Los fenómenos geográficos se desarrollan de manera continúa sobre una extensión de la superficie terrestre (en, sobre o bajo ella), y ha sido un desafío constante crear un modelo de representación tan simple para ser almacenado, procesado y visualizado con facilidad y tan complejo que permita perder el mínimo de información crítica y versátil que permita mantener el nivel de detalle proporcional a la riqueza del mundo real.

Ese modelo se ha denominado **modelo ráster** y es el tema principal del presente libro, su origen, fundamentos, propiedades y algunos usos serán descritos en detalle.

Para ello, se basará en la **Plataforma R** y **RStudio**, unas de las herramientas informáticas más poderosas en la actualidad.

Desde la instalación de las aplicaciones, pasando por una guía básica de su utilización, hasta una detallada descripción del trabajo con dicho modelo.

ISBN: 978-956-404-972-4

Ráster con R

Danilo A . Verdugo Chaura

Ráster con R

Esto va de Filas, Columnas, Cotas y Píxeles

Danilo A . Verdugo Chaura

DANILO A. VERDUGO CHAURA

RÁSTER CON R

ESTO VA DE FILAS, COLUMNAS, COTAS Y PÍXELES

ISBN: 978-956-404-972-4 (Versión papel)

Inscrito en el registro de propiedad intelectual en el Departamento de Derechos Intelectuales (DDI), Servicio Nacional del Patrimonio Cultural.

Isabel, Alonso y Sofía, gracias por su apoyo y paciencia.

Índice general

Índice de figuras

Índice de cuadros

NOTA DEL AUTOR: El libro que tienes en tus manos a sido escrito, diseñado, corregido y editado *íntegramente* con la aplicación **RStudio** (RStudio Team [2020]), usando el estándar **Rmarkdown** (Allaire et al. [2021], Xie et al. [2018a], Xie et al. [2020]) basado en el increíble trabajo del proyecto LaTeX, utilizando el diseño y formato del proyecto **tufte** (Xie and Allaire [2021]). Las citas bibliográficas se mantienen y formatean con el excelente programa de código abierto *JabRef* (JabRef Development Team [2021]).

Prefacio

Mensaje Preliminar al libro

El presente texto existe gracias a la buena y muy sana costumbre de *documentar* todos los procesos, tareas, búsquedas y reflexiones acerca de un tema académico o de un proceso práctico; cada nota debe ser descrita en tus propias palabras y siempre asociando las fuentes. Así contarás con material suficiente para enfrentar cualquier objetivo que te impongas.

Área temática del libro

Los fenómenos geográficos se desarrollan de manera continúa sobre una extensión de la superficie terrestre (en, sobre o bajo ella), y ha sido un desafío constante crear un modelo de representación tan simple para ser almacenado, procesado y visualizado con facilidad y tan complejo que permita perder el mínimo de información crítica y versátil que permita mantener el nivel de detalle proporcional a la riqueza del mundo real.

Ese modelo se ha denominado **modelo ráster** y es el tema principal del presente libro, su origen, fundamentos, propiedades y algunos usos serán descritos en detalle.

Para ello, se basará en la **Plataforma R** y **RStudio**, unas de las herramientas informáticas más poderosas en la actualidad.

Desde la instalación de las aplicaciones, pasando por una guía básica de su utilización, hasta una detallada descripción del trabajo con dicho modelo.

Organización y Características del Libro

Este libro es una introducción a conceptos, técnicas y aplicaciones de los modelos ráster. Este libro se centra en la creación de modelos, descripción y propiedades cubriendo conceptos teóricos y prácticos.

Todos los capítulos presentan nuevos conceptos que se ilustran con casos prácticos utilizando datos reales y lo más actualizados posibles. En cada capítulo se presenta un área del conocimiento geográfico y como un modelo ráster se adapta para su uso particular. Los pasos para cumplir con este objetivo se implementan mediante el uso de la herramienta R y es presentado en cuadros con código debidamente comentados. Esto permite al lector aprender resolviendo problemas que pueden fácilmente ser generalizados a otros problemas.

Este libro no pretende cubrir todo el conjunto de usos y técnicas ni mucho menos proporcionar una colección completa de referencias.

Actualmente, el uso de los modelos ráster es un ***campo creciente y emergente***, por lo que se anima a los lectores a buscar permanentemente métodos y referencias utilizando palabras clave en la red.

Público objetivo

Este libro está dirigido a estudiantes de *pregrado* y *graduados recientes* de disciplinas técnicas. Además, este libro también está dirigido a profesionales que sigan cursos cortos de educación continua y a investigadores de diversas áreas de las geociencias que se encuentren siguiendo cursos de autoaprendizaje.

Se requieren habilidades básicas en **informática**, **matemáticas** y **estadística**, como conocimientos básicos en ciencias geomáticas y geográficas. El código de programación en R siempre es beneficioso, sin embargo, incluso si el lector es nuevo en R, no debería ser un problema, ya que adquirir los conceptos básicos son manejables en un corto período de tiempo. Toda la evaluación del código ha sido realizada en plataformas **Windows**, aunque se insta a los usuarios explorar las versiones disponibles para otros

SO.

Usos previos de los materiales

Partes de los materiales presentados se han utilizado en clases de pre grado de *Sensores Remotos y Teledetección* en la Universidad Tecnológica Metropolitana de Santiago de Chile.

Usos sugeridos del libro

Este libro se puede utilizar en cualquier curso de introducción a las geociencias y/o geoinformática. También servirá de apoyo al enfoque basado en áreas temáticas y casos reales para introducir nuevos conceptos.

Las soluciones implementadas en código R desarrolladas para diferentes problemas son un buen conjunto de ejercicios para los estudiantes. Además, estos códigos pueden servir **como base** cuando los estudiantes enfrenten proyectos más grandes.

Recursos suplementarios

Este libro va acompañado de un conjunto de archivos de datos, nativos y preprocesados necesarios para resolver los casos prácticos. El material se encuentra disponible en el siguiente repositorio de GitHub:

github.com/daniloverdugo/raster_con_R

Reconocimientos

Me gustaría agradecer tanto a mi familia por su amor y apoyo durante todo el tiempo transcurrido que pasé al escribir este texto y más difícil aún, cuando no escribía este libro.

También deseo expresar un público reconocimiento a los excelentes profesionales y estimados amigos que han aportado su tiempo valioso y ordenados estrictamente por alfabeto, Gisella Araya, Nicolás Cea, Adrián Morgado, Francisco Páez, Tomás Pizarro, Diego Soza, Patricio Lamparein y Ana María Marangunic.

Todas sus valiosas observaciones, su cuidadosa revisión, han logrado convertir una serie de textos a una obra práctica y de

utilidad general. Les ofrezco mi más profundo y sincero agradecimiento por su tiempo, que hoy es uno de los tesoros más grandes que disponemos.

Retroalimentación

Los comentarios y críticas constructivas del lector son muy apreciados. Para informar errores, sugerencias o comentarios por favor escribir: *dynageo@gmail.com*.

Danilo Verdugo, Malloco, Chile

I PLATAFORMA R

Antecedentes Generales

1.1 Historia y Origen

R es un *sistema* o una *plataforma* que surge de un entorno computacional diseñado originalmente para el cálculo estadístico. Nace como un dialecto de S el cual es un lenguaje de programación desarrollado por John Chambers (Figura 1.1) y otros en la antigua "Bell Telephone Laboratorios", originalmente miembro de AT & T Corp. S se inicia en 1976 como un programa interno de análisis estadístico. En 1988 el sistema fue reescrito desde FORTRAN a C y empezó a parecerse al sistema que tenemos hoy en día (esto fue en la versión 3 del lenguaje S). Lo importante de destacar es el espíritu original por el cual fue diseñado el sistema: facilitar el análisis de datos, primero para ellos y eventualmente, para otros.

Figura 1.1: John Chambers creador de S.

John Chambers describe el anhelo al momento de diseñar el lenguaje S con que "los usuarios comenzaran en un entorno interactivo, donde no pensaran conscientemente en sí mismos como programadores. Entonces, cuando sus necesidades fuesen más claras y su sofisticación aumentara, deberían ser capaces de transitar gradualmente a la programación, cuando lenguaje y sistema serían más importantes". La clave fue la transición de usuario a desarrollador. Ellos querían construir un lenguaje que fácilmente podría dar servicio a ambos tipos de usuario. Técnicamente, necesitaban construir un lenguaje que fuese adecuado para el análisis de datos interactivo (basada en línea de comandos), así como para la escritura de programas (como los lenguajes tradicionales de programación).

Figura 1.2: Ross Ihaka y Robert Gentleman, los creadores de R.

En 1991, Ross Ihaka y Robert Gentleman (Figura 1.2) en el Departamento de Estadística de la Universidad de Auckland crean R. Inspirado en la filosofía de S pero como un proyecto totalmente abierto. En 1993, se hace público. La experiencia de Robert y Ross sobre el desarrollo de R está documentada en un interesante artículo de 1996 en el Journal of Computational and Graphical Statistics [Ihaka and Gentleman, 1996].

Hoy R se ejecuta en casi cualquier sistema operativo o plataforma informática estándar ¡incluso en una playstation 3! Una característica interesante en muchos proyectos de código abierto son las versiones frecuentes. Estos días se produce una liberación anual importante, por lo general en octubre, donde se incorporan nuevas características principales y lanzadas al público. A lo largo del año, se realizarán lanzamientos de corrección de errores en pequeña escala, según sea necesario. Los lanzamientos frecuentes y ciclo de liberación regular indican un permanente y activo desarrollo del software donde se asegura que los errores serán tratados de una manera oportuna. Por supuesto, el árbol principal de código fuente de R se encuentra bajo el control de un pequeño grupo de desarrolladores. Al momento de escribir estas líneas (mayo 2021) nos encontramos en la versión 4.0.5 o **Shake and Throw**.

Es interesante notar el dato anecdótico que los nombres propios con que se han designado desde la primera versión liberada *Great Pumpkin* (versión 2.14.0, noviembre 2011) o *Trick or Treat* (versión 2.15.2, octubre 2012) entre muchas otras hacen referencia a un capítulo específico de la serie animada *Peanuts* (Figura 1.3).

Figura 1.3: Serie animada Peanuts (Snoopy) basada en los comics de Charles Schulz.

Otra de las ventajas clave que R posee avanzadas capacidades gráficas con *calidad de publicación* cuya existencia se ha asegurado desde el principio del proyecto mediante el control muy fino en todos los aspectos de la composición gráfica.

1.2 Diseño de la Plataforma R

El éxito alcanzado por el proyecto R no tiene nada que ver con las herramientas en sí mismas, sino más bien con lo activa que sea la comunidad de usuarios, hoy miles de ellos en todo el mundo se han unido para realizar contribuciones como también ayudar a otros a usar R para todo tipo de nuevas e insospechadas aplicaciones.

El sistema R base o core o núcleo de la plataforma se compone de las herramientas mínimas para su funcionamiento que se encuentra disponible para varios sistemas operativos: Linux, Windows, Mac e incluso su Código Fuente en la denominada *Red Exhaustiva de Archivos R* (Comprehensive R Archive Network, CRAN.

cran.r-project.org, es una red de servidores ftp y web en todo el mundo que almacenan versiones idénticas y actualizadas de código y documentación para R)

1.3 RStudio

Aunque R posee un diseño muy avanzado e inteligente a medida que se realizan rutinas para análisis de datos, gráficos, o se generan archivos temporales de esos mismos procesos o se descargan e instalan diversos paquetes de código adicionales se hace muy difícil de mantener y menos aún generar procedimientos claros, organizados y replicables en distintos entornos informáticos,

siendo esta última capacidad crítica para la aproximación científica al análisis de datos. RStudio también es una herramienta libre que hace más fácil el trabajo con R, van der Loo and De Jonge [2012] definen sus principales características como:

- Editor de texto, explorador de archivos, visualizador de gráficos todo en el mismo entorno.
- Trabaja directo con una instalación subyacente de R.
- Organiza el código y mantiene múltiples proyectos.
- Mantiene mi investigación reproducible.
- Mantiene los paquetes en la instalación de R.
- Crea y comparte reportes.
- Comparte el código y colabora con otros usuarios.

RStudio hoy es una aplicación y una fundación dedicada al soporte de sus productos de código abierto [1] como al servicio comercial de capacitación y consultoría en temas estadísticos y R. Su fundador J.J. Allaire también inventa el lenguaje de programación web *ColdFusion*, *Windows Live Writer* (programa para escribir y publicar blog), *FitNow* y *LoseIt* ambas aplicaciones móviles para ejercicio y pérdida de peso entre otros proyectos tecnológicos.

[1] www.rstudio.com

1.4 Características Principales

Algunas propiedades que hacen de RStudio el editor ideal para trabajar con R (tomado de van der Loo and De Jonge [2012]):
Integración con la consola R: Escribe comandos directamente en R dentro de RStudio.
Ejecución de código: Ejecuta código directo desde el editor.
Paréntesis inteligente: Cierre automático de paréntesis, ilumina selección, cierre de comillas automático. Termina de escribir la palabra previamente iluminada de un menú intellisense.
Ayuda en línea: Acceso directo a la ayuda de lenguaje y sintaxis. ***Atajos de teclado:*** Tareas repetitivas son asociadas a combinación de teclas. Integra Ayuda, Permite navegar y buscar en las páginas de ayuda nativas de R.
Explorador de objetos e Historia: Se puede inspeccionar cada objeto creado en la sesión actual de R y su explorador de

historia permite recorrer los comandos utilizados desde el actual hasta el primero de la sesión.
Navegar por el código: Saltar entre funciones, llamadas y reportes, visor de datos y visualización tipo grilla para explorar el contenido de los objetos en la sesión actual.
Menús de Importar datos: Para los tipos más comunes de archivo, ofrece un sistema de menú que genera el código R necesario. Integración gráfica con zoom, paneo y capacidades de exportación.
Gestión de proyectos: Facilita el manejo y control de varios proyectos. Control de versión e integra los sistemas de control git y svn.
Generación de Documentos: Genera pdf, html y otros formatos de reporte usando RMarkdown, Sweave o knitr.
Publicación: Publicar reportes y rutinas directo a la web de *Rpubs.com.*

1.5 Instalación de Plataforma R y RStudio

Primero se debe descargar el paquete de instalación desde el sitio web www.r-project.org siguiendo el link **CRAN** y buscar en el listado de servidores espejo el más cercano a la ubicación del usuario. Luego, seleccionar la versión correspondiente al sistema operativo de la máquina a instalar. A continuación, prestar atención y ubicar el link **install R for the first time** en el apartado *base.*

Finalmente, encontramos el link para la descarga del paquete instalador, siempre será le versión más reciente. Ejecutar el programa y seguir los pasos indicados por el asistente.[2]

Una vez terminada la instalación de R procedemos a instalar el programa RStudio desde el sitio www.rstudio.com y buscar en la lista de productos *RStudio Desktop.* A continuación, seleccionar la versión *Open Source Edition* que es libre de pago. Finalmente seguir la secuencia de botones correspondiente hasta llegar al punto de descarga del paquete de instalación. Poner especial atención en seleccionar el que corresponda a su versión de sistema operativo. Ejecutar el programa y seguir los pasos indicados

[2] Es importante señalar que el *nombre de usuario* de la sesión Windows donde se realizará la instalación **no existan** espacios e idealmente no contenga tildes. De ser así, es recomendable crear una nueva sesión con un nombre sin espacios ni tildes.

por el asistente.

1.6 Consideraciones Preliminares

El entorno de trabajo de RStudio posee una configuración por defecto que presenta una distribución de funcionalidades orientada al uso como entorno interactivo de cálculo y proceso de datos mediante línea de comandos contenido en la ventana *Console* y una vista general de los objetos y variables creadas en la sesión actual en la ventana *Environment* (Figura 1.4).

El tipo de trabajo descrito en el presente texto utiliza metodologías más complejas y extensas que trabajar solo como calculadora o proceso de datos. Por lo tanto, vamos a sugerir una nueva distribución de ventanas que optimicen el trabajo en un tipo de archivos que permiten ir escribiendo largas secuencias de comandos que van desarrollando las tareas de manera secuencial y por seguridad buscamos grabar en disco un archivo que la próxima vez me permita repetir dicha secuencia y continuar o modificar según el caso. Los archivos creados tendrán la extensión *.R* y los denominaremos *scripts*.

Una mejor disposición de ventanas es la sugerida en la figura 1.5. Donde se privilegia el panel izquierdo para extender a altura completa el editor de texto y maximizar el área disponible para escribir y editar scripts y mantiene el panel derecho la consola con su línea de comando.

Otro aspecto importante es el uso de un color de fondo obscuro que reduce la fatiga visual cuando se sienta frente al monitor durante períodos muy prolongados.

Otra característica configurada por defecto es que al momento de salir de una sesión, es guardar el *espacio de trabajo* lo que significa que se crea una imagen de las variables y funciones actuales en un archivo llamado ".RData".

Cuando se vuelve a abrir R desde el mismo directorio de trabajo, el espacio de trabajo se cargará y todas estas cosas estarán disponibles. Pero no recomendamos ese comportamiento. Cargar un espacio de trabajo guardado convierte el script cuidadosamen-

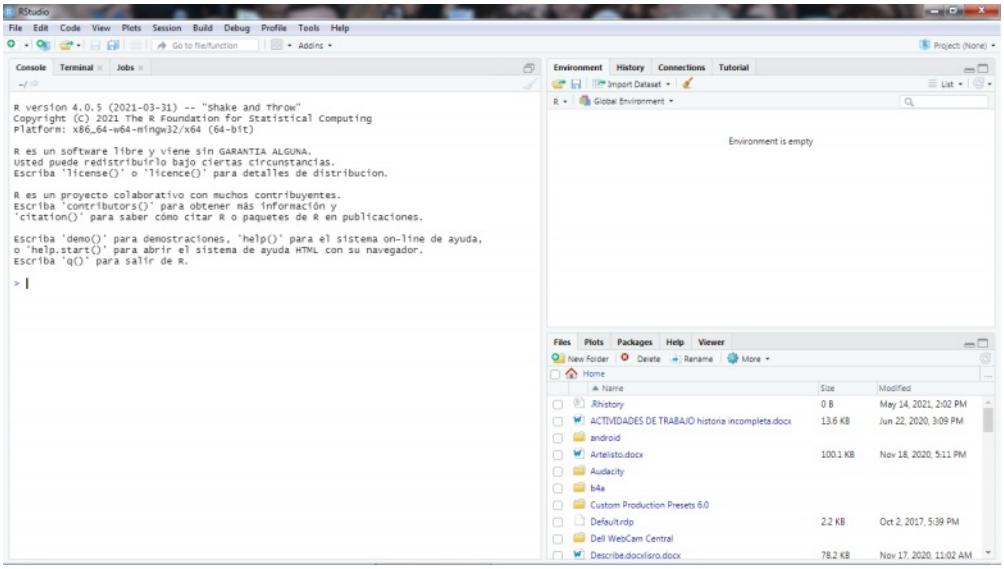


Figura 1.4: Configuración por defecto.

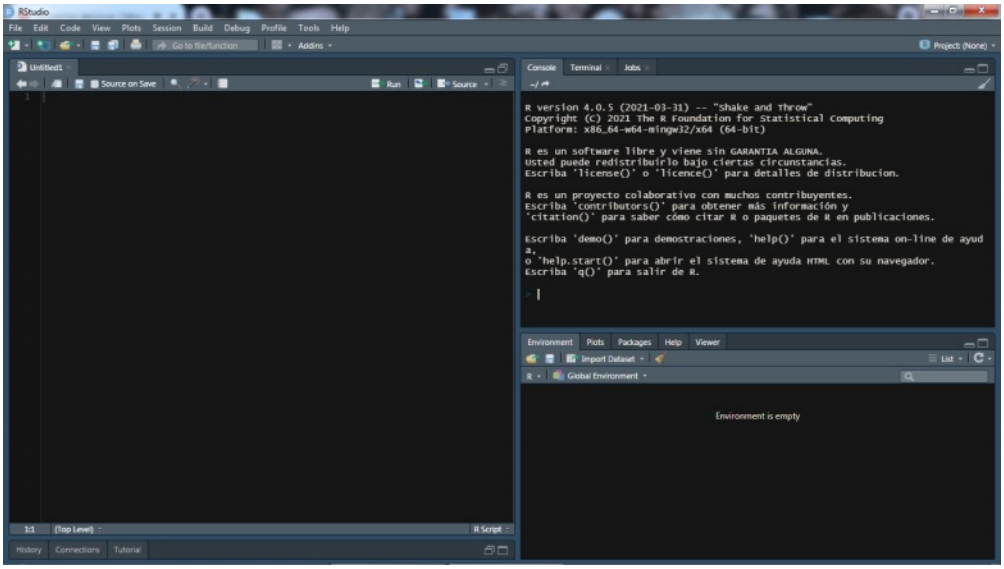


Figura 1.5: Configuración sugerida.

te escrito donde todo sucede lógicamente de acuerdo con un plan a algo parecido a un cajón de sastre, lleno de páginas y cuadernos variados que pueden ser o no pertenecientes al trabajo actual.

Para configurar nuestro programa recomendamos cambiar algunos parámetros en la configuración general de la aplicación, para ellos debemos ir a la opción de menú ***Tools/Global Options***. (Figura 1.6).

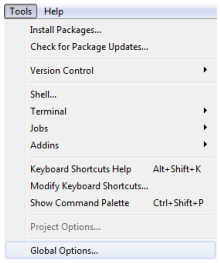


Figura 1.6: Opción de menú para configuración general.

El cuadro de dialogo resume un gran número de parámetros que abarcan detalles del funcionamiento general de RStudio (Figura 1.7).

Los siguientes cambios son necesarios para configurar los aspectos mencionados en punto anterior:

- *General:* Para quitar la carga automática de variables y funciones de la última sesión, debemos en el grupo **Workspace** quitar el check de *Restore .RData into...* y seleccionar la opción ***Never*** de la lista *Save workspce to .RData...* (Figura 1.7).
- *Appearance:* La configuración de colores en el editor de texto y los otros cuadros se selecciona en la lista *Editor theme:* y se debe tener especial cuidado en elegir aquel con un fondo negro o muy obscuro (Figura 1.8).
- *Pane Layout:* La distribución de los espacios disponibles para optimizar el editor de texto y mantener la línea de comando visible se configura siguiendo las opciones en indicadas en la figura 1.9).

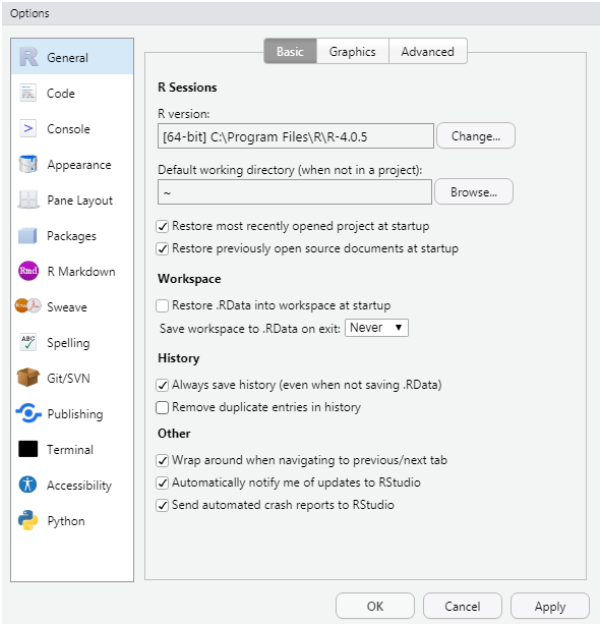


Figura 1.7: Panel de configuración general.

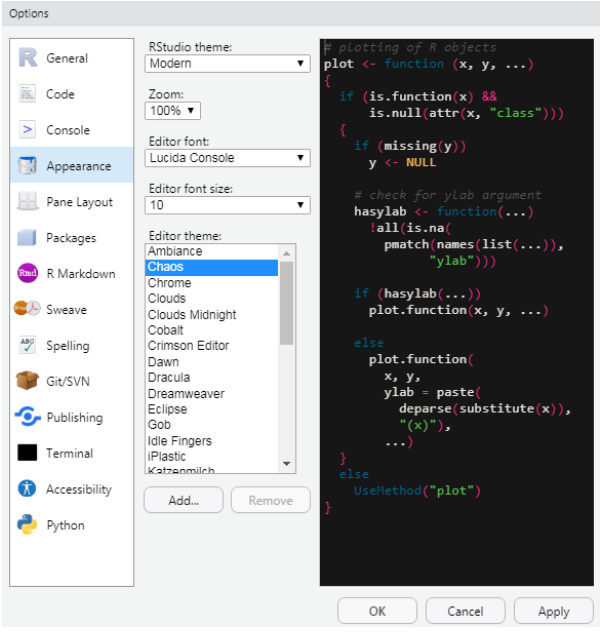


Figura 1.8: Selección para configuración de colores.

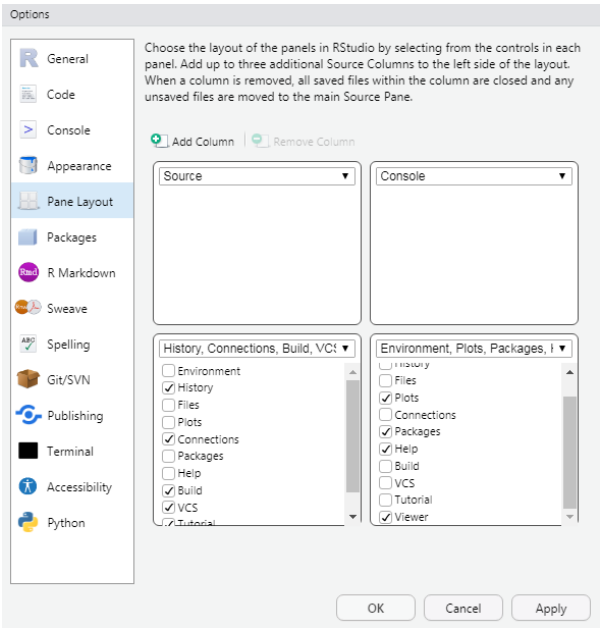


Figura 1.9: Configuración de paneles y cuadros.

Paquetes

El sistema base, tal como se describe en 1.2, es capaz de ejecutar rutinas y código encapsulado en forma de módulos o paquetes (*packages*) que amplifican y diversifican las posibilidades de cómputo, gráfica, conexiones remotas, generación de documentos y un gran, gran etc.

El **sistema base o core** también se forma de un conjunto de paquetes.

Además de las funciones más fundamentales el sistema base incluye los paquetes utils, stats, datasets, graphics, grDevices, grid, methods, tools, parallel, compiler, splines, tcltk, stats4 y algunos paquetes *recomendados*: boot, class, cluster, codetools, foreign, KernSmooth, lattice, mgcv, nlme, rpart, survival, MASS, spatial, nnet, Matrix.

Cuando se descarga una instalación nueva de R desde CRAN, se obtiene todos los paquetes mencionados, que representa un gran porcentaje de la funcionalidad del sistema.

1.7 Búsqueda

Además de la gran cantidad de paquetes instalados por defecto, existe una gran cantidad de paquetes opcionales disponibles.

- A noviembre 2021 existen 18.407 paquetes en CRAN que han sido desarrollados por los usuarios y programadores alrededor del mundo cada uno de ellos diseñado para un campo en especial.

- También hay muchos paquetes asociados con el proyecto **Bioconductor**[3], (detalles en 9.8.1).
- Las personas a menudo hacen sus propios paquetes y los hacen disponibles en sus sitios web personales; en realidad no hay manera confiable de realizar un seguimiento de cuántos paquetes están disponibles de este modo.
- Hay una serie de paquetes que se desarrollan en los repositorios de GitHub y BitBucket pero no existe ninguna lista fiable de todos estos paquetes.

[3] www.bioconductor.org

El mejor método para buscar en la web, nuevos paquetes es utilizar el sitio rseek.org (Figura 1.10) usando palabras claves y potenciado por el motor de búsqueda de Google.

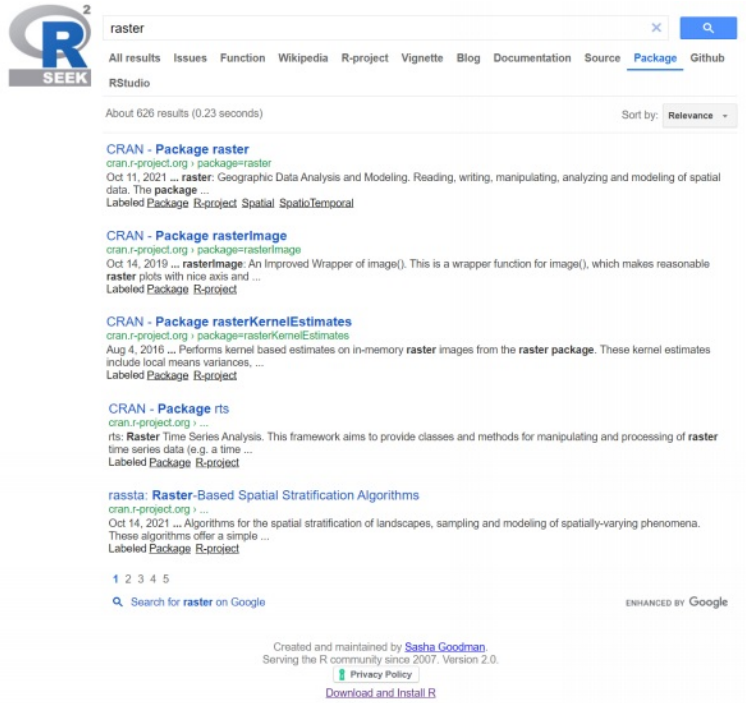


Figura 1.10: Sitio web de búsqueda de contenido asociado a R, modo 'Package'.

Generalmente los resultados siempre serán múltiples y variados.

1.8 Instalación

Una vez identificado un paquete que ofrece las funcionalidades deseadas se debe instalar en nuestro ambiente R y dejarlo disponible para cuando se necesite. El entorno RStudio cuenta con un panel específico (Figura 1.11).

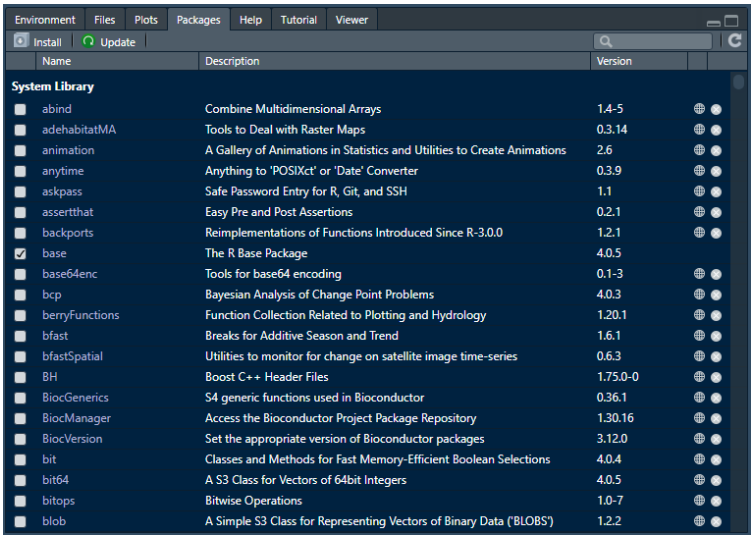


Figura 1.11: Panel Packages.

Que junto al listado de paquetes ya instalados se dispone de opciones para instalar Install y actualizar Update (1.9) los existentes.

1.8.1 Cuadro Diálogo

El botón **Install** levanta un cuadro de diálogo con las opciones de configuración (Figura 1.12) para instalar paquetes disponibles en CRAN o los paquetes descargados en formato *.zip* de sitios no oficiales.

1.8.2 Función install.packages()

Un segundo método de instalación es utilizar la función R escrita directamente en la consola y reemplazando *nombre* por el paquete deseado.

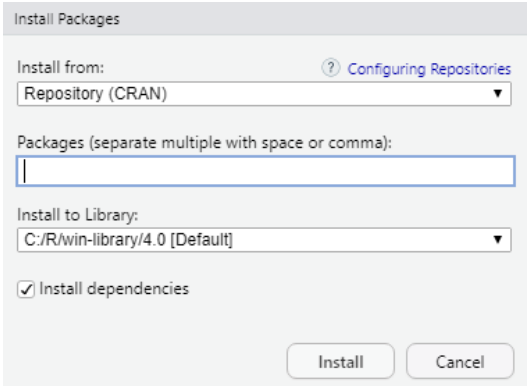


Figura 1.12: Opciones de Instalación vía RStudio.

```
install.packages("nombre")
```

1.9 Actualización

El botón Update del panel paquetes levanta el cuadro de diálogo diseñado para la selección de los paquetes a actualizar. La lista solo muestra aquellos disponibles de actualización (Figura 1.13).

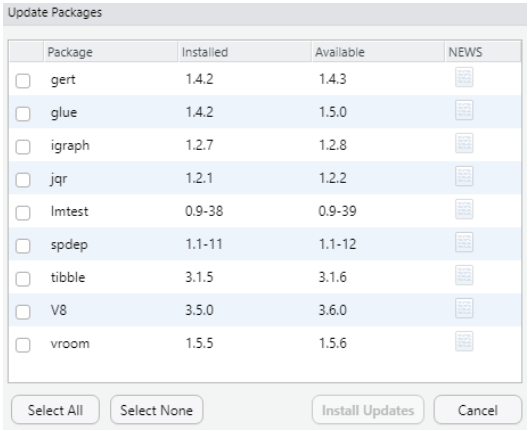


Figura 1.13: Opciones de Instalación vía RStudio.

Sintaxis Básica de R

1.10 Operaciones Numéricas

De acuerdo con la configuración de ventanas descrita en el punto anterior la ventana *Console* se encuentra en la esquina superior derecha junto a las ventanas *Terminal* y *Jobs*. En ella trabajaremos interactivamente mediante la línea de comando, su uso puede ser comparado con el uso de una calculadora electrónica que siempre está a la espera del ingreso de una instrucción a continuación del símbolo >.

La dinámica de uso es ir ingresando el texto que define la operación o instrucción a realizar y finalizar con la tecla **Intro**:

```
1 + 2 + 3
```

Y R responde con la evaluación inmediata de la operación:

```
[1] 6
```

Los espacios en blanco son removidos automáticamente y la orden es evaluada normalmente:

```
1+2          +        3
```

```
[1] 6
```

Cuando la instrucción se extiende por más de una línea, R se encarga de interpretar la estructura y completitud para ser evaluada:

```
12 + 3 -
  5 +
  5
```

```
[1] 15
```

También puedes usar el símbolo 'punto y coma' (*;*) para separar secuencias de instrucciones, pero escritas en la misma línea de código.

El editor ofrece algunas facilidades al usuario como el *historial de órdenes* de las instrucciones digitadas en orden reverso para evitar el retipeo de una instrucción muy extensa. Recorrer con las flechas verticales hasta encontrar el texto buscado y solo editar los cambios requeridos. También cuenta con un potente *motor de finalización inteligente* que permite obtener ayuda a medida que se escribe nombre de funciones o comandos y cuando se trabaja con scripts también nombres de funciones o variables. Por último, para limpiar todo el texto escrito presionar la tecla *escape*.

Se pueden utilizar los símbolos aritméticos tradicionales +, -, *, / y ^ [4]. Además se encuentran disponibles las funciones *log*, *exp*, *sin*, *cos*, *tan*, *sqrt* entre muchas otras bien conocidas. También existe el valor de *pi*.

[4] El acento circunflejo ^ se utiliza para la potenciación (se obtiene combinando las teclas alt+94) también se puede reemplazar con **. Además, existe la *operación módulo* (%%) y la *división entera* (%/%).

```
1 + 2 - 3 * 4 / 5**6; sin(0.33); acos(0.253) ; sqrt(4)
```

```
[1] 2.999232
[1] 0.324043
[1] 1.315016
[1] 2
```

```
9 %% 4
```

```
[1] 1
```

```
13.5 %/% 2
```

```
[1] 6
```

```
pi
```

```
[1] 3.141593
```

También se debe tener en cuenta que los números muy grandes o pequeños serán automáticamente convertidos a notación científica.

```
1/100000000000000000
```

```
[1] 1e-17
```

```
1*100000000000000000
```

```
[1] 1e+17
```

En operaciones más complejas se recomienda el uso de *()* para fijar el orden o *precedencia* de las operaciones. Se acepta *anidar* los paréntesis (()).

```
3 * 5 + 1
```

```
[1] 16
```

```
3 * (5 + 1)
```

```
[1] 18
```

```
1 + (3 + (1 + 2) / 3)
```

```
[1] 5
```

Las operaciones lógicas se realizan utilizando ciertos símbolos o combinaciones de ellos para obtener resultados lógicos:

```
2 == 3
```

```
[1] FALSE
```

```
2 < 3
```

```
[1] TRUE
```

Revisa cuadro 2.1 para una lista completa de los caracteres y su significado en las operaciones lógicas.

Otros componentes del lenguaje R son: valor lógico verdadero *TRUE* y falso *FALSE*, infinitud positiva **Inf** y negativa **-Inf**, no disponible **NA**, no es un número **NaN** y nulo o vacío **NULL**.

Finalmente, no podemos intercambiar mayúsculas o minúsculas al momento de escribir los nombres de instrucciones (por ej. *sin* no es igual a *Sin*). Y en R todo es un *objeto* que pertenecen a cierto tipo de *clases*. Así cada uno de ellos tendrá comportamientos y usos diferentes.

```
class(1)
```

```
[1] "numeric"
```

```
class("palabra")
```

```
[1] "character"
```

```
class(sqrt)
```

```
[1] "function"
```

```
class(1 != 1)
```

```
[1] "logical"
```

```
class(TRUE)
```

```
[1] "logical"
```

```
class(Sys.Date())
```

```
[1] "Date"
```

1.11 Asignación

El proceso de almacenar cualquiera de los objetos ya descritos, o el resultado de las operaciones en memoria RAM y hacerlo persistente a lo largo de la sesión actual de trabajo se denomina *asignación*. Resultados intermedios se deben ir asignado con un nombre fácilmente identificable y memorizables.[5]

[5] Los nombres de los objetos deben en lo posible *resumir* el contenido, evitar nombres *genéricos* como *a* u *X* y no ambiguos como *variable* o *constante*. Pueden contener letras, números, puntos y guiones o *barra baja* (_). Siempre debe comenzar con una letra o punto.

Es conveniente adoptar un estilo de escritura para los nombres de los objetos y para ellos existen varios métodos entre los cuales podemos destacar:

- ***lowercase:*** todominusculas.
- ***UPPERCASE:*** TODOMAYUSCULAS.
- ***UpperCamelCase:*** PrimeraMayusculaSiguientesBajas.
- ***lowerCamelCase:*** primeraMinusculaSiguientesMayusculas.
- ***period.separated:*** separada.por.punto.
- ***snake_case:*** separada_por_guion_bajo.

En el sitio web style.tidyverse.org donde se discuten buenas prácticas para el uso de R se recomienda el último tipo.

La forma de realizar una asignación tiene la forma general:

$$nombre = valor \tag{1.1}$$

Donde *nombre* es el identificador o nombre del objeto o variable que se busca crear y almacenar en memoria, = el operador

de asignación (no confundir con ==) y *valor* es el contenido a almacenar.[6]

```
radio_estimado = 135.45
```

nombre será sintácticamente válido cuando consta de letras, números o punto y comienza con **una letra o el punto**, pero *no seguido de un número*. Los nombres como ".2var" no son válidos, ni tampoco las palabras reservadas.

En la actualidad el uso <- o = es casi indistinto, pero en casos muy específicos (y técnicos) existen diferencias. En nuestro libro recomendamos el uso de <- ya que es un poco más cómodo de tipear que el carácter =.

```
variable <- (112.33 - 45) / 105
```

Una vez escrita la asignación y presionado *intro* se realiza la operación, se crea el objeto en memoria con nombre *variable* y la línea de comando se limpia. Para conocer el resultado de la operación podemos:

[6] En R se ha utilizado tradicionalmente como operador de asignación <- que viene desde la primera versión de R, dado que su origen proviene del lenguaje **S** y a la vez éste fue inspirado por el lenguaje *APL* (A Programming Language, *Kenneth Iverson en 1957-62*) que utilizaba dicho carácter ya que el símbolo = se utilizaba solo para prueba de igualdad.

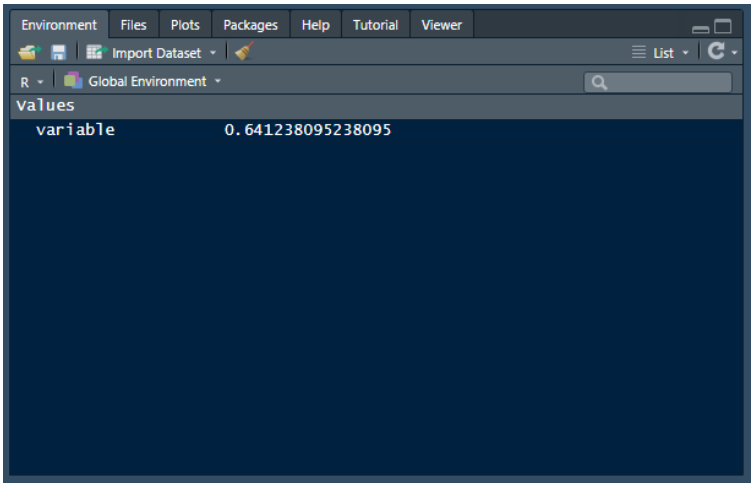


Figura 1.14: Panel Environment.

1. Buscar el resultado en el panel ***Environment*** en la esquina inferior derecha (Figura 1.14).

2. Escribir la asignación dentro de un juego de paréntesis redondos () que obliga imprimir el resultado de la evaluación de la

expresión.

```
(variable <- (112.33 - 45) / 105)
```

```
[1] 0.6412381
```

3. Volver a escribir el nombre y presionar intro.

```
variable
```

```
[1] 0.6412381
```

4. Utilizar el comando *print()*.

```
print(variable)
```

```
[1] 0.6412381
```

El principal objetivo de almacenar en memoria es la reutilización de dichos contenidos:

```
radio_inicial <- 15.43 + 36.315 * (3 * 5)
(area_base <- pi * radio_inicial ** 2)
```

```
[1] 985748.9
```

Al asignar un nuevo contenido con el mismo nombre, el anterior se elimina de memoria y se crea un nuevo objeto.

```
x <- 5
x <- "hola"
x
```

```
[1] "hola"
```

El comando `ls()` me permite listar todos los objetos almacenados en mi sesión actual y `rm(list = ls())` borrará la totalidad de objetos de mi sesión actual.

1.12 Naturaleza Vectorial

En todas las ocasiones que hemos escrito una operación aritmética o impreso un resultado R crea automáticamente un *vector*[7] con una longitud que corresponde a los valores por almacenar.

Si estudiamos el ejemplo siguiente:

[7] Un ***Vector*** es el tipo de objeto más básico del lenguaje R. Un *vector* puede contener cero o más objetos, y siempre serán de la misma clase.

```
5
```

```
[1] 5
```

```
ls()
```

```
[1] "area_base"     "radio_inicial" "variable"      "x"
```

Vemos un [**1**] antecediendo al número 5 que indica que se trata del primer (y único en este caso) elemento, mientras que en el caso del comando *ls()* la lista de respuesta por cada renglón que se extienda en la consola antecederá su correspondiente [] señalando la posición en el vector generado automáticamente.

1.12.1 Función `c()` para crear vectores

La creación manual de vectores usando elementos individuales se realiza mediante la función `c()`.

En el siguiente ejemplo hemos incorporado dos nuevos elementos:

- `;` *(punto y coma)* que nos permite separar dos instrucciones en la misma línea.
- `#` *(almohadilla o numeral)* sirve para crear COMENTARIOS: textos que no serán evaluados por R.

```
c(1, 4, 8, 11) #Sin asignar a nombre de objeto
```

```
[1]  1  4  8 11
```

```
x <- c(2, 4, 6, 8); print(x)
```

```
[1] 2 4 6 8
```

```
y <- c(x, 100, 101, 102); y
```

```
[1]   2   4   6   8 100 101 102
```

```
z <- c("azul", "rojo", "verde", "amarillo"); (z)
```

```
[1] "azul"     "rojo"     "verde"    "amarillo"
```

*1.12.2 Uso de **Secuencias** para crear vectores*

Una secuencia numérica se construye usando el operador `:`. Por ejemplo:

```
1:30
```

Es equivalente a `c(1, 2, 3, ... , 30)`, también es capaz de crear secuencias negativas y/o usar valores decimales. Comprueba las siguientes operaciones y sus resultados:

```
30 : 1
-10 : -8
1.5 : 5
```

La prioridad del comando `:` es máxima y siempre será evaluado al inicio. Compara las secuencias obtenidas entre `1:5-1` y `1:(5-1)`.

*1.12.3 Usar **Repetición** para la creación de vectores*

El comando `rep()` genera un vector con la repetición de un valor un número determinado de veces. También se puede repetir un texto, una fecha o una secuencia definida con la opción **:**.

Revisar usando alguna de las formas de ayuda en línea los argumentos, nombres y usos para el comando *rep*:

```
rep(1, 5); rep('a', 5)
```

```
[1] 1 1 1 1 1
[1] "a" "a" "a" "a" "a"
```

```
rep(c(1:3, 7), 3); rep(x = 1:2, times = 3, each=3)
```

```
 [1] 1 2 3 7 1 2 3 7 1 2 3 7
 [1] 1 1 1 2 2 2 1 1 1 2 2 2 1 1 1 2 2 2
```

*1.12.4 Uso de **Patrones** para crear vectores*

El comando `seq()` genera patrones regulares. Posee múltiples parámetros o *argumentos con nombre* para configurar el patrón. Los dos primeros *desde* **(from)**, *hasta* **(to)** tendrían el mismo comportamiento que los parámetros del comando **:**, un tercero *paso* **(by)** define el incremento y posee un valor por defecto de *1*

y un cuarto argumento define el número de elementos que debe contener la secuencia *(length.out)*[8].

En los ejemplos siguientes vemos el uso del comando usando los argumentos en orden preciso ***sin utilizar sus nombres***:

```
seq(1, 10) #from=2, to=10
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

```
seq(1, 10, 2) #from=2, to=10, by=2
```

```
[1] 1 3 5 7 9
```

También al poseer valores por defecto se puede abreviar su uso:

```
seq(to = 5)
```

```
[1] 1 2 3 4 5
```

```
seq(from = 3)
```

```
[1] 1 2 3
```

Pero la verdadera utilidad de los argumentos con nombres es la libertad en el orden de uso:

```
seq(to = 9, from = 1)
```

```
[1] 1 2 3 4 5 6 7 8 9
```

```
seq(by = 1.5, from = 1, to = 9)
```

```
[1] 1.0 2.5 4.0 5.5 7.0 8.5
```

```
seq(from = 10, by = -3, length.out= 5)
```

```
[1] 10  7  4  1 -2
```

```
seq(by = 3, length.out= 6)
```

```
[1]  1  4  7 10 13 16
```

La misma lógica se aplica al trabajo con fechas:

```
seq(as.Date("1910/1/1"), as.Date("1912/1/1"), "years")
```

```
[1] "1910-01-01" "1911-01-01" "1912-01-01"
```

[8] La mayoría de funciones en R poseen parámetros o argumentos con nombre que permiten ajustar el funcionamiento. El orden y sus nombres lo encuentras usando la ayuda en línea. En R el sistema es muy poderoso y fácil de utilizar, basta ubicar el cursor sobre el nombre de comando y presionar **F1** y un panel con información detallada con:
Descripción: Breve descripción de funcionamiento.
Uso: Sintaxis con el orden y nombres de argumentos.
Argumentos: Lista detallada y en orden correspondiente de los argumentos.
Detalles: Información adicional.
Valor: El objeto que retorna.
Ejemplos: Código R con alternativas de uso.
Será desplegada en el panel inferior derecho. También puedes escribir en la consola el símbolo de interrogación y a continuación el nombre del comando: `?seq` o usar el comando `help('seq')`.

```
seq(from= as.Date("2000/1/1"), by = "month", length.out = 3)
```

```
[1] "2000-01-01" "2000-02-01" "2000-03-01"
```

```
inicio <- as.Date("1999-12-17")
fin <- as.Date("2000-3-7")
seq(from= fin,
    to= inicio,
    by = "-1 month")
```

```
[1] "2000-03-07" "2000-02-07" "2000-01-07"
```

1.12.5 Vectores Lógicos y Caracteres

Además de los vectores numéricos y fechas R soporta del tipo **lógico** y **texto**. En el caso de vectores lógicos solo puede tomar dos valores: **FALSE** (falso) y **TRUE** (verdadero), también son válidos **F** y **T**.
Generalmente son construidos a partir del uso de los operadores lógicos (ver Cuadro 2.1) en *condiciones*. Por ejemplo:

```
enteros <- 1:6
(prueba <- enteros <=3)
```

```
[1]  TRUE  TRUE  TRUE FALSE FALSE FALSE
```

Donde `prueba` almacena un vector de la misma longitud que enteros y cuyo contenido será tanto *True* o *False* dependiendo si o no cumplen la condición.
También vectores de texto o caracteres pueden ser creados. Los textos o frases deben ser escritos entre comillas (") o comillas simples ('), por ejemplo `'Este texto está entre comillas.'`. Generalmente utilizada en la construcción de vectores con `c()`.

```
rep(x=c('x','y'), times= 4)
```

```
[1] "x" "y" "x" "y" "x" "y" "x" "y"
```

Operación Vectorial

1.13 Operaciones Numéricas

Toda operación en R se realiza utilizando los vectores elemento a elemento. Si poseen distinta longitud el resultado tendrá la longitud del mayor y los restantes serán *reciclados*[9].

```
x <- c(100, 800)
y <- 2:7
y + x + 3
```

```
[1] 105 806 107 808 109 810
```

Donde el vector `x` se repite tres veces, el número `3` seis veces todo para igualar la longitud del vector `y`.

Algunas funciones se aplican a cada uno de los elementos y otros devuelven un cálculo que se aplica sobre todo el vector (Ver listado de las más comunes en Cuadro 1.1):

```
x <- 3:6; x
```

```
[1] 3 4 5 6
```

```
x**2 #Elevar al cuadrado
```

```
[1]  9 16 25 36
```

```
sum(x) #Suma el contenido
```

```
[1] 18
```

Podemos mencionar que existen comandos que retornan el vector con un nuevo orden, `sort(x)` que retorna un vector del mismo largo de x ordenados ascendente o descendente vía parámetro `decreasing` y para revertir el orden de los elementos: `rev(X)`

[9] **Reciclar** es repetir un vector de longitud menor tantas veces como sea necesario para alcanzar la longitud del mayor utilizado en la operación. Puede usar una fracción para coincidir la longitud y será informado con un mensaje de alerta.

Cuadro 1.1: Funciones más Comunes

Función	Definición
lenght()	Largo o número de elementos
min()	Mínimo
max()	Máximo
range()	Rango
mean()	Promedio
median()	Mediana
sum()	Suma
prod()	Producto
var()	cuasi-Varianza
sd()	Desviación Estándar

1.14 Valores Faltantes

Puede ocurrir que no todos los componentes de un vector sean conocidos, para R son *valor no disponible* o **NA**. Generalmente una operación que incluya un valor NA dará como resultado un NA. La razón es simple, si no se puede definir o expresar en términos concretos todos los elementos que participan en la operación no se puede conocer o expresar en términos concretos el resultado.

```
x <- c(1, 2, 3, 4, NA, 6)
mean(x)
```

```
[1] NA
```

Para resolver el problema, existe el parámetro `na.rm=TRUE` (remover NA) que precisamente indica que la operación se lleve a cabo e ignore cualquier NA (o NaN[10]):

```
mean(x, na.rm = TRUE)
```

```
[1] 3.2
```

La función `is.na(x)` retorna un vector lógico de longitud similar a X con *TRUE* si corresponde a NA y *FALSE* en caso contrario.

[10] **NaN** corresponden a otro tipo de valores *faltantes*, hace referencia a una expresión numérica *indefinida* como por ej. `0/0` o `Inf-Inf`.

```
is.na(x)
```

```
[1] FALSE FALSE FALSE FALSE  TRUE FALSE
```

Finalmente, no confundir el término *NA* con un texto *"NA"*:

```
is.na(c("NA", NA))
```

```
[1] FALSE  TRUE
```

1.15 Indexado y Subconjunto

R posee varios métodos para acceder a elementos individuales o subconjuntos a través de las *operaciones de indexación*, la forma de alcanzar el ***i-ésimo*** elemento de un vector **x** es mediante la sintáxis **x[i]**.

1.15.1 Índice Numérico Positivo

El índice *i* puede ser un valor, una secuencia o un vector de números enteros positivos[11] que definen los elementos a retornar:

[11] Puede contener índices repetidos.

```
x <- 11:19
x[2]
```

```
[1] 12
```

```
x[c(2, 2, 2, 5:9)]
```

```
[1] 12 12 12 15 16 17 18 19
```

Si el índice es mayor que la longitud del vector retornará NA:

```
x[10]
```

```
[1] NA
```

1.15.2 Índice Numérico Negativo

Si los índices son negativos los valores deben ser *excluidos*:

```
x[-2]
```

```
[1] 11 13 14 15 16 17 18 19
```

```
x[c(-2, -4, -(6:9))]
```

```
[1] 11 13 15
```

1.15.3 Vector Lógico

Debe ser un vector de la misma longitud que el vector a evaluar y contiene valores verdaderos o falsos y serán retornados aquellos que corresponden a valores TRUE[12].

```
x <- 1:5
test <- c(TRUE, TRUE, TRUE, FALSE, FALSE)
x[test]
```

```
[1] 1 2 3
```

Una forma de construir un vector lógico es en respuesta a *condiciones lógicas* donde los elementos que cumplen la condición son devueltos:

```
x[x < 3]
```

```
[1] 1 2
```

```
x[x > 4 | x < 3]
```

```
[1] 1 2 5
```

```
x[x**2 > 4 & x < 5]
```

```
[1] 3 4
```

[12] Para invertir la selección (rescatar los elementos de índices *falsos*) debe aplicar el operador de negación **!**, `x[!test]`.

El proceso inverso que devuelve los *índices* de los elementos que cumplen una condición se realiza con el comando ***wich()***:
> c(12, 5, 6, 9, 11)
> wich(x <= 9)
[1] 2 3 4
Y las versiones más especializadas ***which.min()*** y ***which.max()*** que retornan el índice del menor y mayor valor del vector.

1.16 Edición de Contenido

Tal como un vector puede ser indexado para obtener valores determinados, podemos usar los mismos métodos para modificar sus valores:

```
x <- 1:5; x[2] <- 1000
x[c(4, 5)] <- 100
x
```

```
[1]    1 1000    3  100  100
```

1.17 Opciones Avanzadas

Un vector puede recibir información adicional que construye un objeto más completo y que facilita su uso y comprensión de su contenido.

1.17.1 Nombrar los Componentes de un Vector

El manejo de un índice numérico puede ser complicado relacionarlo con el contenido, así la función `names()` permite asociar cada elemento del vector con un nombre propio, en el siguiente ejemplo asignamos una letra mayúscula o minúscula, nombres de mes por ejemplo:

```
x <- 1:5; names(x) <- LETTERS[1:5]; x[c('B', 'D')]
```

```
B D
2 4
```

Mediante el uso, por ejemplo, de alguna de las constantes internas:

- **LETTERS:** Las 26 letras mayúsculas del alfabeto latino.
- **letters:** Las 26 letras minúsculas del alfabeto latino.
- **month.abb:** Abreviación tres letras para nombres de los meses del año en inglés.
- **month.name:** Nombre completo meses del año en inglés.

1.17.2 Información Adicional

R permite incorporar un texto informativo a un vector con el objeto de asociar algún antecedente importante para el mismo u otros usuarios:

```
comment(x) <- "texto adicional aclaratorio."
```

Para ver el contenido asociado al vector x use :

```
attributes(x)
```

```
$names
[1] "A" "B" "C" "D" "E"

$comment
[1] "texto adicional aclaratorio."
```

Factores

Un dato que adopta un valor desde un conjunto finito de valores se denomina *data categórica*, por ejemplo, el color de ojos, el estado civil o país de origen. Ellos siempre serán *discretos* (toman un valor específico).

Cuando no implican una relación de orden (color azul no es más importante que color negro, o casado es menos importante que soltero) hablamos de **data nominal** y cuando si existe una relación de orden de algún tipo (sano es *mejor* que enfermo, joven es *menor* que anciano) hablamos de **data ordinal**, ambas son categóricas, pero se diferencian en su relación de orden y en R ese tipo de datos se denominan ***factores***.

Un vector de caracteres puede ser convertido en factores utilizando la función `factor()`:

```
niveles <- c("A", "B", "C", "B")
(factores <- factor(niveles))
```

```
[1] A B C B
Levels: A B C
```

1.18 Factores Nominales

El resultado de la conversión a *factor* fue:

```
[1] A B C B
Levels: A B C
```

Donde vemos además de la lista de valores ahora sin las *""* ya que **no** son *textos*. Aparece el ítem `Levels` que señala los niveles o categorías disponibles en el nuevo vector de tipo *nominal*. El

orden con que los presenta es en estricto orden alfabético. Vemos que la clase o categoría base es *dos*, el método para reordenar nuestros niveles es incorporar el parámetro `levels`:

```
factores <- factor(niveles, levels = c("A", "B", "C"))
```

o ejecutar la función:

```
levels(factores) <- c("A", "B", "C")
```

Y al explorar su contenido vemos la secuencia de niveles como se buscaba:

```
factores
```

```
[1] A B C B
Levels: A B C
```

1.19 Factores Ordinales

Hasta ahora tenemos un vector de clases o categorías en la secuencia que deseamos, pero no contiene información relativa a orden o magnitud, por lo tanto no es capaz de ser evaluada por operaciones lógicas:

```
factores > 'B'
```

```
Warning in Ops.factor(factores, "B"): '>' not meaningful for factors
[1] NA NA NA NA
```

Para convertir nuestra data a formato *ordinal* debemos agregar el parámetro `ordered = TRUE` que indica a R que existirá un orden definido por los niveles de *levels*.

```
factores <- factor(niveles,levels= c("A","B","C"), ordered= TRUE)
```

Y así nuestro vector tendrá la capacidad de ser evaluado:

```
factores < 'B'
```

```
[1]  TRUE FALSE FALSE FALSE
```

```
factores == 'C'
```

```
[1] FALSE FALSE  TRUE FALSE
```

Matrices

Un vector posee un índice por lo tanto es *unidimensional*, si agregamos una segunda dimensión se convierte en *bidimensional* y se denomina **matriz**.

La función en R para construir una matriz es `matrix()` y requiere dos parámetros de dimensión, además de los datos. La primera dimensión corresponde al número de *filas* y la segunda dimensión corresponde al número de *columnas*.

```
matrix(1:8, 2, 4)
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
```

Aunque se recomienda utilizar los parámetros con nombre por facilidad de uso:

```
matrix(nrow = 5, ncol = 3)
```

```
     [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA
[3,]   NA   NA   NA
[4,]   NA   NA   NA
[5,]   NA   NA   NA
```

```
matrix(1:15, ncol = 3, nrow = 5)
```

```
     [,1] [,2] [,3]
[1,]    1    6   11
[2,]    2    7   12
[3,]    3    8   13
```

```
[4,]    4    9   14
[5,]    5   10   15
```

El parámetro `byrow` cambia la forma de asignar los datos de columna en columna a fila por fila:

```
matrix(1:15, ncol = 3, nrow = 5, byrow = T)
```

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]   10   11   12
[5,]   13   14   15
```

Si necesita convertir un vector *x* evitando duplicar el objeto a una matriz puede intentarlo así:
> dim(x) <- c(num_filas, num_columnas)
Donde ***dim*** es una función que redefine las *dimensiones* de la matriz.

1.20 Operaciones Numéricas

Las matrices responden a las mismas reglas de operación que los *vectores* realizando las operaciones elemento a elemento:

```
(mi_matriz <- matrix(1:6,nrow=2, ncol=3))
```

```
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

```
mi_matriz * 2
```

```
     [,1] [,2] [,3]
[1,]    2    6   10
[2,]    4    8   12
```

```
mi_matriz + 2 / mi_matriz ** 3
```

```
     [,1]     [,2]     [,3]
[1,] 3.00 3.074074 5.016000
[2,] 2.25 4.031250 6.009259
```

Y así podemos aplicar las funciones detalladas en 1.1:

```
length(mi_matriz) #largo del vector
```

```
[1] 6
```

```
min(mi_matriz) #encuetra valor mínimo
```

```
[1] 1
```

```
max(mi_matriz) #encuentra valor máximo
```

```
[1] 6
```

```
range(mi_matriz) #encuentra el rango
```

```
[1] 1 6
```

```
mean(mi_matriz) #promedio
```

```
[1] 3.5
```

```
median(mi_matriz) #mdiana
```

```
[1] 3.5
```

```
sum(mi_matriz) #sumatoria
```

```
[1] 21
```

```
prod(mi_matriz) #producto de todos elementos
```

```
[1] 720
```

```
var(mi_matriz) #varianza
```

```
     [,1] [,2] [,3]
[1,]  0.5  0.5  0.5
[2,]  0.5  0.5  0.5
[3,]  0.5  0.5  0.5
```

```
sd(mi_matriz) #desviación estándar
```

```
[1] 1.870829
```

Y existe una serie de funciones especialmente diseñadas para matrices.

```
dim(mi_matriz) #Dimensión de la matriz, fila x columna
```

```
[1] 2 3
```

```
nrow(mi_matriz) #Número de filas
```

```
[1] 2
```

```
ncol(mi_matriz) #Número de columnas
```

```
[1] 3
```

```
rowSums(mi_matriz, na.rm = T) #Suma elementos por filas
```

```
[1]  9 12
```

```
colSums(mi_matriz, na.rm = T) #Suma elementos por columnas
```

```
[1]  3  7 11
```

```
rowMeans(mi_matriz) #Promedio por filas
```

```
[1] 3 4
```

```
colMeans(mi_matriz) #Promedio por columnas
```

```
[1] 1.5 3.5 5.5
```

1.21 Indexado y Subconjunto

Las mismas reglas aplicadas a los vectores son aplicables a las matrices, considerando sí que el índice ***i*** en ***x[i,]*** será la i-ésima ***fila***, el índice ***j*** en ***x[, j]*** será la j-ésima ***columna*** y ***x[i, j]*** refiere al ***ij-ésimo*** elemento de ***x***. Los índices ***i*** y ***j*** deben ser vectores numéricos.

```
(mi_matriz <- matrix(1:6, 3, 2, byrow = T))
```

```
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
```

```
mi_matriz[2, 1]
```

```
[1] 3
```

```
mi_matriz[c(1, 3), 2]
```

```
[1] 2 6
```

```
mi_matriz[1, ]
```

```
[1] 1 2
```

```
mi_matriz[2:3, ]
```

```
     [,1] [,2]
[1,]    3    4
[2,]    5    6
```

```
mi_matriz[, 1]
```

```
[1] 1 3 5
```

```
mi_matriz[, 1:2]
```

```
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
```

1.22 Edición de Contenido

Modificar valores se hace aplicando las mismas técnicas del punto anterior, se selecciona un subconjunto y se asignan los valores:

```
mi_matriz[3, 2] <- 45.66 #modificar un valor
mi_matriz[, 1] <- 333 #modificar toda la columna
mi_matriz[c(1, 2), ] <- 777 #modificar todas las filas
mi_matriz
```

```
     [,1]   [,2]
[1,]  777 777.00
[2,]  777 777.00
[3,]  333  45.66
```

1.23 Opciones Avanzadas

1.23.1 Nombres de Filas y Columnas

Para facilidad de uso también contamos con un método para asignar nombres a filas y columnas y usar esos nombres como índices:

```
rownames(mi_matriz) <- LETTERS[1:3]

colnames(mi_matriz) <- c("uno", "dos")

mi_matriz
```

```
  uno    dos
A 777 777.00
B 777 777.00
C 333  45.66
```

```
mi_matriz[,'uno']
```

```
  A   B   C
777 777 333
```

```
mi_matriz['B','uno']
```

```
[1] 777
```

También podemos usar las mismas funciones y un índice para cambiar su nombre:

```
rownames(mi_matriz)[3] <- "Letra C"
mi_matriz
```

```
        uno    dos
A       777 777.00
B       777 777.00
Letra C 333  45.66
```

1.23.2 Modificar Tamaño de Matrices

Agregar información (cambiando el tamaño de la matriz) se realiza mediante las funciones `cbind()` y `rbind()`, la primera agrega a continuación un nueva columna de datos y la segunda filas:

```
datos_columna <- c(100, 200,300)
mi_matriz <- cbind(mi_matriz, nombre_col= datos_columna)
datos_fila <- c(1000, 2000, 3000)
(mi_matriz <- rbind(mi_matriz, nombre_fil= datos_fila))
```

```
           uno     dos nombre_col
A          777  777.00        100
```

```
B           777  777.00        200
Letra C     333   45.66        300
nombre_fil 1000 2000.00       3000
```

1.23.3 Información Adicional

Por último, también contamos con la función `comment()` para asociar información adicional al objeto.

```
comment(mi_matriz) <- "Información Adicional"
attributes(mi_matriz)
```

```
$dim
[1] 4 3

$dimnames
$dimnames[[1]]
[1] "A"          "B"          "Letra C"    "nombre_fil"

$dimnames[[2]]
[1] "uno"        "dos"        "nombre_col"


$comment
[1] "Información Adicional"
```

Dataframes

Un *vector* y una *matriz* pueden guardar solo datos del mismo tipo. Para resolver esta limitación se ha definido el objeto *dataframe*.

Un **dataframe** es una lista de *variables* del *mismo número de filas* con *nombres de fila únicos*. Si no se incluyen variables, los nombres de las filas determinan el número de filas.

Los nombres de las columnas no *deben estar vacíos* y los intentos de utilizar nombres vacíos no es admitido. Se permiten nombres de columna duplicados, pero debe usar la opción `check.names = FALSE` al momento de crearlo. Sin embargo, no todas las operaciones conservarán los nombres de columna duplicados: por ejemplo, un subconjunto tipo matriz forzará que los nombres de columna en el resultado sean únicos.

Si se construye un dataframe a partir de varios objetos deben tener el mismo número de filas, aunque factores y textos se reciclarán un número entero de veces si fuese necesario.

Si los nombres de fila no se proporcionan al momento de creación, los nombres de fila se toman del primer componente que tenga nombres adecuados, por ejemplo, un vector con nombre o una matriz con nombres de fila o un dataframe. Si se proporcionó como NULL o no se encontró ningún componente adecuado, los nombres de fila son la secuencia entera que comienza en uno (y dichos nombres de fila se consideran 'automático').

1.24 Creación

Un dataframe, es una **estructura matricial cuyas columnas pueden ser de diferentes tipos** (numérico, lógico, factor y carácter, etc.).

Los vectores deben tener la misma longitud:

```
numeros <- c(2, 3, 5)
textos <- c("aa", "bb", "cc")
logicos <- c(TRUE, FALSE, TRUE)
dataframe <- data.frame(numeros, textos, logicos)
dataframe
```

```
  numeros textos logicos
1       2     aa    TRUE
2       3     bb   FALSE
3       5     cc    TRUE
```

Los nombres de cada columna pueden ser definidos al momento de crear el objeto:

```
dataframe <- data.frame("numericos"= numeros,
                        "textual"= textos,
                        "bool"= logicos)
dataframe
```

```
  numericos textual  bool
1         2      aa  TRUE
2         3      bb FALSE
3         5      cc  TRUE
```

También es posible convertir una matriz a dataframe de manera directa:

```
m1 <- matrix(1:12, nrow = 4, ncol = 3)
m1
```

```
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
```

```
as.data.frame(m1)
```

```
  V1 V2 V3
1  1  5  9
2  2  6 10
3  3  7 11
4  4  8 12
```

Para revisar el detalle y su estructura:

```
str(dataframe)
```

```
'data.frame':   3 obs. of  3 variables:
 $ numericos: num  2 3 5
 $ textual  : chr  "aa" "bb" "cc"
 $ bool     : logi  TRUE FALSE TRUE
```

1.25 Operaciones Numéricas

Como el objeto dataframe acepta distintos tipos de datos, las operaciones aritméticas deben ser realizadas sobre las columnas numéricas:

```
dataframe$numericos * 2
```

Aprovechamos de mencionar que un mecanismo de creación de columna y con nombre en la misma operación:

```
dataframe$cuadrado <- dataframe$numericos ** 2
dataframe
```

```
  numericos textual  bool cuadrado
1         2      aa  TRUE        4
2         3      bb FALSE        9
3         5      cc  TRUE       25
```

1.26 Indexado y Subconjunto

Subconjunto por número de filas:

```
dataframe[2:3, ]
```

```
  numericos textual  bool cuadrado
2         3      bb FALSE        9
```

```
3         5      cc  TRUE       25
```

Subconjunto por nombre de columnas:

```
dataframe[ , c("bool","textual")]
```

```
   bool textual
1  TRUE      aa
2 FALSE      bb
3  TRUE      cc
```

Subconjunto por número de fila y nombre de columnas:

```
dataframe[c(1,2), c("bool","cuadrado")]
```

```
   bool cuadrado
1  TRUE        4
2 FALSE        9
```

1.26.1 Opciones Avanzadas

También podemos usar condiciones lógicas para crear subconjuntos:

```
dataframe[dataframe$bool==TRUE,]
```

```
  numericos textual bool cuadrado
1         2      aa TRUE        4
3         5      cc TRUE       25
```

```
dataframe[dataframe$bool==TRUE & dataframe$numericos <5, ]
```

```
  numericos textual bool cuadrado
1         2      aa TRUE        4
```

La función `subset()` provee un formato más económico para realizar subconjuntos:

```
subset(dataframe, bool == T & numericos < 5)
```

```
  numericos textual bool cuadrado
1         2      aa TRUE        4
```

1.26.2 Modificar Tamaño o Estructura

Las formas de consultar el tamaño y geometría de un objeto:

```
dim(dataframe)
```

```
[1] 3 4
```

```
nrow(dataframe)
```

```
[1] 3
```

```
ncol(dataframe)
```

```
[1] 4
```

Podemos agregar una columna siempre y cuando contenga el mismo número de filas:

```
vector <- c("uno","dos","tres")
dataframe <- cbind(dataframe, "adicional"= vector)
str(dataframe)
```

```
'data.frame':   3 obs. of  5 variables:
 $ numericos: num  2 3 5
 $ textual  : chr  "aa" "bb" "cc"
 $ bool     : logi  TRUE FALSE TRUE
 $ cuadrado : num  4 9 25
 $ adicional: chr  "uno" "dos" "tres"
```

Y el correspondiente comando para agregar filas:

```
vector <- c(7,"dd", NA, 0,"cuatro")
str(rbind(dataframe,vector))
```

```
'data.frame':   4 obs. of  5 variables:
 $ numericos: chr  "2" "3" "5" "7"
 $ textual  : chr  "aa" "bb" "cc" "dd"
 $ bool     : chr  "TRUE" "FALSE" "TRUE" NA
 $ cuadrado : chr  "4" "9" "25" "0"
 $ adicional: chr  "uno" "dos" "tres" "cuatro"
```

Si estudiamos con cuidado el resultado, notaremos que la columna **numéricos** aparece como *carácter* eso se produce si intentamos agregar una fila usando `rbind()` y `c()`, ya que convierte **todas las columnas** en una clase de carácter. Esto se debe a que todos los elementos del vector creado por c() deben ser de la misma clase, por lo que todos son forzados a la clase de caracteres que obliga a todas las variables en el marco de datos a la clase

de caracteres.

Para agregar filas de manera adecuada, necesitamos convertir los elementos que se agregan a un dataframe y asegurarnos de que las columnas sean de la misma clase que el dataframe destino.

```
agregar_fila <- data.frame(numericos= 9,
                           textual= "RR",
                           bool= FALSE,
                           cuadrado= 110,
                           adicional= "cuatro")
(dataframe <- rbind(dataframe, agregar_fila))
```

```
  numericos textual  bool cuadrado adicional
1         2      aa  TRUE        4       uno
2         3      bb FALSE        9       dos
3         5      cc  TRUE       25      tres
4         9      RR FALSE      110    cuatro
```

1.26.3 Nombres de Filas y Columnas

```
rownames(dataframe) <- c("fila1", "fila2", "fila3", "fila4")
dataframe
```

```
      numericos textual  bool cuadrado adicional
fila1         2      aa  TRUE        4       uno
fila2         3      bb FALSE        9       dos
fila3         5      cc  TRUE       25      tres
fila4         9      RR FALSE      110    cuatro
```

```
colnames(dataframe) <- c("col_1", "col_2", "col_3")
dataframe
```

```
      col_1 col_2 col_3  NA     NA
fila1     2    aa  TRUE   4    uno
fila2     3    bb FALSE   9    dos
fila3     5    cc  TRUE  25   tres
fila4     9    RR FALSE 110 cuatro
```

1.26.4 Información Adicional

```
comment(dataframe) <- "Texto informativo adicional"

attributes(dataframe)
```

```
$names
[1] "col_1" "col_2" "col_3" NA      NA

$row.names
[1] "fila1" "fila2" "fila3" "fila4"

$class
[1] "data.frame"

$comment
[1] "Texto informativo adicional"
```

Array

Un vector posee un índice por lo tanto es *unidimensional*, una matriz posee dos dimensiones y es *bidimensional* y para objetos de una dimensión superior R provee la función `array()`. Siempre todos los datos deben ser del mismo tipo.
Esta nueva dimensión la llamaremos *capas*, *niveles* o *layers* indistintamente en este libro.

```
(y <- array(data = 1:3**3, dim = c(3, 3, 3)))
```

```
, , 1

     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

, , 2

     [,1] [,2] [,3]
[1,]   10   13   16
[2,]   11   14   17
[3,]   12   15   18

, , 3

     [,1] [,2] [,3]
[1,]   19   22   25
[2,]   20   23   26
[3,]   21   24   27
```

1.27 Operaciones Numéricas

Un array responde a las mismas reglas de operación que los *vectores* o *matrices* realizando las operaciones elemento a elemen-

to:

```
y/3 + y**3
```

```
, , 1

          [,1]      [,2]     [,3]
[1,]  1.333333  65.33333 345.3333
[2,]  8.666667 126.66667 514.6667
[3,] 28.000000 218.00000 732.0000

, , 2

         [,1]     [,2]     [,3]
[1,] 1003.333 2201.333 4101.333
[2,] 1334.667 2748.667 4918.667
[3,] 1732.000 3380.000 5838.000

, , 3

         [,1]     [,2]     [,3]
[1,] 6865.333 10655.33 15633.33
[2,] 8006.667 12174.67 17584.67
[3,] 9268.000 13832.00 19692.00
```

Y así podemos aplicar las funciones detalladas en 1.1:

```
length(y); min(y); max(y); range(y)
```

```
[1] 27
[1] 1
[1] 27
[1]  1 27
```

```
mean(y); median(y); sum(y); prod(y)
```

```
[1] 14
[1] 14
[1] 378
[1] 1.088887e+28
```

```
var(y); sd(y)
```

```
[1] 63
[1] 7.937254
```

1.28 Indexado y Subconjunto

Si en las matrices, el índice ***i*** en ***x[i,,]*** será la i-ésima ***fila***, el índice ***j*** en ***x[,j,]*** será la j-ésima ***columna*** el índice ***k*** en ***x[,,k]*** corresponde al k-ésimo nivel, así ***x[i,j,k]*** refiere al ***ijk-ésimo*** elemento de ***x***.

Los índices ***i***, ***j***, ***k*** deben ser vectores numéricos.

```
mean(y[, , 1]) #Promedio de nivel 1
```

```
[1] 5
```

```
mean(y[, , 2:3]) #Promedio de niveles 2 y 3
```

```
[1] 18.5
```

```
mean(y[, , c(1,3)]) #Promedio de nivel 1 y 3
```

```
[1] 14
```

```
mean(y[1, ,]) #Promedio de la primera fila (todos los niveles)
```

```
[1] 13
```

1.29 Edición de Contenido

Modificar valores se hace aplicando las mismas técnicas del punto anterior, se selecciona un subconjunto y se asignan los valores:

```
y[1, 2, 2] <- 45.66 #modificar un valor
y[,,1] <- 0 #modificar todo el nivel
```

```
y
```

```
, , 1

     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0
[3,]    0    0    0

, , 2

     [,1]  [,2] [,3]
[1,]   10 45.66   16
[2,]   11 14.00   17
```

```
[3,]   12 15.00   18

, , 3

     [,1] [,2] [,3]
[1,]   19   22   25
[2,]   20   23   26
[3,]   21   24   27
```

1.30 Opciones Avanzadas

1.30.1 Nombres de Filas y Columnas

Para facilidad de uso también contamos el método **dimnames()** para asignar nombres por dimensión:

```
dimnames(y)[[1]] <- LETTERS[1:3] #Nombre de Filas
dimnames(y)[[2]] <- LETTERS[12:14] #Nombre de Columnas
dimnames(y)[[3]] <- c("UNO", "DOS", "TRES") #Nombre de Niveles
```

```
y
```

```
, , UNO

  L M N
A 0 0 0
B 0 0 0
C 0 0 0

, , DOS

   L     M  N
A 10 45.66 16
B 11 14.00 17
C 12 15.00 18

, , TRES

   L  M  N
A 19 22 25
B 20 23 26
C 21 24 27
```

```
y[,'M','DOS']
```

```
    A     B     C
45.66 14.00 15.00
```

1.30.2 Información Adicional

Por último, también contamos con la función `comment()` para asociar información adicional al objeto.

```
comment(y) <- "Información Adicional"
attributes(y)
```

```
$dim
[1] 3 3 3

$dimnames
$dimnames[[1]]
[1] "A" "B" "C"

$dimnames[[2]]
[1] "L" "M" "N"

$dimnames[[3]]
[1] "UNO"  "DOS"  "TRES"


$comment
[1] "Información Adicional"
```

Fechas

Una **fecha** denota un *día en particular*, que denota un período de tiempo que tiene un comienzo y un final diferente. Aunque sus demarcaciones exactas y su duración pueden ser controvertidas, normalmente pensamos en una fecha como un período de 24 horas en un lugar en particular.

La instrucción `Sys.Date()` se utiliza para obtener la fecha actual. Y pertenece a una clase particular *Date*, que define un objeto que posee una serie de propiedades específicas que vamos a revisar a continuación.

1.31 Operaciones con Fecha

Las fechas son almacenadas internamente como el número de días transcurridos a partir del 1 de enero de 1970. Un objeto tipo *date* es convertido a número con `as.numeric()`.

Para reconocer un texto como objeto fecha debe ser *entendido por el sistema* y debido a la diversidad de formatos utilizados existe una manera de 'explicar' al sistema cual es el que se utiliza o deseamos utilizar (Cuadro 1.2).

1.31.1 Formato de Fechas

En el siguiente ejemplo vamos a convertir varios textos a objeto fecha.

```
as.Date('07/ene/23',format= '%d/%b/%y')
```

```
[1] "2023-01-07"
```

Cuadro 1.2: caracteres para Formatear Fechas

Símbolo	Definición
%d	Día
%m	Mes Numérico
%b	Mes 3 letras
%B	Mes Nombre
%y	Año Dos Dígitos
%Y	Año 4 Dígitos

```
as.Date('07/enero/2023',format= '%d/%B/%Y')
```

```
[1] "2023-01-07"
```

```
as.Date('enero-07-2023',format= '%B-%d-%Y')
```

```
[1] "2023-01-07"
```

Puede ocurrir que las fechas a procesar no estén en el lenguaje de la máquina y no puede reconocer nombres o abreviaturas.

```
as.Date('07/jan/23',format= '%d/%b/%y')
```

```
[1] NA
```

```
as.Date('07/january/23',format= '%d/%B/%y')
```

```
[1] NA
```

Para ello R soluciona el problema aplicando una configuración local mediante el comando `Sys.setlocale()`.

```
Sys.setlocale("LC_TIME", locale = "C")
```

```
[1] "C"
```

```
as.Date('07/jan/23',format= '%d/%b/%y')
```

```
[1] "2023-01-07"
```

```
as.Date('07/january/23',format= '%d/%B/%y')
```

```
[1] "2023-01-07"
```

Y para cambiar de regreso a español:

```
Sys.setlocale("LC_TIME", locale = "Spanish")
```

```
[1] "Spanish_Spain.1252"
```

```
as.Date('07/ene/23',format= '%d/%b/%y')
```

```
[1] "2023-01-07"
```

```
as.Date('07/enero/23',format= '%d/%B/%y')
```

```
[1] "2023-01-07"
```

Para la extracción de los componentes de una fecha, se pueden utilizar los comandos `weekdays`, `months`, `days`, `quarters`.

```
weekdays(as.Date('07/ene/23',format= '%d/%b/%y'))
```

```
[1] "sábado"
```

Y como la mayoría de los demás objetos se pueden aplicar operaciones aritméticas.

```
as.Date(x = "13/Abril/2021",
        format = "%d/%B/%Y") - as.Date(x = "07/Ago/20",
                                       format = "%d/%b/%y")
```

```
Time difference of 249 days
```

```
fechas <- c(as.Date(x = "13/Abril/2021", format = "%d/%B/%Y"),
            as.Date(x = "07/Ago/20", format = "%d/%b/%y"),
            as.Date(x = "23/Dic/19", format = "%d/%b/%y"))
```

```
mean(fechas)
```

```
[1] "2020-08-14"
```

```
min(fechas)
```

```
[1] "2019-12-23"
```

```
max(fechas)
```

```
[1] "2021-04-13"
```

1.31.2 Cambio de Formato

Si desea usar fechas en un formato distinto al que ya posee o tiene definido, puede hacerlo usando la función `format()` del

paquete base.

La fecha actual como se encuentra definida por el sistema:

```
(fecha <- Sys.Date())
```

```
[1] "2021-12-17"
```

La puedo expresar utilizando los caracteres de formato definidos en el cuadro 1.2:

```
format(fecha, "%d %B %Y")
```

```
[1] "17 diciembre 2021"
```

Hora

El *tiempo* denota un instante o momento particular dentro de un período de *tiempo* más largo. Por lo tanto, el tiempo se suele considerar como un punto dentro de un día. Mientras que un día típico dura 24 horas, un punto en el tiempo no tiene duración (es decir, su comienzo y final son idénticos). La medición y los informes de tiempos inevitablemente plantean muchos problemas sobre convenciones (AM o PM), su precisión (Minutos, segundos o milisegundos) y ubicación (Husos o zonas horarias).

Para especificar con precisión una hora en particular también se requiere especificar la fecha dentro de la cual está incrustado. Por tanto, nuestras definiciones de objetos de tiempo serán necesariamente más complejas que las de objetos de fecha.

El paquete *base*, responde a la pregunta de San Agustín ¿Qué es entonces el tiempo? con no menos de tres clases de objetos para representar fechas y horas:

- **Date**: representa la fecha de los días sin considerar los tiempos (revisada en el punto anterior).
- **POSIXct**: la principal clase de fecha y hora para representar la *hora del calendario*.
- **POSIXlt**: una clase adicional de fecha y hora para representar la *hora local* (como listas).

"POSIXct" es más conveniente para incluir en objetos de datos como un dataframe, y "POSIXlt" está más cerca de los formularios legibles por humanos. Existe una clase virtual "POSIXt" de la que heredan ambas clases: se utiliza para permitir operaciones como la resta para mezclar ambas clases.

La hora actual se obtiene con la función:

```
Sys.time()
```

```
[1] "2021-12-17 01:13:25 -03"
```

Que retorna un objeto POSIXct que se compone de una fecha y una hora en una zona horaria determinada.

```
ahora <- Sys.time()
class(ahora)
```

```
[1] "POSIXct" "POSIXt"
```

El tiempo se cuenta desde "1970-01-01 00:00:00 UTC" (una fecha arbitraria).

Para conocer la zona horaria existe la función[13]:

[13] Para conocer la lista zonas horarias por su nombre usar *OlsonNames()*.

```
Sys.timezone()
```

```
[1] "America/Santiago"
```

1.32 Formato POSIX

El acrónimo POSIX significa "Interfaz de sistema operativo portátil" y define estándares para mantener la compatibilidad entre diferentes sistemas operativos de computadora (con la "X" implicando Unix, debido a su independencia de un fabricante en particular). Define un estándar para expresar y acordar especificaciones de conversión.

En el código, se usa dentro de una cadena de caracteres y se introduce con el símbolo %, generalmente seguido de una sola letra (cuadro 1.3).

Cuadro 1.3: caracteres para Formatear el tiempo

Símbolo	Definición
%H	Hora dos dígitos (0-23)
%I	Hora dos dígitos (0-12)
%M	Minutos dos dígitos (0-59)
%p	Indicador AM/PM
%S	Segundos dos dígitos (0-59)
%Z	Zona horaria o desplazamiento UTC

Estos símbolos existen y comparten con los símbolos de fe-

chas[14]:

[14] Para revisar la extensa lista de códigos escribe en la consola *?strftime*.

```
format(ahora, "%A %I:%M")
```

```
[1] "viernes 01:13"
```

```
format(ahora, "%H:%M:%S (%Z)")
```

```
[1] "01:13:25 (-03)"
```

```
format(ahora, "%H:%M:%S")
```

```
[1] "01:13:25"
```

1.33 Operando con el tiempo

El objeto tiempo también es capaz de realizar operaciones algebraicas.

```
ahora
```

```
[1] "2021-12-17 01:13:25 -03"
```

Sumar 30 segundos:

```
(despues_s <- ahora + 30)
```

```
[1] "2021-12-17 01:13:55 -03"
```

Sumar 30 minutos:

```
(despues_m <- ahora + (30 * 60))
```

```
[1] "2021-12-17 01:43:25 -03"
```

Sumar 30 horas (notese el ajuste de la fecha):

```
(despues_h <- ahora + (30 * 60 * 60))
```

```
[1] "2021-12-18 07:13:25 -03"
```

```
despues_h - ahora
```

```
Time difference of 1.25 days
```

La función *difftime()* soporta las siguientes unidades "auto", "secs", "mins", "hours", "days" y "weeks".

```
difftime(despues_h, ahora, units = "weeks")
```

```
Time difference of 0.1785714 weeks
```

Una función útil que también trabaja con tiempos es `trunc()`. Toma un argumento `x`, permite especificar el tiempo `units` de interés y trunca los enteros de cualquier unidad más pequeña a cero:

```
trunc(ahora, units= "mins")
```

```
[1] "2021-12-17 01:13:00 -03"
```

```
trunc(ahora, units= "months")
```

```
[1] "2021-12-01 -03"
```

```
trunc(ahora, units= "years")
```

```
[1] "2021-01-01 -03"
```

1.34 Conversión entre zonas horarias

En general, hemos visto que la base R proporciona un amplio soporte para crear y computar con fechas y horas. Al mismo tiempo, tratar con tres clases diferentes y sus relaciones con otros tipos de datos (específicamente: números) es un desafío. Aunque la mayoría de las complicaciones provienen de las complejidades inherentes de fechas y horas, las clases y los comandos R básicos para fechas y horas se han ido desarrollando con el tiempo y se vuelven cada vez más poderosos.

Convertir entre zonas horarias, una operación muy compleja se realiza de manera muy práctica. Por ejemplo, deseo conocer la hora de término de un viaje de 15 horas en una zona horaria diametralmente opuesta (note que usamos la versión de hora local *lt*), entonces, ¿qué hora sería en 15 horas más en Shangái?

```
as.POSIXlt(Sys.time() + (15 * 60 * 60), tz = "Asia/Shanghai")
```

```
[1] "2021-12-18 03:13:25 CST"
```

Colores

Todo proceso, mapa o datos finalmente debe ser desplegado en pantalla o papel mediante una salida gráfica, ya sea en formato de cartografía, reportes, diagramas o gráficos y el color juega un rol importante en la capacidad de transmitir el mensaje apropiado incluso puede ser más importante que la calidad de la data *"buena data, mala salida, error de lectura"*.

HSV (valor de saturación de tono) es una simple transformación del espacio RGB (rojo-verde-azul) que ha sido una opción conveniente para las paletas de colores en muchos sistemas de software. Sin embargo, los colores HSV aunque capturan las propiedades perceptivas de tono, colorido, saturación, croma las propiedades de luminosidad, brillo, luminancia y valor lo hace de manera deficiente y, en consecuencia, las paletas correspondientes no suelen ser una buena opción para gráficos estadísticos y visualización de datos.

Por el contrario, los colores HCL (tono-croma-luminancia) son más adecuados para *capturar la percepción humana del color*, así se pueden derivar mejores conjuntos de colores en función de estas coordenadas.

Un sistema básico de manejo de colores es usar su nombre propio para la definición, R de manera interna maneja un listado de 657 colores con nombres propios y provee de las herramientas para revisarlos.

Sistemas más convenientes se han diseñado, como el conjunto de colores que tienen una relación lógica entre ellos, las **paletas** de colores.

Conceptualmente, se distinguen tres tipos de paletas:

- **Cualitativo**: para codificar información *categórica*, es decir, cuando no se dispone de un orden particular de categorías y cada color debe recibir el mismo peso perceptual.
- **Secuencial**: para codificar información *ordenada* o *numérica*, es decir, donde los colores van de mayor a menor (o viceversa).
- **Divergente**: diseñado para codificar información numérica alrededor de un *valor neutral central*, es decir, donde los colores divergen de neutral a dos extremos.

En R por la importancia y relevancia que tienen las capacidades gráficas se han desarrollado varias herramientas para el manejo del color siendo los paquetes destacables:

- *grDevices* (R Core Team [2021]) **Paquete componente de Plataforma R**.
- *RColorBrewer* (Neuwirth [2014])
- *colorspace* (Zeileis et al. [2020], Zeileis et al. [2009], Stauffer et al. [2009])

1.35 Colores en R

1.35.1 Sistema por Nombres

R provee nombres de colores que se pueden utilizar con una especificación al parámetro `col=` en las funciones gráficas. Ver figura 1.15 para revisar la lista y colores.

El color "transparent" no es un color y, por lo tanto, no figura en la lista, pero se **acepta** como especificación de color.

Pare revisar un vector de caracteres que contiene todos los nombres de colores integrados usar:

```
colors()
```

white	aliceblue	antiquewhite	antiquewhite1	antiquewhite2	antiquewhite3	antiquewhite4	aquamarine	aquamarine2	aquamarine3
aquamarine4	azure	azure2	azure3	azure4	beige	bisque	bisque2	bisque3	bisque4
black	blanchedalmond	blue	blue2	blue3	blue4	blueviolet	brown	brown1	brown2
brown3	brown4	burlywood	burlywood1	burlywood2	burlywood3	burlywood4	cadetblue	cadetblue1	cadetblue2
cadetblue3	cadetblue4	chartreuse	chartreuse2	chartreuse3	chartreuse4	chocolate	chocolate1	chocolate2	chocolate3
chocolate4	coral	coral1	coral2	coral3	coral4	cornflowerblue	cornsilk	cornsilk2	cornsilk3
cornsilk4	cyan	cyan2	cyan3	cyan4	darkgoldenrod	darkgoldenrod1	darkgoldenrod2	darkgoldenrod3	darkgoldenrod4
darkgray	darkgreen	darkkhaki	darkmagenta	darkolivegreen	darkolivegreen1	darkolivegreen2	darkolivegreen3	darkolivegreen4	darkorange
darkorange1	darkorange2	darkorange3	darkorange4	darkorchid	darkorchid1	darkorchid2	darkorchid3	darkorchid4	darkred
darksalmon	darkseagreen	darkseagreen1	darkseagreen2	darkseagreen3	darkseagreen4	darkslateblue	darkslategray	darkslategray1	darkslategray2
darkslategray3	darkslategray4	darkturquoise	darkviolet	deeppink	deeppink2	deeppink3	deeppink4	deepskyblue	deepskyblue2
deepskyblue3	deepskyblue4	dimgray	dodgerblue	dodgerblue2	dodgerblue3	dodgerblue4	firebrick	firebrick1	firebrick2
firebrick3	firebrick4	floralwhite	forestgreen	gainsboro	ghostwhite	gold	gold2	gold3	gold4
goldenrod	goldenrod1	goldenrod2	goldenrod3	goldenrod4	gray	gray0	gray1	gray2	gray3
gray4	gray5	gray6	gray7	gray8	gray9	gray10	gray11	gray12	gray13	gray14
gray15	gray16	gray17	gray18	gray19	gray20	gray21	gray22	gray23	gray24
gray25	gray26	gray27	gray28	gray29	gray30	gray31	gray32	gray33	gray34
gray35	gray36	gray37	gray38	gray39	gray40	gray42	gray43	gray44	gray45
gray46	gray47	gray48	gray49	gray50	gray51	gray52	gray53	gray54	gray55
gray56	gray57	gray58	gray59	gray60	gray61	gray62	gray63	gray64	gray65
gray66	gray67	gray68	gray69	gray70	gray71	gray72	gray73	gray74	gray75
gray76	gray77	gray78	gray79	gray80	gray81	gray82	gray83	gray84	gray85
gray86	gray87	gray88	gray89	gray90	gray91	gray92	gray93	gray94	gray95
gray96	gray97	gray98	gray99	green	green2	green3	green4	greenyellow	honeydew
honeydew2	honeydew3	honeydew4	hotpink	hotpink1	hotpink2	hotpink3	hotpink4	indianred	indianred1
indianred2	indianred3	indianred4	ivory	ivory2	ivory3	ivory4	khaki	khaki1	khaki2
khaki3	khaki4	lavender	lavenderblush	lavenderblush2	lavenderblush3	lavenderblush4	lawngreen	lemonchiffon	lemonchiffon2
lemonchiffon3	lemonchiffon4	lightblue	lightblue1	lightblue2	lightblue3	lightblue4	lightcoral	lightcyan	lightcyan2
lightcyan3	lightcyan4	lightgoldenrod	lightgoldenrod1	lightgoldenrod2	lightgoldenrod3	lightgoldenrod4	lightgoldenrodyellow	lightgray	lightgreen
lightpink	lightpink1	lightpink2	lightpink3	lightpink4	lightsalmon	lightsalmon2	lightsalmon3	lightsalmon4	lightseagreen
lightskyblue	lightskyblue1	lightskyblue2	lightskyblue3	lightskyblue4	lightslateblue	lightslategray	lightsteelblue	lightsteelblue1	lightsteelblue2
lightsteelblue3	lightsteelblue4	lightyellow	lightyellow2	lightyellow3	lightyellow4	limegreen	linen	magenta	magenta2
magenta3	maroon	maroon1	maroon2	maroon3	maroon4	mediumorchid	mediumorchid1	mediumorchid2	mediumorchid3
mediumorchid4	mediumpurple	mediumpurple1	mediumpurple2	mediumpurple3	mediumpurple4	mediumseagreen	mediumslateblue	mediumspringgreen	mediumturquoise
mediumvioletred	midnightblue	mintcream	mistyrose	mistyrose2	mistyrose3	mistyrose4	moccasin	navajowhite	navajowhite2
navajowhite3	navajowhite4	navy	oldlace	olivedrab	olivedrab1	olivedrab2	olivedrab3	olivedrab4	orange
orange2	orange3	orange4	orangered	orangered2	orangered3	orangered4	orchid	orchid1	orchid2
orchid3	orchid4	palegoldenrod	palegreen	palegreen1	palegreen3	palegreen4	paleturquoise	paleturquoise1	paleturquoise2
paleturquoise3	paleturquoise4	palevioletred	palevioletred1	palevioletred2	palevioletred3	palevioletred4	papayawhip	peachpuff	peachpuff2
peachpuff3	peachpuff4	peru	pink	pink1	pink2	pink3	pink4	plum	plum1
plum2	plum3	plum4	powderblue	purple	purple1	purple2	purple3	purple4	red
red2	red3	rosybrown	rosybrown1	rosybrown2	rosybrown3	rosybrown4	royalblue	royalblue1	royalblue2
royalblue3	royalblue4	salmon	salmon1	salmon2	salmon3	salmon4	sandybrown	seagreen	seagreen1
seagreen2	seagreen3	seashell	seashell2	seashell3	seashell4	sienna	sienna1	sienna2	sienna3
sienna4	skyblue	skyblue1	skyblue2	skyblue3	skyblue4	slateblue	slateblue1	slateblue2	slateblue3
slateblue4	slategray	slategray1	slategray2	slategray3	slategray4	snow	snow2	snow3	snow4
springgreen	springgreen2	springgreen3	springgreen4	steelblue	steelblue1	steelblue2	steelblue3	steelblue4	tan
tan1	tan2	tan4	thistle	thistle1	thistle2	thistle3	thistle4	tomato	tomato2
tomato3	tomato4	turquoise	turquoise1	turquoise2	turquoise3	turquoise4	violet	violetred	violetred1
violetred2	violetred3	violetred4	wheat	wheat1	wheat2	wheat3	wheat4	yellow	yellow2
yellow3	yellow4								

Figura 1.15: Tabla de Colores por Nombre.

1.35.2 RColorBrewer

Paquete diseñado para la selección de paletas de color especialmente diseñadas para cartografía temática y basadas en los estudios de percepción del color realizados por *Cynthia Brewer* (Figura 1.16) en la universidad de PennState durante el 2001-2.

Figura 1.16: Cynthia 'Cindy' Brewer. Geográfo.

Para obtener detalles y herramientas de selección de paleta interactivas, consultar http://colorbrewer.org.

El paquete RColorBrewer (Neuwirth [2014]) presenta una serie de comandos que permiten revisar las paletas y modificarlas para obtener un vector de colores.

Por primera y única vez instalar el paquete `RColorBrewer`:

```
install.packages("RColorBrewer")
```

Y en cada sesión de trabajo se debe cargar el paquete con el comando `library()`.

```
library(RColorBrewer)
```

Revisar cuadro 1.4 para ver nombres y propiedades básicas de paletas disponibles y figura 1.17 para explorar los colores disponibles.

Y para mostrar gráficamente colores y nombres en el panel *Plots*: `display.brewer.all()`.

La sintaxis para obtener el listado de colores en R es:

```
mi_paleta <- brewer.pal(n = 4, name = "Greens")
```

El objeto **mi__paleta** contiene el número de colores definido y no pueden ser más que los descritos en el cuadro 1.4. El comando `colorRampPalette(mi_paleta)(n)` donde el valor *n* corresponde al número final de colores obtenidos por interpolación de los colores definidos en el parámetro *mi_paleta*.

```
colores <- colorRampPalette(mi_paleta)(10)
```

Cuadro 1.4: Propiedades de Paletas en Paquete ColorBrewer.

	N° Colores	Categoría	Contra Daltonismo
BrBG	11	div	TRUE
PiYG	11	div	TRUE
PRGn	11	div	TRUE
PuOr	11	div	TRUE
RdBu	11	div	TRUE
RdGy	11	div	FALSE
RdYlBu	11	div	TRUE
RdYlGn	11	div	FALSE
Spectral	11	div	FALSE
Accent	8	qual	FALSE
Dark2	8	qual	TRUE
Paired	12	qual	TRUE
Pastel1	9	qual	FALSE
Pastel2	8	qual	FALSE
Set1	9	qual	FALSE
Set2	8	qual	TRUE
Set3	12	qual	FALSE
Blues	9	seq	TRUE
BuGn	9	seq	TRUE
BuPu	9	seq	TRUE
GnBu	9	seq	TRUE
Greens	9	seq	TRUE
Greys	9	seq	TRUE
Oranges	9	seq	TRUE
OrRd	9	seq	TRUE
PuBu	9	seq	TRUE
PuBuGn	9	seq	TRUE
PuRd	9	seq	TRUE
Purples	9	seq	TRUE
RdPu	9	seq	TRUE
Reds	9	seq	TRUE
YlGn	9	seq	TRUE
YlGnBu	9	seq	TRUE
YlOrBr	9	seq	TRUE
YlOrRd	9	seq	TRUE

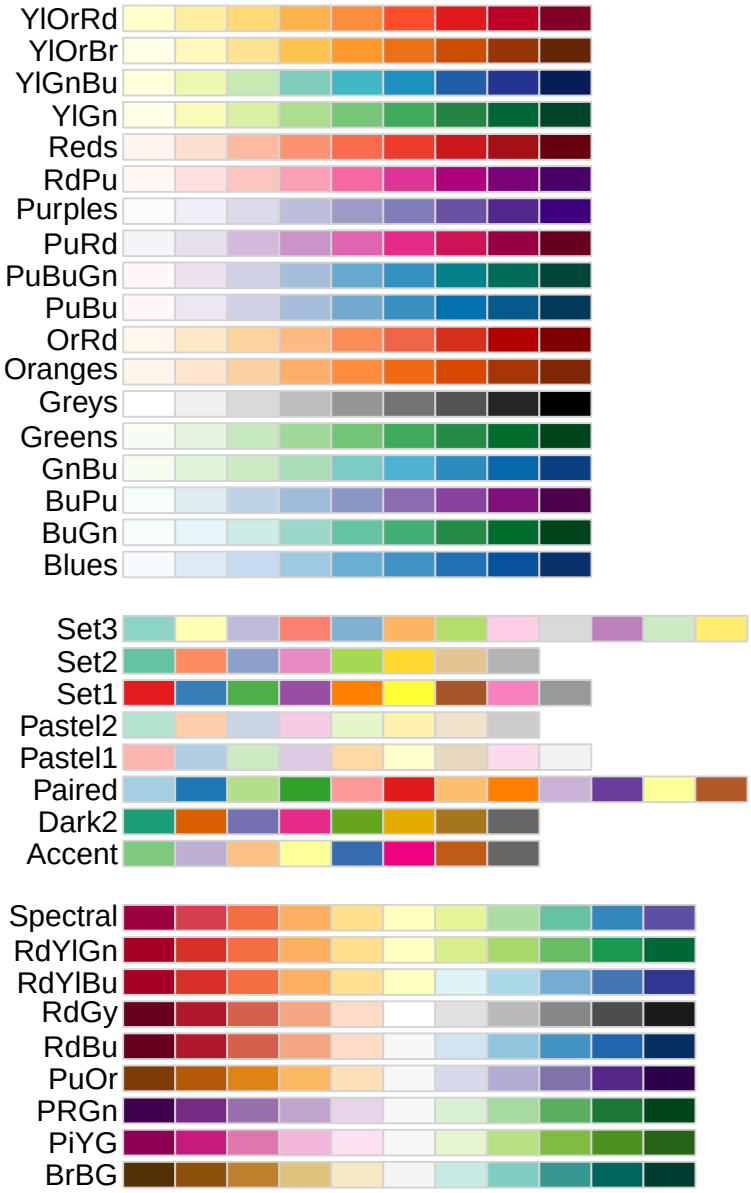


Figura 1.17: Paletas de colores, ColorBrewer.

1.36 Paquete colorspace

Por primera y única vez instalar el paquete `colorspace` utilizando la sintaxis:

```
install.packages("colorspace")
```

Y en cada sesión de trabajo se debe cargar el paquete con el comando `library`.

```
library(colorspace)
```

El paquete cuenta con una serie de comandos que facilitan la exploración de secuencias colores (que llamaremos *paletas*) y que permiten ver directamente en el panel *Plots* los colores que la componen y mediante el parámetro n chequear distintos niveles de detalles:

```
hcl_palettes(palette = "vik", plot = TRUE, n = 5)
```

Figura 1.18: Paleta Vik, 5 colores.

```
hcl_palettes(palette = "vik", plot = TRUE, n = 10)
```

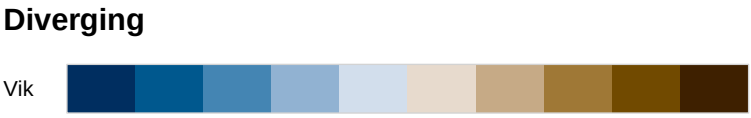


Figura 1.19: Paleta Vik, 10 colores.

```
hcl_palettes(palette = "Lisbon", plot = TRUE, n = 20)
```

Figura 1.20: Paleta Lisbon, 20 colores.

Se cuenta con un gran número de paletas para cada uno de los tres tipos de paletas descritas en el punto anterior, así el comando `hcl_palettes(plot=FALSE)` imprime un listado de nombres para efectos de búsqueda y selección:

```
hcl_palettes(plot=FALSE)
```

```
HCL palettes

Type:  Qualitative
Names: Pastel 1, Dark 2, Dark 3, Set 2, Set 3, Warm, Cold, Harmonic, Dynamic

Type:  Sequential (single-hue)
Names: Grays, Light Grays, Blues 2, Blues 3, Purples 2, Purples 3, Reds 2,
       Reds 3, Greens 2, Greens 3, Oslo

Type:  Sequential (multi-hue)
Names: Purple-Blue, Red-Purple, Red-Blue, Purple-Orange, Purple-Yellow,
       Blue-Yellow, Green-Yellow, Red-Yellow, Heat, Heat 2, Terrain,
       Terrain 2, Viridis, Plasma, Inferno, Rocket, Mako, Dark Mint,
       Mint, BluGrn, Teal, TealGrn, Emrld, BluYl, ag_GrnYl, Peach,
       PinkYl, Burg, BurgYl, RedOr, OrYel, Purp, PurpOr, Sunset,
       Magenta, SunsetDark, ag_Sunset, BrwnYl, YlOrRd, YlOrBr, OrRd,
       Oranges, YlGn, YlGnBu, Reds, RdPu, PuRd, Purples, PuBuGn, PuBu,
       Greens, BuGn, GnBu, BuPu, Blues, Lajolla, Turku, Hawaii, Batlow

Type:  Diverging
Names: Blue-Red, Blue-Red 2, Blue-Red 3, Red-Green, Purple-Green,
       Purple-Brown, Green-Brown, Blue-Yellow 2, Blue-Yellow 3,
       Green-Orange, Cyan-Magenta, Tropic, Broc, Cork, Vik, Berlin,
       Lisbon, Tofino
```

Y desde el listado podemos explorar los nombres y reemplazando el parámetro `hcl_palettes(plot=TRUE)` se genera una carta de colores y buscar por paletas. Además se pueden generar listas de paletas por tipo (ver figuras 1.23, 1.24, 1.25, 1.26)

1.36.1 Uso de Colores

Los comandos `qualitative_hcl(n, palette)`, `sequential_hcl(n, palette)` y `diverging_hcl(n, palette)` retornan un vector de n colores correspondientes a la paleta definida por su nombre en *palette*, los nombres de color serán expresados en códigos hexadecimales.

```
qualitative_hcl(5, palette = "Pastel 1")
```

```
[1] "#FFC5D0" "#E2D4A8" "#A8E1BF" "#A4DDEF" "#E4CBF9"
```

```
sequential_hcl(4, palette = "Purples 2")
```

```
[1] "#3C2692" "#7C76AB" "#BCBAD1" "#F1F1F1"
```


Figura 1.21: Paleta colores 'qualitative_hcl()'.

Una función que requiere definir un color o un vector de colores aceptan un vector de códigos hexadecimales (Figura 1.21) generado por las funciones para la construcción de las paletas.

```
#carga de un modelo raster
volcan <- raster(volcano)
#define paleta de colores mediante paleta
colores <- qualitative_hcl(30, palette = "Pastel 1")
#genera plano usando colores
plot(volcan, col= colores)
```

También un vector con *'nombres propios'* (Figura 1.22) se puede utilizar .

```
plot(volcan, col=c("blue","red","magenta"))
```

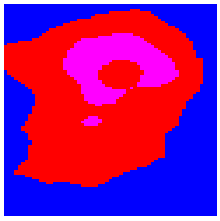


Figura 1.22: Nombres propios para crear paleta.

1.36.2 Detalle de Paletas

A continuación, observamos en detalle cada paleta por tipo con sus colores y nombre como el comando que la genera para ser recreada en su panel *Plots*.

Figura 1.23: Paleta de Colores Cualitativos, generado con: hcl_palettes('qualitative',n= 6, plot= TRUE)

Figura 1.24: Paleta de Colores Secuenciales de único matiz, generado con: hcl_palettes('sequential (single-hue)',n= 7, plot= TRUE)

Figura 1.25: Paleta de Colores Secuenciales de múltiple matiz, generado con: hcl_palettes('sequential (multi-hue)',n= 6, plot= TRUE)

Figura 1.26: Paleta de Colores Divergentes, generado con: hcl_palettes('diverging',n= 6, plot= TRUE)

1.36.3 Usando Modificadores

Usando la figura 1.22 vamos a revisar una serie de funciones del paquete *colorspace*, dedicadas a modificar una paleta de colores.

- *darken/lighten*: Funciones para modificar una paleta para oscurecer o iluminar la paleta de colores, el parámetro `amount`, monto o magnitud del efecto (Figura 1.27).

```
#Aplicación de modificador 'lighten'
plot(volcan,
     col= lighten(col = c("blue","red","magenta"),
                  amount = 0.75))
```

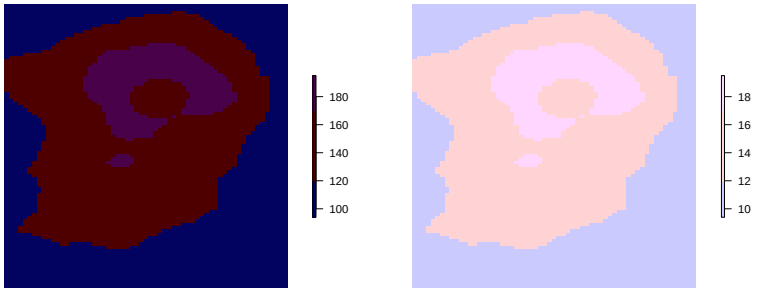


Figura 1.27: Oscurecer (izquierda) o Iluminar (derecha) una paleta de colores.

- *desaturate*: Función que modifica el monto de croma (cantidad de color o tono) de una paleta, el parámetro `amount` monto o magnitud del efecto, *0* no efecto y *1* total desaturación (Figura 1.28).

```
#Aplicación modificador 'desaturate'
plot(volcan,
     col= desaturate(col = c("blue","red","magenta"),
                     amount = 0.4))
```

- *adjust_transparency*: Función que agrega un factor de translucencia a los colores, con valor de *1* para opaco y *0* para transparencia total (Figura 1.29).

```
#ajuste transparencia por modificador"
plot(volcan,
     col= adjust_transparency(terrain.colors(n = 255),
```

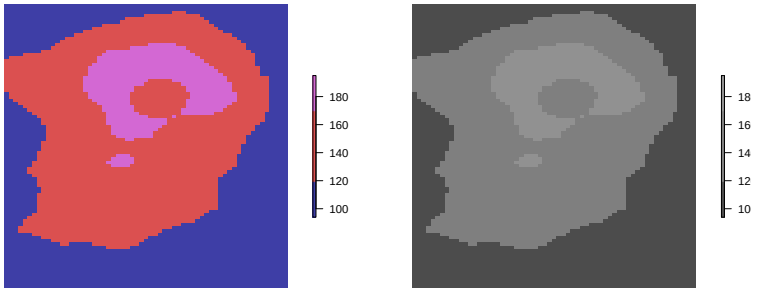


Figura 1.28: Desaturación de colores un 40% (izquierda) y 100% (derecha).

```
                alpha = 0.5),
add=T)
```

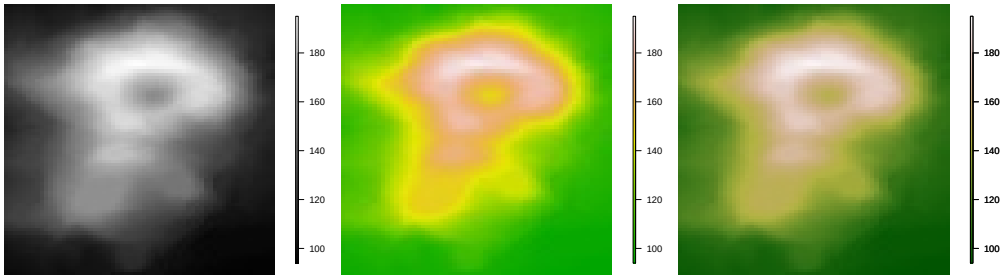


Figura 1.29: Aplicando transparencia, Mapa base en blanco y negro (izquierda), mapa en colores (centro) y superposición con translucencia de un 50% (derecha).

II PROYECTOS EN R

Desarrollando un Proyecto R

Al iniciar todo trabajo más complejo que unas líneas en R, es buena práctica trabajar con el concepto de **Proyecto** en RStudio. Facilita la organización del trabajo todo dentro de una carpeta y mediante subcarpetas dividir por temas o niveles de proceso.

En una sesión inicial ir a menú *File - New Project...* (Figura 2.1) y seguir los pasos del asistente.

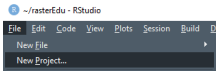


Figura 2.1: Opción Menú Nuevo Proyecto.

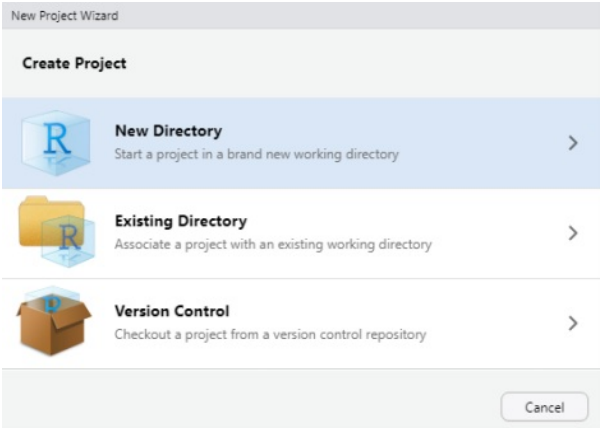


Figura 2.2: Opciones de Creación de proyectos.

1. Crear un proyecto en una carpeta nueva, existente o asociado a un repositorio de control web como *git*, programa de *control de versiones* diseñado por Linus Torvalds, para la eficiencia, confiabilidad y compatibilidad en el mantenimiento de versiones (Wikipedia) o *Apache Subversion*,o SVN, programa para el control de versiones open source basado en un repositorio similar a un sistema de ficheros (Wikipedia). Seleccionar una carpeta nueva (Figura 2.2).

2. Seleccionar *New Proyect* (otras opciones generan proyectos para crear un paquete o una aplicación web) (Figura 2.3).

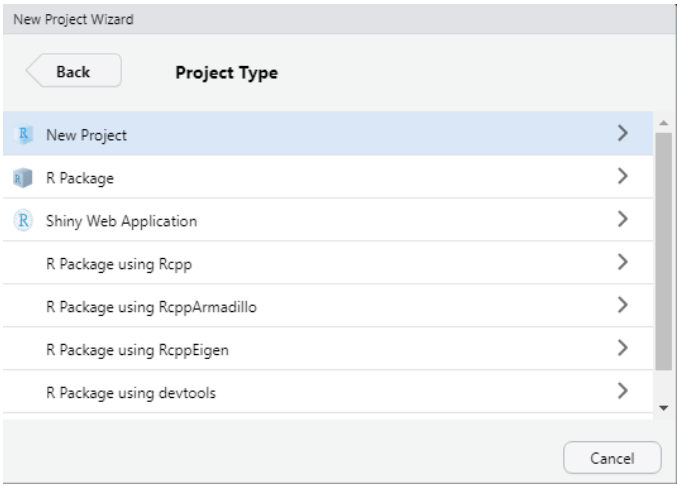


Figura 2.3: Tipo de Proyecto.

3. Se debe escribir el nombre y seleccionar la carpeta superior (Figura 2.4).

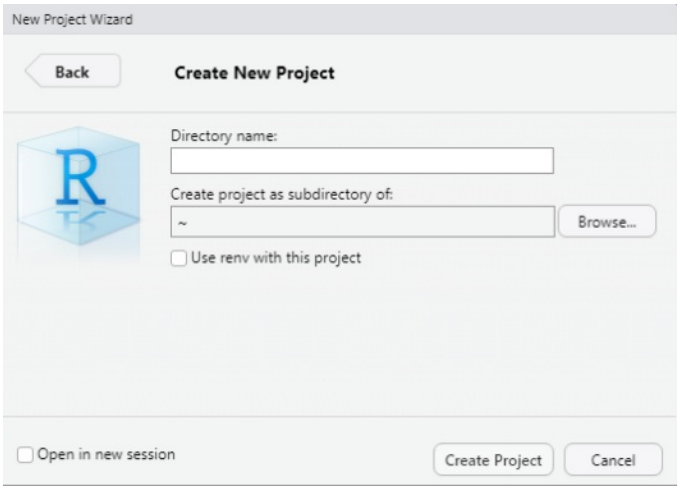


Figura 2.4: Opciones de Ubicación y nombre de Proyecto.

2.1 Organización de Archivo

La forma en que se *organizan los archivos* ayuda a que el código sea más **legible**. De manera similar, la forma en que se *organiza el código* dentro de un archivo tiene un impacto significativo en la **legibilidad**.

Los archivos también pueden tener propósitos específicos. Algunos pueden contener solo funciones que serán utilizadas por otros archivos, algunos pueden usarse para actualizar paquetes, etc.

2.1.1 ¿Cómo organizar los archivos dentro de un proyecto?

- Utilizar la función de proyectos de RStudio cada vez que comience a trabajar en un nuevo proyecto.
- Mantenga todos los archivos de origen de un proyecto en un directorio y use rutas relativas para acceder a ellos.
- Separe los archivos que contienen funciones que serán utilizadas por otras partes del código del núcleo del código.
- Tenga en cuenta en qué directorio de trabajo se encuentra al obtener un script.

2.1.2 ¿Cómo organizar el código dentro de cada archivo?

- Comience cada archivo con un comentario que diga quién lo escribió y cuándo, qué contiene y cómo encaja en el programa más grande.
- Luego cargar todos los paquetes requeridos.
- A continuación, los archivos requeridos de origen (otro script *r*), si los hubiera.

Entonces agregar nuestro código y dividirlo en archivos separados (generalmente <2000–3000 líneas).

La instalación de cada paquete debe ser realizado en la ***consola*** ya que se realiza una primera vez y estará disponible para las futuras sesiones.

2.2 Manejo de Carpetas

Se deben crear carpetas donde organizar la información y productos, se pueden crear por medio de Windows o directamente utilizando código R. La función `dir.create(nombre)` crea una carpeta o chequear si existe, `dir.exists(nombre)`.

Crear una carpeta llamada *nombre* dentro de la carpeta raíz (1.36.3, paso 1):

```
dir.create("nombre")
```

Y para crear una carpeta dentro la recién creada *nombre* (si existe, genera un mensaje de alerta):

```
dir.create("nombre/nombre1")
```

Para chequear la existencia de una carpeta:

```
dir.exists("nombre")
```

```
[1] TRUE
```

2.2.1 here

En la carpeta seleccionada se genera un archivo de extensión '.Rproj'. Recomendamos instalar y cargar en la sesión actual el paquete *here* (Müller [2020]) que permite **crear rutas de directorios** o **localiza los archivos de manera relativa a la carpeta raíz**.

```
library(here)
```

Un ejemplo de sintaxis y uso (solo para referencia):

```
(here("carpeta", "subcarpeta", "nombre.ext"))
```

2.3 Archivos Script

Desde este punto vamos a escribir código R que se reutilizará para replicar metodologías o tareas de análisis y el modo más práctico es crear un archivo *scripts* (.r) (Figura 2.5).

Y usando el **editor de texto** (Figura 2.6) (¡y no la consola!) vamos construyendo nuestros programas o rutinas.

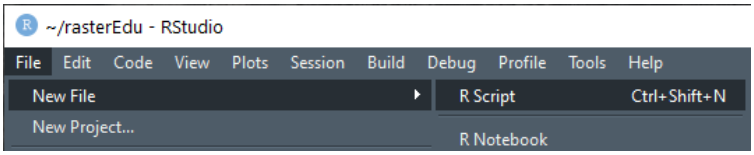


Figura 2.5: Opción Menú Nuevo Archivo Script.

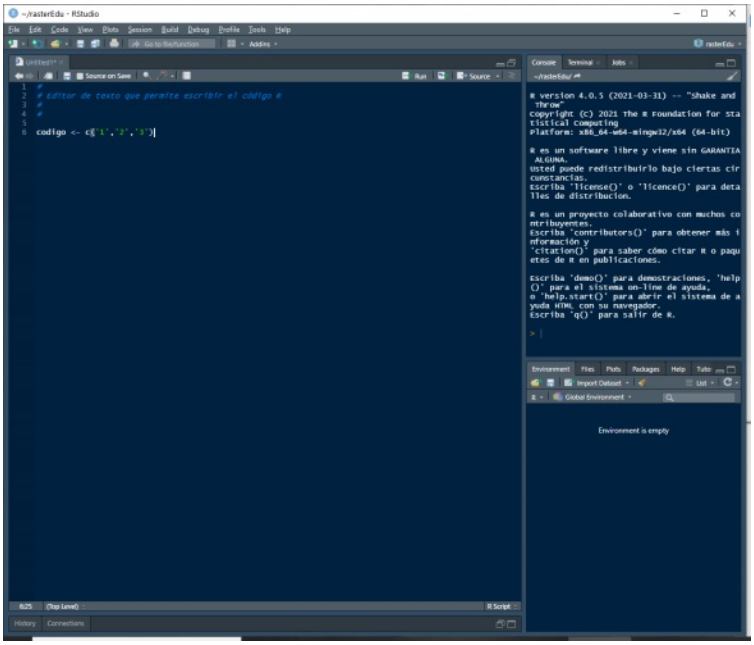


Figura 2.6: Editor de Textos.

Para ejecutar el código, existen varias alternativas:

- Posicionar el cursor en la línea y presionar la combinación *control + enter*.
- Iluminar el texto (seleccionar) y presionar la combinación *control + enter*.
- Ejecutar todo el código del archivo script con la combinación *control + shift + s*.

2.4 Manejo de objetos

Cuando se crea un objeto muy grande (en términos de data) o que tarda en ser descargado de la web o el proceso de construcción sea muy complejo se recomienda almacenarlo en disco y la próxima vez que se ejecute el archivo script (ver 2.3) el objeto se carga directo desde disco con el consiguiente ahorro de tiempo.

El método ideal es identificar el objeto que se desea almacenar y leer desde disco, R provee de dos funciones básicas, para grabar en disco `saveRDS()` y para leer desde disco `readRDS()`.

En el siguiente ejemplo, un supuesto *objeto* creado en el procesamiento será **almacenado en la carpeta** (carpeta del proyecto)/*RAW*/ con el nombre de ***objeto.rds***⊛.

El símbolo ⊛ que aparece adjunto al nombre hace referencia que dicho archivo se encuentra disponible en el sitio *github* del texto.

```
objeto <- c(1:1000)
saveRDS(objeto, here("raw", "objeto.rds"))
```

En este ejemplo guardamos *objeto* en la carpeta de salida definida en la función *here* (que debe existir previamente). Y es **here** quien se encarga de realizar todas las operaciones internas de chequear el carácter separador, ubicar la carpeta raíz del proyecto y validar toda la dirección.

El proceso inverso, leer **desde disco a un objeto R**, el comando requiere la dirección y nombre del archivo que contiene el *objeto a leer*⊛. Se recomienda utilizar el mismo nombre del objeto original:

```
objeto <- readRDS(here("raw", "objeto.rds"))
```

2.5 Consejos Prácticos para escribir R

Algunas recomendaciones básicas para escribir código R, para facilitar la lectura y comprensión del código.

- Colocar espacios alrededor de todos los operadores (=, +, -, <-, etc.).
- Utilizar <-, no =, para la asignación.
- Siempre use comentarios para marcar secciones de código.
- Comentar el código detalladamente. Los comentarios deben explicar **por qué**, no **el qué**.
- Cada línea de un comentario debe comenzar con el símbolo de comentario y un solo espacio.
- Una llave de apertura nunca debe ir en su propia línea y siempre debe ir seguida de una nueva línea; una llave de cierre siempre debe ir en su propia línea, a menos que vaya seguida de otra.
- Siempre sangra el código dentro de las llaves.
- Mantener sus líneas de menos de 80 caracteres. Esta es la cantidad que cabrá cómodamente en una página impresa en un tamaño razonable.

Si nota que se está quedando sin espacio, esto probablemente sea una indicación de que debe encapsular parte del trabajo en una función separada.

R como Lenguaje de Programación

Hasta este punto se ha escrito y utilizado código paso a paso, utilizando R desde la consola como una calculadora, con cada línea de código ejecutándose individualmente, y alguna función ocasional se ha aplicado individualmente a un conjunto de datos o atributos específicos.

En nuestro recién creado archivo *Script* (***.r***) y utilizando el editor de texto vamos a escribir desde ahora en adelante nuestro código r expresado en varias líneas. Usando *identación.*

Muy a menudo, en el uso de data ráster, nos gustaría *hacer lo mismo repetidamente*, pero *ajustando algunos de los parámetros en cada iteración*, por ejemplo, aplicando el mismo algoritmo a diferentes datos, diferentes atributos o usando diferentes umbrales.

El objetivo de este capítulo es introducir algunos principios básicos de programación que permitirán hacer muchas cosas repetidamente en un solo bloque de código.

Estos son los conceptos básicos para escribir programas de computadora.

En el análisis y mapeo de datos ráster, con frecuencia queremos aplicar el mismo conjunto de comandos una y otra vez, para recorrer los datos o listas de datos, o modificando los datos dependiendo de si se cumple alguna condición o no. Estos tipos de acciones repetidas están respaldados por **funciones**, **bucles** y **condiciones lógicas**.

Unos pocos ejemplos simples sirven para ilustrar cómo la **programación en R** combina estas ideas a través de funciones con comandos condicionales, bucles y variables.

Por ejemplo, considere la siguiente variable altura_arbol:

```
altura_arbol <- c(5.2, 9.1, 4.3, 3.2, 6.2)
```

Es posible que deseemos imprimir el primer elemento de esta variable si *tiene un valor menor que 6*: este es un *comando condicional* ya que la operación (en este caso imprimir un contenido) se realiza de forma condicional (es decir, si se cumple la condición).

```
if (altura_arbol[1] < 6)  {
  cat('árbol pequeño\n')
} else  {
  cat('árbol grande\n')
}
```

```
árbol pequeño
```

Alternativamente, es posible que deseemos examinar todos los elementos de la variable `altura_arbol` y, dependiendo de si cada valor individual cumple con la condición, realizar la misma operación.

Podemos realizar operaciones repetidamente utilizando una *estructura de bucle* de la siguiente manera, observe la construcción del bucle *for* en la forma:

```
for (variable in secuencia) {código R}
```

Esto se ilustra en el siguiente código:

```
for (i in 1:5) {
  if (altura_arbol[i] < 6)  {
    cat('árbol', i, 'es pequeño\n')
  } else  {
    cat('árbol', i, 'es grande\n')
  }
}
```

```
árbol 1 es pequeño
árbol 2 es grande
árbol 3 es pequeño
árbol 4 es pequeño
árbol 5 es grande
```

Una tercera situación es cuando deseamos realizar el mismo conjunto de operaciones, grupo de funciones condicionales o en bucle una y otra vez, quizás a *diferentes datos*. Podemos hacer esto agrupando código y definiendo nuestras propias *funciones*.

```
medir_altura_arbol <- function(lista, marca) {
  for (i in 1:length(lista)) {
    if (lista[i] < marca)  {
      cat('árbol', i, 'es pequeño\n')
    } else  {
      cat('árbol', i, 'es grande\n')
    }
  }
}

medir_altura_arbol(altura_arbol, 5)

árbol 1 es grande
árbol 2 es grande
árbol 3 es pequeño
árbol 4 es pequeño
árbol 5 es grande

medir_altura_arbol(altura_arbol, 8)

árbol 1 es pequeño
árbol 2 es grande
árbol 3 es pequeño
árbol 4 es pequeño
árbol 5 es pequeño

otro_arbol <- c(4.5, 9.44,11.3,3.66)
medir_altura_arbol(lista = otro_arbol,
                   marca = 8)

árbol 1 es pequeño
árbol 2 es grande
árbol 3 es grande
árbol 4 es pequeño
```

Note como el código de la función `medir_altura_arbol(lista, marca)` modifica el ciclo original desde `for (i in 1:5)` a `for`

`(1:length(lista))` para definir el número de veces que vamos a repetir el ciclo. El parámetro `marca` se define para dar mayor flexibilidad al usuario.

2.6 Estructuras de Programación

En los ejemplos de arriba, se han introducido un número de conceptos claves para la programación. Ahora vamos a explicar algunos aspectos importantes.

2.6.1 Comandos Condicionales

Un comando condicional prueba por una *condición* si es verdadera (TRUE) o falsa (FALSE), y si la respuesta es verdadera algunas acciones deben ser realizadas.

Las operaciones condicionales se componen por `if` y `else`. La instrucción **if** es seguida por la *condición*, la expresión que es evaluada, y una *consecuencia* será ejecutada si la condición resulta verdadera.

```
x <- 9
if (x > 0) cat("es positivo")
```

```
es positivo
```

En el siguiente ejemplo, no se cumple la condición y nada ocurre:

```
x <- -9
if (x > 0) cat("es positivo")
```

También la instrucción *if* posee una *alternativa* que será ejecutada o evaluada cuando la condición sea falsa.

```
x <- -9
if (x > 0) cat("es positivo") else cat("es negativo")
```

```
es negativo
```

Las operaciones condicionales son compuestas por uno o más operadores lógicos (Cuadro 2.1). Además, R contiene una serie de *funciones lógicas* que también se pueden usar para evaluar condiciones, un ejemplo en cuadro 2.2) pero existen muchas otras.

Cuadro 2.1: Operadores Lógicos

Símbolo	Definición
==	Es igual
!=	Distinto
>	Mayor
>=	Mayor o igual
<	Menor
<=	Menor o igual
&	Y
|	O
!	Negación

Cuadro 2.2: Funciones Lógicas

Función	Descripción
any(x)	TRUE si alguna condición es verdadera
all(x)	TRUE si todas las condiciones son verdaderas
is.numeric(x)	TRUE si x es valor numérico
is.logical(x)	TRUE si x es TRUE or FALSE
is.character(x)	TRUE si x es carácter

Un ejemplo de su uso:

```
listado <- c(4.5, 9.44,11.3,3.66)
if (all(listado > 0)) cat("todos los valores positivos")
```

```
todos los valores positivos
```

```
listado <- c(4.5, 9.44, -11.3, -3.66)
if (any(listado > 0)) cat("algunos valores son positivos")
```

```
algunos valores son positivos
```

```
any(listado == 0)
```

```
[1] FALSE
```

2.6.2 Bloque de Código

Grupos de instrucciones son llamados **bloque de código** y en R deben ser contenidos entre símbolos *corchetes* { y }.

2.6.3 Funciones

En la introducción creamos la función:

```
medir_altura_arbol <- function(lista, marca) {
  for (i in 1:length(lista)) {
    if (lista[i] < marca)  {
      cat('árbol', i, 'es pequeño\n')
    } else  {
      cat('árbol', i, 'es grande\n')
    }
  }
}
```

Que podemos diagramar de la siguiente forma:

nombre <- **function**(lista de argumentos) {expresiones R}

Las *expresiones R* conforman un bloque de código que recibe un *nombre* y puede recibir una *lista de argumentos* que nos permite evaluar el código con distintos valores que el usuario desea utilizar.

Dentro del bloque de código se pueden crear nuevos objetos (variables) que incluso pueden tener el mismo nombre que objetos externos al bloque, solo *existen dentro del bloque*.

2.6.4 Ciclos y Repeticiones

En la introducción creamos un ciclo controlados por la secuencia *for* (1:length…) {} que define un cierto número de veces (cada uno de los componentes del vector).

R posee por su naturaleza vectorial (1.12) varias formas para crear secuencias (1.12.2), definir patrones (1.12.4) o repeticiones (1.12.3) que pueden ser usados para el control de un ciclo.

```
for (val in seq(0, 1, by= 0.25)){
}
```

Otra forma de controlar los ciclos es mediante el uso de *condiciones*, útiles cuando se desea ejecutar un bloque de código hasta que una condición lógica sea verdadera.

Las funciones `repeat()` y ‘kreak“ permiten la construcción de dicho tipo de ciclo:

```
i <- 1; n <- 200
repeat{
  cuadrado <- i * i
  if (cuadrado > n) break
  i <- i + 1
}
cat("El primer número que excede",n, "es", cuadrado)
```

```
El primer número que excede 200 es 225
```

También es posible incluir ciclo dentro de las funciones como en el siguiente ejemplo:

```
primer_mayor_cuadrado <- function(n){
  i <- 1
  repeat{
    cuadrado <- i * i
    if (cuadrado > n) break
    i <- i + 1
  }
  return (cuadrado)
}

primer_mayor_cuadrado(13256)
```

```
[1] 13456
```

III INFORMACIÓN RÁSTER

Introducción al Concepto Ráster

Los fenómenos geográficos se desarrollan de manera continua sobre una extensión de la superficie terrestre (en, sobre o bajo ella) aunque convertir la realidad a una expresión simple que se pueda ser incorporado a un programa informático no es nada de simple.

El primer paso es describir variables independientes de la manera más precisa posible y para ello se requiere la construcción de un *modelo* de aquella variable que **represente** esa realidad y pueda servir para estudiar en profundidad ya no en la realidad sino sobre el modelo en sí.

El problema básico es como reducir una cantidad casi infinita de detalles en un monto que permita su almacenamiento y su representación. El método ideal es extraer una serie de valores representativos que siguiendo una estrategia permita replicar el fenómeno.

Si podemos enumerar los pasos o niveles que llevan de la realidad a un conjunto de datos numéricos dependiendo del nivel de abstracción requerido son tres:

- *Modelo Geográfico:* Modelo conceptual de la realidad geográfica.
- *Modelo de Datos:* Tomar el modelo anterior con sus características y reducirlo a una serie finita de elementos.
- *Modelo de Almacenamiento:* Esquemas de almacenamiento físico del modelo de datos.

3.1 Modelo Geográfico

El modelo geográfico es la base del sistema ya que modela teóricamente la realidad geográfica y su comportamiento. Es una conceptualización o esquema mental que permita entender el fenómeno geográfico en sí, el espacio que ocupa, la variable tratada y cómo se comporta en ese espacio. Existen varios modelos y basados en Couclelis [1992] podemos mencionar:

- Entidades Discretas
- Campos

3.1.1 Entidades Discretas

Se basa en el supuesto de que el espacio geográfico se encuentra vacío y sobre él existen distintos elementos que lo rellenan. Dichos elementos poseen características propias y constantes dentro de límites bien definidos que pueden extenderse desde la dimensión cero (*punto*), dimensión uno (*arco* o *línea*), dimensión dos (*polígono*) y superiores.

3.1.2 Campos

Si consideramos que sobre el espacio geográfico se desarrollan los elementos de manera continua (principalmente variables físicas) vamos a encontrar que para cada punto existe su correspondiente magnitud.

Así una vía o un curso de agua se adapta naturalmente al modelo de entidades discretas y la altitud o temperatura se comporta de manera más natural en el modelo de campos.

3.2 Modelo de Datos

Vamos a clasificar los modelos detallados en dos grupos, modelo de representación *ráster* y modelo de representación *vectorial.*(Figura 3.1, basado en Galton [2001]).

El modelo ráster se basa en una subdivisión regular del espacio geográfico generalmente cuadradas ordenadas en una grilla

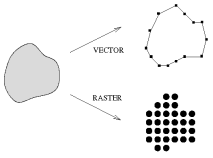


Figura 3.1: Representación de Modelos Vector y Ráster

regular continua como una matriz. Pueden tener más de un valor (atributos o bandas) asociado a cada celda.

El modelo vectorial no divide el espacio en su totalidad sino por medio de una serie de elementos geométricos con valores (atributos) asociados y cada uno de ello relacionado con los objetos geográficos presentes.

3.3 Modelo de Almacenamiento

Los modelos de almacenamiento plantean básicamente un esquema de cómo convertir dichas unidades en valores numéricos de la forma más eficiente. Es decir, cómo escribir dichos valores en un soporte digital o guardarlos en la memoria del ordenador de la mejor manera posible.

Los modelos de almacenamiento deben atender principalmente a dos necesidades básicas, que son las que definirán su idoneidad para cada tarea y tipo de dato:

- Minimizar el espacio ocupado por los datos.
- Maximizar la eficiencia de cálculo.

En R el modelo de almacenamiento ráster se implementa vía el *paquete*[1] ***raster*** Hijmans [2021] que permite:

> "leer, escribir, manipular, analizar y modelar datos espaciales en grilla cuadricular. El paquete implementa funciones básicas y de alto nivel. Se admite el procesamiento de archivos muy grandes."

Como primer paso si no se ha realizado anteriormente debemos instalar el paquete **solo la primera vez** *raster*:

```
install.packages('raster')
```

Luego (y en las sesiones futuras directamente avanzar a este paso) se debe activar el código contenido en el paquete para la sesión actual con:

```
library(raster)
```

[1] Todas las funciones y conjuntos de datos de R se almacenan en ***paquetes***. Solo cuando se carga un paquete, su contenido está disponible. Esto se hace tanto por eficiencia (la lista completa tomaría más memoria y tardaría más en buscar que en un subconjunto).

3.4 Conversión de Objeto a Ráster

3.4.1 Objeto RasterLayer

Cuando a partir de un objeto bidimensional (*matriz*) se crea un objeto especialmente diseñado para el análisis de datos ráster en una variable:

```
mat <- matrix(1:36, 6, 6)
mat
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    7   13   19   25   31
[2,]    2    8   14   20   26   32
[3,]    3    9   15   21   27   33
[4,]    4   10   16   22   28   34
[5,]    5   11   17   23   29   35
[6,]    6   12   18   24   30   36
```

Utilizando la función *raster* construimos un objeto *RasterLayer* (Figura 3.2):

```
base <- raster(mat)
base
```

```
class      : RasterLayer
dimensions : 6, 6, 36  (nrow, ncol, ncell)
resolution : 0.1666667, 0.1666667  (x, y)
extent     : 0, 1, 0, 1  (xmin, xmax, ymin, ymax)
crs        : NA
source     : memory
names      : layer
values     : 1, 36  (min, max)
```

Información fundamental que es almacenada que describe la información contenida es:
dimensions: Descripción geométrica del modelo, número de filas, columnas.
resolution: Tamaño (en m.) de las celdas en eje x e y.
extent: Coordenadas extremas del área cubierta por el modelo.
crs: Sistema de referencias de coordenadas. Proyección del mapa.

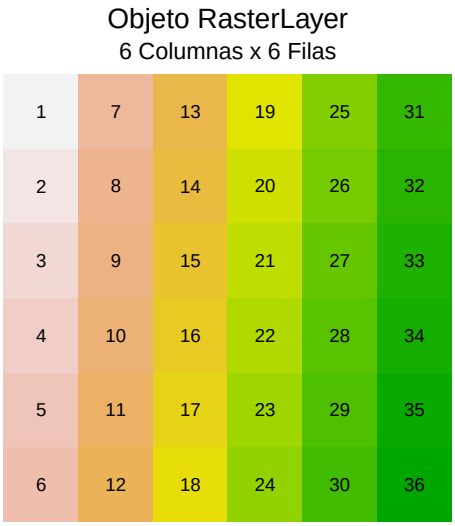


Figura 3.2: Modelo de Representación Ráster.

source: Dónde reside la información (en archivo o memoria) de las celdas contenidas en el modelo.
names: Nombre de la capa o variable.
values: Rango numérico del contenido en los valores del modelo.

3.4.2 Objeto RasterBrick

Un RasterBrick es un objeto ráster de **varias capas**. Por lo general, se crean a partir de un archivo de varias capas (bandas); pero también pueden existir enteramente en la memoria. Son similares a un **RasterStack** (que se puede crear con stack, ver 3.4.3).

El tiempo de procesamiento *debería ser más corto cuando se usa un RasterBrick*. Sin embargo, son menos flexibles ya que solo pueden apuntar a **un único archivo**.

Se puede crear un RasterBrick a partir de objetos RasterLayer, desde un RasterStack o desde un archivo (multicapa). También se puede crear a partir de un array tridimensional:

```
arr <- array(data = 1:6**3, dim = c(6, 6, 3))
arr
```

```
, , 1

     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    7   13   19   25   31
[2,]    2    8   14   20   26   32
[3,]    3    9   15   21   27   33
[4,]    4   10   16   22   28   34
[5,]    5   11   17   23   29   35
[6,]    6   12   18   24   30   36

, , 2

     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]   37   43   49   55   61   67
[2,]   38   44   50   56   62   68
[3,]   39   45   51   57   63   69
[4,]   40   46   52   58   64   70
[5,]   41   47   53   59   65   71
[6,]   42   48   54   60   66   72

, , 3

     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]   73   79   85   91   97  103
[2,]   74   80   86   92   98  104
[3,]   75   81   87   93   99  105
[4,]   76   82   88   94  100  106
[5,]   77   83   89   95  101  107
[6,]   78   84   90   96  102  108
```

El comando *brick* convierte el objeto array a *RasterBrick* que contendrá la misma información que el rasterlayer incorporando el detalle para una de las capas contenidas en él (Figura 3.3).

```
multi <- brick(arr)
multi
```

```
class       : RasterBrick
dimensions  : 6, 6, 36, 3  (nrow, ncol, ncell, nlayers)
resolution  : 0.1666667, 0.1666667  (x, y)
extent      : 0, 1, 0, 1  (xmin, xmax, ymin, ymax)
crs         : NA
source      : memory
names       : layer.1, layer.2, layer.3
min values  :       1,      37,      73
max values  :      36,      72,     108
```

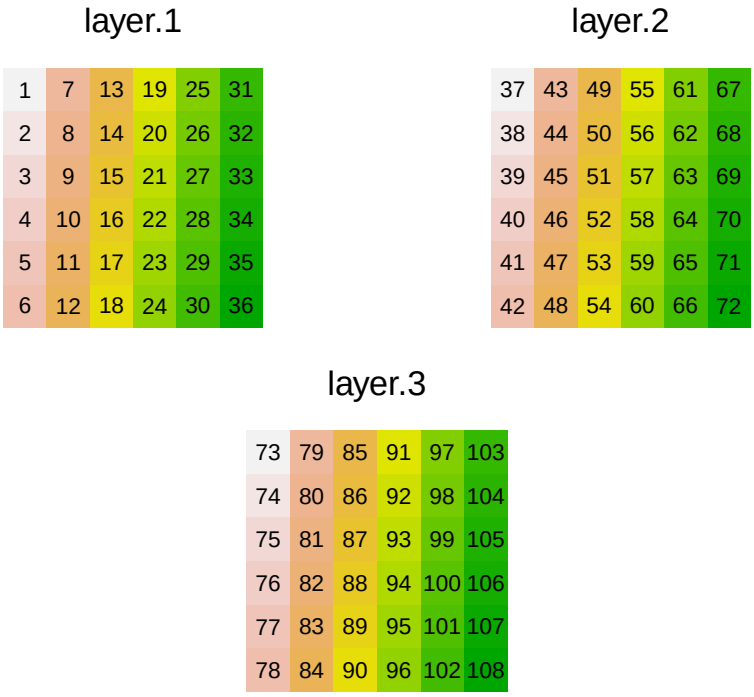


Figura 3.3: Conceptualización de RasterBrick.

3.4.3 Objeto RasterStack

Un RasterStack es una *colección de objetos RasterLayer* con la misma **extensión espacial** y **resolución**. Se puede crear un RasterStack a partir de objetos RasterLayer, o de **distintos ar-**

chivos ráster, o **ambos**.

El tiempo de procesamiento *debería ser más largo cuando se usa un RasterStack*. Sin embargo, son más flexibles ya que pueden apuntar a **distintos archivos**.

Revisaremos en detalle el uso del comando `stack()`, en elcaso práctico de carga de datos de población (ver 6.2).

3.4.4 addLayer y dropLayer

Comandos especializados en agregar una capa a un objeto *Raster* o remover una capa desde *RasterStack* o *RasterBrick*.

Siempre el objeto devuelto es un RasterStack (a menos que no se haya proporcionado *nada para agregar o quitar*, en cuyo caso se devuelve el objeto original).

Sintaxis :

- `addLayer (x, ...)`
- `dropLayer (x, i, ...)`

Donde `x` es objeto ráster e `i` un valor entero que define el o los índices de las capas que se eliminarán.

`...` Siempre definen argumentos adicionales. Para las capas que se agregarán por addLayer. Ninguno implementado para dropLayer).

Operaciones Básicas

No olvidemos que si en nuestra sesión actual no hemos cargado el paquete *raster* debemos iniciar con este paso:

```
library(raster)
```

A diferencia del capítulo anterior vamos a crear un objeto *RasterLayer* desde cero, utilizando parámetros con nombre para definir tanto el número de columnas y filas como la ubicación geográfica:

```
(base <- raster(ncol= 9,
                nrow= 9,
                xmx= -70,
                xmn= -71,
                ymx= -32,
                ymn= -33
                )
 )
```

```
class      : RasterLayer
dimensions : 9, 9, 81  (nrow, ncol, ncell)
resolution : 0.1111111, 0.1111111  (x, y)
extent     : -71, -70, -33, -32  (xmin, xmax, ymin, ymax)
crs        : +proj=longlat +datum=WGS84 +no_defs
```

Así tenemos la definición geométrica, tamaño, resolución y ubicación pero no contiene datos. Para efectos de nuestra demostración vamos a asociar un vector de *valores aleatorios*[2] de la misma longitud que el número de celdas (Figura 3.4):

[2] `runif` genera números aleatorios en el rango *min*, *max*.

```
values(base) <- as.integer(runif(ncell(base),
                                 min = 1,
                                 max=6)
                           )
base
```

```
class       : RasterLayer
dimensions  : 9, 9, 81  (nrow, ncol, ncell)
resolution  : 0.1111111, 0.1111111  (x, y)
extent      : -71, -70, -33, -32  (xmin, xmax, ymin, ymax)
crs         : +proj=longlat +datum=WGS84 +no_defs
source      : memory
names       : layer
values      : 1, 5  (min, max)
```

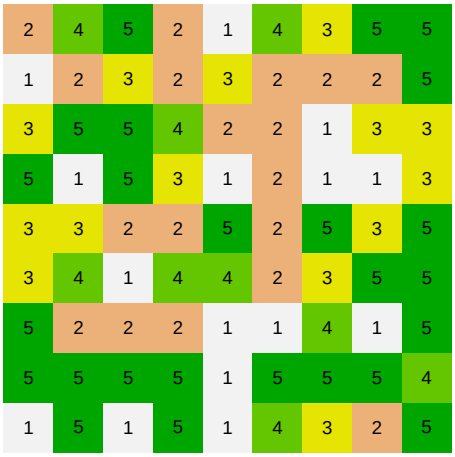


Figura 3.4: Objeto rás-ter (los números indi-can el valor contenido en cada celda).

3.5 Estructura Interna slots

El objeto *RasterLayer* creado en el punto anterior, automáti-camente e internamente asocia una serie de atributos y propieda-des que en R se denominan **slots** (ranuras). Cada ranura tiene un *nombre* y una *clase* asociada. La extracción de un slot devuelve un objeto de esa clase.

Tenga en cuenta que el operador `@` (arroba) para extraer o modificar el contenido de un slot, función primitiva y reside en el

paquete base.

Para listar los slots que componen un objeto debemos usar:

```
slotNames(base)
```

```
 [1] "file"     "data"     "legend"   "title"    "extent"   "rotated"
 [7] "rotation" "ncols"    "nrows"    "crs"      "history"  "z"
```

Y para revisar el contenido basta usar el nombre:

```
slot(base, name = 'extent')
```

```
class       : Extent
xmin        : -71
xmax        : -70
ymin        : -33
ymax        : -32
```

O usar el formato `objeto@nombre`:

```
base@file
```

```
An object of class ".RasterFile"
Slot "name":
[1] ""

Slot "datanotation":
[1] "FLT4S"

Slot "byteorder":
[1] "little"

Slot "nodatavalue":
[1] -Inf

Slot "NAchanged":
[1] FALSE

Slot "nbands":
[1] 1
```

```
Slot "bandorder":
[1] "BIL"

Slot "offset":
[1] 0

Slot "toptobottom":
[1] TRUE

Slot "blockrows":
[1] 0

Slot "blockcols":
[1] 0

Slot "driver":
[1] ""

Slot "open":
[1] FALSE
```

3.6 Propiedades Geométricas

Nuestro objeto rasterlayer posee una serie de propiedades que se pueden revisar mediante los mismos comandos que ya hemos estudiado para las matrices:

```
dim(base); ncol(base); nrow(base)
```

```
[1] 9 9 1
[1] 9
[1] 9
```

Y algunos especialmente diseñado para él:

```
xres(base) #Tamaño eje x
```

```
[1] 0.1111111
```

```
yres(base) #Tamaño eje y
```

```
[1] 0.1111111
```

```
res(base) #Tamaño en ambos ejes
```

```
[1] 0.1111111 0.1111111
```

```
ncell(base) #Número total de elementos
```

```
[1] 81
```

```
extent(base) #retorna objeto extensión
```

```
class      : Extent
xmin       : -71
xmax       : -70
ymin       : -33
ymax       : -32
```

3.7 Estadísticas Generales

Los valores contenidos en el objeto ráster pueden ser explorados utilizando estadísticos globales mediante el comando `cellStats`:

```
cellStats(base,stat=min) #valor mínimo
```

```
[1] 1
```

```
cellStats(base,stat=max) #valor máximo
```

```
[1] 5
```

El comando acepta el paso del parámetro mediante un 'texto' que funciona más rápido en ráster grandes.

```
cellStats(base, 'mean') #Promedio de los valores
```

```
[1] 3.135802
```

```
cellStats(base,'sd') #Desviación estándar
```

```
[1] 1.52277
```

```
cellStats(base, 'sum') #Sumatoria
```

```
[1] 254
```

Para extraer el contenido y su frecuencia existen `unique` y `freq` que retorna un vector con todos los elementos duplicados removidos y el segundo construye una tabla de frecuencia para los valores contenidos (realiza por defecto un redondeo a número entero y se puede controlar con el parámetro *digits*).

```
unique(base)
```

```
[1] 1 2 3 4 5
```

```
freq(base, digits= 0)
```

```
     value count
[1,]     1    15
[2,]     2    18
[3,]     3    14
[4,]     4     9
[5,]     5    25
```

3.8 Nombre de Capas

También podemos acceder a los nombres de las capas y modificarlas por claridad u ordenamiento de la data:

```
names(base) <- 'mi_variable'
base
```

```
class      : RasterLayer
dimensions : 9, 9, 81  (nrow, ncol, ncell)
resolution : 0.1111111, 0.1111111  (x, y)
extent     : -71, -70, -33, -32  (xmin, xmax, ymin, ymax)
crs        : +proj=longlat +datum=WGS84 +no_defs
source     : memory
names      : mi_variable
values     : 1, 5  (min, max)
```

3.9 Indexado y Subconjuntos

3.9.1 Número de Celdas

Un objeto ráster se almacena internamente como un vector con sus elementos ordenados secuencialmente, iniciando en 1 en la esquina superior izquierda, aumenta a la derecha y hacia abajo. El último número es igual al número de celdas. La organización se ilustra en la Figura 3.5.

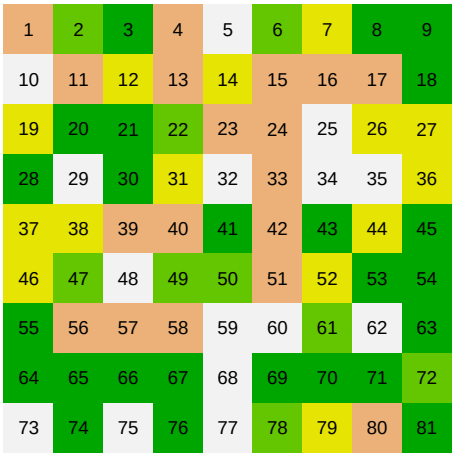


Figura 3.5: Objeto ráster con el número de orden de celdas.

Así podemos extraer información por su *ids*:

```
base[10]
```

```
[1] 1
```

```
base[2:6]
```

```
[1] 4 5 2 1 4
```

Y usando este tipo de numeración encontramos una serie de funciones que permiten obtener coordenadas reales a partir del número de celda:

```
xFromCell(base,1) #Coordenada x de celda 1
```

```
[1] -70.94444
```

```
yFromCell(base,1) #Coordenada y de celda 1
```

```
[1] -32.05556
```

```
xyFromCell(base,1:3) #Vector coordenada de celda 1 a 3
```

```
             x         y
[1,] -70.94444 -32.05556
[2,] -70.83333 -32.05556
[3,] -70.72222 -32.05556
```

Y su recíproco; a partir de coordenadas x e y obtener el número de *celda*.

```
cellFromXY(base, c(-70.4,-32.8)) #Número de celda
```

```
[1] 69
```

Y la forma de realizar una prueba de validez de ids:

```
validCell(base,0) #id fuera del vector
```

```
[1] FALSE
```

```
validCell(base,82) #id fuera del vector
```

```
[1] FALSE
```

```
validCell(base,1)
```

```
[1] TRUE
```

3.9.2 Filas y Columnas

El siguiente método de identificación de elementos es el par de valores [fila, columna]. Las filas incrementan hacia abajo y columnas hacia la derecha. De 1 a *ncol* y 1 a *nrow*. Ver figura 3.6.

El método de acceso es similar al detallado para las matrices:

```
base[2,5]
```

```
[1] 3
```

```
base[1:3,1:3]
```

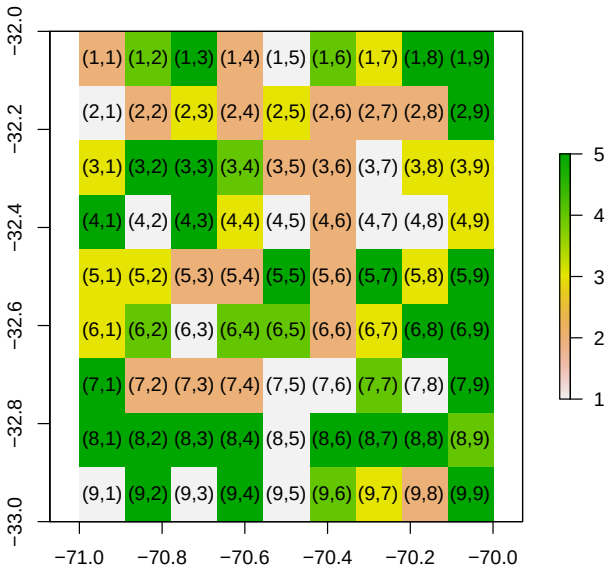


Figura 3.6: Objeto ráster (los números indican la combinación (fila, columna) para cada).

```
[1] 2 4 5 1 2 3 3 5 5
```

Y utilizando el parámetro *drop=FALSE* retorna un objeto rasterlayer (Figura 3.7):

```
base[1:3,1:3, drop=FALSE]
```

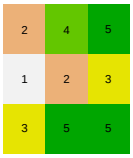


Figura 3.7: Objeto ráster producto del subconjunto.

```
class       : RasterLayer
dimensions  : 3, 3, 9  (nrow, ncol, ncell)
resolution  : 0.1111111, 0.1111111  (x, y)
extent      : -71, -70.66667, -32.33333, -32  (xmin, xmax, ymin, ymax)
crs         : +proj=longlat +datum=WGS84 +no_defs
source      : memory
names       : layer
values      : 1, 5  (min, max)
```

Con este tipo de numeración tenemos una serie de funciones que permiten obtener coordenadas reales a partir de la información de filas y columnas:

```
xFromCol(base,1) #Coordenada x de columna 1
```

```
[1] -70.94444
```

```
yFromRow(base,1) #Coordenada y de fila 1
```

```
[1] -32.05556
```

El inverso:

```
colFromX(base, -70.15) #Columna para coordenada x
```

```
[1] 8
```

```
rowFromY(base, -32.18) #Fila para coordenada y
```

```
[1] 2
```

Mientras que conversión de fila y columna a número de celdas:

```
cellFromRowCol(base,2,2)
```

```
[1] 11
```

Y número de celda a fila y columna:

```
colFromCell(base, 28)
```

```
[1] 1
```

```
rowFromCell(base, c(1,5,15))
```

```
[1] 1 1 2
```

```
rowColFromCell(base, c(1,5,15))
```

```
     row col
[1,]   1   1
[2,]   1   5
[3,]   2   6
```

Sistemas Coordenados de Referencia (CRS)

3.10 Fundamentos

En los atributos destacados del objeto base aparece el ítem *crs* con el texto `+proj=longlat +datum=WGS84 +no_defs` que corresponde al sistema de coordenadas utilizado por defecto, el sistema mundial para dispositivos GPS o WGS84. El **WGS84** (*World Geodetic System 1984*) es un sistema geodésico de coordenadas geográficas usado mundialmente, que permite localizar cualquier punto de la Tierra (sin necesitar otro de referencia) por medio de tres unidades dadas (x,y,z). El texto esta formateado de acuerdo con el estándar que utiliza la aplicación **PROJ** [3] https://proj.org/ PROJ contributors [2021].

Los sistemas de coordenadas son muy variados y han sido sistematizados bajo el nombre genérico de **EPSG**[4] y esta estructura es la manera práctica y recomendada de uso en R; el mismo sistema WGS84 corresponde bajo esta nomenclatura al código EPSG:4326. El sitio de la iniciativa es www.spatialreference.org y se recomienda su visita y estudio.

Cuando se utiliza un CRS, debes considerar que existen los sistemas de **coordenadas proyectadas** y **geográficas**, también que existen muchos tipos de sistemas de coordenadas.

Los CRS se definen básicamente por:

- Marco de medición, que puede ser *geográfico* (coordenadas esféricas) o *planimétrico* (coordenadas proyectadas sobre un plano).

- Unidad de medición, generalmente serán metros (¡o pies!) para

[3] La sintaxis **PROJ** es una lista de parámetros antecedidos por *+*.

[4] **European Petroleum Survey Group** o **EPSG** (1986–2005) fue una organización científica vinculada a la industria del petróleo europea. Formada por especialistas del campo de la geodesia, topografía y cartografía aplicadas en la exploración petrolífera. EPSG compiló y difundió el conjunto de parámetros geodésicos **EPSG** como una base de datos que contiene elipsoides, datums, sistemas de coordenadas, proyecciones cartográficas, etc.

sistemas de coordenadas proyectadas o grados decimales para latitud/longitud).

Encontramos otras propiedades, como un **esferoide de referencia**, un **datum** y **parámetros de proyección** como uno o más paralelos estándar, un meridiano central o posibles cambios en las direcciones de x e y.

Existen dos tipos comunes de sistemas de coordenadas que se utilizan en el análisis espacial:

- **Un sistema de coordenadas geográficas** utiliza una superficie esférica, de tres dimensiones para definir ubicaciones en la tierra (en donde las coordenadas se miden desde el centro de la tierra). La *latitud* y la *longitud* definen un sistema de coordenadas esféricas o globales.

- **Un sistema de coordenadas proyectadas** se define en una superficie plana, de dos dimensiones, como un mapa impreso o visualizado, por lo tanto, también se denomina proyección cartográfica. Una proyección cartográfica utiliza fórmulas matemáticas para relacionar las coordenadas esféricas del globo con coordenadas planares, planas.

A diferencia de un sistema de coordenadas geográficas, un sistema de coordenadas proyectadas posee longitudes, ángulos y áreas constantes en las dos dimensiones. Sin embargo, todas las proyecciones cartográficas que representan la superficie de la tierra en un mapa plano crean distorsiones en algún aspecto de la **distancia**, el **área**, la **forma** o la **dirección** ya que se trata de poner datos en tres dimensiones en un plano de dos dimensiones. Cuando selecciona una referencia espacial para almacenar y remuestrear los datos ráster, debe elegir la proyección que minimiza el tipo de distorsión que más le preocupa.

Como las proyecciones producen distintos tipos de distorsiones, se diseñan con **fines específicos**. Datos a gran escala en un área limitada, otra se puede utilizar para un mapa a pequeña escala del mundo o minimizar la distorsión de una o más características de datos (distancia, área, forma o dirección), etc.

Minimizar estas limitaciones utilizamos proyecciones cartográficas que se adapten al **uso**, la **ubicación geográfica** y la **ex-**

tensión deseados.

3.10.1 Identificación y Extracción

Para identificar el CRS de un objeto ráster usamos:

```
crs(base)
```

```
CRS arguments: +proj=longlat +datum=WGS84 +no_defs
```

y para definir en un objeto que carece de él (por ejemplo, construido manualmente como en el capítulo anterior) se utiliza con la sintaxis de asignación:

```
crs(objeto) <- crs('descripción PROJ o código EPSG')
```

3.10.2 Transformación Ráster

Un ráster consta de celdas rectangulares del mismo tamaño (en términos de las unidades del CRS aunque su tamaño real puede variar) y no es posible transformar celda a celda. Así, las estimaciones de los valores de las celdas nuevas deben realizarse *en función de los valores de las celdas antiguas*. Si los valores son datos de clase o categóricos, normalmente se utiliza el *vecino más cercano* (en R es método *'ngb'*), si la información es continúa se realiza algún tipo de *interpolación* (en R el método es bilineal *'bilinear'*). Figura 3.8.

Para definir el CRS destino debemos usar el código EPSG en un texto con formato `+init=epsg:CODIGO_EPSG` y automáticamente la función construirá y agregará todos los parámetros.

```
utm19  <- crs("+init=epsg:32719") # UTM 19S WGS84
utm19
```

```
CRS arguments:
 +proj=utm +zone=19 +south +datum=WGS84 +units=m +no_defs
```

Y así la transformación de CRS se realiza con el comando `projectRaster` (Figura 3.9).

```
base_proj <- projectRaster(from = base,
                           crs = utm19,
```

Figura 3.8: Diferencias en el método de estimación de valores.

```
                                    method= 'ngb') # method= 'bilinear'
```

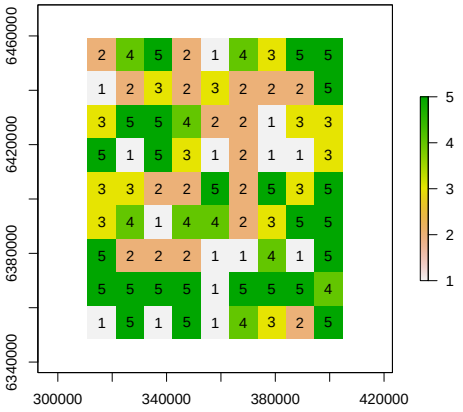


Figura 3.9: Objeto ráster proyectado a nuevo CRS.

En la figura 3.8 tenemos los valores obtenidos con distintos métodos de interpolación y podemos comparar sus resultados con los valores originales en la figura 3.4).

En el caso anterior hemos tomado un objeto ráster y es proyectado a un nuevo CRS, un segundo método que da más control sobre el proceso es agregar a la operación de transformación un objeto existente para que así la alineación del nuevo objeto sea correcta y precisa. También en el momento de realizar el proceso, se puede cambiar la resolución (el tamaño de la celda) del producto final.

En nuestro caso vamos a crear un objeto ráster a partir del objeto original:

```
base_h19s <- projectExtent(base, utm19)
```

Y modificamos su atributo de resolución:

```
res(base_h19s) <- 10000 # también (x,y) ej. c(10000,15000)
```

Y usamos este objeto como molde de destino donde verter la información del objeto fuente y crear el nuevo objeto de salida (Figura 3.10:

```
base_h19s_10k <- projectRaster(from = base,
                               to = base_h19s,
                               method = 'ngb')
```

```
base_h19s_10k
```

```
class      : RasterLayer
dimensions : 11, 10, 110  (nrow, ncol, ncell)
resolution : 10000, 10000  (x, y)
extent     : 311072.4, 411072.4, 6349127, 6459127  (xmin, xmax, ymin, ymax)
crs        : +proj=utm +zone=19 +south +datum=WGS84 +units=m +no_defs
source     : memory
names      : mi_variable
values     : 1, 5  (min, max)
```

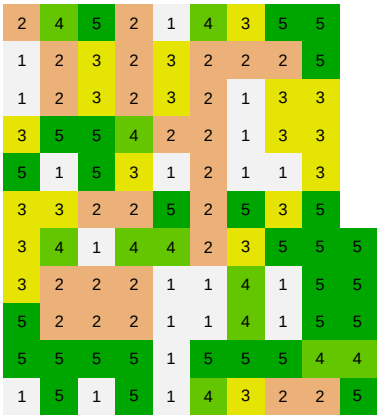


Figura 3.10: Objeto ráster proyectado a nuevo CRS y resolución menor al original.

3.10.3 Transformación Vectorial

Hemos revisado la transformación de objetos ráster y ahora vamos a transformar objetos vectoriales, la función `extent(x)`

extrae de un rasterLayer o un objeto espacial (vector) un objeto *extent* pero no contiene información definida de un CRS.

```
ext <- extent(base)
crs(ext)
```

```
[1] NA
```

La función `as(objeto, clase)` convierte un *objeto* al tipo *clase*.

```
polygon <- as(object = ext,
              Class =  'SpatialPolygons')
polygon
```

```
class       : SpatialPolygons
features    : 1
extent      : -71, -70, -33, -32  (xmin, xmax, ymin, ymax)
crs         : NA
```

Y se debe asignar un crs() válido para las coordenadas que contiene (Figura 3.11, izquierda):

```
crs(polygon) <- crs("+proj=longlat +datum=WGS84 +no_defs")
polygon
```

```
class       : SpatialPolygons
features    : 1
extent      : -71, -70, -33, -32  (xmin, xmax, ymin, ymax)
crs         : +proj=longlat +datum=WGS84 +no_defs
```

La función `spTransform(objeto, crs)` transforma el *objeto* a un nuevo sistema definido por el *crs*. En el ejemplo vamos a convertir nuestro objeto *polygon* por ejemplo a la proyección **sinusoidal** (EPSG: 6974) (Figura 3.11, derecha):

```
proj_sinu <- "+proj=sinu +lon_0=0 +x_0=0 +y_0=0 +ellps=WGS84 +datum=WGS84 +units=m +no_defs"

ext_sinusoidal <- spTransform(x = polygon,
                              crs(x = proj_sinu)
                              )
```

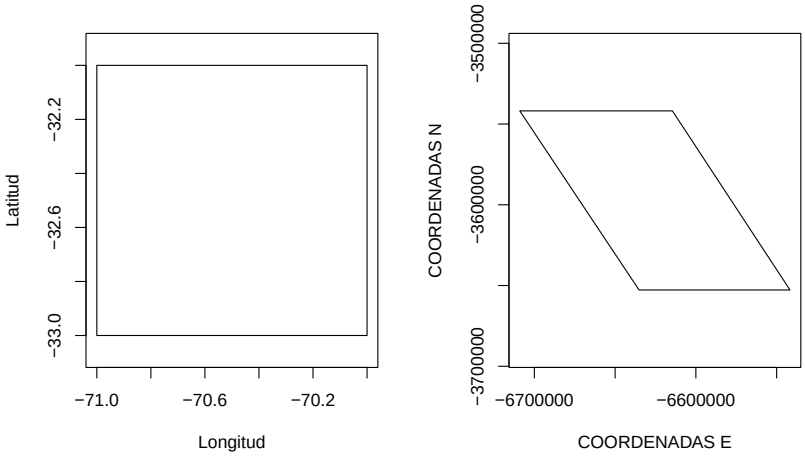


Figura 3.11: Sistema coordenado de la extensión de la imagen base. Sistema Coordenado Geográfico (izquierda). Extensión de la imagen proyectada al sistema Sinusoidal (derecha).

Área de Estudio

Al comenzar a trabajar en los conceptos básicos del procesamiento digital del terreno debemos definir el área de estudio y vamos a desarrollar tres métodos para definir esa área.

3.11 Vector de Coordenadas

El método más simple y directo es crear el área de estudio mediante una lista de coordenadas y comandos del paquetes *Simple Features* o *sf* (Pebesma [2018]) diseñado para trabajar con objetos geográficos en formato vectorial codificados de manera estándar (ver en Wikipedia Simple Features).

1. Instalar si fuese necesario y cargar en memoria:

```
library(sf)
```

2. Crear dos vectores con las coordenadas del polígono:

```
lng <- c(-71.37663, -71.31071, -71.17888,
         -71.19535, -71.32719, -71.37663)
lat <- c(-34.77340, -34.68528, -34.73500,
         -34.81855, -34.84337, -34.77340)
```

3. Convertir objeto *Vector* a *lista* combinando ambos vectores:

```
lista <- list(cbind(lng, lat))
```

4. Convertir el objeto *lista* en un objeto *sf* estándar:

```
pol <- st_polygon(lista)
```

5. Incorporar información de proyección y convertir al objeto final (Figura 3.12):

```
poligono <- st_sf(st_sfc(pol), crs= 4326)
```

El parámetro *crs= 4326* configura el *sistema coordenado de referencia* al sistema mundial para dispositivos GPS, los datos se expresan en latitud y longitud.

Para conocer más detalles, alternativas y usos de los sistemas coordenados, revisar el punto 3.9.2.

Y el objeto final *poligono* será utilizado posteriormente para la obtención de la información topográfica (Ver 4.3.1).

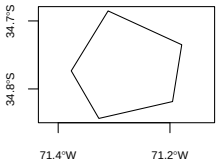


Figura 3.12: Polígono por Coordenadas

3.12 Definición Interactiva

Un segundo método, requiere una *conexión a internet* al momento de utilizar una interfaz gráfica interáctiva para definir el área sobre un mapa web utilizando diversas herramientas de dibujo.

Primero debemos cargar los paquetes *mapview* y *mapedit* (Appelhans et al. [2020a], Appelhans et al. [2020b]) que son los encargados de la creación del entorno gráfico para el despliegue del mapa base, herramientas de dibujo, coordenadas entre otros.

Por primera y única vez instalar los paquetes utilizando la sintaxis apropiada:

```
install.packages("mapview")
install.packages("mapedit")
```

1. Leer en memoria los paquetes:

```
library(mapview)
library(mapedit)
```

2. Algunos parámetros de configuración importantes de la herramientas es definir las capas de cartografía base las cuales sirven de referencia para el trazado de elementos (vector de textos en parámetro `basemaps`), desde un gran número de opciones disponibles en sitio web *leaflet providers preview*[5]. En el ejemplo siguiente se han seleccionado tres tipos bases cartográficas, dos de ESRI y uno del proyecto OSM.
   Además se configura el color y transparencia de los elementos trazados (`vector.palette`).
   Por último, un parámetro para ubicar en la ventana el cuadro de herramientas con parámetro: `layers.control.pos`.

[5] `https://leaflet-extras.github.io/leaflet-providers/preview/`

```
mapviewOptions(basemaps = c('Esri.WorldImagery',
                            "Esri.WorldShadedRelief",
                            "OpenStreetMap.DE"
                          ),
               vector.palette = rgb(50,220,230,
                                    alpha = .90,
                                    maxColorValue = 255
                                  ),
               layers.control.pos = "topright")
```

Finalmente se define el nombre del objeto que recibirá el producto de la sesión interáctiva realizada en **la pestaña Viewer** de RStudio (Figura 3.13):

```
area_vista <- editMap()
```

Para terminar la sesión y recuparar los elementos trazados se debe clicar en el botón **Done** en la esquina inferior derecha.

Y el objeto final *area_vista* será utilizado posteriormente para la obtención de la información topográfica (Ver punto 4.3.2).

3.13 Conexión Remota a Unidades Políticas

En 2009 es creada por Robert J. Hijmans una base de datos global de límites administrativos con el objeto de contar con todos los países en alta resolución https://gadm.org/ y hoy mediante el

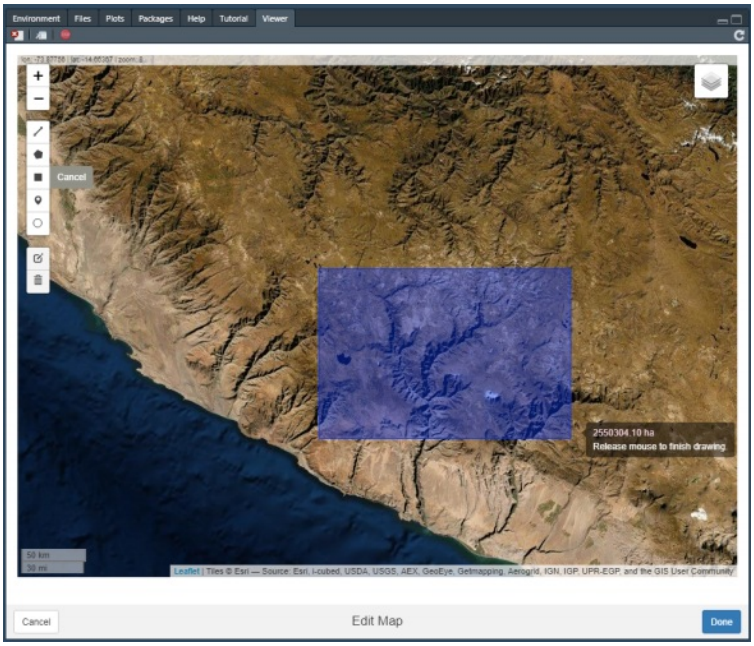


Figura 3.13: Captura de pantalla del funcionamiento de la sesión interáctiva Mapview, al mover el ratón actualiza la geometría y los datos asociados.

paquete *GADMTools* Decorps [2021] esta disponible para usuarios de R y obtener directamente el objeto poligonal para crear mapas, o en nuestro caso definir el área de estudio.

1. Instalar (si no se ha realizado previamente) y cargar el paquete en la sesión actual:

```
library(GADMTools)
```

2. Definir los parámetros de configuración antes de realizar la descarga:

- 1. Código ISO:

La Organización Internacional de Estandarización (ISO) representa países, territorios dependientes y áreas de interés especial bajo el código **ISO_3166-1_alpha-3** que se compone de tres caracteres alfabéticos[6] que intentan resumir o representar el nombre original. Por ejemplo: *Argentina ARG*, *Bangladesh BGD*, *Chile CHL*, *Francia FRA*. El parámetro *fileNames* corresponde a ese código.

[6] Listado completo de países y código en: en.wikipedia.org/wiki/ISO_3166-1_alpha-3

Una función de ayuda para buscar los códigos, nombres y estándares se utiliza la función `ccodes()` que devuelve un detallado objeto *dataframe*.

```
ccodes()
```

- 2. Nivel de Detalle:

La información se organiza por niveles o jerarquía política y de acuerdo con el país reciben distintos nombres (cuadro 3.1). El parámetro *level* debe incluir el nivel que corresponda al nivel de detalle de los objetos a descargar y hará variar el tamaño del archivo resultante (Figura 3.14).

Cuadro 3.1: Nombre de Unidades por Nivel de Detalle.

Nivel	Perú	Chile	Francia
1	Región	Región	Région
2	Provincia	Provincia	Département
3	Distrito	Comuna	Arrondissement
4			Canton
5			Commune

- 3. Definir Carpeta de Salida:

La sesión actual se encuentra ubicada en la carpeta raíz del proyecto y si no definimos el nombre de una carpeta en el parámetro *basefile* el archivo será descargado allí, se recomienda utilizar siempre una carpeta especial dentro del proyecto dedicada exclusivamente a información base o *cruda*.

Si previamente no has creado la carpeta debes utilizar el comando: `dir.create(nombre)` que crea la carpeta *nombre* dentro de la carpeta del proyecto, si existe informa con un mensaje y termina sin realizar cambios. Descargando directamente el archivo con los datos.

El texto que especifica la ruta debe separar usamos la función `here()`.

3. Realizar la descarga:

```
peru_nivel_1 <- gadm_sp_loadCountries("PER",
                                      level = 1,
                                      basefile = here("RAW"))

peru_nivel_2 <- gadm_sp_loadCountries("PER",
                                      level = 2,
                                      basefile = here("RAW"))

peru_nivel_3 <- gadm_sp_loadCountries("PER",
                                      level = 3,
                                      basefile = here("RAW"))
```

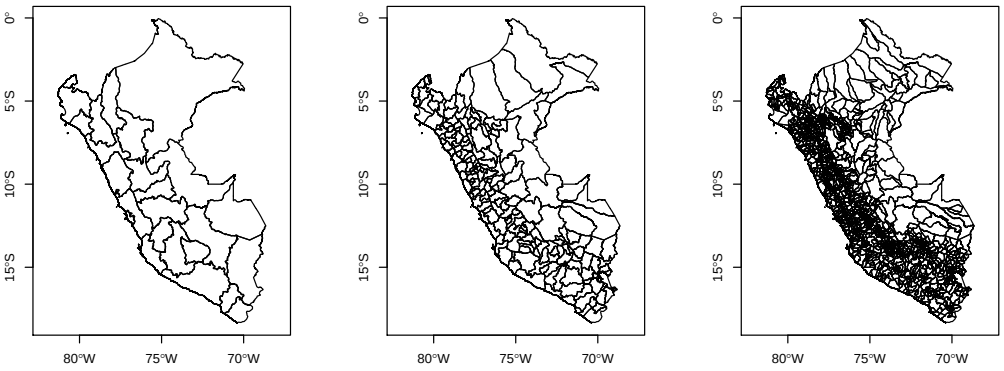


Figura 3.14: División Político Administrativa. Perú Niveles 1, 2 y 3 (izquierda a derecha)

Si un nivel no existe, el sistema informa mediante un mensaje de alarma, intenta descargar un nivel 4 de Perú.

```
peru_nivel_4 <- gadm_sp_loadCountries("PER",
                                      level = 4,
                                      basefile = here("RAW"))
```

4. Procesos post descarga:

- Para generar **un mapa básico** utilizar el comando: `gadm_plot()` (Figura 3.15).

- Revisar el **contenido de los nombres** de las unidades políticas: `listNames()` del nivel correspondiente (los comando *sort* corresponde a una función del paquete R base que ordena ascendente o descendentemente el contenido de un vector, *head* un comando del paquete utils que retorma el inicio de un vector y *tail* retorna registros al final del vector):

```
head(sort(listNames(peru_nivel_1,level = 1)))
```

```
[1] "Amazonas"   "Ancash"     "Apurímac"   "Arequipa"   "Ayacucho"   "Cajamarca"
```

- **Selección y filtraje** de unidades por su nombre: `gadm_subset()`.

```
puno_piura <- gadm_subset(peru_nivel_1,
                          level = 1,
                          regions = c("Puno","Piura"))
```

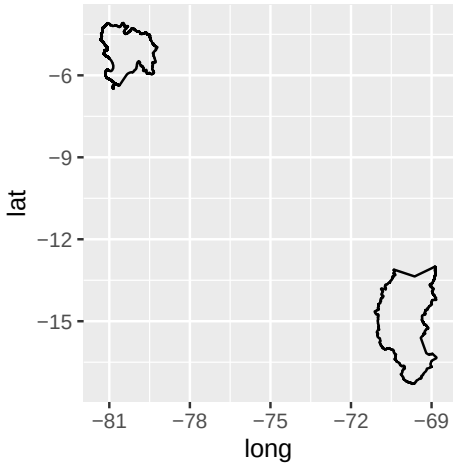


Figura 3.15: Mapa generado con la función gadm_plot()

Y el objeto final *puno_piura*® será utilizado posteriormente para la obtención de la información topográfica (Ver 4.3.3).

Tal como se describe en 2.4 almacenamos la data en disco usando nuestra carpeta de datos *raw* y tal como se recomienda, usamos el mismo nombre del objeto original:

```
saveRDS(puno_piura, here("raw", "puno_piura.rds"))
```

Y para el proceso inverso, cargar en memoria desde disco:

```
puno_piura <- readRDS(here("raw", "puno_piura.rds"))
```

IV GEOMORFOMETRÍA

Introducción a la Geomorfometría

4.1 Definición de Conceptos

***Geomorfometría* es la ciencia del análisis *cuantitativo* del relieve**. Esta es una aproximación cartográfica y computacional al manejo de los valores de altitud de la superficie terrestre (Tobler [1976]) que hereda sus grandes principios de las matemáticas, ciencia de la tierra y más recientemente de la ciencia de la informática. Es importante destacar que el término *geomorfometría* se ha acuñado para diferenciarlo de la *morfometría* usada en otras ciencias tales como la biología o medicina, y podemos encontrar sus primeras referencias a mediados de la década del 40.

Cabe señalar que podemos encontrar dos corrientes en el análisis geomorfométrico: *específico*, que analiza las formas del relieve de una manera discreta (landforms) y *general*, tratando la superficie de una manera continua.

Donde nuestros pies se apoyan es universalmente comprendido por ser la interfaz entre el suelo o la roca desnuda y la atmósfera. Pero encontrar un término aceptado transversalmente para nombrar dicha capa y la ciencia que se encarga de su medición geométrica ha sido más difícil, la **representación numérica** de la superficie terrestre es descrita por términos tales como *terrain modelling*, *terrain analysis* o *science of topography*. Mientras que la **descripción cuantitativa**, o **medidas de la forma de la superficie terrestre** se presenta en la bibliografía con términos tales como *topographic attributes* o *properties*, *land-form parameters*, *morphometric variables*, *terrain information*, *terrain at-*

tributes, *geomorphometric parameters*, *land-surface parameters* o *geomorphometric attributes*. Siendo este último el usado en el presente texto: ***atributo geomorfométrico***, por sus implicaciones cualitativas pero mensurables inherentes en la definición del término atributo.

A pesar del amplio uso del término *terreno*[1] resulta impreciso. El término terreno significa distintas cosas según el especialista, desde la superficie terrestre, suelo, vegetación, incluso un punto de vista socioeconómico de un área y no tiene precisamente implicaciones de la forma. Entonces el término tan usado como *terrain analysis* es confuso ya que implica la larga tradición del trabajo cualitativo o la interpretación de los procesos de interpretación de imágenes. Entonces consideramos el uso del término **relieve** (relief) como sinónimo de la morfología de la superficie terrestre con especial acento en la variación de la elevación y podemos recomendar que dicho parámetro se ajusta mucho mejor al concepto de procesamiento entonces, un poco más preciso es el término **digital relief modelling** que involucra los conceptos de *procesamiento informático* y el acceso a los *valores de cota* o a *su procesamiento*.

[1] **Terreno** viene del latín *terrenum*, que puede ser traducido como *de la tierra*.

Así, el término **Modelo Digital de Relieve, MDR** (digital relief model) se convertirá en el sujeto de estudio mediante el uso de **atributos geomorfométricos**, mientras que la **Geomorfometría** será la ciencia encargada de su estudio.

4.2 Historia

La geomorfometría inicia con la medida sistemática de la elevación sobre el nivel del mar de puntos del terreno, sabemos con seguridad que en el antiguo Egipto se realizaron los primeros trabajos, como también el cálculo de alturas por la proyección de sombras descrito por el filósofo griego Tales de Mileto (625-546 A.C), pero el concepto actual de contorno de elevación usado para describir superficies continuas del terreno data de 1584 con el trabajo pionero del topógrafo holandés *Pieter Bruinz* que dibuja líneas de igual profundidad (en un primitivo intento de curva de nivel) en el Río Spaarne (Figura 4.1).

Figura 4.1: Trabajo de Pieter Bruinz.

Con su aporte, fueron en los siglos siguientes muchos los trabajos realizados con la misma técnica, pudiendo destacar por su envergadura y ambición el trabajo de *Dupain-Triel* en 1791, que consistió en la publicación de un primitivo mapa de curvas de nivel de Francia (Figura 4.2).

Figura 4.2: Mapa de curvas de nivel de Francia, Dupain-Triel 1791.

En 1774 el matemático británico *Charles Hutton* (Figura 4.3) fue asignado para resumir los trabajos realizados por Charles Manson, astrónomo que buscaba estimar la masa de la Tierra. Hutton uso un lápiz para conectar los puntos de igual altitud en las montañas escocesas de Schiehallion, desarrollando el concepto de **isohipsa**.

Este método a probado ser una muy efectiva forma de repre-

Figura 4.3: Charles Hutton (1737–1823).

sentar la topografía y es una de las más importantes innovaciones en la historia de la cartografía, por su conveniencia, exactitud, y fácil comprensión.

Afines del siglo XVIII e inicio del XIX, en Europa surge un explosivo auge de trabajos de levantamientos topográficos de mayor precisión, los estudios de esas elevaciones y su comportamiento fueron estudiados cualitativamente, principalmente por ingenieros militares, fundando la **orografía**[2], mientras que las cotas y los parámetros derivados de ellas dan paso a la **orometría**[3].

[2] **Orografía** viene del latín *montañas* y *para dibujar*.

[3] **Orometría** viene del latín *montañas* y *para medir*.

Desde Alemania y Austria, la cuna de la geomorfometría, surgen dos hitos principales. El breve (19 páginas) capítulo de orometría en el manual de Von Sonklar de 1873 que presenta doce medidas cuantitativas de la morfología de las montañas, el cual estimula un sinnúmero de publicaciones sobre la caracterización de la superficie terrestre. Y uno de los mejores compendios de la temprana ciencia de la geomorfometría (¡que incluye críticas al texto de Sonklar!), fue el capítulo más grande y completo en el texto de 1894 de Albrecht Penck (Figura 4.4).

Figura 4.4: Manual de Von Sonkla (1873) y Texto de Penck 1894 (Fotografía cortesía de R.Pike)

Ya a mediados del siglo XIX los mapas de curvas de nivel han alcanzado un mayor desarrollo y disponibilidad, el análisis del relieve florece. Medidas del mayor y menor punto en un área de estudio (generalmente un cuadrado o círculo) cuantifican la dimensión vertical del relieve, expresando la altura relativa. En 1911 se desarrolla el primer mapa cuantitativo de relieve local (o relativo), usando los rangos de elevación de una grilla de 5

x 5 km cuadrados. Geógrafos y luego geomorfólogos miden las áreas encerradas por curvas de nivel para generar gráficos de elevación versus área. Esta curva hipsométrica puede ser acumulada e integrada para estudios comparativos de diversas regiones.

Con el auge de la computación digital a mediados de la década de los cincuenta la geomorfometría tuvo un acelerado progreso, mientras que el arreglo de grillas de elevaciones era preparadas por los geofísicos para la corrección por gravedad, los ingenieros civiles proyectaban autopistas y el ejercito investigaba tácticas de combate, el concepto de modelo digital fue descrito abiertamente en el MIT (en 1958) pero no tuvo un uso generalizado hasta finales de los 60.

Su potencial e importancia se vio restringido por las limitaciones de los computadores digitales de ese período. Algunos modelos digitales fueron preparados fotogramétricamente o levantados en terreno, incluso laboriosamente interpolados a mano de mapas de contornos. La digitalización semi automática de toda la superficie de EE.UU. a una resolución de 63 m a partir de los mapas de contornos escala 1:250.000 en el período 1963-1972 distribuido finalmente por el USGS marcó la explosión de disponibilidad de dichos productos (U.S. Army Map Service, 1963).

Los primeros mapas y diagramas de los resultados geomorfométricos fueron limitados a pantallas de baja resolución (tubos de rayos catódicos) y solo 128 caracteres alfanuméricos impresos por línea en papel de 38 cm de ancho (Figura 4.5) utilizando sobreimpresión de caracteres para lograr mayor número de tonos de gris.

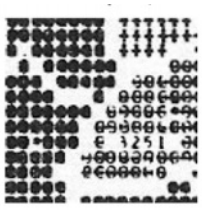


Figura 4.5: Impresión de un Modelo Digital.

El procesamiento digital de la información ha sido implementado por diversos programas computacionales de una manera secuencial, donde cada celda es calculada utilizando alguna fórmula y a continuación la celda siguiente, y así hasta completar la zona de interés.

Hoy con la capacidad de acceder a los núcleos de procesamiento gráfico de las tarjetas de video, el poder de cómputo de un número masivo de procesadores en paralelo aumenta las posibilidades de acelerar los algoritmos de cálculo.

Pasos Preliminares

En este capítulo revisaremos los pasos iniciales para preparar nuestra sesión en R para comenzar posteriormente con los análisis y usaremos como área de estudio los definidos en el punto 3.10.3.

El paquete *elevatr* (Hollister et al. [2020]) provee acceso a servicios web de información topográfica y será utilizado para descarga.

Como primer paso debemos instalar el paquete que recomendamos realizarlo siempre en la **consola** y luego cargar en memoria el paquete requerido con *library(elevatr)* en el script.

4.3 Descargar Información Topográfica

La función principal para descargar la información es:

```
area_mdr <- get_elev_raster(locations, z, src)
```

Los parámetros de configuración son los siguientes:

locations: Un polígono que defina el área de estudio. **z**: Valor del cual depende el tamaño de la celda y se estima a partir de la ecuación 4.1.

$$celda = cos(\phi * \pi/180) * 2 * \pi * 6378137m)/(256 * 2^{ZOOM}) \quad (4.1)$$

Donde ϕ corresponde a la latitud geográfica promedio la región a descargar, $ZOOM$ el nivel de detalle de la información, que es un valor que va de *1* a *14*. Un ejemplo para latitudes media y alta la podemos revisar en el cuadro 4.1.

src: La fuente de datos vía servicios web, hoy (junio 2021) conecta con:

- opentopography.org
- Amazon Web Services Terrain Tiles

- USGS Elevation Point Query Service

Cuadro 4.1: Tamaño de Celda(m) por Nivel de Zoom y Latitud

Zoom	0°	-45°	-60°
1	78271	55346	39135
2	39135	27673	19567
3	19567	13836	9783
4	9783	6918	4891
5	4891	3459	2445
6	2445	1729	1222
7	1222	864	611
8	611	432	305
9	305	216	152
10	152	108	76
11	76	54	38
12	38	27	19
13	19	13	9
14	9	6	4

4.3.1 Por Lista de Coordenadas

En el punto 3.11 se construye el objeto *poligono* mediante una lista de coordenadas y descargamos el MDR desde Amazon Web Services y un zoom de 11 ($\sim 60\ m$).

```
area_dtm <- get_elev_raster(locations = poligono,
                            z= 11,
                            src= "aws")
```

Y obtenemos un objeto *RasterLayer*:

```
class      : RasterLayer
dimensions : 1379, 2240, 3088960  (nrow, ncol, ncell)
resolution : 0.0003138821, 0.0003138821  (x, y)
extent     : -71.54297, -70.83987, -35.02988, -34.59704  (xmin, xmax, ymin, ymax)
crs        : +proj=longlat +datum=WGS84 +no_defs
source     : file1aac69927956.tif
names      : file1aac69927956
values     : -32768, 32767  (min, max)
```

```
plot(area_dtm)
plot(poligono, add=T)
```

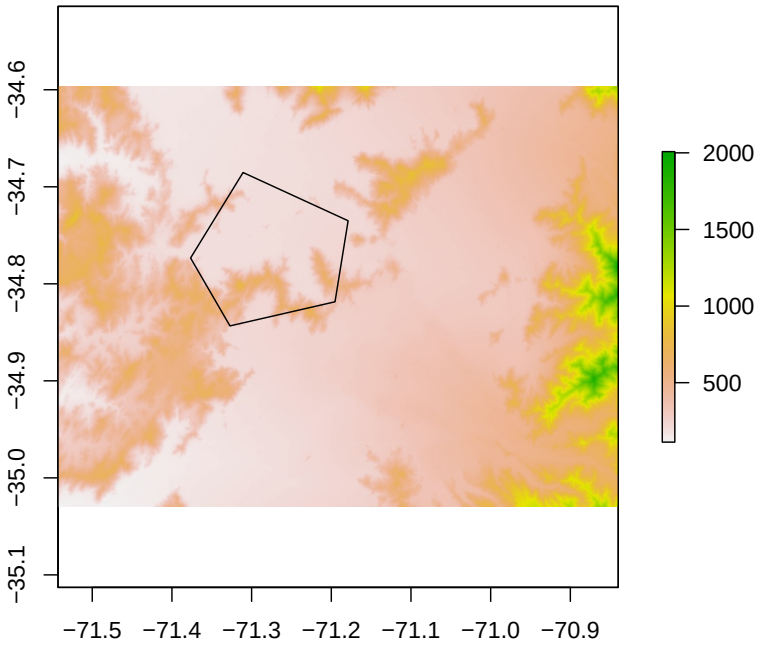


Figura 4.6: MDR y objeto poligono creado por coordenadas.

Nótese el parámetro `add=T` de la función `plot` agrega contenido al gráfico creado en el paso anterior (Figura 4.6).

Que se encuentra almacenado en una carpeta temporal donde fue descargado con un nombre generado automáticamente:

```
filename(area_dtm)
```

```
[1] "C:\\Users\\danilo.verdugo\\AppData\\Local\\Temp\\RtmpOEJ3A8\\file1aac69927956.tif"
```

Conviene por seguridad almacenar en disco con nombre y ruta conocido®:

```
writeRaster(area_dtm, here("raw", "area_dtm.grd"))
```

Para leer desde disco y construir un nuevo objeto®:

```
area_dtm <- raster(here("raw", "area_dtm.grd"))
```

4.3.2 Definición Interactiva

En el punto 3.12 se construye el objeto *area_vista* mediante una sesión *mapview* y descargamos el MDR desde Amazon Web Services y un zoom de 12 ($\sim 36\ m$).

```
dtm <- get_elev_raster(area, z= 12, src="aws")
```

Y obtenemos el correspondiente RasterLayer:

```
dtm
```

```
class      : RasterLayer
dimensions : 8055, 12004, 96692220  (nrow, ncol, ncell)
resolution : 0.0001684019, 0.0001684019  (x, y)
extent     : -73.91602, -71.89452, -15.96132, -14.60485  (xmin, xmax, ymin, ymax)
crs        : +proj=longlat +datum=WGS84 +no_defs
source     : dtm.grd
names      : file1a5f86daa1349
values     : 425, 6405  (min, max)
```

```
plot(dtm)
plot(area_vista, add=TRUE)
```

4.3.3 División Política

En el punto 3.13 se construye el objeto *puno_piura*® con data proveniente del sitio gadm.org y descargamos el MDR desde Amazon Web Services y un zoom de 9 (~300m).

El objeto descargado y procesado por el paquete GADMTools (Decorps [2021]) pertenecen a un "tipo" particular, especialmente diseñado para su uso:

```
class(puno_piura)
```

```
[1] "gadm_sp"
```

Debemos extraer la información espacial en un formato estándar y compatible para los otros paquetes y se realiza utilizando el símbolo **$** que permite explorar identificadores con *nombre* y almacenarlo en un nuevo objeto ahora compatible.

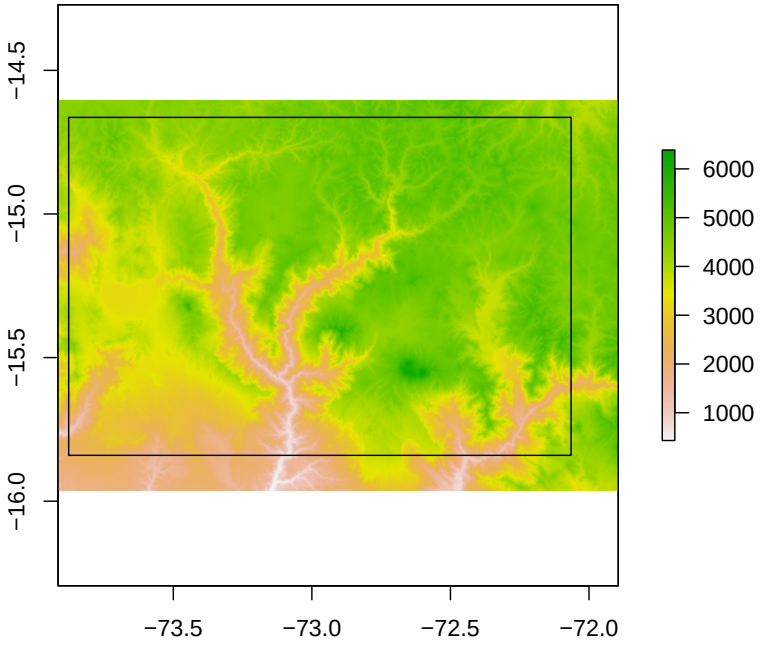


Figura 4.7: MDR y polígono generado en sesión mapview.

```
area_puno_piura <- puno_piura$spdf

elev <- get_elev_raster(area_puno_piura, z= 9, src='aws')
```

Y obtenemos el correspondiente RasterLayer (Figura 4.8):

```
elev

class      : RasterLayer
dimensions : 10784, 10484, 113059456  (nrow, ncol, ncell)
resolution : 0.001341376, 0.001341376  (x, y)
extent     : -81.5625, -67.49952, -17.97882, -3.513421  (xmin, xmax, ymin, ymax)
crs        : +proj=longlat +datum=WGS84 +no_defs
source     : puno.grd
names      : file2072835f83c54
values     : -7409, 8657  (min, max)

plot(elev)
plot(area_puno_piura, add=TRUE)
```

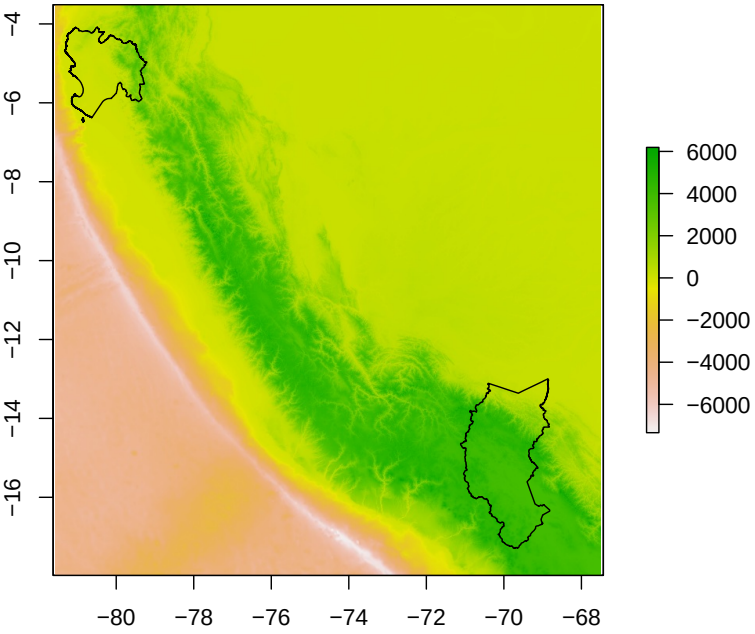


Figura 4.8: MDR y polígonos GDAM.

4.4 Preparación del MDR

El MDR del área de trabajo que ha sido obtenido directo desde el servicio web tiene algunas características que deben ser corregidos con el objeto de optimizar y hacer más compatible los productos a realizar.

4.4.1 Recorte del MDR

No siempre el área de estudio corresponde exactamente a un MDR ya que se compara con los productos almacenados en los servidores y se obtienen aquellos que cubren el área solicitada por lo tanto puede ocurrir que el área de estudio sea más pequeña que el área cubierta por el MDR.

Para ello vamos a realizar un ***recorte*** (*crop*) obteniendo así un RasterLayer que coincide con la *extensión del polígono*, pasado a la función (Figura 4.9, superior).

```
dtm_crop <- crop(area_dtm, poligono)
```

También se puede realizar un recorte además de la extensión

del área de estudio usar la *forma exacta* del área de estudio[4] (Figura 4.9, inferior).

```
dtm_mask <- mask(dtm_crop, poligono)
```

[4] El comando *mask* (o cualquier otro) también podría existir en otros paquetes (y el nombre podría chocar con alguno de los cargados en la sesión actual) se puede especificar de qué paquete desea ejecutar la función con la sintaxis ***paquete::función***.

4.4.2 Cambio de Proyección

```
dtm_mask
```

```
class      : RasterLayer
dimensions : 504, 630, 317520  (nrow, ncol, ncell)
resolution : 0.0003138821, 0.0003138821  (x, y)
extent     : -71.37661, -71.17887, -34.84344, -34.68524  (xmin, xmax, ymin, ymax)
crs        : +proj=longlat +datum=WGS84 +no_defs
source     : memory
names      : file1aac69927956
values     : 184, 828  (min, max)
```

El MDR está definido WGS84 un sistema coordenado universal (ver punto 3.9.2) basado en *latitud* y *longitud* donde las coordenadas son expresadas en grados, minutos, segundos y cualquier información generada se expresa usando esas mismas unidades.

Por ejemplo, la información relativa al tamaño de la celda *(resolution : 0.0003138821, 0.0003138821 (x, y))* son difíciles de apreciar o entender, vamos a convertir a un sistema de referencia donde las unidades de medida sean metros (m) y el proceso lo llamaremos **Proyección** (Figura 4.10).

La primera tarea es seleccionar el sistema coordenado de referencia de destino (revisar detalles en 3.9.2) y buscar su código EPSG para realizar la proyección:

```
utm19  <- crs("+init=epsg:32719")
dtm_proj <- projectRaster(dtm_mask, crs = utm19)
```

```
class      : RasterLayer
dimensions : 526, 657, 345582  (nrow, ncol, ncell)
resolution : 28.7, 34.8  (x, y)
extent     : 282108.7, 300964.6, 6141570, 6159875  (xmin, xmax, ymin, ymax)
crs        : +proj=utm +zone=19 +south +datum=WGS84 +units=m +no_defs
source     : memory
names      : file1aac69927956
values     : 184.0229, 827.2889  (min, max)
```

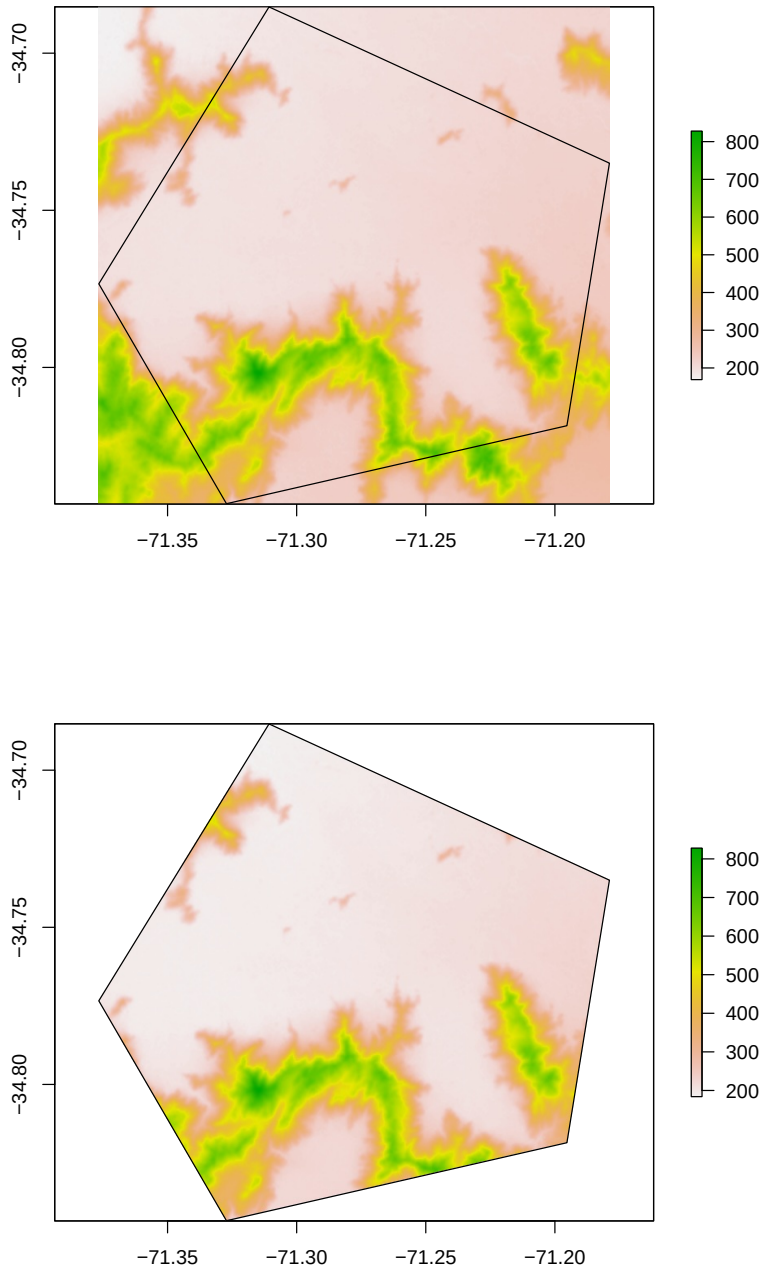


Figura 4.9: MDR recortado por extensión del área de estudio (arriba). MDR enmascarado por el área de estudio (abajo).

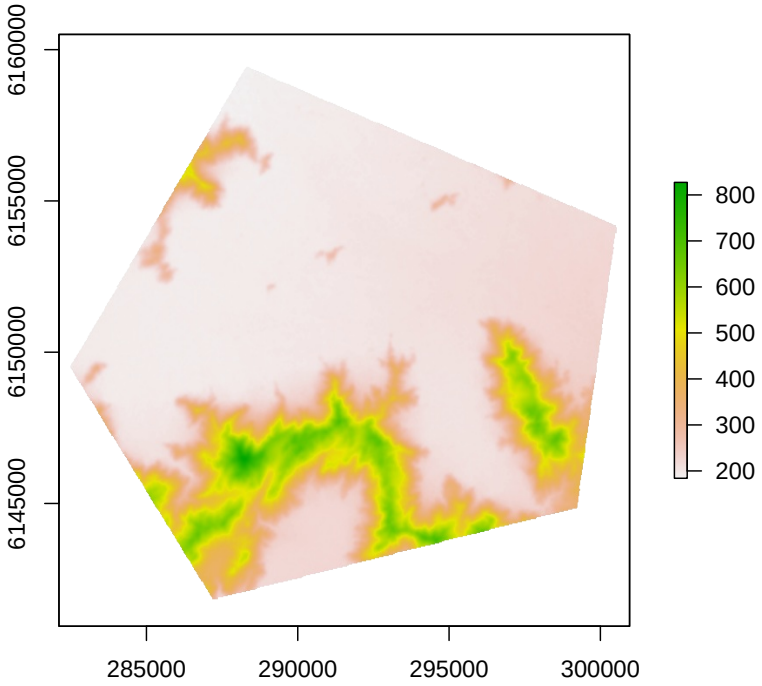


Figura 4.10: MDR Proyectado a UTM H19S.

Al revisar el MDR resultante vemos que ahora está en un nuevo sistema coordenado y los valores de resolución son $x = 28.7m$ e $y = 34.8m$ con las coordenadas de extensión expresados en m.

4.4.3 Cambio de Resolución

El próximo paso es ajustar el tamaño de la celda a valores enteros (e idealmente iguales en ambos ejes):

```
dtm_proj_ext <- projectExtent(dtm_proj, crs = utm19)
res(dtm_proj_ext) <- 30

dtm_proj_30m <- projectRaster(from = dtm_proj,
                              to= dtm_proj_ext,
                              method = 'bilinear')
```

Algunos pequeños ajustes que mejoran nuestro objeto:

```
names(dtm_proj_30m) <- 'DTM'
print(dtm_proj_30m)
```

```
class       : RasterLayer
dimensions  : 610, 629, 383690  (nrow, ncol, ncell)
resolution  : 30, 30  (x, y)
extent      : 282108.7, 300978.7, 6141575, 6159875  (xmin, xmax, ymin, ymax)
crs         : +proj=utm +zone=19 +south +datum=WGS84 +units=m +no_defs
source      : memory
names       : DTM
values      : 184.5201, 825.2792  (min, max)
```

Ahora es el turno de transformar el objeto *poligono* que define el área de estudio que sigue en coordenadas geográficas. Nuestro polígono se construyó a partir de un listado de coordenadas (ver 3.11) construido mediante el paquete *sf* que también posee una función para transformar su CRS.

```
poligono_geom <- sf::st_transform(x = poligono, crs = utm19)
```

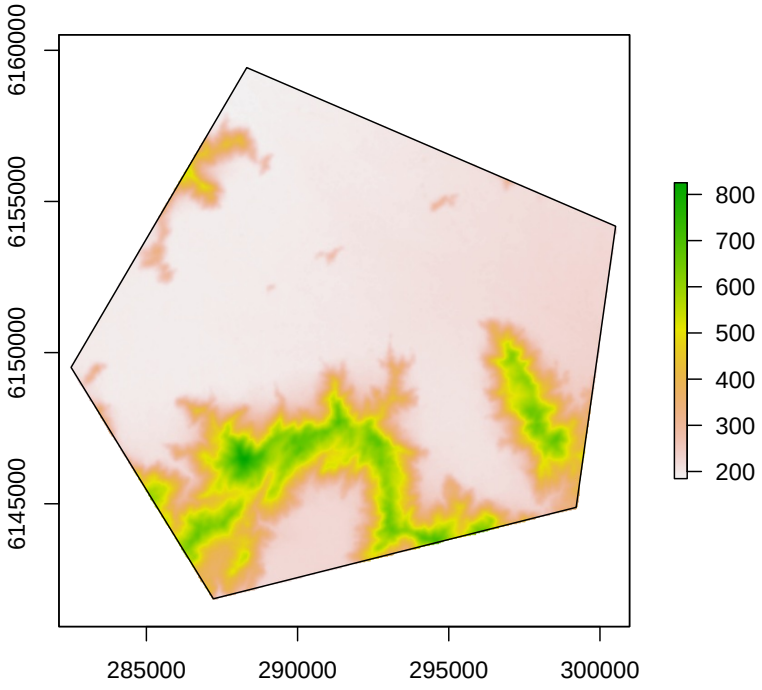


Figura 4.11: Objeto poligono transformado a CRS del MDR.

Ahora, mismo procedimiento para el caso de dibujo del área de estudio mediante una sesión interactiva, ver punto 3.12 (Figura 4.12):

```
dtm_ext <- projectExtent(dtm, utm19)
res(dtm_ext) <- 50
dtm_20m <- projectRaster(from = dtm,
                         to= dtm_ext,
                         method = 'bilinear')
pol_proj <- sf::st_transform(x = pol, utm19)
```

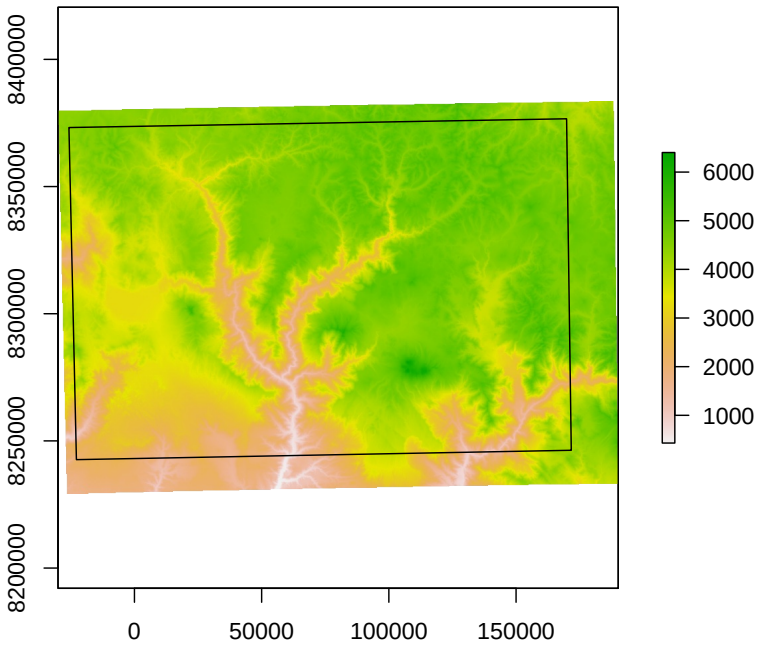


Figura 4.12: MDR y área de estudio, generado con Sesión Interáctiva.

Y finalmente, el caso de la división política descargada del servicio GADM, ver punto 3.13 (Figura 4.13):

```
elev_proj_ext <- projectExtent(elev, utm19)
res(elev_proj_ext) <- 1500
elev_proj_150m <- projectRaster(from = elev,
                                to= elev_proj_ext,
                                method = 'bilinear')
area_puno_piura_proj <- spTransform(x = area_puno_piura, utm19)
```

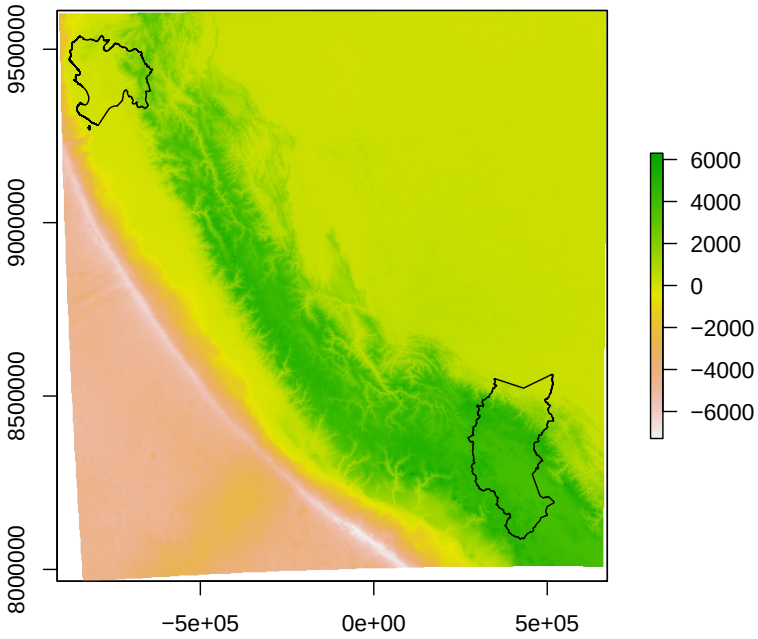


Figura 4.13: MDR y área de estudio, generado con Descarga de División Política.

4.5 Funciones Adicionales Importantes

Una vez trazado un mapa con el comando `plot()` se despliega en la pestaña **Plots** y R posee una serie de comandos que permiten interactuar con dicho dibujo o *ploteo*. Vamos a repasarlos en detalles ya que facilitan mucho el trabajo siempre dentro de la misma plataforma R.

4.5.1 Extraer Información con el Cursor

Mediante la función `click()` permite extraer sobre un mapa valores de la ubicación del click y opcionalmente sus coordenadas o número de celda.

En el siguiente ejemplo se inicia con el trazado del mapa sobre cual extraer la información, luego realizando tres click del ratón, extraer tres *(n=3)* muestras junto a sus coordenadas *(xy=T)* y el número de celdas *(cell=T)*[5].

```
plot(elev_ext)
cl <- click(elev_ext, n=3, xy=T, cell=T)
```

[5] Función con interacción del usuario se finalizan, 1) Presionado tecla **ESC**, 2) Presionando botón **Finish** (barra de título de la pestaña Plots).

```
cl
```

```
       x       y cell     value
1 292662 8565406 1597  493.4250
2 285162 8548906 2329  929.0102
3 232662 8548906 2294 1899.7384
```

La función retorna un objeto de clase *dataframe* y por comodidad podemos convertir a matriz con el comando:

```
m <- as.matrix(cl)
```

Y así podemos utilizar los mecanismos estudiados en punto 1.21.

```
m[,'value'] #valores de celdas
```

```
[1]  493.4250  929.0102 1899.7384
```

```
m[1,] #Primer registro
```

```
          x           y        cell       value
 292662.010 8565405.751    1597.000     493.425
```

```
m[2:3, 3] #números de celda
```

```
[1] 2329 2294
```

4.5.2 Dibujar Elementos Gráficos en Mapa

Usando el mismo mecanismo podemos realizar dibujos de tres tipos básicos:

- **Polígono**: Por defecto retorna un objeto *SpatialPolygons* del paquete **sp**.
- **Línea**: Por defecto retorna un objeto *SpatialLines* del paquete **sp**.
- **Extensión**: Por defecto retorna un objeto *Extent* del paquete **raster**.

```
plot(elev_ext)
objeto <- drawPoly()
objeto <- drawLine()
objeto <- drawExtent()
```

4.5.3 Recortar Mapa Creado por Coordenadas

Si cuento con las coordenadas que definen una ventana rectangular de corte puedo usar el comando `extent()` para crear un objeto *Extent*. El comando también acepta varias formas de definir sus coordenadas: extraer límites de un objeto ráster, puede crear a partir de una matriz de 2x2 o un vector de longitud 4 con la estructura *xmin, xmax, ymin, ymax* (Figura 4.14).
El objeto Extent se usa para extraer un subconjunto de celdas y usando el parámetro *drop= FALSE* devuelve un nuevo RasterLayer.

```
ext <- extent(c(210000,310000,8500000,8600000))
mdt_ext <- elev_proj_150m[ext, drop= FALSE]
plot(mdt_ext)
```

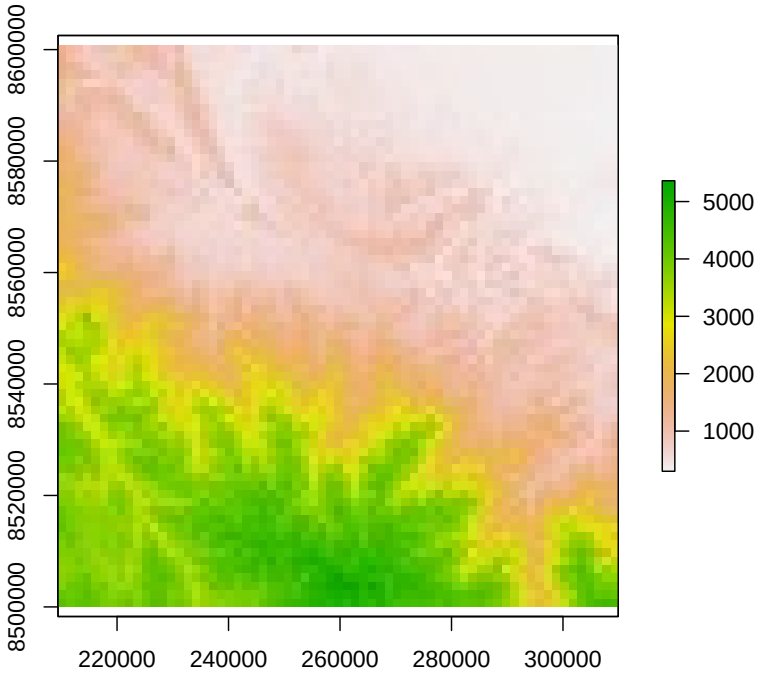


Figura 4.14: Recorte de MDR vía coordenadas.

4.5.4 Recortar Geométricamente un objeto Ráster

Recortar directamente el objeto trazado en pantalla se realiza con el comando `select()` que retorna directamente un objeto RasterLayer (Figura 4.15).

```
plot(elev_ext)
elev1 <- select(elev_ext, use="pol") # también use='rec'
plot(elev1)
```

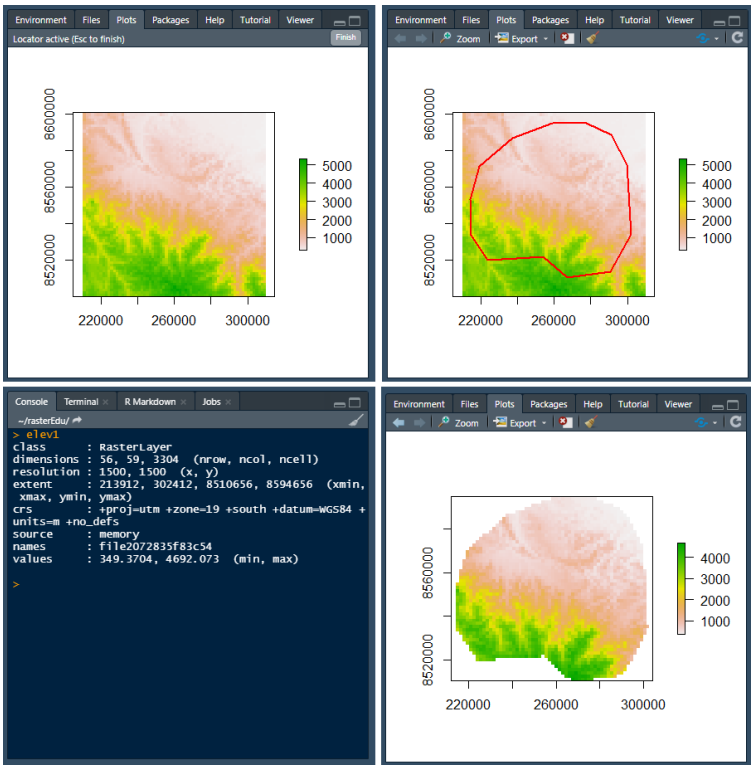


Figura 4.15: Recorte geométrico de objeto ráster. (izq.arrriba) Clicar en ventana y para terminar presione 'Finish' o 'ESC' (der.arriba) Presentación del polígono dibujado (izq.abajo) Objeto RasterLayer que retorna (der.abajo) Plot del RasterLayer retornado.

4.5.5 Realizar Zoom

Sobre el mapa definir un área con dos clics y se levanta una ventana con el zoom correspondiente, allí puede exportar a diversos formatos o pasar contenido a una variable entre otras op-

ciones.

```
plot(elev_ext)
zoom(elev_ext)
```

4.6 Manejo de Archivos Ráster

4.6.1 Grabar

El modelo ráster construido se debe guardar en disco, para efectos de respaldo, ahorro de tiempo de proceso, evitar recrear cada vez, etc.

La sintaxis para guardar archivo es: `writeRaster(modelo, nombre_archivo, overwrite, format, bylayer, suffix)`

Donde los parámetros son:

- **`modelo`**: El objeto ráster a guardar.
- **`nombre_archivo`**: Nombre y ruta del archivo de salida. Se recomienda el uso del comando *here* (ver 2.2.1).

Opcionales:

- **`format`**: Formato de salida (ver 4.6.3). Si no se incluye el sistema intenta extraerlo de la extensión del nombre del archivo, si no es capaz se usará el formato por defecto.
- **`overwrite`**: Parámetro lógico (TRUE o FALSE) que permite sobre escribir un archivo existente.
- **`bylayer`**: Parámetro lógico (TRUE o FALSE) que define escribir un archivo separado por cada capa o layer (Si el objeto es un rasterBrick o rasterStack).
- **`suffix`**: Lista de nombres o números de misma longitud que las capas contenidas en el modelo. Si se usa *bylayer=TRUE*. (Valores entre `nlayers(x)` o `names(x)`). Si un objeto es guardado en disco, adjunto al nombre se incluye el símbolo ⊛ que indica que se encuentra disponible en el sitio *github* del texto.

4.6.2 Leer

Leer un archivo ráster se realiza con la función `raster()` o `brick()` dependiendo si el archivo contiene una o más capas.

La sintaxis es muy simple y similar en ambos casos:

```
objeto <- raster(nombre_archivo)
objeto <- brick(nombre_archivo)
```

- `nombre_archivo`: Nombre y ruta del archivo a ser leído. Se recomienda el uso del comando *here* (ver 2.2.1).

> Es importante señalar que el mismo comando `raster()`, `brick()`o `stack()` posee una serie de funciones adicionales, por ej. crear un objeto vacío 'desde la nada' o definir la geometría básica pero sin datos entre varias otras.

4.6.3 Formatos

Cuando una función escribe un archivo en el disco, el formato del archivo está determinado por el argumento `format` si se proporciona, o por la extensión del archivo (si se conoce la extensión).

En otros casos, se utiliza el formato predeterminado es *raster*, pero esto se puede cambiar usando `rasterOptions`. Los formatos disponibles se pueden revisar en el cuadro 4.2.

```
lst <- writeFormats()
```

Cuadro 4.2: Tipos de Archivo de Salida.

Nombre	Detalle
raster	R-raster
SAGA	SAGA GIS
IDRISI	IDRISI
IDRISIold	IDRISI (img/doc)
BIL	Band by Line
BSQ	Band Sequential
BIP	Band by Pixel
ascii	Arc ASCII
CDF	NetCDF
ADRG	ARC Digitized Raster Graphics
BAG	Bathymetry Attributed Grid
BMP	MS Windows Device Independent Bitmap
BT	VTP .bt (Binary Terrain) 1.3 Format
BYN	Natural Resources Canada's Geoid
CTable2	CTable2 Datum Grid Shift
EHdr	ESRI .hdr Labelled
ELAS	ELAS
ENVI	ENVI .hdr Labelled
ERS	ERMapper .ers Labelled
FITS	Flexible Image Transport System
GPKG	GeoPackage
GS7BG	Golden Software 7 Binary Grid (.grd)
GSBG	Golden Software Binary Grid (.grd)
GTiff	GeoTIFF
GTX	NOAA Vertical Datum .GTX
HDF4Image	HDF4 Dataset
HFA	Erdas Imagine Images (.img)
IDA	Image Data and Analysis
ILWIS	ILWIS Raster Map
INGR	Intergraph Raster

Cuadro 4.3: Continuación...

Nombre	Detalle
ISCE	ISCE raster
ISIS2	USGS Astrogeology ISIS cube (Version 2)
ISIS3	USGS Astrogeology ISIS cube (Version 3)
KRO	KOLOR Raw
LAN	Erdas .LAN/.GIS
Leveller	Leveller heightfield
MBTiles	MBTiles
MRF	Meta Raster Format
netCDF	Network Common Data Format
NGW	NextGIS Web
NITF	National Imagery Transmission Format
NTv2	NTv2 Datum Grid Shift
NWT_GRD	Northwood Numeric Grid Format .grd/.tab
PAux	PCI .aux Labelled
PCIDSK	PCIDSK Database File
PCRaster	PCRaster Raster File
PDF	Geospatial PDF
PDS4	NASA Planetary Data System 4
PNM	Portable Pixmap Format (netpbm)
RMF	Raster Matrix Format
ROI_PAC	ROI_PAC raster
RRASTER	R Raster
RST	Idrisi Raster A.1
SAGA	SAGA GIS Binary Grid (.sdat, .sg-grd-z)
SGI	SGI Image File Format 1.0
Terragen	Terragen heightfield
VICAR	MIPL VICAR file

Descripción Básica del Relieve

4.7 Análisis Exploratorio de Datos

A partir del trabajo de Tukey [1977] se han comenzado a afianzar y difundir las técnicas del análisis exploratorio de datos (EDA) como herramientas que permiten realizar un estudio inicial de los mismos en aproximaciones previas a la realización de procedimientos estadísticos de mayor complejidad, las aplicaciones incluidas en el Análisis Exploratorio de Datos Espaciales tienen por objetivo que el investigador cuente con información estructural del comportamiento de una variable.

Aplicación que además de este comportamiento estructural, permite descubrir errores en la codificación de los datos, determinar casos anómalos y la posibilidad de comprobar supuestos necesarios para la aplicación de la mayoría de los análisis posteriores.

Como un modelo MDR se compone de un arreglo matricial de datos podemos utilizarlos para realizar dichos análisis exploratorio sobre la muestra contenida y así describir el área que representa a escala **global**.

En Smith et al. [2018] presenta los siguientes grupos de estadísticas descriptivas:

- Recuento y Valores Específicos.
- Medidas de Tendencia Central.
- Medidas de Propagación.
- Medidas de Forma de Distribución.

Como primer paso vamos a seleccionar un área de estudio y preparar un set de datos a modo de ejemplo.

4.8 Preparación de los Datos

En la siguiente secuencia de comandos R vamos a descargar la división administrativa de Argentina Nivel 2, seleccionamos el Departamento Azul y descargamos el MDR, lo recortamos y enmascaramos por el polígono. Usamos la proyección UTM H20S y cambiamos la resolución a 60m (Figura 4.16).

```
arg_nivel_2 <- gadm_sp_loadCountries("ARG",
                                     level = 2,
                                     basefile = "RAW/")
azul <- gadm_subset(arg_nivel_2,
                        level = 2,
                        regions = c("Azul"))
azul <- azul$spdf

dtm_azul <- get_elev_raster(azul, z= 10, src = 'aws')

dtm_azul_f <- crop(dtm_azul, azul)
dtm_azul_f <- mask(dtm_azul_f, azul)
utm20  <- crs("+init=epsg:32720")
dtm_azul_proj <- projectExtent(dtm_azul_f, utm20)
res(dtm_azul_proj) <- 60

dtm_azul_utm <- projectRaster(from = dtm_azul_f,
                                    to= dtm_azul_proj,
                                    method = 'bilinear')

azul_proj <- spTransform(x = azul, utm20)
```

4.9 Recuento y Valores Específicos

Son las medidas más simples y directas que podemos realizar sobre una serie finita de valores. Nos permiten evaluar el monto de la data disponible.

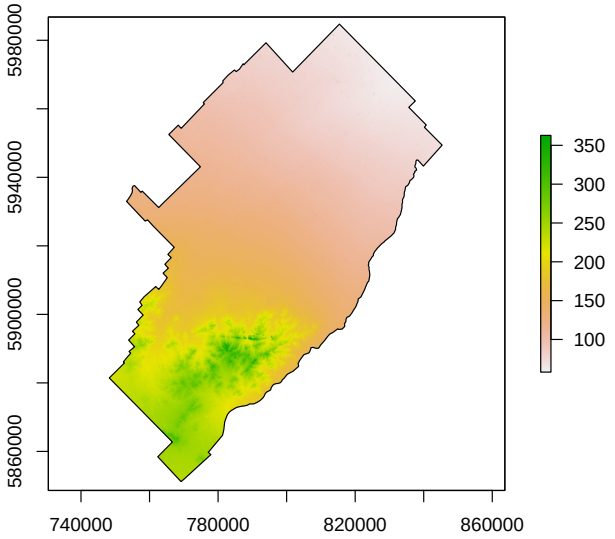


Figura 4.16: División Administrativa de Segundo Nivel. Partido Azul (ARG).

4.9.1 Contar

El total de celdas del MDR obtenido con `ncell()` y el total de celdas NA mediante el cálculo de frecuencia de NAs `freq(x, value=NA)`. Y por diferencia tenemos las `celdas` con información. Un dato derivado de los totales son el área del MDR (en nuestro caso *metros* ya que se encuentra proyectado).

```
nna <- freq(dtm_azul_utm,value=NA)
celdas <- ncell(dtm_azul_utm) - nna
```

El área ráster la obtenemos calculando el tamaño de la celda `res(x)[1] * res(x)[2]` por el número de celdas con información y dividimos por *1e6* para expresar en km^2.

```
area_raster <- (res(dtm_azul_utm)[1] *
                res(dtm_azul_utm)[2]) *
                celdas
(area_raster_km2 <- area_raster/1e6)
```

```
[1] 6578.568
```

Para fines de comparación calculamos el área del polígono mediante el comando `area(x)` dónde x corresponde a un objeto geométrico.

```
(area_polygon_km2 <- area(azul_proj)/1e6)
```

```
[1] 6571.835
```

¿Por qué la diferencia?

4.9.2 Máximo

```
(maximo <- cellStats(dtm_azul_utm, 'max'))
```

```
[1] 363.5457
```

4.9.3 Mínimo

```
(minimo <- cellStats(dtm_azul_utm, 'min'))
```

```
[1] 57.20885
```

4.10 Medidas de Tendencia Central

Entre las medidas estadísticas básicas se encuentran varias formas de *promedios* o *valores medios*. Dado que estas estadísticas intentan proporcionar una única medida para resumir una gran cantidad de valores, se centran en representar todo el conjunto de datos a través de *valores centrales* en lugar de valores extremos y, por lo tanto, se denominan **medidas de tendencia central**.

4.10.1 Promedio

O *media aritmética* a menudo se indica con el símbolo μ. En muchos casos es la muestra mejor estimada (sin sesgo) de valor medio.

```
(promedio <- cellStats(dtm_azul_utm, 'mean'))
```

```
[1] 141.7225
```

4.10.2 Mediana

La mediana es el valor que corresponde a la mitad del set ordenado de valores.

```
(mediana <- median(dtm_azul_utm, na.rm = T))
```

```
[1] 126.9042
```

4.11 Medidas de Propagación o Extensión

La medida básica de propagación o extensión de un conjunto de datos es el *rango*, la diferencia entre el máximo y mínimo, aunque da una información limitada con respecto al patrón o forma de propagación.

4.11.1 Rango

```
(rango <- maximo - minimo)
```

```
[1] 306.3369
```

4.11.2 Cuartil Inferior y Superior

En la secuencia ordenada de datos el 25% de los elementos son menores o iguales que este valor (*cuartil inferior*) y el 75% de los datos son menores o iguales al valor correspondiente (*cuartil superior*). Con el mismo comando se pueden extraer otros valores agregando el porcentaje al vector definido por el parámetro `probs` y si el objeto ráster es muy grande controlar el número de celdas a muestrear con el parámetro `ncells`.

```
(quantiles <- quantile(dtm_azul_utm, probs=c(0.25, 0.75)))
```

```
25% 75%
 93 182
```

4.11.3 Rango Intercuantil

La diferencia entre los valores del cuartil superior e inferior cubre el 50% medio de la distribución. Es una medida sólida de la propagación ya que no se ve afectada por valores atípicos en las colas superior e inferior de la muestra.

```
rango_interq <- quantiles[2] - quantiles[1]
names(rango_interq) <- "" #Remover nombre del valor
rango_interq
```

```
89
```

4.11.4 Varianza

La varianza es la diferencia cuadrática promedio de los valores en un conjunto de datos de la media de su población, μ, o de la media de la muestra (también conocida como varianza de la muestra, donde los datos son una muestra de una población más grande).

Esta medida de dispersión es una de las más utilizadas y, a menudo, se describe como la *desviación cuadrática media* (DME). Las diferencias se elevan al cuadrado para eliminar el efecto de los valores negativos (de lo contrario, la suma sería 0). Sin embargo, el proceso de cuadratura tiene otros resultados. Estos incluyen aumentar considerablemente la ponderación otorgada a valores grandes (positivos o negativos), con el resultado de que la varianza puede ser grande como resultado de la contribución de unos pocos valores atípicos por lo tanto no se considera una estadística sólida. El tamaño de la varianza también está fuera de escala con los datos originales y normalmente se ajusta tomando la raíz cuadrada para cambiar la escala de la medida a los datos, dando la *desviación cuadrática media de la raíz* (RMSD) o **desviación estándar** (σ).

La varianza de la población a menudo se indica con el símbolo σ^2.

```
(varianza  <- var(as.vector(dtm_azul_utm), na.rm = T))
```

```
[1] 3540.502
```

4.11.5 Desviación Estándar

Es la raíz cuadrada de la varianza. También se podrá calcular con `sqrt(varianza)`. Se representa con la letra σ.

```
(desviacion  <- cellStats(dtm_azul_utm, 'sd'))
```

```
[1] 59.50212
```

4.11.6 Coeficiente de Variación

Cuando se desea hacer referencia a la relación entre el tamaño del promedio y la variabilidad de la variable, se utiliza el coeficiente de variación. Su fórmula expresa la desviación estándar como porcentaje del promedio, mostrando una interpretación relativa del grado de variabilidad, Por ejemplo, si el Coeficiente de variación es menor o igual al 80%, significa que la media aritmética es representativa del conjunto de datos, por ende, el conjunto de datos es *Homogéneo*. Por el contrario, si el Coeficiente de variación supera al 80%, el promedio no será representativo del conjunto de datos (por lo que resultará *Heterogéneo*).

```
(cv <- desviacion/promedio * 100)
```

```
[1] 41.98494
```

4.12 Medidas de Forma de Distribución

Las medidas de forma de distribución permiten conocer que forma tiene la curva que representa la serie de datos de la muestra.

4.12.1 Asimetría

Mide si la curva tiene una forma simétrica, es decir, si respecto al centro de esta (centro de simetría) los segmentos de curva que quedan a derecha e izquierda son similares (Figura 4.17).

Los resultados pueden ser los siguientes:

- ***Cero***: *distribución simétrica*. Existe la misma concentración de valores a la derecha y a la izquierda de la media.
- ***Positiva***: *distribución asimétrica positiva*. Existe mayor concentración de valores a la derecha de la media que a su izquierda.
- ***Negativa***: *distribución asimétrica negativa*. Existe mayor concentración de valores a la izquierda de la media que a su de-

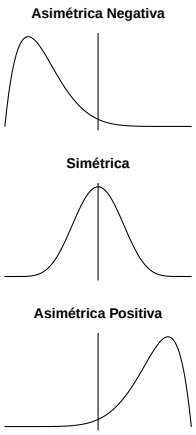


Figura 4.17: Tipos de Distribución

recha.

```
(asimetria <- cellStats(dtm_azul_utm,'skew',na.rm=T))
```

```
[1] 0.7094864
```

4.12.2 Curtosis

Mide si los valores de la distribución están más o menos concentrados alrededor de los valores medios de la muestra. Señala cuan aguda es la zona media de la por lo que también es conocida como *medida de apuntamiento* (Figura 4.18).

Figura 4.18: Tipos de Curtosis: (a) Leptocúrtica. (b) Mesocúrtica. (c) Platicúrtica.

- ***Cero***: Una distribución normal perfecta. Se denomina Mesocúrtica.
- ***Positiva***: Una distribución con las colas más pesadas y la zona media más aguda, más en "punta". Se denomina curva Leptocúrtica.
- ***Negativa***: Colas más livianas, zona media más achatada. Se denomina curva Platicúrtica.

```
(curtosis <- moments::kurtosis(dtm_azul_utm,na.rm = T))
```

```
[1] 2.437521
```

4.13 Comparación de Estadísticas Básicas

Usando las estadísticas descriptivas se realiza la comparación entre dos áreas distintas (Figura 4.19). El cuadro 4.4 muestra los valores obtenidos. Se invita al lector realizar el mismo ejercicio.

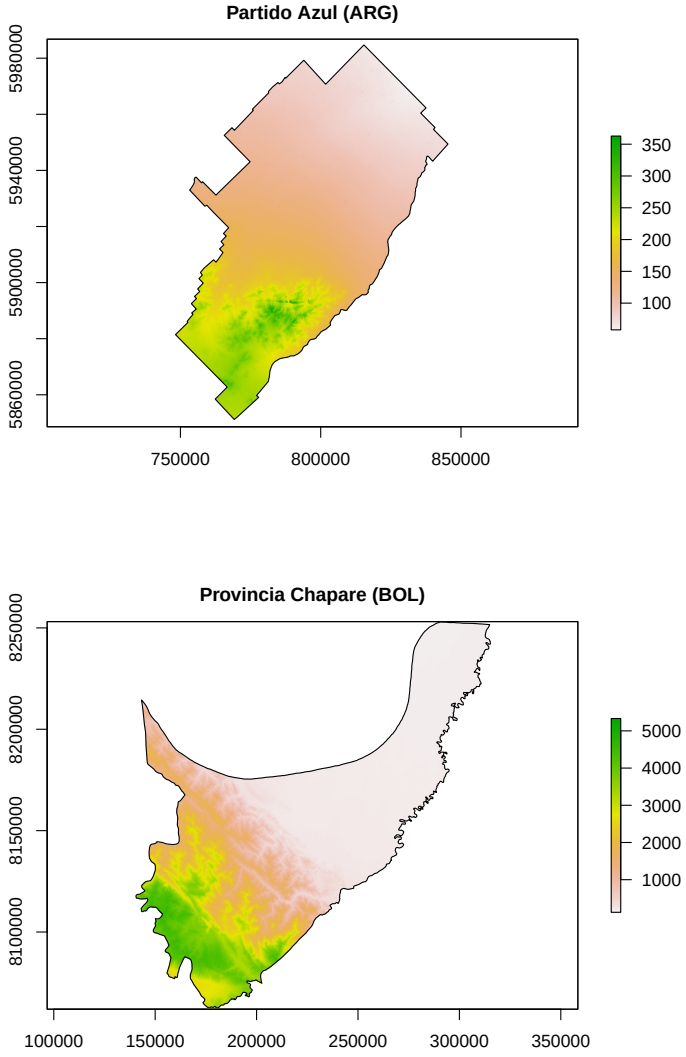


Figura 4.19: Divisiones Administrativas Comparadas.

Cuadro 4.4: Comparativa de Estadísticas Descriptivas.

	Azul(ARG)	Chapare(BOL)
Recuento y Valores Específicos		
Número de Celdas	1827380.00	3580775.00
Área (km²)	6578.57	12890.79
Mínimo	57.21	-100.87
Máximo	363.55	6141.14
Tendencia Central		
Promedio	141.72	1272.38
Mediana	126.90	591.70
Propagación o Extensión		
Rango	306.34	6242.01
Quartil Inferior	93.00	213.45
Quartil Superior	182.00	2145.60
Rango Interquartil	89.00	1932.14
Varianza	3540.50	1713605.33
Desviación Estándar	59.50	1309.05
Coeficiente Variación	41.98	102.88
Forma y Distribución		
Asimetría	0.71	0.98
Curtosis	2.44	2.62

Análisis Geomorfométrico del Relieve

4.14 Algoritmo Básico

La realización de operaciones geomorfométricas dentro de un modelo ráster implica calcular cantidades intermedias (sobre el mismo conjunto de celdas de interés) que luego se utilizan para calcular el producto final. La mayoría de los algoritmos geomorfométricos funcionan a través de **operación de vecindad**: un procedimiento que *mueve una pequeña matriz regular de celdas* (denominada tanto *sub-ventana*, *filtro*, *kernel*) sobre todo el modelo desde la parte superior izquierda a la esquina inferior derecha y repite una fórmula matemática en cada ubicación de esta cuadrícula de muestreo.

Las celdas o píxeles vecinos en una ventana de muestreo se definen comúnmente en relación a un **píxel central**, ubicación para la que un parámetro es derivado. En principio, hay varias formas de designar píxeles vecinos, se utiliza una forma secuencial de las celdas desde arriba a abajo e izquierda derecha (Figura 4.20 superior) o forma de un sistema de coordenado con el origen en la celda central (Figura 4.20 inferior).

Z1	Z2	Z3
Z4	Z5	Z6
Z7	Z8	Z9

Z(−1,1)	Z(0,1)	Z(1,1)
Z(−1,0)	Z(0,0)	Z(1,0)
Z(−1,−1)	Z(0,−1)	Z(1,−1)

Figura 4.20: Formas de designación de las celdas vecinas en una ventana 3x3, por Orden (superior) o Coordenado (inferior).

Calcular un valor derivado de un modelo puede ser una simple repetición de una fórmula dada sobre el área de interés. Considere un modelo ráster muy pequeño de solo 6 × 6 píxeles. Realizamos un zoom en los valores (en este ejemplo elevaciones) una ventana de 3 filas y 3 columnas (un *kernel* de 3x3) y derivar un parámetro deseado (promedio) con una calculadora de bolsillo, figura 4.21.

```
set.seed(4)
ras <- raster(matrix(rnorm(36, mean = 5, sd = 1),
```

rnorm(n, mean, sd): función para generar *n* números aleatorios que siguen la distribución normal, centrada en *mean* y una dispersión *sd*.

```
                    nrow = 6, byrow = T),
xmn=0, xmx=6, ymn=0, ymx=6)
```

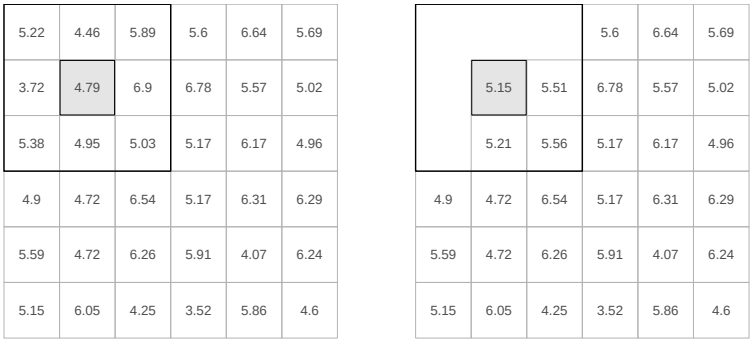


Figura 4.21: Ejemplo Numérico de Algoritmo Geomorfométrico, cálculo del Promedio.

1 Extraer los valores dentro de la ventana 3x3 centrada en el píxel central:

$$muestra = 5.22, 4.46, 5.89, 3.72, 4.79, 6.9, 5.38, 4.95, 5.03 \quad (4.2)$$

2 Aplicar cálculo:

$$\bar{x} = \frac{z1 + z2 + z3 + z4 + z5 + z6 + z7 + z8 + z9}{9} \quad (4.3)$$

3 Reemplazar el Valor del píxel central.
4 Mover la ventana al píxel central siguiente.
5 Barrer toda la extensión del modelo.

Un caso especial es el tratamiento de las celdas del borde o margen del modelo. Generalmente se ignora el proceso en donde la ventana no obtiene valores para todos los vecinos o aplicar una solución alternativa, un *padding* o *relleno virtual*, con el promedio del modelo o repetir el valor adyacente o con una constante, etc.

A continuación presentaremos diferentes formulaciones (cada una de ellas con un grado diferente de complejidad matemática) para extraer atributos geomorfométricos básicos del MDR, luego discutir la importancia y la interpretación de cada uno.

4.15 Atributos Geomorfométricos Básicos

Tal como se describe en 4.1 el estudio cuantitativo del relieve se realiza a través de *atributos geomorfométricos* que no requieren información adicional al contenido en el modelo, ni conocimiento de los procesos formativos ni los procesos que afectan al relieve. El valor refleja la morfometría local en cada celda y serán divididos en dos grupos, geométricos y medidas estadísticas.

Los **parámetros geométricos** son aquellos basados en un análisis de las propiedades geométricas de la superficie terrestre. Esto incluye *pendiente*, *aspecto* y *curvaturas* junto a otros valores derivados de ellos. Entre todos los parámetros que se utilizan para caracterizar una superficie terrestre, las que se describen en esta parte del capítulo son probablemente aquellos con la base matemática más sólida. No debemos olvidar que la superficie terrestre en sí misma es una superficie en un sentido matemático y, como tal, puede analizarse utilizando todos los conceptos de geometría diferencial. Los parámetros como la pendiente o el aspecto son fáciles para calcular con unas pocas ideas geométricas básicas.

Para realizar este análisis matemático, primero necesitamos una función matemática describiendo la superficie terrestre y de ésta extraer nuevos parámetros.

El trabajo fundacional para definir las primeras y segundas derivadas de una superficie (que son necesarias para el cálculo de pendiente, aspecto y curvaturas) se basan en los trabajos de Evans [2019] , Shary [1995], Zevenbergen and Thorne [1987] y el método modificado Evans-Young (Shary et al. [2002]).

El algoritmo de cálculo fija un polinomio de segundo orden a una ventana de 3x3 (Evans [2019], Young and Evans [1978], Pennock et al. [1987]):

$$z = \frac{rx^2}{2} + sxy + \frac{ty^2}{2} + px + qy + z_0 \tag{4.4}$$

Donde p, q, r, s, t, z_0 son coeficientes determinados por:

$$p = \frac{\partial z}{\partial x} = \frac{z3 + z6 + z9 - z1 - z4 - z7}{6d} \tag{4.5}$$

$$q = \frac{\partial z}{\partial y} = \frac{z1 + z2 + z3 - z7 - z8 - z9}{6d} \tag{4.6}$$

$$r = \frac{\partial^2 z}{\partial x^2} = \frac{z1 + z3 + z4 + z6 + z7 + z9 - 2(z2 + z5 + z8)}{3d^2} \tag{4.7}$$

$$s = \frac{\partial^2 z}{\partial x \partial y} = \frac{-z1 + z3 + z7 - z9}{4d^2} \tag{4.8}$$

$$t = \frac{\partial^2 z}{\partial y^2} = \frac{z1 + z2 + z3 + z7 + z8 + z9 - 2(z4 + z5 + z6)}{3d^2} \tag{4.9}$$

$$z_0 = \frac{5z5 + 2(z2 + z4 + z6 + z8) - (z1 + z3 + z7 + z9)}{9} \tag{4.10}$$

Horn [1981] propone utilizar un estimador de diferencias finitas de tercer orden, de modo que las derivadas este-oeste y sur-norte son[6]:

$$p = \frac{\partial z}{\partial x} = \frac{(z3 + 2z6 + z9) - (z1 - 2z4 - z7)}{8d} \tag{4.11}$$

$$q = \frac{\partial z}{\partial y} = \frac{(z1 + 2 * z2 + z3) - (z7 - 2z8 - z9}{8d} \tag{4.12}$$

[6] El paquete *raster* utiliza para el cálculo de la pendiente esta modificación.

4.16 Funciones de la Primera Derivada

Las primeras propiedades morfométricas de un terreno que se pueden estudiar son resultantes de las primeras derivadas parciales de la superficie. Un concepto básico del cálculo vectorial implica la primera derivada parcial es la *gradiente*. Dado un campo escalar (similar las elevaciones contenidas en el MDR) el gradiente es un campo vectorial apuntando en la dirección en la que la variación máxima en los valores de ese campo escalar ocurre. La expresión del gradiente en dos dimensiones es:

$$\nabla \bar{Z} = (\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}) \tag{4.13}$$

Las dos principales propiedades geométricas se pueden derivar del gradiente son su longitud (módulo) y su dirección. Cuando estos conceptos se aplican a la geomorfometría, constituyen dos de los parámetros más importantes de la superficie terrestre: **pendiente** y **aspecto**.

```
utm19  <- crs("+init=epsg:32719")
crs(ras) <- utm19
pendiente <- terrain(x= ras, opt= "slope", unit="degrees
```

39.9	34.9	16.16	27.73
27.58	25.02	20.07	11.05
24.95	25.06	11.57	21.82
15.07	31.55	40.09	32.57

Figura 4.22: Mapa de pendiente (°) para los datos del ejemplo anterior (Notar la reducción de filas y columnas).

4.16.1 Pendiente

La **pendiente del gradiente** refleja la tasa máxima de cambio de los valores de elevación:

$$pendiente = \arctan(|\nabla \bar{Z}|) \tag{4.14}$$

y reemplazando en 4.14 las expresiones 4.11 y 4.12 resulta:

$$pendiente = \arctan(\sqrt{p^2 + q^2}) \tag{4.15}$$

que indica el ángulo entre el plano horizontal y el tangencial a la superficie. Tenga en cuenta que el plano tangencial está definido por el propio vector de gradiente y es, por tanto, normal a la superficie. También es normal que la línea de contorno pase a través del punto.

Si tomamos los datos de muestra de la figura 4.21 (izquierda) y aplicamos la fórmula de pendiente 4.15 resulta el mapa de pendientes en grados (°) de la figura 4.22.

En R utilizamos el comando `terrain()` del paquete *raster* con la opción `opt='slope'` y configuramos la salida en grados sexagesimales con el parámetro `unit='degrees'`

Se recomienda usar la opción *unit='degrees'* cuando el producto sea visualización de los datos, pero considerar que si la utilización de este dato en otros procesos análiticos sea expresado en unidad de *radianes*, *unit='radians'*.

El objeto RasterLayer debe tener definido su sistema coordenado `crs`.

```
pendiente <-terrain(RasterLayer, opt='slope', unit='degrees')
```

El uso de la **pendiente** es importante en los más diversos campos de la ciencia y su uso tan extendido a generado una serie de parámetros derivados para usos específicos como erosión, ecología, humedad del suelo, estabilidad de taludes y un largo etc.

Como ejemplo del proceso de utilización de un atributo para derivar uno nuevo revisaremos el concepto de *superficie*.

4.16.2 Superficie

Si consideramos una celda con resolución d, su área es calculada como $area = d^2$ pero es una área planimétrica o proyectada, no considera el desarrollo de la *superficie* por efecto de la pendiente, aquella es un *área_real* y muy útil, por ejemplo el área a reforestar no será la misma superficie, en estudio de habitat el análisis de cantidad de alimento puede ser afectada por las diferencias entre área versus superficie.

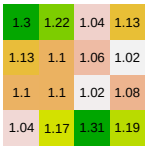


Figura 4.23: Mapa de Superficie. El valor representa la superficie (área proyectada es 1).

Luego, la $superficie$ de una celda con $AREA$ y $pendiente$ será (Berry [1996]):

$$superficie = \frac{AREA}{\cos(pendiente)} \tag{4.16}$$

Donde $AREA$ es la superficie de la celda y $pendiente$ debe ser expresado en radianes.

En R aplicamos la fórmula 4.16 directo sobre nuestro RasterLayer de pendiente (°) y obtenemos un modelo ráster donde cada celda representa la superficie (Figura 4.23):

```
superficie <- 1/cos(pendiente * (pi/180))
plot(superficie)
```

4.16.3 Aspecto

El segundo atributo básico es el **aspecto** que se define como el ángulo entre la dirección del Norte y la proyección horizontal del vector de la gradiente contado en sentido horario en cualquier punto de la superficie (Shary et al. [2002]).

El aspecto es un valor no negativo entre 0° y 360°.

$$aspecto = \arctan(\frac{q}{p}) \tag{4.17}$$

En R utilizamos el comando `terrain()` del paquete *raster* con la opción `opt='aspect'` y configuramos la salida en grados sexagesimales con el parámetro `unit='degrees'` (Figura 4.24).

```
aspecto <-terrain(RasterLayer, opt='aspect',
                  unit='degrees')
```

272.6	252.9	160.2	120
279.4	230.1	193.8	43
284.9	322	68.1	227.6
287	183.5	153.7	214.1

Figura 4.24: Mapa de aspecto (°) para los datos del ejemplo anterior (Notar la reducción de filas y columnas).

Se recomienda usar la opción *unit='degrees'* cuando el producto sea visualización de los datos, pero considerar que si la utilización de este dato en otros procesos análiticos sea expresado en unidad de *radianes*, *unit='radians'*.

El objeto RasterLayer debe tener definido su sistema coordenado `crs`.

4.16.4 Sombreado

Con la combinación de pendiente y el aspecto se puede utilizar para generar los denominados mapas de relieve sombreado. Estos mapas intentan crear imágenes que reflejan la topografía del MDR, por lo que son más claras y más intuitivo para el ojo humano. Estos se han utilizado de forma rutinaria desde la década de 1980 para mejorar la apariencia visual del relieve (Horn [1981]).

Los mapas en relieve sombreados se pueden utilizar como imágenes independientes o, mejor, se pueden ir utilizando para alterar la representación de otras imágenes que contienen diferentes parámetros de diferentes tipos de información por superposición

(Figuras 5.1, 4.37). Este efecto se logra utilizando un mapa de relieve sombreado y modificando el brillo de los píxeles de acuerdo con los valores que contiene. Así da el usuario una impresión del espacio 3D.

Los valores de cada celda se calculan aplicando modelos de insolación simplificados, ignorando la influencia del relieve circundante y considerar la insolación como un parámetro local (Figura 4.25).

Considerando al sol en una posición definida por su **azimut** ϕ y **elevación** θ sobre el horizonte, la insolación de una celda con aspecto a y pendiente s viene dado por Shary et al. [2005]:

$$F = \frac{100\tan(s)}{\sqrt{1+\tan^2(s)}}\left[\frac{\sin(\theta)}{\tan(s)} - \cos(\theta)\sin(\phi - a)\right] \tag{4.18}$$

En R, debemos simplificar los componentes de la formula original en pequeños componentes por facilidad de escritura y facilitar la revisión. Donde `theta` es el ángulo de *elevación del sol* en grados y convertidos a radianes multiplicando por `pi/180` y `alpha` es el *azimut solar* en grados y convertido a radianes multiplicado por `pi/180`. También se consideran las coberturas `pendiente` y `aspecto` expresadas en *radianes*.

```
theta <- 35 * pi/180
phi <- 315 * pi/180
f0 <- 100 * tan(pendiente)
f1 <- sqrt(1+ tan(pendiente)^2)
f2 <- sin(theta)/tan(pendiente)
f3 <-  cos(theta) * sin(phi-aspecto)
f <- f0 / f1 * (f2 - f3)
plot(f)
```

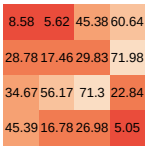


Figura 4.25: Mapa de Insolación para los datos del ejemplo anterior (Notar la reducción de filas y columnas).

Aunque el paquete *raster* provee una función que encapsula todo bajo el comando `hillshade()` que necesita como parámetros de ingreso la pendiente, aspecto el ángulo solar y dirección (azimut) (Figura 4.26):

```
sombreado <-hillShade(slope = pendiente,
                      aspect = aspecto,
```

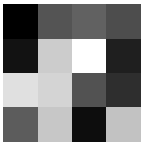


Figura 4.26: Mapa de Sombreado para los datos del ejemplo anterior (Notar la reducción de filas y columnas).

```
                    angle = 35,
                    direction = 315,
                    normalize = F)
plot(sombreado)
```

4.17 Funciones de la Segunda Derivada

El siguiente paso en el análisis de los atributos geomorfométricos locales son las que se desprenden de las segundas derivadas. Estos están relacionados con la *concavidad* y *convexidad* de la superficie.

4.17.1 Curvaturas

El parámetro que describe esto se llama **curvatura**. Un concepto importante para la comprensión es el significado geométrico de las curvaturas es la llamada sección normal de una superficie (Figura 4.27).

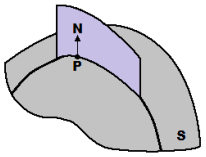


Figura 4.27: Sección Normal en la superficie S en el punto P es una curva que resulta de la intersección de dicha superficie y un plano que pasa a través del vector N ortogonal a S. Modificado de Shary (1995).

Una sección normal es una curva plana, cuya curvatura se define como $1/R$, donde R es el radio de un círculo que mejor se ajusta a esta curva en un punto dado (Figura 4.28).

La curvatura k de una curva plana $z(x)$ viene dada por la fórmula (Hengl and Reuter [2009]):

$$k = \frac{\frac{\partial^2 z}{\partial z^2}}{\left[1 + \left(\frac{\partial y}{\partial x}\right)^2\right]^{1.5}} \tag{4.19}$$

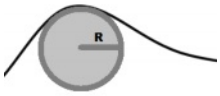


Figura 4.28: En una curva plana un círculo de radio R que es el mejor ajustado a esta curva en un punto dado, la Curvatura será el inverso de R o 1/R.

Una forma general de introducir una curvatura en una superficie es definir una sección curva plana en él y luego usar la fórmula anterior para representar esta curvatura como función $f(p, q, r, s, t)$ de la primera y segunda derivadas parciales de elevación (ver ecuaciones 4.5, 4.6, 4.7, 4.8, 4.9).

El punto de contacto entre el círculo y la superficie se centrará en la celda a calcular, en un punto de la superficie terrestre con pendiente y aspecto. De ese punto de contacto y siguiendo a Shary et al. [2002] se generan cuatro direcciones naturales (ver figura 4.29).

aa' y bb' son físicamente definidas por el campo gravitacional de la tierra, mientras que las otras dos son definidas por la superficie: cc' el valor máximo de la sección curva plana y dd' el valor mínimo.

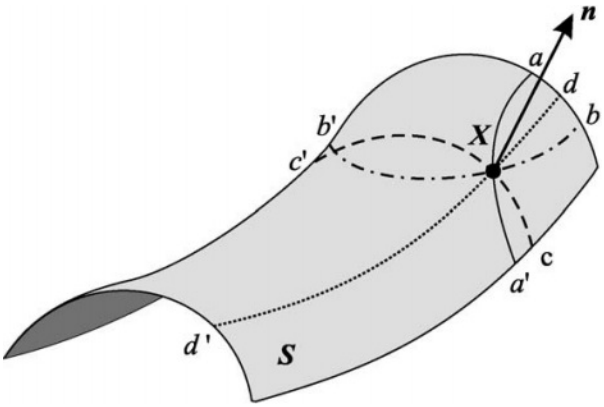


Figura 4.29: Cuatro direcciones generadas naturalmente en la superficie S. Tomado de Hengl and Reuter(2009).

Los nombres de las curvaturas para cada una de las cuatro secciones son:

- **PERFC**: *Curvatura de Perfil (o Vertical)*: Sección aa' (Figura 4.30).
- **PLANC**: *Curvatura Planimétrica (o Plana)*: Sección bb' (Figura 4.31).

Y como información complementaria agregaremos ***Curvatura Máxima***, sección cc' y ***Curvatura Mínima***, Sección dd'.

El cálculo de curvaturas se realizará utilizando el método Zevenbergen and Thorne [1987] que redefine los parámetros de las derivadas parciales como:

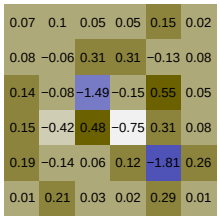


Figura 4.30: Mapa de Curvatura de Perfil (PERFC) para los datos del ejemplo anterior.

$$p = \frac{\partial z}{\partial x} = \frac{z6 - z4}{2d} \tag{4.20}$$

$$q = \frac{\partial z}{\partial y} = \frac{z2 - z8}{2d} \tag{4.21}$$

$$r = \frac{\partial^2 z}{\partial x^2} = \frac{z4 + z6 - 2z5}{d^2} \tag{4.22}$$

$$s = \frac{\partial^2 z}{\partial x \partial y} = \frac{-z1 + z3 + z7 - z9}{4d^2} \tag{4.23}$$

$$t = \frac{\partial^2 z}{\partial y^2} = \frac{z2 + z8 - 2z5}{d^2} \tag{4.24}$$

y las fórmulas para el cálculo de curvaturas son:

$$PERFC = \frac{-2(rq^2 + tq^2 + spq)}{(p^2 + q^2)} \tag{4.25}$$

$$PLANC = \frac{2(rq^2 + tp^2 - spq)}{(p^2 + q^2)} \tag{4.26}$$

En la práctica R vamos a utilizar el paquete *spatialEco* (Evans [2021]) implementa dichas fórmulas con el comando `curvature()` y vía parámetro `type` se configura el tipo de curvatura a calcular:

```
PLANC <- curvature(ras, type = 'planform')
PERFC <- curvature(ras, type = 'profile')
```

La *curvatura de perfil* (PERFC) representa la dirección de la pendiente máxima y la *curvatura de la forma en planta* o *planimetrica* (PLANC) es perpendicular a la dirección de la pendiente máxima.

Los valores negativos en la curvatura del perfil indican que la superficie es *convexa hacia arriba*, mientras que los valores positivos indican que la superficie es *cóncava hacia arriba*.

Los valores positivos en la curvatura de la forma en planta indican que la superficie es *lateralmente convexa*, mientras que los valores negativos indican que la superficie es *lateralmente cóncava*. Un valor cero (0) en una o ambas, indican que la superficie es *plana*.

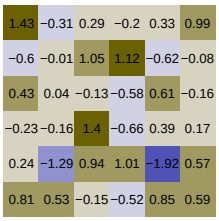


Figura 4.31: Mapa de Curvatura Planimétrica o Plana (PLANC) para los datos del ejemplo anterior.

Como primer paso debemos instalar el paquete que recomendamos realizarlo siempre en la **consola** y luego cargar en memoria el paquete requerido con *library(spatialEco)* en el script.

4.18 Cálculo de Parámetros

Vamos a poner en práctica los atributos recién descritos en un modelo digital de relieve. Creamos nuestro modelo ráster siguiendo los pasos de selección y descarga (ver 4.3.3), proyección, recorte y enmascarado (ver 4.36):

```
utm19  <- crs("+init=epsg:32719")
#Descarga de área de estudio
cl_nivel_2 <- gadm_sp_loadCountries("CHL", level = 2,
```

```
                                    basefile = "RAW/")
#Seleccionamos un polígono específico
cordillera <- gadm_subset(cl_nivel_2, level = 2,
                          regions = c("Cordillera"))
#Extraemos la componente geométrica
cordillera <- cordillera$spdf
#Projectamos CRS
cordillera_proj <- spTransform(cordillera, utm19)
#Descarga de información topográfica
dtm_cordillera <- get_elev_raster(cordillera, z= 10,src= 'aws')
#Recorte y enmascarado por el polígono
dtm_cordillera_f <- crop(dtm_cordillera, cordillera)
dtm_cordillera_f <- mask(dtm_cordillera_f, cordillera)
#Preparamos la proyección
dtm_cordillera_proj <- projectExtent(dtm_cordillera_f, utm19)
#Ajustamos la resolución espacial
res(dtm_cordillera_proj) <- 50
#Realizamos el cambio CRS
dtm_cordillera_utm <- projectRaster(from = dtm_cordillera_f,
                                    to= dtm_cordillera_proj,
                                    method = 'bilinear')
```

Generamos un mapa de relieve basado en un set de colores del paquete *colorspace* (Figura 4.32).

```
colores <- sequential_hcl(n = 100, palette = "YlOrRd", rev = T)
plot(dtm_cordillera_utm, col= colores, asp=1)
```

Aprovechamos de guardar en disco nuestro MDR para ser utilizado en el capítulo de diseño cartográfico (ver 4.23.4) utilizando el formato *.grd* y respetando el nombre del objeto creado®.

```
writeRaster(dtm_cordillera_utm,
            here("raw","dtm_cordillera_utm.grd"),
            overwrite=T)
```

Atributos geomorfométricos usando la función *terrain()* y *hillshade()*:

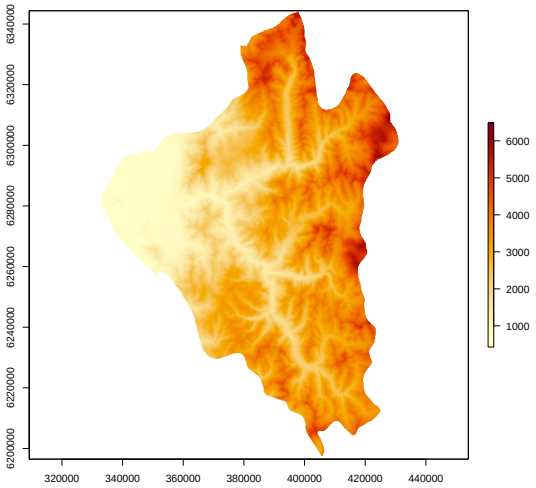


Figura 4.32: Provincia Coordillera, Chile. Modelo Digital de Relieve.

```
pen <- terrain(dtm_cordillera_utm, opt='slope',
               unit='degrees')
aspec <- terrain(dtm_cordillera_utm, opt= "aspect",
                 unit= "degrees")
hill <- hillShade(slope = pen, aspect = aspec,
                  angle=35, direction=315)
```

Mapa de pendientes (Figura 4.33).

```
colores <- sequential_hcl(100, palette = "Reds", rev = T)
plot(pen, col= colores, asp=1)
```

Mapa de Aspecto (Figura 4.34).

```
colores <- qualitative_hcl(100, palette = "Set 2",
                           rev = T, c = 90)
plot(aspec, col= colores, asp=1)
```

Para el cálculo del sombreado se requieren los parámetros de *pendiente* y *aspecto* en radianes.

Mapa de Sombreado utilizando una escala de grises (Figura 4.35).

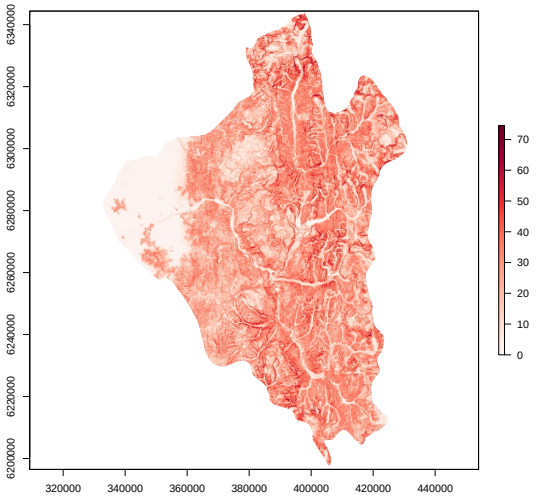


Figura 4.33: Modelo Digital de Pendientes(°).

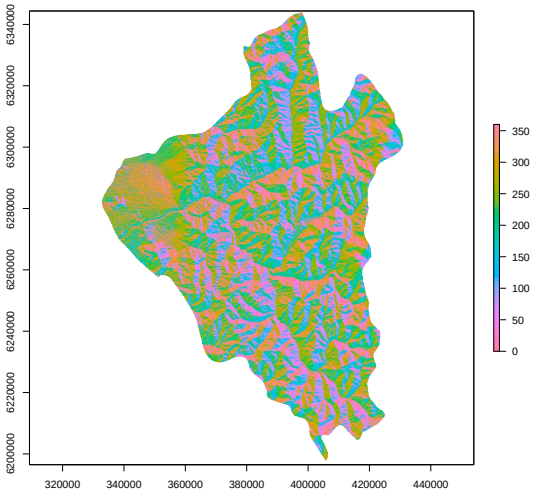


Figura 4.34: Modelo Digital de Aspecto(°).

```
rampa <- brewer.pal(n = 9, name = "Greys")
#rev(), invierte el vector
colores <- rev(colorRampPalette(rampa)(100))
plot(hill, col= colores, asp=1, legend=F)
```

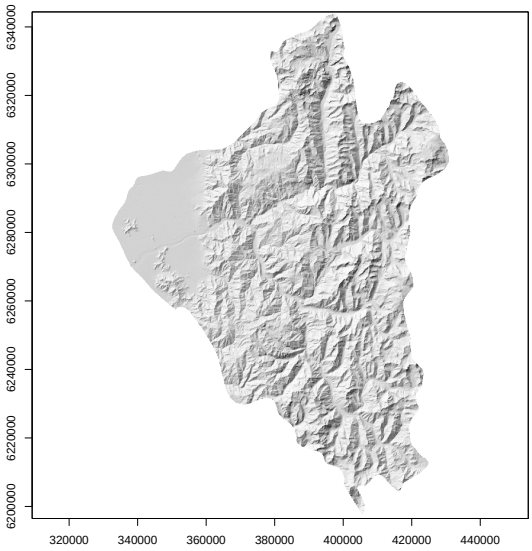


Figura 4.35: Sombreado Digital de Relieve. Notar que la paleta de colores se obtiene del paquete RColorBrewer.

Cálculo de Curvaturas:

```
planform <- curvature(dtm_cordillera_utm, type = "planform")
profilec <- curvature(dtm_cordillera_utm, type = "profile")
```

Diseño de paleta colores (divergente ya que el atributo curvatura posee valores positivos y negativos):

```
rampa <- brewer.pal(n = 11, name = "RdGy")
colores <- colorRampPalette(rampa)(80)
```

Dibujo del área de trabajo y definición de zona de detalle mediante la creación de un objeto *extent* y superpuesto al mapa (Figura 4.36):

```
plot(planform, axes=F, col=colores, asp=1)
ext <- extent(c(383000,385000,6240000,6242000))
plot(ext, add=T)
```

Recorte de la zona de estudio mediante la solución propuesta

en 4.5.3 y ploteo del mapa detalle:

```
plan_clip <- planform[ext, drop=F]
plot(plan_clip,axes=F, legend=F, col=colores, asp=1)
```

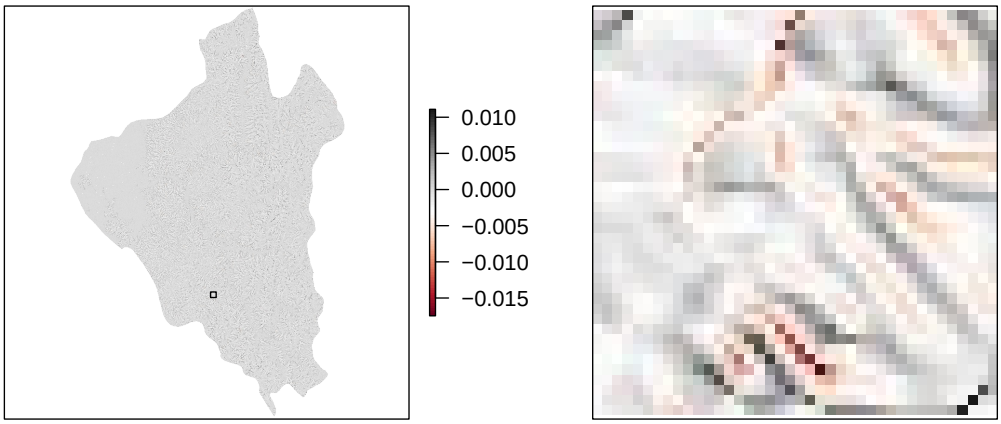


Figura 4.36: Curvatura Planimétrica (izquierda). Detalle del área destacada (derecha).

Dibujo del área de trabajo y definición de zona de detalle mediante la creación de un objeto *extent* y dibujo sobre el mapa (Figura 4.37):

```
plot(profilec, axes=F, col=colores, asp=1)
ext <- extent(c(383000,385000,6240000,6242000))
plot(ext, add=T)
```

Recorte de la zona de estudio mediante la solución propuesta en 4.5.3 y ploteo del mapa detalle:

```
plof_clip <- profilec[ext, drop=F]
plot(plof_clip, axes=F, legend=F, col=colores, asp=1)
```

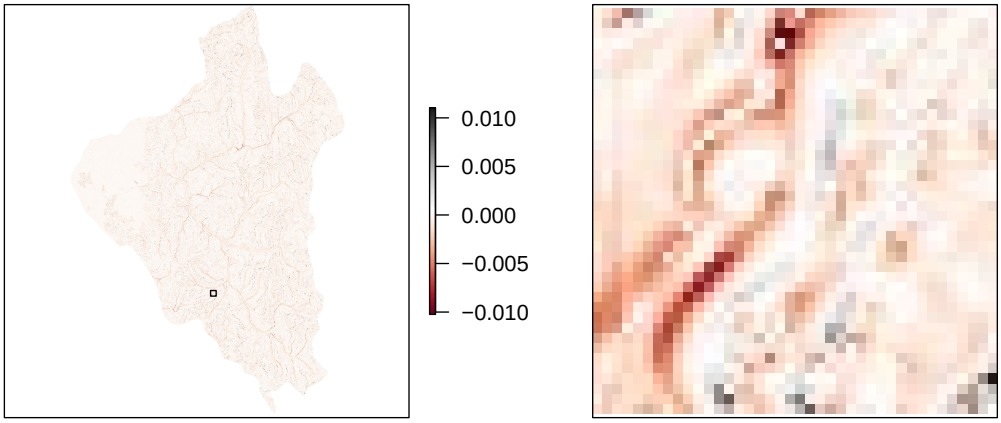


Figura 4.37: Curvatura de Perfil (izquierda). Detalle del área destacada (derecha).

4.19 Medidas Estadísticas

En el punto 4.6.3 se describen los estadísticos básicos que describen en su totalidad nuestro MDR, mientras que en el **Análisis local** (conocido como *vecindario*), implica realizar operaciones con los valores de una celda dada y las celdas que la rodean hasta una cierta distancia (kernel 3x3, 4x4, 5x5 y más) (ver 4.14). Mientras que, como hemos visto, los parámetros geomorfométricos geométricos suelen hacer uso de una ventana cuadrada de 3×3, el análisis local estadístico también puede utilizar vecindarios de diferentes tamaños. Cualquiera que sea el tamaño y la forma del vecindario considerado, el cálculo de los parámetros derivados se lleva a cabo de la misma manera y es simplemente utilizando la totalidad de los valores contenidos dentro de los límites.

Los primeros parámetros a considerar son los primeros *cuatro momentos de la distribución de un valor*[7], a saber, el *valor medio*, la *desviación estándar*, el coeficiente de *asimetría* y el coeficiente de *curtosis*.

Calcular el valor medio de la elevación sirve como filtro para reducir el ruido y eliminar, entre otras cosas, depresiones falsas de una sola celda. Sin embargo, existen métodos más sofisticados para esto, que dan mucho mejores resultados (Jenson and Domingue [1988], Martz and De Jong [1988], Morris and Heer-

[7] Los **momentos** de una función son medidas cuantitativas relacionadas con la forma de la gráfica de la función (Wikipedia).

degen [1988]), ya que no tocan aquellas celdas que constituyen la depresión.

Otros parámetros estadísticos, como la mediana, se puede utilizar para obtener un resultado similar. Tenga en cuenta que muchas medidas estadísticas se superponen con medidas geométricas. Por ejemplo, la desviación estándar está fuertemente correlacionada con la pendiente.

4.19.1 Comando Focal

En el paquete *raster* encontramos la función `focal()`, que mediante una operación de vecindad calcula un ráster de salida en donde el valor para cada celda de salida es **una función** de los valores de todas las celdas de entrada que están en una vecindad especificada alrededor de esa ubicación (un *kernel* de 3x3, 5x5, 7x7, etc). La función que se realiza en la entrada será una estadística, como el *máximo*, el *promedio* o la *suma* de todos los valores que se encuentran en esa vecindad.

La función focal trabaja con una matriz de factores (la ventana móvil) y debe poseer un tamaño impar pero no necesariamente cuadrada (por ej. 3x5, 5x5, 7x3), estos factores definen el **peso** o aporte de la celda en el valor final. Por ejemplo, en un kernel de 3x3, el promedio es la sumatoria de cada uno de los 9 valores multiplicado por su correspondiente peso w fijado en 1/9:

$$promedio = \sum_{i=1}^{i=9} Z_i * w_i \tag{4.27}$$

Y expresado en términos de R debemos construir un objeto `matrix` con el tamaño y valores (pesos) requerido:

```
(matriz <- matrix(1/9, nc=3, nr=3))
```

```
          [,1]      [,2]      [,3]
[1,] 0.1111111 0.1111111 0.1111111
[2,] 0.1111111 0.1111111 0.1111111
[3,] 0.1111111 0.1111111 0.1111111
```

Y pasarlo al comando `focal()`:

```
promedio <- focal(raster_original, w=matriz)
```

También se debe considerar el *efecto borde* (no se puede posicionar un kernel en las filas y columnas extremas) ya que algunas celdas de la ventana "quedan fuera" del modelo y por tanto no se pueden calcular. Debemos decidir que hacer, y tenemos dos opciones, ignorar aquellas celdas y cambiar el tamaño del ráster de salida o proveer al sistema de un valor para "extender" (*padding)* el modelo y aplicar la ventana móvil.

El parámetro `pad` puede ser `TRUE` o `FALSE` dependiendo de si deseo extender o no el modelo. Aplicar la corrección debe proveer un valor para *rellenar* mediante `padValue` (Figura 4.38).

```
promedio <- focal(raster_original, w=matriz, pad=FALSE)
```

o:

```
media <- cellStats(raster_original,'mean') #Usar promedio
promedio <- focal(raster_original, w=matriz,
                  pad=TRUE, padValue= media)
```

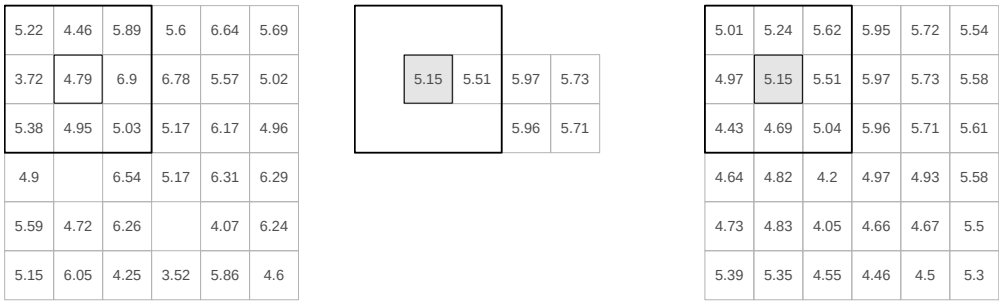


Figura 4.38: Cálculo Promedio Focal. (izq) valores originales (centro) sin padding y remoción de valores NA (der) padding con promedio global sin remover NA.

Si se utiliza una ventana mayor (por ej. 5x5) se debe prestar atención en proveer los pesos correspondientes: `matrix(1/25,nrow=5,ncol=5)`.

También se debe considerar la presencia de valores `NA` y si serán removidos del cálculo (`na.rm=TRUE`) y notar que el cálculo solo devolverá `NA` si **todos** los valores son `NA`.

Generalmente remover los valores nulos no es buena idea por-

que podría desbalancear el efecto de los pesos asignados.

El último parámetro muy importante es `NAonly=TRUE` que solo aquellas celdas que son `NA` serán reemplazadas con el valor calculado (Figura 4.39).

5.22	4.46	5.89	5.6	6.64	5.69
3.72	4.79	6.9	6.78	5.57	5.02
5.38	4.95	5.03	5.17	6.17	4.96
4.9		6.54	5.17	6.31	6.29
5.59	4.72	6.26		4.07	6.24
5.15	6.05	4.25	3.52	5.86	4.6

5.22	4.46	5.89	5.6	6.64	5.69
3.72	4.79	6.9	6.78	5.57	5.02
5.38	4.95	5.03	5.17	6.17	4.96
4.9	4.82	6.54	5.17	6.31	6.29
5.59	4.72	6.26	4.66	4.07	6.24
5.15	6.05	4.25	3.52	5.86	4.6

Figura 4.39: Uso parámetro NAonly= TRUE. Solo se calculan los valores NA.

Además de los pesos, opcionalmente se puede indicar la función a combinar en el cálculo. Esta función debe aceptar múltiples valores (es un conjunto de valores vecinos) y debe retornar un solo valor (el que se asigna a la celda central), por ejemplo:

```
matriz <- matrix(1, nc=3, nr=3) #matriz de unos
promedio <- focal(raster_original, w=matriz, fun=mean)
```

En la figura 4.40 se muestra el resultado de las funciones (en orden izquierda - derecha, arriba abajo): Valores iniciales, *mean*, *min*, *sd*, *sum* y *modal*. Todos aplican padding con el promedio global y no remueven valores *NA*.

La combinación de formula con una matriz de pesos *organizada de manera especial* permiten destacar, ignorar, suavizar o fortalecer el impacto en el cálculo final, por ejemplo necesitamos construir un cálculo focal que busca estimar el valor central a partir del valor máximo de la esquina inferior derecha de una matriz de 3x3:

```
matriz_sur_este = matrix(c(0,0,0,0,1,1,0,1,2), nrow=3)
print(matriz_sur_este)
```

```
     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    1    1
```

Original

5.22	4.46	5.89	5.6	6.64	5.69
3.72	4.79	6.9	6.78	5.57	5.02
5.38	4.95	5.03	5.17	6.17	4.96
4.9		6.54	5.17	6.31	6.29
5.59	4.72	6.26		4.07	6.24
5.15	6.05	4.25	3.52	5.86	4.6

Promedio

5.01	5.24	5.62	5.95	5.72	5.54
4.97	5.15	5.51	5.97	5.73	5.58
4.99	5.28	5.67	5.96	5.71	5.61
5.21	5.42	5.41	5.59	5.55	5.58
5.32	5.43	5.21	5.25	5.26	5.5
5.39	5.35	5.12	5.02	5.06	5.3

Mínimo

3.72	3.72	4.46	5.39	5.02	5.02
3.72	3.72	4.46	5.03	4.96	4.96
3.72	3.72	4.79	5.03	4.96	4.96
4.72	4.72	4.72	4.07	4.07	4.07
4.72	4.25	3.52	3.52	3.52	4.07
4.72	4.25	3.52	3.52	3.52	4.07

Desviación Estándar

0.59	0.89	0.81	0.64	0.59	0.45
0.57	0.89	0.86	0.69	0.67	0.53
0.57	1.01	0.9	0.73	0.68	0.52
0.31	0.67	0.7	0.86	0.83	0.76
0.41	0.81	1.13	1.17	1.09	0.78
0.35	0.62	0.92	0.96	0.92	0.64

Suma

45.13	47.14	50.57	53.53	51.45	49.86
44.69	46.34	49.56	53.73	51.57	50.2
39.91	42.21	45.32	53.62	51.41	50.47
41.72	43.38	37.84	44.71	44.36	50.2
42.59	43.46	36.49	41.97	42.05	49.53
48.47	48.19	40.96	40.12	40.46	47.72

Figura 4.40: Cálculo Focal, distintas funciones.

```
[3,]    0    1    2
```

Y usaremos el valor *máximo global* para realizar el padding y no sufrir efecto de borde:

```
maximo_global <- cellStats(ras,'max')
```

Y pasamos los parámetros al comando focal para obtener nuestro nuevo modelo ráster (Figura 4.41):

```
impacto_sur_este <- focal(raster_original,
                          w= matriz_sur_este,
                          fun= max,
                          pad= T,
                          padValue= maximo_global)
```

5.22	4.46	5.89	5.6	6.64	5.69
3.72	4.79	6.9	6.78	5.57	5.02
5.38	4.95	5.03	5.17	6.17	4.96
4.9	4.72	6.54	5.17	6.31	6.29
5.59	4.72	6.26	5.91	4.07	6.24
5.15	6.05	4.25	3.52	5.86	4.6

9.57	13.79	13.55	11.13	10.03	13.79
9.91	10.07	10.34	12.33	9.91	13.79
9.43	13.08	10.33	12.62	12.58	13.79
9.43	12.51	11.82	8.14	12.48	13.79
12.1	8.49	7.04	11.72	9.19	13.79
13.79	13.79	13.79	13.79	13.79	13.79

Figura 4.41: Mapa Original (izquierda), Mapa de Celdas Estimadas a partir de valores al sur-este de la celda central (derecha).

La facilidad de expresar una función en término de fórmulas y pesos permiten expresar o definir nuevos parámetros geomorfométricos, como ejemplo vamos a revisar tres valores que describen el relieve (basado en Wilson et al. [2007]). Figura \ref{fig:geomorfo-focal-7:

TRI: *Terrain Ruggedness Index* o *índice de rugosidad del terreno* es el promedio de las diferencias absolutas entre el valor de la celda central y sus 8 vecinos.

Expresado en el comando focal vamos primero a definir una matriz de pesos uno (todas las celdas tienen similar aporte):

```
matriz_pesos <- matrix(1, nrow=3, ncol=3)
```

Primero vamos escribir la cabecera o definición:

- **`function(x, ...)`**: Es una función que recibe un objeto `x`

(nuestro modelo ráster) y puede o no recibir parámetros adicionales `...`, símbolo que le indica a R que pueden existir parámetros adicionales.

- **`abs(x[-5]-x[5])`**: Diferencia (resta) absoluta de los valores de la ventana móvil (menos el valor central $x[-5]$) contra el valor central [5] y este resultado se debe:
- **`sum( )/8`**: Promediar; sumando todas las diferencias y dividiendo por 8 (recuerda que removimos el índice $x[5]$ o celda central).

Finalmente todo se pasa como parámetros al comando **`focal`** del paquete ráster:

```
TRI <- focal(raster_original,
             w=matriz_pesos,
             fun=function(x, ...) sum(abs(x[-5]-x[5]))/8
             )
```

TPI: *Topographic Position Index* o *índice de Posición Topográfica* es la diferencia entre el valor de la celda central y el valor medio de de las 8 celdas vecinas. En R quedaría:

```
TPI <- focal(raster_original,
             w=matriz_pesos,
             fun=function(x, ...) x[5] - mean(x[-5])
             )
```

RUGOSIDAD: es la diferencia entre los valores máximo y mínimo en la ventana móvil.

```
rugosidad <- focal(raster_original,
               w=matriz_pesos,
               fun=function(x, ...) max(x) - min(x)
               )
```

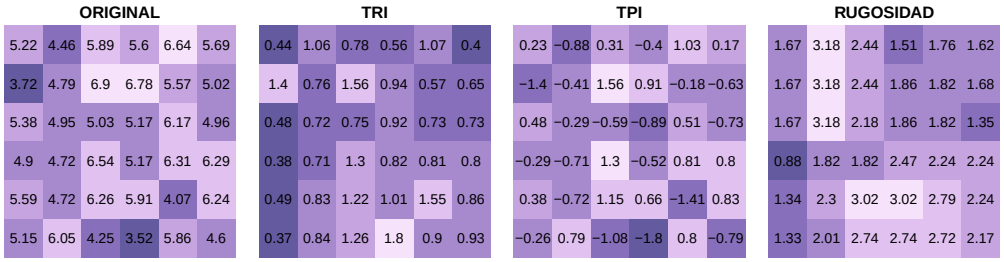


Figura 4.42: MDR original y cuatro parámetros geomorfométricos.

4.20 Cálculo de Parámetros Geomorfométricos

Creamos nuestro modelo ráster siguiendo los pasos de selección y descarga (ver 4.3.3), proyección, recorte y enmascarado (ver 4.37):

```
utm18  <- crs("+init=epsg:32718")
col_nivel_2 <- gadm_sp_loadCountries("COL", level = 2,
                                     basefile = "RAW/")
lerida <- gadm_subset(col_nivel_2, level = 2,
                          regions = c("Lérida"))
lerida <- lerida$spdf
lerida_proj <- spTransform(lerida, utm18)
dtm_lerida <- get_elev_raster(lerida, z= 11,
                                  src = 'aws')
dtm_lerida_f <- crop(dtm_lerida, lerida)
dtm_lerida_f <- mask(dtm_lerida_f, lerida)
dtm_lerida_proj <- projectExtent(dtm_lerida_f, utm18)
res(dtm_lerida_proj) <- 40
dtm_lerida_utm <- projectRaster(from = dtm_lerida_f,
                                    to= dtm_lerida_proj,
                                    method = 'bilinear')
```

El área de estudio (Figura 4.43).

```
#paleta de colores
colores <- sequential_hcl(100, palette = "YlOrRd", rev = T)
plot(dtm_lerida_utm, col= colores, asp=1)
```

Cálculo de la desviación estándar mediante la función *focal()* (Figura 4.44).

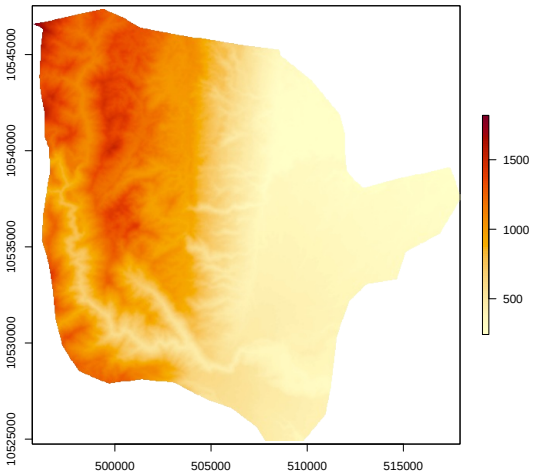


Figura 4.43: Área de estudios, Municipio de Lérida, Colombia. Modelo Digital de Relieve.

```
#Estimación y mapa de la desviación estándar
d_estandar <- focal(dtm_lerida_utm, w=matrix(1,3,3), fun=sd)
colores <- sequential_hcl(100, palette = "GnBu", rev = F)
plot(d_estandar, col= colores, asp=1)
```

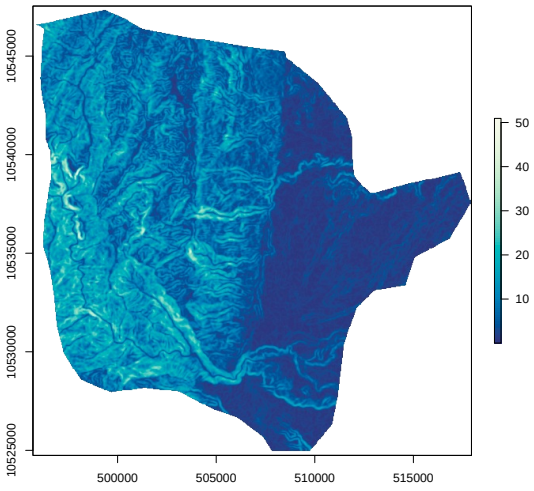


Figura 4.44: Zoom Área de Estudio, cálculo de la desviación estándar.

Cálculo de la media usando un kernel de 5x5 con *focal()* (Figura 4.45).

```
d_estandar <- focal(dtm_lerida_utm, w=matrix(1,25,25), fun='mean')
colores <- sequential_hcl(100, palette = "YlOrRd", rev = T)
plot(d_estandar, col= colores, asp=1)
```

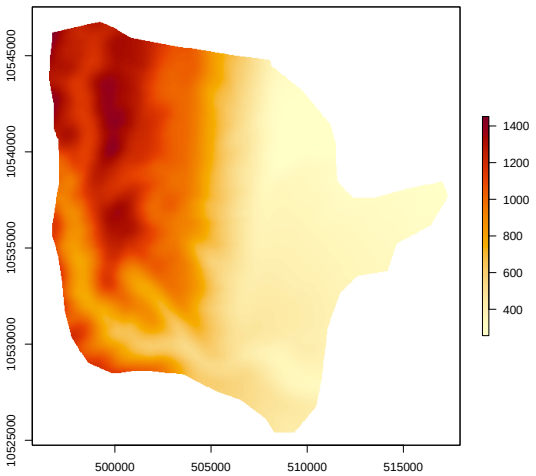


Figura 4.45: Área de Estudio, cálculo del promedio con un kernel 5x5.

Cálculo del factor TPI (Figura 4.46).

```
TPI <- focal(dtm_lerida_utm, w=matrix(1,5,5),
             fun=function(x,...) x[5] - mean(x[-5]))
colores <- diverging_hcl(100, palette = "Red-Green")
plot(TPI, col= colores, asp=1)
```

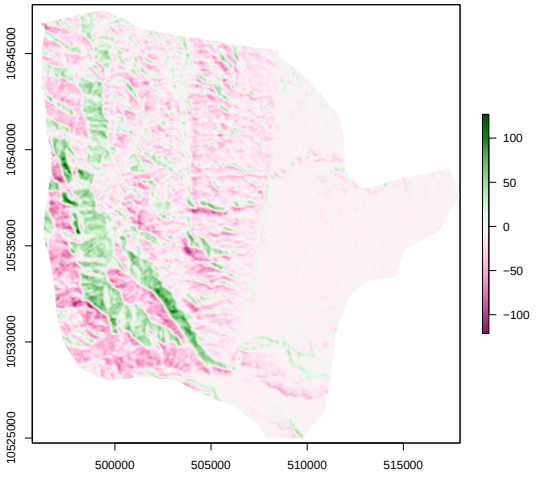


Figura 4.46: Zoom Área de Estudio, cálculo TPI.

Cálculo de la rugosidad (Figura 4.47).

```
rug <- focal(dtm_lerida_utm, w=matrix(1,5,5),
             fun=function(x,...) max(x) - min(x))
colores <- diverging_hcl(100, palette = "Red-Green")
plot(rug, col= colores, asp=1)
```

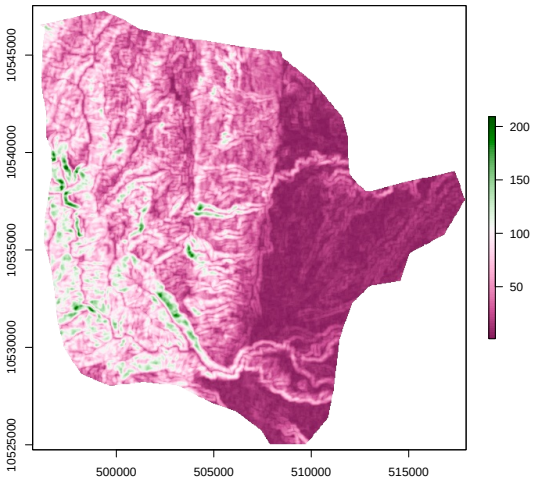


Figura 4.47: Zoom Área de Estudio, cálculo RUGOSIDAD.

Batimetría

La batimetría es el equivalente submarino de la altimetría. El nombre proviene del griego *profundo*, y *medida*. Es el estudio de las profundidades marinas, de la tercera dimensión de los fondos lacustres o marinos. Un mapa batimétrico (o carta batimétrica) normalmente muestra el relieve del fondo o terreno como *isóbatas*[8], y puede también dar información adicional de navegación en superficie.

Originalmente, *batimetría* se refería a la medida de la profundidad oceánica. Las primeras técnicas usaban segmentos de longitud conocida de cable o cuerda pesada, descolgadas por el lateral de un barco. La mayor limitación de esta técnica es que mide la profundidad en un solo punto cada vez, por lo que es muy ineficiente. También es muy imprecisa, ya que está sujeta a los movimientos del barco, las mareas, y las corrientes que puedan afectar al cable.

Los datos usados hoy en día para la confección de mapas batimétricos provienen normalmente de un sonar montado bajo la quilla o en el lateral de un buque, lanzando una onda de sonido hacia el fondo marino (Figura 4.48). La cantidad de tiempo que tarda el sonido en ir a través del agua, rebotar en el fondo y volver, informa al equipo de la profundidad real.

Años atrás, se podía calcular la media de cada uno de los impulsos individuales de un sonar para confeccionar un mapa continuo en lugar de una medición de puntos. Hoy día se puede usar un sonar de barrido ancho, consistente en docenas de ondas simultáneas, muy estrechas y adyacentes entre sí, formando un abanico de entre 90° y 180°.

[8] La **isóbata** es una curva que se utiliza para la representación cartográfica de los puntos de *igual profundidad* en el océano y en el mar, así como en lagos de grandes dimensiones. También es usado para representar la *profundidad* de formaciones geológicas, tanto su techo como su base (Wikipedia).

Figura 4.48: Levantamiento batimétrico (cortesía www.shoa.cl).

El abanico de ondas sonoras formado por los sonares de barrido ancho permite una resolución y precisión muy altas. En general, aunque depende de la profundidad, permite a un buque cubrir mucha más superficie del fondo marino que a base de mediciones individuales.

4.21 marpap

El paquete **marmap** (Pante and Simon-Bouhet [2013]) está diseñado para *descargar*, *trazar* y *manipular* **datos batimétricos** y **topográficos** en R.

Puede consultar la base de datos de batimetría y topografía ETOPO1 alojada por la NOAA (Amante and Eakins [2009]).

El paquete ofrece funciones para consultar datos (batimetría, información de muestreo, etc.) que están disponibles de forma interactiva haciendo clic en los mapas de marmap. Los datos batimétricos y topográficos también se pueden utilizar para calcular las áreas de superficie proyectadas dentro de los intervalos de profundidad / altitud especificados, y restringir el cálculo de distancias de trayectoria más cortas realistas (Pante and Bouhet [2015]).

4.21.1 Descarga de Datos Batimétricos

La función principal para obtener la data es `getNOAA.bathy()` que consulta la base de datos ETOPO1 alojada en el sitio web de la NOAA[9], dadas las coordenadas del área de interés y la resolución deseada.

[9] www.ngdc.noaa.gov

Los usuarios tienen la opción de escribir directamente los datos descargados en un archivo (`keep=TRUE`) con el objeto de si se realiza una consulta idéntica (es decir, utilizando resolución y latitud idéntica), *getNOAA.bathy()* cargará datos del archivo escrito previamente en el disco en lugar de consultar la base de datos NOAA.

Este comportamiento debe usarse preferentemente por:

1. Para reducir la cantidad de consultas innecesarias al sitio web de la NOAA.

2. Para reducir el tiempo de carga de datos.

Si el usuario desea realizar varias consultas idénticas al sitio web de la NOAA sin cargar los datos escritos en el disco, el usuario debe modificar el nombre del archivo de datos. La estructura de los archivos es compuesto por una cabecera *marmap_coord*, las coordenadas que definen el área, el detalle de la resolución y la extensión del archivo '.csv'. Un ejemplo basado en nuestro próximo ejercicio:

marmap_coord_-92.724609;-2.88211;-86.616211;2.722527_res_1.csv

Alternativamente, el archivo de datos se puede mover fuera del directorio de trabajo actual y así preservarlo. R provee una serie de funciones para el trabajo con archivos, entre los más importantes podemos destacar:

```
file.create(..., showWarnings = TRUE)
file.exists(...)
file.remove(...)
file.rename(from, to)
file.append(file1, file2)
```

Como primer paso debemos instalar el paquete, recomendamos realizarlo siempre en la **consola** y luego cargar en memoria el paquete requerido con *library()* en el inicio del script.

El área de estudio se define por las coordenadas **geográficas** extremas, longitud y latitud mínimas y longitud y latitud máxima y un valor de *resolución* (entero entre 1 y 4) que define el tamaño de la grilla en minutos a descargar.

Primero vamos a definir el área geográfica de estudio y definir el sistema de proyección donde llevaremos nuestros datos y productos gráficos:

```
utm16 <- crs("+init=epsg:32716")
marco <- extent(c(-92.72461, -86.61621, -2.88211, 2.722527))
```

Se recomienda usar la técnica de 4.3.2 para definir interactivamente la zona.

Y opcionalmente la carpeta `path`, para almacenar los datos descargados (solo si `keep=TRUE`):

```
bati_galapagos <- getNOAA.bathy(lon1 = marco[1],
                                lon2 =  marco[2],
                                lat1 = marco[3],
                                lat2 = marco[4],
                                resolution = 1,
                                keep = TRUE,
                                path = here("raw"))
```

El objeto esta definido como una clase especialmente diseñada para el paquete:

```
summary(bati_galapagos)
```

```
Bathymetric data of class 'bathy', with 366 rows and 336 columns
Latitudinal range: -2.9 to 2.7 (2.9 S to 2.7 N)
Longitudinal range: -92.7 to -86.6 (92.7 W to 86.6 W)
Cell size: 1 minute(s)

Depth statistics:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  -3860   -3063   -2464   -2335   -1988    1581

First 5 columns and rows of the bathymetric matrix:
                   -2.9 -2.88328358208955 -2.8665671641791
-92.7              -3585             -3567            -3544
-92.6832876712329 -3573             -3553            -3523
-92.6665753424658 -3562             -3538            -3505
-92.6498630136986 -3551             -3528            -3491
-92.6331506849315 -3543             -3518            -3481
                  -2.84985074626866 -2.83313432835821
-92.7                       -3524             -3515
-92.6832876712329           -3496             -3482
-92.6665753424658           -3473             -3453
-92.6498630136986           -3454             -3430
-92.6331506849315           -3442             -3414
```

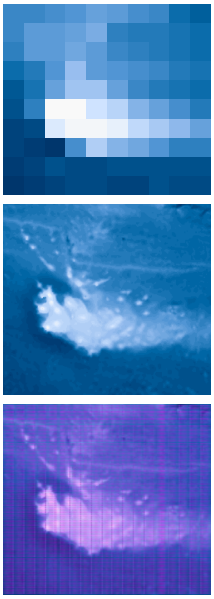


Figura 4.49: Es importante elegir bien parámetros nlon y nlat en comando griddify. 10x10 (arriba), 250x250 (centro) y 500x500 (inferior). Color magenta indica valores NA.

Convertimos a clase de objeto *data.frame* (objeto estándar R) que permite aplicar funciones directamente:

```
bati_xyz <- fortify.bathy(bati_galapagos)

head(bati_xyz)
```

```
          x   y     z
1 -92.70000 2.7 -2512
2 -92.68329 2.7 -2514
3 -92.66658 2.7 -2528
4 -92.64986 2.7 -2534
5 -92.63315 2.7 -2513
```

```
6 -92.61644 2.7 -2492
```

A continuación, removemos los datos con elevación o cota mayor a cero convirtiendo en registros NA:

```
bati_xyz[bati_xyz$z > 0,] <- NA
```

Ahora contamos con una lista de puntos coordenados con *longitud*, *latitud* y *elevación* (*negativo* para valores bajo cero y NA sobre el nivel del mar).

Puede ocurrir que el objeto resulte ser demasiado grande o pesado para su manejo óptimo. Algunos consejos en el punto 4.23.4.

El siguiente paso es convertir ese conjunto de datos xyz espaciados irregularmente y transformarlos en un objeto ráster con dimensiones especificadas por el usuario `griddify`.

griddify() se basa en varias funciones del paquete ráster, especialmente `rasterizer()` que transfiere valores asociados a objetos vectoriales a celdas ráster y transfiere dichos valores de las celdas a otro sistema coordenado o nuevo tamaño de celda con `resample()`.

Si una celda no contiene ningún valor de profundidad o altitud en la matriz xyz se agrega un NA en esa celda. Luego se aplica una *interpolación bilineal* para completar la mayoría de las celdas que faltan.

En el caso que una celda llegue a contener más de un valor de profundidad o altitud en la matriz xyz se calcula el valor medio.

La resolución final del modelo ráster son los parámetros número de columnas `nlon` y número de filas `nlat`. Se debe considerar la resolución de descarga, el número y distribución de puntos xyz y el tamaño del área de estudio para definirlos (Figura 4.49).

En la conversión se ignoran los valores NA `na.omit()` para mantener así la tierra emergida sin modelo:

```
bati_raster <- griddify(na.omit(bati_xyz),
                        nlon= 250,
                        nlat= 250)
```

Y graficamos nuestro modelo con una lista de colores prove-

niente de la paleta *Blues* del paquete *colorbrewer* (ver 1.35.2) y un nuevo parámetro muy oportuno del comando *plot* `colNA='magenta'` que iluminan celdas NA (Figura 4.50) que ha sido proyectado a sistema coordenado UTM H16S (ver 3.10.2) y cambio de resolución a $1000 \times 1000\ m$:

```
bati_exten <- projectExtent(object = bati_raster,
                            crs = utm16)
res(bati_exten) <- 1000
bati_reproj <- projectRaster(from = bati_raster,
                             to = bati_exten,
                             crs=utm16,
                             method = "bilinear")
```

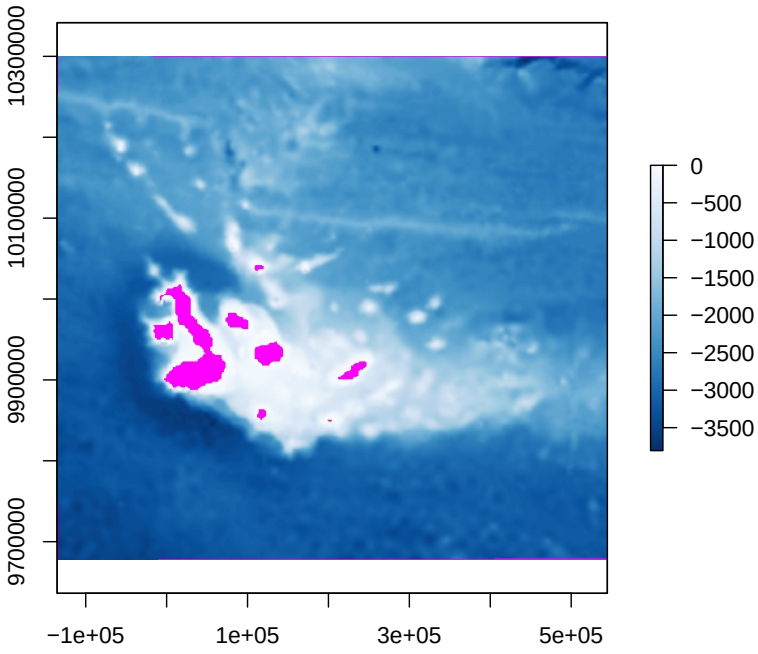


Figura 4.50: Islas Galapagos, UTM H16S. (color magenta indica valores NA).

4.22 Descarga Datos de Relieve

La descarga de data topográfica se realiza usando la extensión del modelo batimétrico `bati_raster` y bajamos el nivel de zoom (en cuadro 4.1 aproximadamente el nivel de zoom 7 corresponde

al pixel de batimetría).

En la figura 4.51 se muestra el modelo descargado y superpuesta la extensión del área de estudio se aprecia que el primero es más extenso.

```
islas_galap <- get_elev_raster(bati_raster,
                               src="aws",
                               z=7,
                               neg_to_na = T)
mdr_crop <- crop(islas_galap,
                 y = bati_raster)
```

Figura 4.51: Modelo de Relieve, zoom 5. Área de estudio destacada en rojo.

A diferencia de las ocasiones anteriores **recortamos primero y luego proyectamos**. Así tenemos un modelo más pequeño sobre el cual realizar las operaciones posteriores con el objeto de optimizar los recursos de proceso.

Con el modelo más pequeño (más liviano) proyectamos para hacer coincidir con el modelo de batimetría (Figura 4.50) **usando el mismo modelo de batimetría** como referencia.

ATENCIÓN: Se usa el **modelo de destino** *'to = bati_reproj'* como referencia para realizar la proyección.

```
mdr_reproj <- projectRaster(from = mdr_crop,
                            to = bati_reproj, #modelo batimétrico!!
                            method = "bilinear")
```

4.23 Fusión MDR - Batimetría

*4.23.1 Usando función **merge()***

`merge()`: *Fusionar objetos ráster* para formar un nuevo objeto ráster con una extensión espacial más grande. Si los objetos se superponen, los valores obtienen prioridad en el mismo orden que los argumentos (se superponen). Los valores NA se ignoran (excepto cuando se define el parámetro `overlap = FALSE`).

Los objetos ráster deben tener **el mismo origen y resolución**. En las áreas donde los objetos ráster se superponen, se

conservan los valores del objeto ráster que está primero en la secuencia de argumentos. Si prefiere usar el promedio de valores de celda o hacer otro cálculo, puede usar `mosaic()` en lugar de *merge*.

```
mdr_final <- merge(x= bati_reproj, y= mdr_reproj, tolerance= 0.25)
```

4.23.2 Usando función ***cover()***

`cover(x, y)`: *Reemplaza valores NA* de un objeto ráster `x` con los valores de otro ráster `y` para formar un nuevo objeto ráster con la misma extensión espacial.

```
mdr_final <- raster::cover(x = bati_reproj, y = mdr_reproj)
```

Figura 4.52: MDR batimétrico (izquierda), topografía (centro) y MDR producto de función 'cover()' con los valores NA en magenta.

4.23.3 Corregir NA

La corrección de posibles valores NA resultantes en cualquiera de los procesos descritos (Figura 4.53, izquierda) se corrige utilizando el promedio de los valores vecinos.

```
mdr_corr <- focal(mdr_final, w=matrix(1, nrow=3, ncol=3),
                  na.rm=T, NAonly=T, fun='mean')
```

Figura 4.53: Errores de Alineación o celdas NA en verde (izquierda). Modelo Digital del Relieve incluye batimetría (derecha).

El MDR creado (Figura 4.53, derecha) será almacenado en disco para su posterior uso en el punto 5.3.5 para la creación de productos cartográficos más refinados®.

```
writeRaster(x = mdr_corr,
            filename = here("raw","bati_galapagos.grd"),
            overwrite=T)
```

4.23.4 Número de Datos muy grande

Los archivos de datos que contienen información batimétrica pueden volverse enormes rápidamente (por ej. decenas a cientos de megabytes en memoria o millones de registros de coordenadas), especialmente si se descargan datos alta resolución registrados en grandes áreas. **marmap** normalmente puede importar archivos xyz tan grandes y crear un objeto **bathy** pero trabajar con tales objetos puede ser difícil (si no imposible) dependiendo de la cantidad de RAM disponible en su computadora (Pante and Bouhet [2015]).

Para revisar el tamaño estimado de la memoria que usa el objeto existe la función `object.size()`:

```
print(object.size(bati_xyz), units="MB")
```

```
2.3 Mb
```

O revisar el número de registros:

```
nrow(bati_xyz)
```

```
[1] 122976
```

En tal caso, se recomienda extraer un subconjunto de datos:

1. seleccionar una región más pequeña de un objeto *bathy* conservando su resolución.
2. Reducir la resolución del objeto *bathy* en toda el área.
3. Una combinación de las 2 primeras, es decir, disminuir la resolución del objeto *bathy* seleccionando un área más pequeña.

Para la opción 1, puede usar `get.box()` o `subsetBathy()` para seleccionar un área más pequeña de un objeto más grande.

`subsetBathy()` permite la selección de un área no rectangular dentro de un gran objeto *bathy* para crear un objeto nuevo más pequeño aunque de la misma resolución. Esta función también tiene un **modo interactivo** para que pueda seleccionar un área de interés haciendo clic en un mapa.

Para la opción 2, no hay una solución incorporada en marmap. Sin embargo, es sencillo disminuir la resolución de un objeto *bathy* ya que es sólo un matriz con una clase especial. Si tienes un gran objeto llamado `data`, una posible solución sería:

Bajar la resolución de los datos por un factor de n:

```
n <- 2
data_baja_res <- data[seq(1, nrow(data), by = n),
                      seq(1, ncol(data), by = n)]
```

Y volver a convertir a clase `bathy`:

```
class(data_baja_res) <- "bathy"
```

`data_baja_res` es ahora un nuevo objeto con una resolución 2 veces menor que el objeto original.

La opción 3 es solo una combinación de los 2 métodos anteriores: primero, crear un objeto de baja resolución y luego usar `get.box()` o `subsetBathy()` para extraer una región más pequeña de él.

V PRODUCTOS CARTOGRÁFICOS

Visión Cartográfica

5.1 Composición

R además de las opciones gráficas básicas que hemos revisado hasta ahora, posee capacidades avanzadas de composición cartográfica.

El comando `plot()` posee el parámetro `add=` que le indica a R que el gráfico actual se debe superponer sobre el último ploteo activo con una transparencia definido en `alpha=` con valor 1 opaco y 0 transparencia total. Así logramos componer un mapa con el sombreado como base y las alturas sobreimpresas.

Usando la misma técnica se superpone el trazado del polígono de la unidad administrativa; el orden del primer **plot** define la base y los subsiguientes escriben sobre el anterior.

En el siguiente ejemplo incorporamos dos funciones que no han sido revisado hasta ahora: `scalebar()` y `compassRose()`. El primero incorpora una escala gráfica que posee una serie de parámetros de configuración que permiten afinar su presentación y diseño. El segundo pertenece al paquete *sp* que inserta una rosa náutica y posee parámetros básicos de ubicación y tamaño.

Utilizaremos el MDR guardado en disco en el punto 4.18 es cargado en memoria:

```
dtm_cordillera_utm <- raster(here("raw", "dtm_cordillera_utm.grd"))
```

Y calculamos el sombreado:

```
pen <- terrain(dtm_cordillera_utm, opt='slope', unit='radians')
aspec <- terrain(dtm_cordillera_utm, opt= "aspect", unit= "radians")
hill <- hillShade(pen, aspec, angle=35, direction=315)
```

5.1.1 Caso 1, Provincia Cordillera, Chile.

La composición cartográfica se construye agregando capas de dibujo. En la figura 5.1 se dibuja el sombreado, sobre ella la altitud coloreada con un factor de transparencia. Finalmente se incorpora una escala gráfica y un símbolo de norte.

```
#Creación y manejo de colores
rampa <- brewer.pal(n = 9, name = "Greys")
colores <- rev(colorRampPalette(rampa)(100))
colores1 <- sequential_hcl(100, palette = "YlOrRd", rev = T)
#Agregar plot sobre plot
plot(hill,
     legend=F,
     axes=T,
     box=F,
     col= colores,
     asp=1)
plot(dtm_cordillera_utm,
     col=colores1,
     legend=F,
     add=TRUE,
     alpha=0.55,
     asp=1)
scalebar(40000,
         xy=c(340000, 6204000), #coordenadas de inserción
         type='bar', #'line'
         divs=3,
         below="kilometros",
         label= c("0","20","40"),
         adj=c(0.5,-1.25))
sp::compassRose(340000,6330000, cex = 0.75)
```

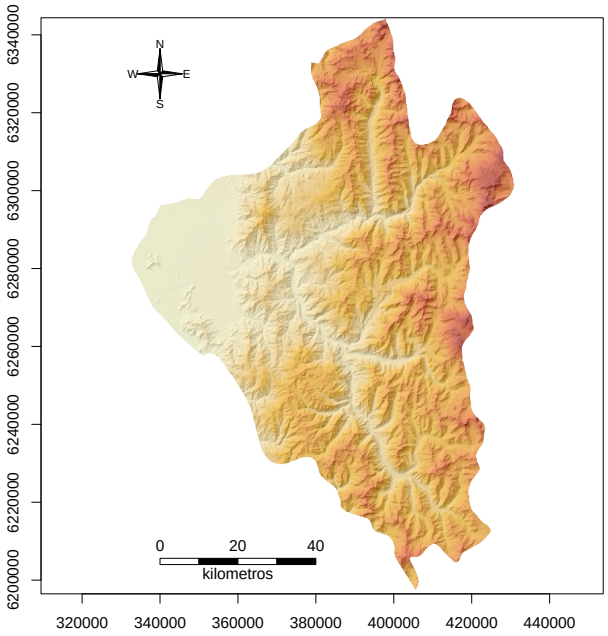


Figura 5.1: Área de estudios, Provincia Cordillera, Chile. Sombreado y Modelo Digital del Relieve.

5.1.2 Caso 2, Mapa Pendiente.

El siguiente ejemplo incluye funciones para agregar curvas de nivel, filtrar una cobertura por valor (pendiente) y aplicar un recorte mediante objeto *extent* para realizar una vista de detalle.

Especial atención requiere el uso del modificador de color (ver 1.36.3) `adjust_transparency()` que incorpora un factor de *transparencia* o (*alpha*) a la paleta de colores permitiendo la superposición de varias capas de gráficos combinando el color de salida.

La composición cartográfica se realiza paso a paso:

1. Configuración de colores.

```
rampa <- brewer.pal(n = 9, name = "Greys")
colores <- rev(colorRampPalette(rampa)(100))
paleta <- colorRampPalette(c("deepskyblue3"))
paleta1 <- colorRampPalette(c("indianred4"))
```

2. Definir el área de interés y recortar por su extensión.

```
ext <- extent(c(380000,390000,6240000,6250000))
hill_clip <- hill[ext, drop=F]
pen_clip <- pen[ext, drop=F]
dtm_clip <- dtm_cordillera_utm[ext, drop=F]
```

3. Preparar la capa de pendiente donde cada menor o igual a 45° (convertidos a radián) asignar como NA y las celdas restantes asignados a 1.

```
pen_clip[pen_clip <= (45 * pi/180)] <- NA
pen_clip[!is.na(pen_clip)] <- 1
```

4. Generar la imagen por la superposición de cada uno de los ploteos.

```
plot(hill_clip, axes= T, legend= F, col= colores, asp= 1, box= F)
plot(pen_clip, add= T, alpha= 0.45, axes= F, col= paleta(1),
     legend= F, asp= 1)
contour(dtm_clip, asp= 1, axes= F, add= T,
        col= adjust_transparency(paleta1(1),0.6),
        nlevels= 20)
scalebar(4000, xy= c(385700, 6240300),
         type= 'bar', divs= 3,
         below= "kilometros",
         label= c("0","2","4"),
         adj= c(0.5,-1.75)
         )
```

La función `contour(x, asp, axes, add, col, nlevels)` del paquete *raster* que genera a partir de un ráster `x` un conjunto de `nlevels` *isolíneas*, trazadas en el color `col`.

5.2 Caso 3, Islas Galapagos

5.2.1 Extrayendo datos con geonames

La base de datos geográfica de **GeoNames** está disponible para descargar de forma gratuita bajo una licencia de atribución de *creative commons*[1]. Contiene más de 25 millones de nombres

[1] **Creative commons** es una organización sin fines de lucro dedicada a promover el acceso y el intercambio de cultura (Wikipedia).

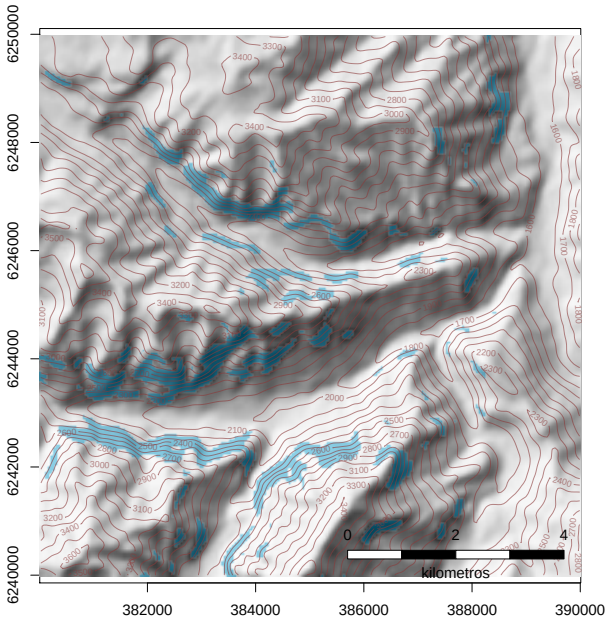


Figura 5.2: Área de estudios, Provincia Cordillera, Chile. Sombreado y Modelo Digital del Relieve se agregan curvas de nivel y destacan celdas con pendiente mayor a 45°.

geográficos y consta de más de 11 millones de características únicas, de las cuales 4,8 millones corresponde a lugares poblados y 13 millones de nombres alternativos.

Se puede acceder a los datos de forma gratuita a través de varios servicios web y una exportación diaria de la base de datos. GeoNames ya está atendiendo hasta más de 150 millones de solicitudes de servicios web por día.

GeoNames está integrando datos geográficos como nombres de lugares en varios idiomas, elevación, población y otros de diversas fuentes. Todas las coordenadas lat/long están en WGS84 (World Geodetic System 1984). Los usuarios pueden editar, corregir y agregar nuevos nombres manualmente utilizando una interfaz wiki fácil de usar.

El paquete *geonames* (Rowlingson [2019]) requiere crear un *id* de usuario registrando en el sitio del proyecto [2] y así tener acceso al servicio web en el cual se pueden consultar datos geográficos globales como *áreas administrativas*, *lugares poblados*, *datos meteorológicos*, etc.

Luego de instalado y cargado en memoria, el primer paso es

[2] http://www.geonames.org/login/

Como primer paso debemos instalar el paquete que recomendamos realizarlo siempre en la **consola** y luego cargar en memoria el paquete requerido con *library(geonames)* en el script.

configurar una variable global con el nombre de usuario registrado:

```
options(geonamesUsername="XXXXXXXXXXX" )
```

El paquete ofrece varias opciones que permiten realizar consultas y recuperar datos, entre los que podemos destacar:

- **`GNcities()`**: Buscar ciudades dentro de un área definida por coordenadas geográficas extremas.
- **`GNfindNearbyPlaceName()`**: Buscar localidades pobladas hasta cierta distancia de la coordenada.
- **`GNfindNearByWeather()`**: Regresa información del clima de la estación más cercana a las coordenadas dadas.
- **`GNearthquakes()`**: Recientes temblores dentro del área geográfica definida por sus coordenadas extremas.
- **`GNwikipediaSearch()`**: Busca términos existentes en la plataforma Wikipedia relacionados con el texto buscado.

Ejemplos de Uso y sintaxis mínima:

```
ciudades <- GNcities(1,-90,-1,-91, lan="es", maxRows = 20)
lugares <- GNfindNearbyPlaceName(-0.1,-90.5,300,maxRows = 50)
clima <- GNfindNearByWeather(-0.1,-90.5)
temblor <- GNearthquakes(1,-90,-1,-91, maxRows = 20)
wiki <- GNwikipediaSearch("Islas Galapagos")
```

Vamos a descargar datos de localidades cercanas y las agregaremos a nuestro mapa, la técnica se puede aplicar a los resultados de los otros comandos detallados anteriormente:

```
lugares <- GNfindNearbyPlaceName(-0.1,-90.5,300,maxRows = 50)
```

Vamos a crear una lista con los **nombres de columnas** y su **posición** (o índice) que componen la data:

```
names(lugares)
```

```
 [1] "adminCode1"            "lng"
 [3] "distance"              "geonameId"
 [5] "toponymName"           "countryId"
 [7] "fcl"                   "population"
 [9] "countryCode"           "name"
[11] "fclName"               "adminCodes1.ISO3166_2"
[13] "countryName"           "fcodeName"
```

```
[15] "adminName1"           "lat"
[17] "fcode"
```

Para revisar el **contenido** de los campos revisaremos los tres primeros registros y así confirmar que sea data que necesitamos rescatar:

```
head(lugares, n=3)
```

```
  adminCode1       lng distance geonameId toponymName
1         01  -90.2718 43.87898   3651163     Seymour
2         01 -90.25289 54.69618   3651449  Santa Cruz
3         01 -90.31713 70.17233   3660250  Bellavista
```

(continuación:)

```
  countryId fcl population countryCode       name
1   3658394   P          0          EC    Seymour
2   3658394   P          0          EC Santa Cruz
3   3658394   P          0          EC Bellavista
```

(continuación:)

```
           fclName adminCodes1.ISO3166_2 countryName
1 city, village,...                     W     Ecuador
2 city, village,...                     W     Ecuador
3 city, village,...                     W     Ecuador
```

(continuación:)

```
       fcodeName adminName1      lat fcode
1 populated place  Galápagos -0.42356   PPL
2 populated place  Galápagos -0.52755   PPL
3 populated place  Galápagos -0.70733   PPL
```

Revisado el contenido se debe crear un vector con los índices de los campos mínimos necesarios e idealmente en el orden *longitud*, *latitud*, *var1*, *var2*, ..., *varn* y usar el vector para crear un subconjunto (ver 1.21):

```
etiquetas <- lugares[,c(2,16, 5,8)]
```

Usando las técnicas de indexado podemos revisar el resultado y confirmar nuestro conjunto final de datos:

```
etiquetas[1:3,]
```

```
        lng      lat toponymName population
```

```
1  -90.2718 -0.42356     Seymour          0
2 -90.25289 -0.52755  Santa Cruz          0
3 -90.31713 -0.70733  Bellavista          0
```

5.2.2 Convirtiendo a SpatialPointsDataFrame

Antes de poder agregar nuestros datos a los mapas debemos convertir el objeto R de clase *geoname* a un objeto de clase **SpatialPointsDataFrame** del paquete *sp* (Pebesma and Bivand [2005], Bivand et al. [2013]) que proporciona clases y métodos para tratar con datos espaciales en R. Las estructuras de datos espaciales implementadas incluyen puntos, líneas, polígonos y ráster; cada uno de ellos con o sin datos de atributos.

Primero vamos a extraer las columnas de coordenadas asegurándonos que el formato sea numérico y creando con ellas un objeto *matrix*, usando los nombres de las columnas en el objeto organizado en el paso anterior.

```
lng <-as.numeric(etiquetas[ ,"lng"])
lat <-as.numeric(etiquetas[ ,"lat"])
coords <- cbind(lng,lat)
```

Luego crear el componente de atributos a partir de las columnas restantes:

```
data  <- etiquetas[, c(3,4)]
```

Y finalmente, asociar el texto estándar del sistema coordenado WGS84:

```
proy  <- crs("+proj=longlat +datum=WGS84 +no_defs")
```

Para integrar todo en el comando que construye el objeto buscado:

```
spdf  <- SpatialPointsDataFrame(coords= coords, data= data,
                                proj4string = proy)
```

Si desea mejorar la calidad del objeto final se pueden cambiar los nombres de los atributos asociados:

```
names(spdf) <- c("nombre", "poblacion")
```

```
spdf
```

```
class       : SpatialPointsDataFrame
features    : 14
extent      : -91.03333, -89.55706, -1.34853, -0.42356  (xmin, xmax, ymin, ymax)
crs         : +proj=longlat +datum=WGS84 +no_defs
variables   : 2
names       :            nombre, poblacion
min values  :       Bellavista,         0
max values  : Tomás de Berlanga,      8996
```

5.2.3 Configuración de Elementos Gráficos: Puntos

Ya tenemos los datos que deseamos agregar a nuestro mapa, para ello debemos configurar los parámetros con los que deseamos darle su forma, tamaño, color entre otras propiedades.

La mayoría de los comandos gráficos (Figura 5.3) comparten los mismos parámetros de configuración:

- **`lwd`**: Grosor de línea.
- **`lty`**: Tipo de línea.
- **`cex`**: Tamaño de punto y funciona no como valor absoluto sino como factor de aumento o reducción del valor por defecto.
- **`col`**: Color de elemento, puede usr el código, el nombre propio, componentes RGB o valores hexadecimales.
- **`pch`**: Forma del punto.

El trazado de elementos gráficos se realiza agregando "capa" sobre "capa" de un elemento base que define el área gráfica disponible, el sistema coordenado y las propiedades físicas del dibujo: tamaño del gráfico, ubicación en el canvas, etc.

Las funciones de trazado deben ser consecutivas e ininterrumpidas. Sino, se debe volver a iniciar la secuencia de dibujo.

Vamos a usar como base el mapa de batimetría construido en el punto 4.21.1 y usaremos una paleta divergente:

```
colores <- diverging_hcl(100, palette = "Vik", rev = F)
plot(bati_raster, col=colores, legend=F, axes=F, box=F)
```

Y agregaremos sobre el mapa los puntos creados en 5.2.2:

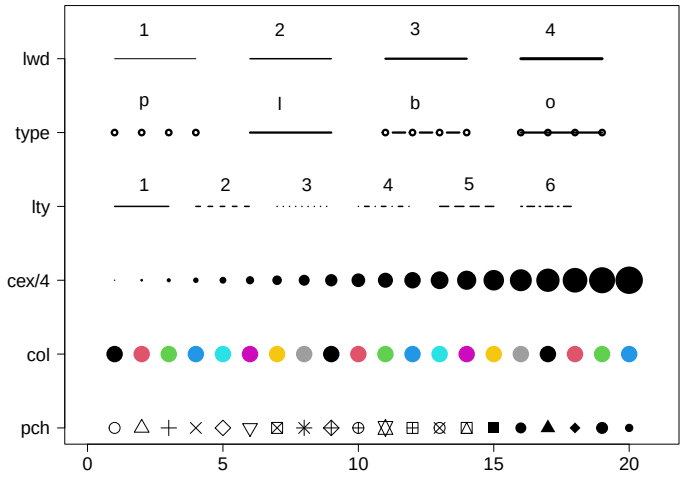


Figura 5.3: Parámetros de Configuración de Gráficos

```
points(spdf, pch=17, col="green", cex=.75)
```

Finalmente, Agregamos los textos utilizando una serie de propiedades específicas para textos (revisar la ayuda en la plataforma RStudio):

```
text(spdf, spdf$nombre, cex=1, pos=3, offset=c(0.25,6),
     family="serif", font=3, halo=T)
```

Resultado de los pasos anteriores:

Un problema es la superposición de textos y/o símbolos, producto de la cercanía geográfica, escala del mapa o tamaño de elementos gráficos.

Una de tantas posibles soluciones es detectar aquellos elementos cercanos y darle otras propiedades gráficas, para ello vamos a utilizar la función:

`spDists()` que devuelve una matriz de distancias utilizando métodos de distancia euclidiana o **círculo máximo** que es el círculo resultante de una sección a una esfera por un plano que pase por su centro y la divide en dos hemisferios. Así, la distancia más corta entre dos puntos de la superficie de una esfera siempre será el *arco de círculo máximo que los une* (Wikipedia), si **`x`** es un **objeto espacial** y se encuentra proyectado.

Estimación de las distancias:

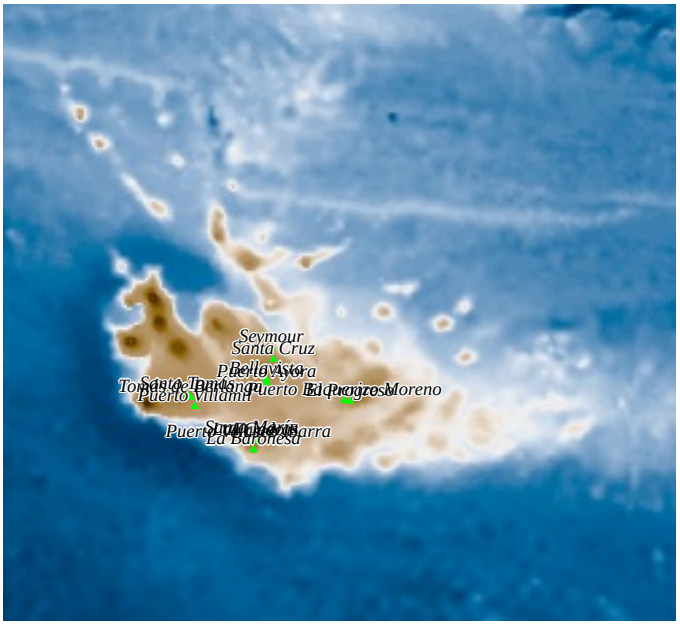


Figura 5.4: Islas Galapagos, Ecuador.

```
dist_matrix <- spDists(spdf)
```

Agregamos los nombres de fila y columnas por los nombres de las localidades para facilitar su análisis:

```
rownames(dist_matrix) <- spdf$nombre
colnames(dist_matrix) <- spdf$nombre
```

De la inspección de los valores podemos seleccionar aquellos que se encuentren más cerca y removerlos del trazado u aplicar otras soluciones.

Cuadro 5.1: Cuadro Parcial Distancias (km).

	Seymour	Santa Cruz	Bellavista
Seymour	0.0	11.7	31.8
Santa Cruz	11.7	0.0	21.1
Bellavista	31.8	21.1	0.0

5.2.4 Límites Internacionales en Alta Resolución

Como ya revisamos en 6.12.1, el paquete *rnaturalearth* (South [2017]) provee un detallado set de datos espaciales para agregar

a nuestros productos cartográficos.

A continuación, descargamos los datos en alta resolución (si no se ha instalado previamente, debe instalar el paquete **rnaturalearthhires** y agregar a la sesión actual con:

`library(rnaturalearthhires)` que agrega la data a la instalación estándar.

```
mundo <- rnaturalearth::ne_countries(scale = "large")
```

Y recortamos (crop) los polígonos usando la extensión (extent) de la capa ráster, la cual es extendida dos filas y columnas en cada dirección (para asegurar una buena cobertura de los datos recortados) mediante el uso de la función `extend(x, y)` que toma el objeto `x` y aumenta usando el parámetro `y` como medida de la extensión.

```
mapa <- crop(mundo,
             extend(extent(bati_raster),
                    y= 2))
```

Finalmente se compone el producto realizando en una sola llamada las capas gráficas.

```
#Preparar los colores
colores <- diverging_hcl(100, palette = "Vik", rev = F)
# Impresión MDR
plot(bati_raster, col=colores, legend=F, axes=F, box=F, asp=1)
# Impresión de Cobertura Poligonal
plot(mapa, col= c("#8B3626"), add=T)
# Impresión de puntos importantes
points(spdf[c(2,3,7,12,14),],
       pch=17,
       col="green",
       cex=.75, asp=1)
#Agregar el texto
text(spdf[c(2,3,7,12,14),],
     spdf[c(2,3,7,12,14),]$nombre,
     cex=1,pos=3,
     offset=c(0.25,6),
     family="serif",
```

```
font=3, halo=T, asp=1)
```

Figura 5.5: Islas Galapagos, Ecuador.Textos Organizados.

5.3 Caso 4, Mejoras Cartográficas

5.3.1 Trabajando Con Cuadrículas

Las cuadrículas son las líneas de longitud y latitud que se muestran en un mapa proyectado, y definir y dibujar estas líneas no ha sido fácil de automatizar.

El paquete R **graticule** (Sumner [2021]) proporciona las herramientas para crear y dibujar estas líneas mediante una serie de parámetros ajustables.

Primero calculamos la extensión del área de estudio (nuestro caso es el MDR desarrollado en 4.18) y realizar los ajustes y configuración de la cuadrícula.

```
#projectamos la extensión del modelo a coor. geográfica
ext <- projectExtent(object = dtm_cordillera_utm,
                     crs = crs("+proj=longlat"))
```

```
#extraemos solo objeto extenaión
(bb <- ext@extent)
```

```
class       : Extent
xmin        : -70.83107
xmax        : -69.7324
ymin        : -34.37077
ymax        : -33.02605
```

Las coordenadas extremas son redondeadas para ajustar los decimales.

```
xini <- round(bb[1], 1)
xfin <- round(bb[2], 1)
yini <- round(bb[3], 1)
yfin <- round(bb[4], 1)
```

Otro aspecto es importante la cantidad de divisiones en ambos ejes, (depende de la 'forma' de la extensión), en nuestro caso vamos a realizar 2x2 cuadrángulos.

```
lons <- seq(xini, xfin, length = 3)
lats <- seq(yini, yfin, length = 3)
```

Para efecto de diseño estético agregamos una 'pestaña' para indicar los rótulos de las coordenadas.

```
xl <- range(lons) + c(-0.01, 0.01)
yl <- range(lats) + c(-0.01, 0.01)
```

Así hemos construido varios vectores, los cuales definen la ubicación y número líneas a trazar:

```
#longitudes, latitudes
lons; lats
```

```
[1] -70.80 -70.25 -69.70
```

```
[1] -34.4 -33.7 -33.0
```

Se generan los vértices para trazar las líneas en el sistema coordenado del MDR con `graticule(lons, lats, proj, xlim, ylim)` para cada longitud contenida en el vector `lons`, latitud `lats` desde las coordenadas extremas `xlim, ylim`.

```
utm19 <- crs("+init=epsg:32719")
grat <- graticule::graticule(lons = lons,
                             lats =  lats,
                             proj = utm19,
                             xlim = xl,
                             ylim = yl)
```

Y la función `graticule_labels()` calcula y compone la posición de los textos para cada línea trazada (Figura 5.6).

```
labs <- graticule::graticule_labels(lons = lons,
                                    lats = lats,
                                    xline = max(xl),
                                    yline = max(yl),
                                    proj = utm19)
```

Así, tenemos trazados y textos de coordenadas geográficas, descritas en coordenadas proyectadas que pueden ser usadas como una capa gráfica adicional a nuestro mapa final.

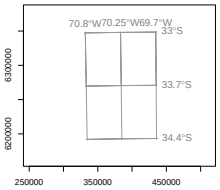


Figura 5.6: Trazado de paralelos y meridianos calculados con 'graticule'.

5.3.2 Distribuyendo Curvas de Nivel

Otro aspecto que vamos a mejorar en este caso práctico, es la creación de curvas de nivel o isohipsas en una secuencia controlada y separar entre primarias y secundarias, como los aspectos gráficos del trazado.

El primer paso es extraer el rango válido de elevaciones del MDR, con ese parámetro se realiza el diseño gráfico de las curvas de nivel, definiendo la **equidistancia** (separación entre curvas sucesivas) en múltiples de 5, 10, 25, 50 o 100 m. Así la legibilidad e interpretación de las magnitudes involucradas en más simple y directa.

```
(zlim <- cellStats(dtm_cordillera_utm, stat = "range"))
```

```
[1]  428.000 6514.183
```

Usando la función `round()` para redondear a magnitudes exactas (idealmente enteros), el parámetro `digits` usa un valor negativo, el redondeo se realiza a la potencia señalada por el índice (por ejemplo -1 a la decena, -2 centena, -3 unidad de mil y así).

```
zini <- round(x = zlim[1], digits = -2)
zfin <- round(x = zlim[2], digits = -3)
```

En función de los valores, se definen los valores de equidistancia, 500 m para las curvas índices o primarias y 100 m para las curvas de nivel secundaria.

```
crv_primaria <- seq(500, zfin, by=500)
crv_secundaria <- seq(zini, zfin, by=100)
```

5.3.3 Símbolos Proporcionales

Por último, vamos a estudiar una forma de representar magnitudes puntuales que sean de fácil lectura.

El tamaño de la representación dependerá de la magnitud de la variable. En términos de datos, la magnitud corresponde proporcionalmente al área de un símbolo del punto.

Por ejemplo, si un valor de un dato es cinco veces el de otro, el área del símbolo de punto será cinco veces más grande. La relación se expresa de la siguiente manera:

$$\frac{\pi r_i^2}{\pi r_{max}^2} = \frac{v_i}{v_{max}} \tag{5.1}$$

Donde r_i es el radio del círculo que se va a dibujar, r_{max} radio del círculo más grande del mapa, v_i valor de la variable por el cual será dibujado y v_{max} el máximo valor de la variable.

Resolviendo para r_i, obtenemos:

$$r_i = \left(\frac{v_i}{v_{max}}\right)^{0.5} \times r_{max} \tag{5.2}$$

Es bien sabido que el área percibida del trazado de símbolos proporcionales no coincide con su *área matemática*; más bien, nos inclinamos a subestimar el área de símbolos más grandes. Como una solución a este problema, es razonable modificar el área de círculos más grandes para que coincida con el *área percibida*.

Flannery [1971] derivó experimentalmente un exponente de función de potencia de 0.5716 para ajustar esta discordancia. Creando una **escala de percepción** (Tanimura et al. [2006]):

$$r_i = \left(\frac{v_i}{v_{max}}\right)^{0.57} \times r_{max} \tag{5.3}$$

Esta solución todavía se cita y se utiliza ampliamente para el mapeo de símbolos proporcionales. Dado que la escala perceptiva ajusta el área de círculos para tener en cuenta la subestimación, el área del círculo más grande en la escala perceptual es mayor que la escala matemática.

En este caso práctico vamos a utilizar información de terremotos obtenidos con la función `GNearthquakes()` del paquete *geonames*.

Para el área de estudio extraído en el punto anterior llamamos la función para rescatar los 100 más recientes terremotos con una magnitud igual o superior a 4,5.

```
temblor <- GNearthquakes(north = bb[3],
                         east = bb[2],
                         south = bb[4],
                         west = bb[1],
                         maxRows = 100,
                         minMagnitude = 4.5)
```

La función devuelve un dataframe con siete variables. El único inconveniente es que a pesar de tener columnas de significado numérico se expresan como caracteres, el primer paso es cambiar de formato.

```
str(temblor)
```

```
'data.frame':   3 obs. of  7 variables:
 $ datetime : chr  "2021-01-24 00:07:09" "2017-08-02 07:26:40" "2020-05-23 13:26:24"
 $ depth    : chr  "110.82" "92" "127.45"
 $ lng      : chr  "-70.2208" "-70.6255" "-69.8723"
 $ src      : chr  "us" "us" "us"
 $ eqid     : chr  "us7000d2s4" "us2000a3js" "us70009p3k"
 $ magnitude: chr  "5.8" "5.4" "5.2"
 $ lat      : chr  "-33.3594" "-33.2117" "-33.5218"
```

Para ello existe el paquete *hablar* (Sjoberg [2020]) contiene una serie de herramientas sencillas para convertir columnas a nuevos tipos de datos. Como funciones intuitivas para columnas con valores perdidos. Luego de instalar el paquete cargamos en memoria con `library(hablar)`. Contiene la función `retype(x)`

convierte el vector o dataframe `x` de carácter a numérico si el total de la columna contiene textos válidos.

```
temblor <- hablar::retype(temblor)
```

```
str(temblor)
```

```
tibble [25 x 7] (S3: tbl_df/tbl/data.frame)
 $ datetime : POSIXct[1:25], format: "2021-01-24 00:07:09" ...
 $ depth    : num [1:25] 110.8 92 127.5 109.3 81.7 ...
 $ lng      : num [1:25] -70.2 -70.6 -69.9 -70.4 -70.5 ...
 $ src      : chr [1:25] "us" "us" "us" "us" ...
 $ eqid     : chr [1:25] "us7000d2s4" "us2000a3js" "us70009p3k" "us20005hz4" ...
 $ magnitude: num [1:25] 5.8 5.4 5.2 5.1 5 4.9 4.9 4.8 4.8 4.8 ...
 $ lat      : num [1:25] -33.4 -33.2 -33.5 -33.9 -33.5 ...
```

Con las columnas convertidas a números podemos, identificando las columnas correspondientes a sus coordenadas, convertir el objeto a un objeto espacial con la función `coordinates(x) <- valor` con un objeto `valor` que representa las coordenadas para cada registro.

```
coordinates(temblor) <- temblor[c("lng", "lat")]
#Es importante asignar el CRS que corresponda
crs(temblor) <- crs("+proj=longlat")
```

Para que coincidan con nuestro mapa proyectamos y cambiamos de CRS.

```
puntos_utm <- spTransform(temblor,
                          CRSobj= crs("+init=epsg:32719"))
```

Utilizando la ecuación 5.3 y basado en el valor de la magnitud del sismo, calculamos un valor de escala para diseñar nuestros símbolos.

```
mag <- temblor$magnitude
scale <- (mag / max(mag) ^ 0.57) * 5
```

5.3.4 Norte Real

El paquete *oce* (Kelley and Richards [2021]) diseñado para el análisis de datos oceanográficos, incluyendo las mediciones 'ADCP', mediciones realizadas con flotadores 'argo', mediciones 'CTD', series cronológicas del nivel del mar, datos topográficos y

costeros, entre otros varios.

Proporciona funciones especializadas para calcular las propiedades del agua de mar, como temperatura potencial en la ecuación de estado 'UNESCO' o 'TEOS-10'. Produce gráficos que se ajustan a las convenciones de *literatura oceanográfica*. Este paquete se analiza ampliamente en el libro de Dan Kelley *Análisis Oceanográfico con R*, publicado en 2018 por 'Springer-Verlag' con ISBN 978-1-4939-8842-6.

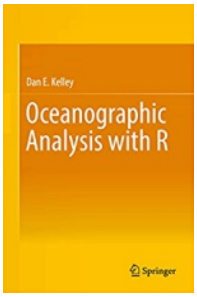


Figura 5.7: Portada del libro 'Oceanographic Analysis with R de Dan Kelley.

Una de las funciones contenidas en el paquete es `magneticField( longitude, latitude, time)` que obtiene el ángulo de declinación magnética para una fecha `time` en la coordenada `longitude`, `latitude`.

Ese ángulo será utilizado para modificar la orientación del símbolo de norte que agregaremos al mapa final.

Creamos el objeto espacial en la coordenada ($long = -70.9,\ lat = -33$).

```
#Creamos el objeto espacial
p <- SpatialPoints(coords = cbind(-70.9,-33),
                   proj4string=crs("+proj=longlat"))
#transformamos al CRS del mapa
ubicar <- spTransform(x = p,
                      CRSobj= CRS("+init=epsg:32719"))
```

Y calculamos el ángulo de declinación para el punto p.

```
decl <- oce::magneticField(longitude = p@coords[1],
                           latitude = p@coords[2],
                           time = Sys.Date())
```

5.3.5 Confección del Plano

Un truco para mejorar el diseño de planos es crear un objeto que extienda el área gráfica, generamos la *extensión* del MDR y *extendemos* su extensión. En este ejemplo 15 km.

```
e <- extend(extent(hill),15000)
```

Creamos los sets de colores para cada capa del mapa final.

```
rampa <- brewer.pal(n = 9, name = "Greys")
colores <- rev(colorRampPalette(rampa)(100))
pal <- colorRampPalette(c("indianred4"))

#Capa base, la extensión del mapa base
plot(e, col="white", xlab="", ylab="")
#Se agrega mapa de sombreado
plot(hill, legend=F, axes=T, box=F, add=T,
     col= colores, alpha= 0.5, asp=1)
#Primer grupo de curvas, secundarias
contour(dtm_cordillera_utm, asp=1, axes=F,
        add=T, col= adjust_transparency(pal(1),0.3),
        levels=crv_secundaria, drawlabels=F,
        method="flattest", lwd=0.25)
#Segundo grupo de curvas
contour(dtm_cordillera_utm, asp=1, axes=F, add=T,
        col= adjust_transparency(pal(1),0.6),
        levels=crv_primaria, labcex=0.45,
        method="flattest", lwd=0.5)
#Símbolos puntuales proporcionales
points(puntos_utm, bg=adjust_transparency(c("#FFAEB9"),0.25),
       col=c("#FF6EB4"), cex=scale * 0.5, pch=21)
#Escribe la magnitud centrado sobre símbolo
text(puntos_utm, labels=temblor$magnitude,
     col=c("#8B3A62"), cex=scale * 0.1)
#Agregamos la informacion de fecha y hora
text(puntos_utm, labels= format(temblor$datetime,
                                format="%d %B %Y\n[%H:%M]"),
     col=c("#8B3A62"), cex= scale * 0.075, offset= 1.1, pos=1)
#Trazado de líneas que definen la cuadrícula
plot(grat, add=T, col = "gray60")
#Rótulos de longitud
text(subset(labs, labs$islon),
     lab = parse(text = labs$lab[labs$islon]),
     pos = 3, col="gray50", cex=0.7)
#Rótulos de latitud
text(subset(labs, !labs$islon),
```

```
     lab = parse(text = labs$lab[!labs$islon]),
     pos = 4, col="gray50", cex=0.7)
#Dibuja escala gráfica
scalebar(20000, xy=c(370000, 6180000),
         type='bar', divs=4, below="kilometros",
         label= c("0","10","20"), adj=c(0.5,-1.5)
         )
#Símbolo de Norte con declinación
sp::compassRose(x = ubicar[1]@coords[1],
                y = ubicar[1]@coords[2],
                cex = 0.75, rot = decl$declination)
```

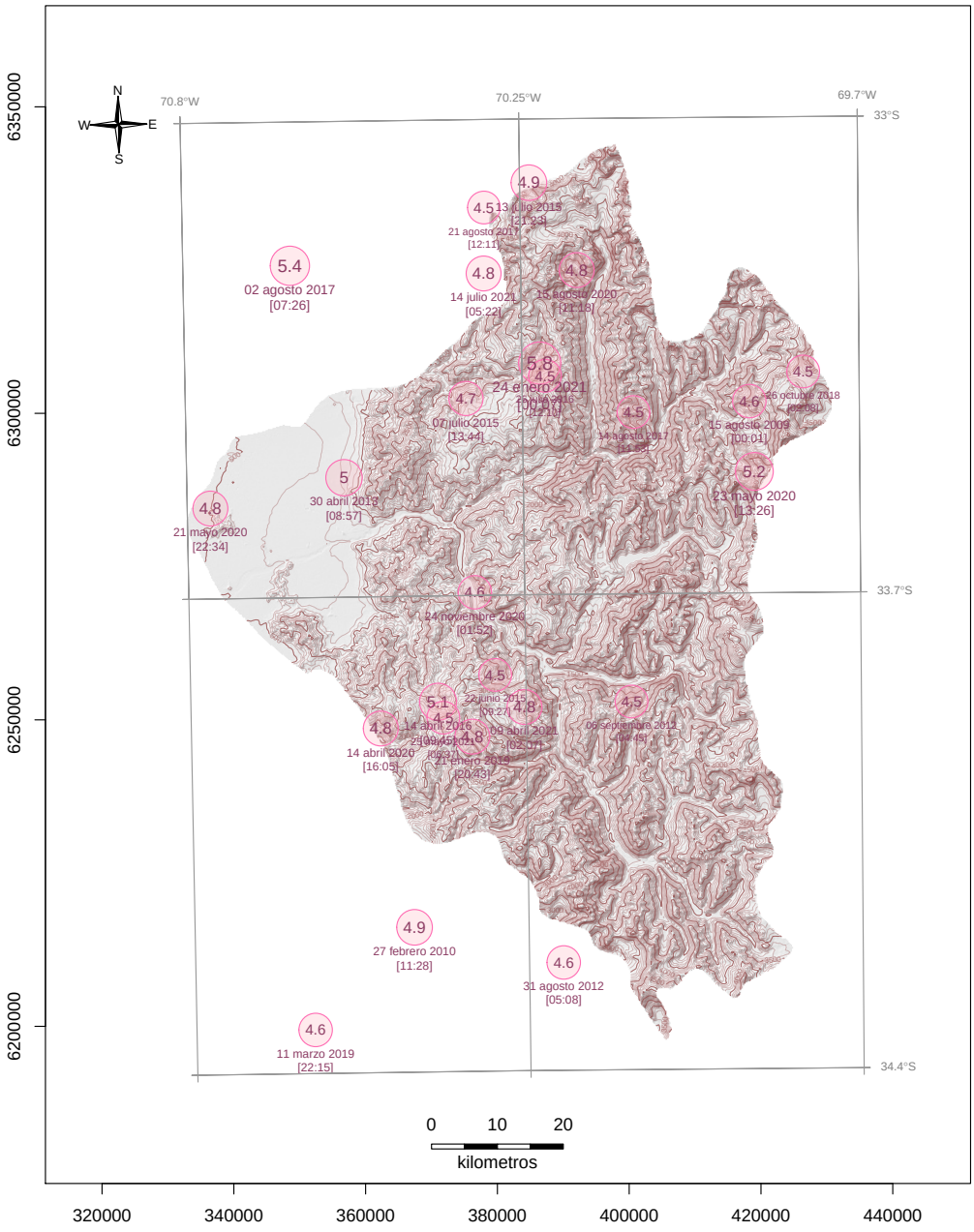


Figura 5.8: Provincia Coordillera, Chile. Se agregan la cuadrícula geográfica, norte con declinación magnética y símbolos graduados.

Gráficos y Productos Adicionales

El modelo digital de relieve y batimetría resultante en 4.23.3 fue grabado en disco y ahora, será utilizado para demostrar las capacidades producción de material gráfico y resúmenes de datos (Figura 5.9).

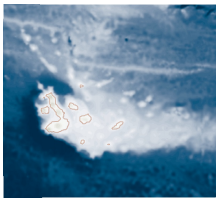


Figura 5.9: Islas Galapagos, UTM H16S.

```
batimetria <- raster(here("RAW","bati_galapagos.grd"))
```

Revisar propiedades básicas del modelo ráster:

```
structure(batimetria)
```

```
class      : RasterLayer
dimensions : 622, 680, 422960  (nrow, ncol, ncell)
resolution : 1000, 1000  (x, y)
extent     : -135323.2, 544676.8, 9677924, 10299924  (xmin, xmax, ymin, ymax)
crs        : +proj=utm +zone=16 +south +datum=WGS84 +units=m +no_defs
source     : bati_galapagos.grd
names      : layer
values     : -3806.533, 1636.28  (min, max)
```

Descripción básica de la data del modelo ráster:

```
summary(batimetria)
```

```
            layer
Min.    -3806.533
1st Qu. -3063.429
Median  -2461.388
3rd Qu. -1984.523
Max.     1636.280
NA's     1392.000
```

5.4 Boxplots

Los **boxplots**, también conocidos como diagramas de cajas y bigotes, son una representación gráfica que permite resumir las

características principales de los datos (posición, dispersión, asimetría, ...) e identificar la presencia de valores atípicos. Fue ideado por John Tukey (Tukey [1977]) de la Universidad de Princeton (USA).

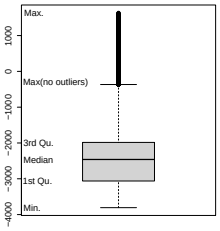


Figura 5.10: Boxplot del modelo batimetria.

La caja de un boxplot comienza en el primer cuartil (25%) y termina en el tercero (75%). Por lo tanto, **la caja representa el 50% de los datos centrales, con una línea dentro que representa la mediana**. A cada lado de la caja se dibuja un segmento con los datos más lejanos sin contar los **valores atípicos** (*outliers*) del boxplot, que, en caso de existir, se representarán con círculos.

Un **dato atípico es aquella observación que está muy distante del resto de los datos**. Se dice que un valor es un valor atípico si es mayor que $Q_3 + 1.5 * IQR$ (valor atípico a la derecha), o es menor que $Q_1 - 1.5 * IQR$ (valor atípico a la izquierda), siendo Q_1 el primer cuartil, Q_3 el tercer cuartil y IQR el rango intercuartil ($Q_3 - Q_1$) que representa el ancho de la caja.

5.5 Histogramas

El **histograma** representa la *distribución de los valores de un conjunto de datos*, en nuestro caso las celdas de un modelo. Esta gráfica es útil para:

1. Identificar valores de datos **atípicos**.
2. Evaluar los valores **mínimo** y **máximo** en sus datos.
3. Explorar la **distribución general** de los valores de elevación en los datos (es decir, si el área es generalmente plana, montañosa, alta o baja).

Un histograma nos muestra cómo se distribuyen los datos. Cada *bin* (contenedor) o barra del gráfico representa el número o la frecuencia de celdas que se encuentran dentro del rango especificado por dicho bin.

El argumento `breaks` permite especificar más o menos bins en el histograma. Tener en cuenta que este argumento no da como resultado *el número exacto de quiebres*, sino los que necesita el histograma.

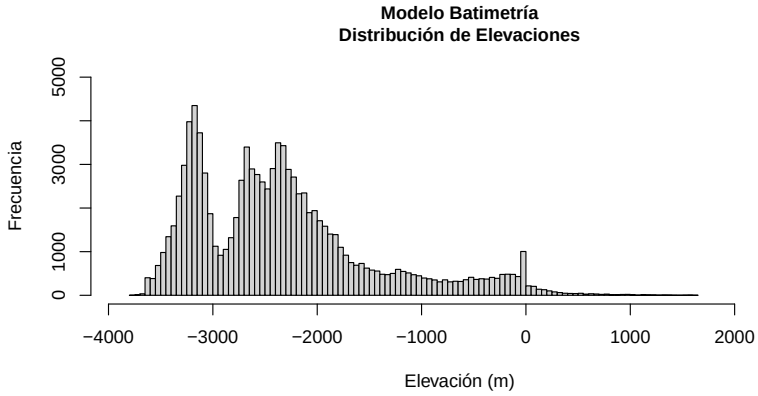


Figura 5.11: Histograma del modelo digital del relieve (incluye batimetría) del Archipielago de las Galápagos.

Vamos a quitar el elemento batimétrico para revisar la estadística sobre el relieve, y para ellos vamos a utilizar el comando `trim()` que ajusta todas las filas y columnas exteriores que solo tengan el valor definido en el parámetro `values`.

Para ello, respaldamos nuestro modelo original, usando técnicas de *subcojunto* (ver 1.21) reemplazamos todos las celdas cuya elevación sea menor a cero por NA. Luego, aplicamos la función *trim*.

```
terreno <- batimetria
terreno[terreno < 0] <- NA
recortado <- trim(terreno, values=NA)
```

Y revisamos propiedades básicas del modelo ráster:

```
structure(recortado)
```

```
class       : RasterLayer
dimensions  : 193, 263, 50759  (nrow, ncol, ncell)
resolution  : 1000, 1000  (x, y)
extent      : -16323.21, 246676.8, 9848924, 10041924  (xmin, xmax, ymin, ymax)
crs         : +proj=utm +zone=16 +south +datum=WGS84 +units=m +no_defs
source      : memory
names       : layer
values      : 0, 1636.28  (min, max)
```

Con la descripción básica de la data del modelo ráster:

```
summary(recortado)
```

```
              layer
Min.        0.00000
```

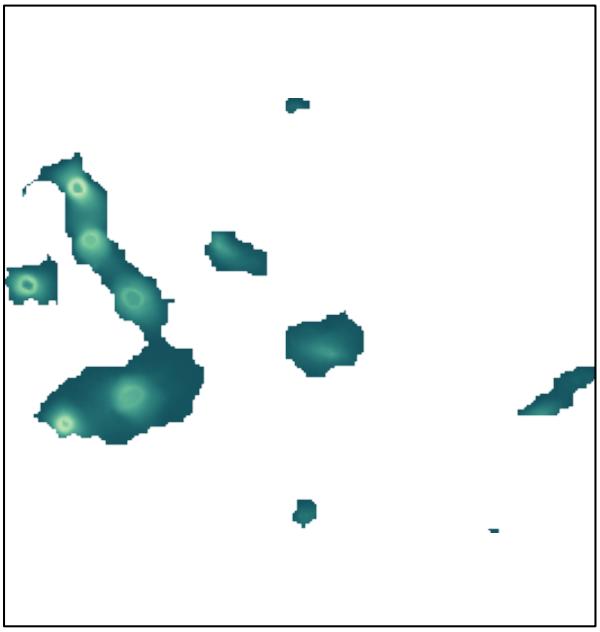


Figura 5.12: Modelo de Relieve. Topografía.

```
1st Qu.     82.49431
Median     211.98929
3rd Qu.    473.63216
Max.      1636.27979
NA's     44907.00000
```

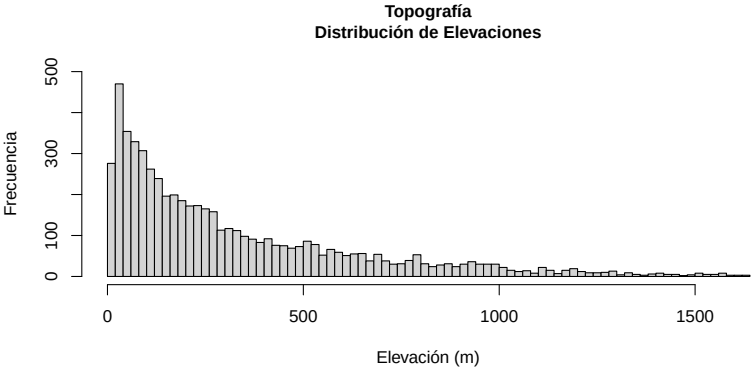


Figura 5.13: Histograma del Relieve del Archipielago de las Galápagos.

Para analizar la variable *pendiente* generamos el modelo en ° y extraemos su histograma:

```
pend <- terrain(batimetria, "slope", "degrees")
```

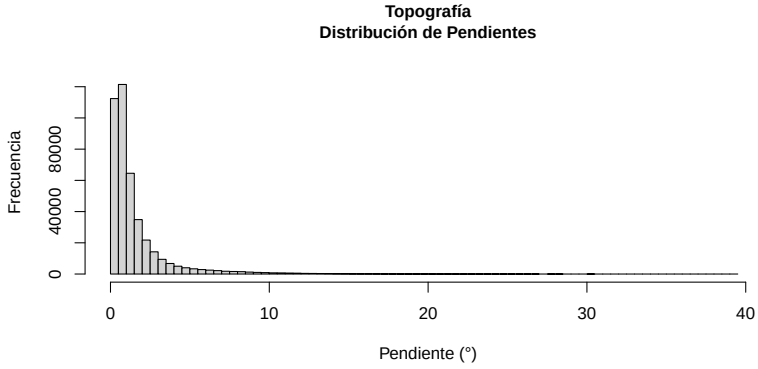


Figura 5.14: Histograma de pendiente (°).

Para la variable *aspecto* también debemos definir la unidad en °:

```
aspecto <- terrain(batimetria, "aspect", "degrees")
```

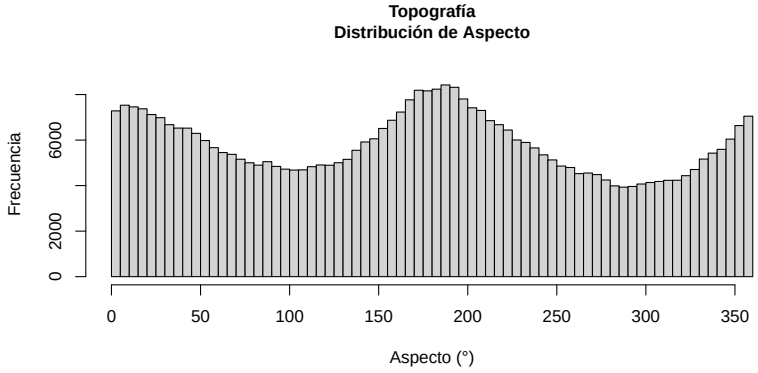


Figura 5.15: Histograma del aspecto (°).

VI APLICACIÓN DE DATOS RÁSTER

POBLACIÓN

6.1 Proyecto Worldpop

Una comprensión del número, características y la ubicación de las poblaciones humanas sustenta el trabajo operativo, análisis de políticas y/o desarrollo científico a nivel mundial en múltiples sectores.

Sin embargo, las fuentes tradicionales de datos de población a menudo están desactualizadas, tienen baja granularidad (detalle) y se actualizan en escalas de décadas.

Nuevas fuentes de datos y los recientes avances metodológicos realizados por el programa WorldPop [1] proporcionan ahora una alta resolución, datos abiertos y contemporáneos sobre la distribución de la población humana, lo que permite la medición precisa de las distribuciones, composiciones, características de la población local, crecimiento y dinámica, a escala nacional y regional.

[1] Visite el sitio web del Proyecto Worldpop: www.worldpop.org

La población humana mundial crece en más de *80 millones al año* y se prevé que alcance la marca de los *10 mil millones en 50 años.*

Se espera que la gran mayoría de este crecimiento se concentre en países de bajos ingresos y principalmente en áreas urbanas. Los efectos de un crecimiento tan rápido están bien documentados, y las economías, el medio ambiente y la salud de las naciones, entre otros, están experimentando cambios significativos.

Los datos contemporáneos de alta resolución sobre la distribución de la población humana y su composición son un requisito previo para la medición precisa de los impactos del crecimiento

de la población, para monitorear los cambios y para planificar las intervenciones.

El proyecto WorldPop se inició en 2013 para unir los proyectos centrados en continentes AfriPop, AsiaPop y AmeriPop, con el objetivo de producir mapas de composición y distribución de población detallados y de libre acceso para toda América Central y del Sur, África y Asia.

Una detallada descripción de los métodos de estimación y cálculo se encuentran en la dirección: www.worldpop.org/methods (Figura 6.1).

En el sitio del proyecto podemos encontrar mapas ráster de resolución variable dependiendo del tema y formato *tif*, estructurados por país, tipo y año (Figura 6.2).

Entre los que podemos destacar:

Modelos ráster por país y años para totales de población (basados en Lloyd et al. [2019]), estructura por sexo y edad (Pezzulo et al. [2017], Tatem et al. [2013]), nacimientos o movimientos migratorios (Figura 6.3).

6.2 Descarga de Datos

Paso a paso para ubicar y descargar la data utilizada. Todo inicia visitando el sitio del proyecto www.worldpop.org.

En la barra de búsqueda escribimos el nombre de un país (en el ejemplo Uruguay) se nos presenta una lista con los datos disponible (Figura 6.4).

Clic en el primer botón disponible **Data & Resources** y accesamos a los detalles de la data disponible (Figura 6.5) y buscamos el botón para iniciar la descarga **Download Entire Dataset/102.39MB** y realizamos un clic derecho sobre él para obtener la dirección de enlace (Figura 6.5).

El enlace de descarga de la base de datos de Uruguay para el año 2020 es:

"`https://data.worldpop.org/GIS/Population/Global_2000_2020/2000/URY/ury_ppp_2000_UNadj.tif`"

Al revisar los enlaces correspondientes a los años siguientes podemos detectar que la estructura de carpeta que varía entre

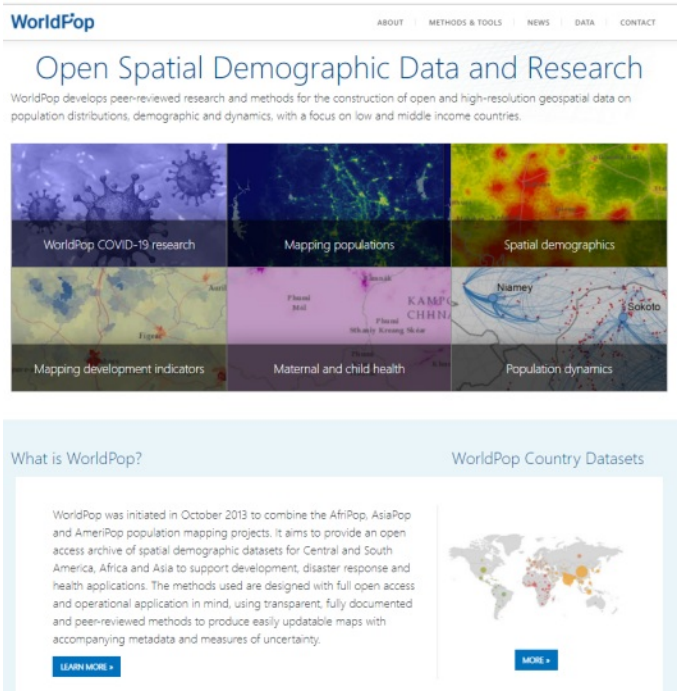


Figura 6.1: Sitio del proyecto WorldPop.

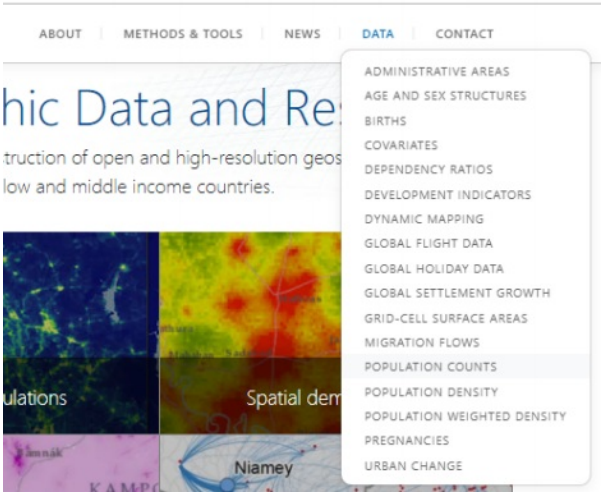


Figura 6.2: Data disponible para descarga y consulta (resaltada en gris la usada en la presente guía).

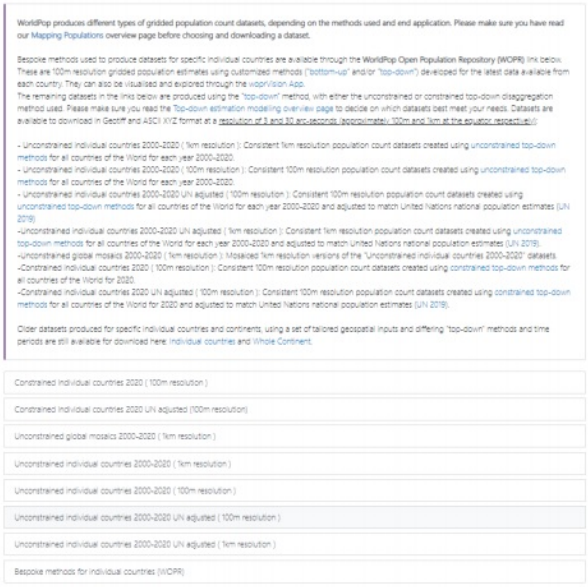


Figura 6.3: Descripción técnica y detalle de los tipos de datos generados y mentenidos (resaltada en gris la utilizada en la presente guía).

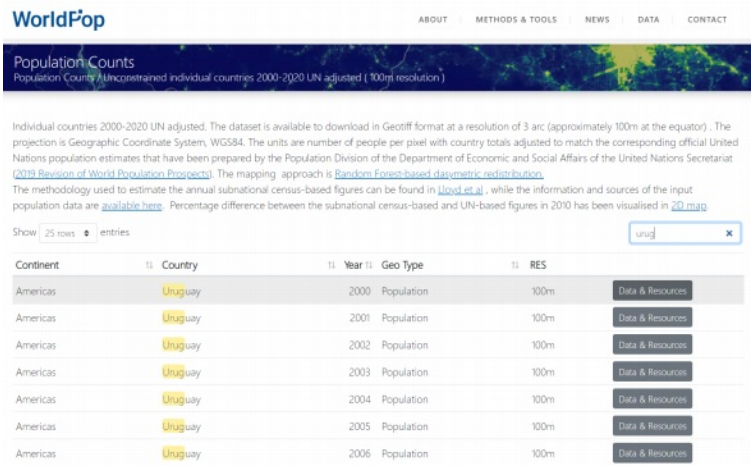


Figura 6.4: Caso Práctico de búsqueda para Uruguay

Figura 6.5: Dónde encontrar el link de descarga (clic derecho).

año y año son dos carpetas; si las separamos en partes fijas y móviles podemos construir una lista de nombres con la siguiente técnica:

```
url <- paste0("https://data.worldpop.org/GIS/Population/Global_2000_2020/",
              2000:2020,
              "/URY/ury_ppp_",
              2000:2020,
              "_UNadj.tif")
```

```
url
```

```
 [1] "https://data.worldpop.org/GIS/Population/Global_2000_2020/2000/URY/ury_ppp_2000_UNadj.tif"
 [2] "https://data.worldpop.org/GIS/Population/Global_2000_2020/2001/URY/ury_ppp_2001_UNadj.tif"
 [3] "https://data.worldpop.org/GIS/Population/Global_2000_2020/2002/URY/ury_ppp_2002_UNadj.tif"
 [4] "https://data.worldpop.org/GIS/Population/Global_2000_2020/2003/URY/ury_ppp_2003_UNadj.tif"
 [5] "https://data.worldpop.org/GIS/Population/Global_2000_2020/2004/URY/ury_ppp_2004_UNadj.tif"
 [6] "https://data.worldpop.org/GIS/Population/Global_2000_2020/2005/URY/ury_ppp_2005_UNadj.tif"
 [7] "https://data.worldpop.org/GIS/Population/Global_2000_2020/2006/URY/ury_ppp_2006_UNadj.tif"
 [8] "https://data.worldpop.org/GIS/Population/Global_2000_2020/2007/URY/ury_ppp_2007_UNadj.tif"
 [9] "https://data.worldpop.org/GIS/Population/Global_2000_2020/2008/URY/ury_ppp_2008_UNadj.tif"
[10] "https://data.worldpop.org/GIS/Population/Global_2000_2020/2009/URY/ury_ppp_2009_UNadj.tif"
[11] "https://data.worldpop.org/GIS/Population/Global_2000_2020/2010/URY/ury_ppp_2010_UNadj.tif"
[12] "https://data.worldpop.org/GIS/Population/Global_2000_2020/2011/URY/ury_ppp_2011_UNadj.tif"
[13] "https://data.worldpop.org/GIS/Population/Global_2000_2020/2012/URY/ury_ppp_2012_UNadj.tif"
[14] "https://data.worldpop.org/GIS/Population/Global_2000_2020/2013/URY/ury_ppp_2013_UNadj.tif"
[15] "https://data.worldpop.org/GIS/Population/Global_2000_2020/2014/URY/ury_ppp_2014_UNadj.tif"
[16] "https://data.worldpop.org/GIS/Population/Global_2000_2020/2015/URY/ury_ppp_2015_UNadj.tif"
[17] "https://data.worldpop.org/GIS/Population/Global_2000_2020/2016/URY/ury_ppp_2016_UNadj.tif"
[18] "https://data.worldpop.org/GIS/Population/Global_2000_2020/2017/URY/ury_ppp_2017_UNadj.tif"
[19] "https://data.worldpop.org/GIS/Population/Global_2000_2020/2018/URY/ury_ppp_2018_UNadj.tif"
[20] "https://data.worldpop.org/GIS/Population/Global_2000_2020/2019/URY/ury_ppp_2019_UNadj.tif"
[21] "https://data.worldpop.org/GIS/Population/Global_2000_2020/2020/URY/ury_ppp_2020_UNadj.tif"
```

Usando la misma técnica podemos construir una lista solo con el nombre de archivo (para realizar la descarga):

```
dest <- paste0("ury_ppp_",2000:2020,"_UNadj.tif")
```

```
dest
```

```
 [1] "ury_ppp_2000_UNadj.tif" "ury_ppp_2001_UNadj.tif"
 [3] "ury_ppp_2002_UNadj.tif" "ury_ppp_2003_UNadj.tif"
 [5] "ury_ppp_2004_UNadj.tif" "ury_ppp_2005_UNadj.tif"
 [7] "ury_ppp_2006_UNadj.tif" "ury_ppp_2007_UNadj.tif"
 [9] "ury_ppp_2008_UNadj.tif" "ury_ppp_2009_UNadj.tif"
[11] "ury_ppp_2010_UNadj.tif" "ury_ppp_2011_UNadj.tif"
[13] "ury_ppp_2012_UNadj.tif" "ury_ppp_2013_UNadj.tif"
[15] "ury_ppp_2014_UNadj.tif" "ury_ppp_2015_UNadj.tif"
[17] "ury_ppp_2016_UNadj.tif" "ury_ppp_2017_UNadj.tif"
[19] "ury_ppp_2018_UNadj.tif" "ury_ppp_2019_UNadj.tif"
```

```
[21] "ury_ppp_2020_UNadj.tif"
```

En R contamos con la función `download.file()` que permite descargar archivos desde internet. Básicamente requiere la *dirección de origen*, el *nombre y dirección de destino*. La url debe iniciar con alguno de los esquemas: `http://`, `https://`, `ftp://`, `file://`.

En nuestro ejemplo vamos a trabajar con 20 archivos, la manera poco eficiente es realizar la operación el mismo número de veces, reemplazando el índice de los vectores. En R (desde el aspecto de lenguaje de programación) posee instrucciones para realizar series de repetición o ciclos de código, entre varios o muchos otros vamos a revisar la sintaxis:

`for(var in seq){expr}`.

Donde:
`var`: El nombre de una variable.
`seq`: Una expresión que retorna un vector.
`expr`: Código R válido.

Nuestra lista de archivos a descargar cuenta con:

```
ncell(url)
```

```
[1] 21
```

Y buscamos descargar desde la dirección web *url* a nuestra carpeta local (definida en *here*) con el nombre *dest*, construimos el ciclo *for*:

```
for(i in 1:21){
    #i será un número entre 1 y 21.
    download.file(url[i], here("raw", dest[i]), mode="wb")
}
```

Dependiendo de la velocidad de descarga de su conexión internet luego de algunos minutos tendrá en la carpeta definida con *here()* los 21 archivos.

Tal como se detalla en 3.4.3 la única forma de construir un objeto compuesto de archivos distintos es *RasterStack*.

```
pop_stack <- stack(here("raw",dest[1:21]))
names(pop_stack) <- as.character(2000:2020)
```

Los nombres asignados tendrán un carácter *x* como prefijo. R lo hace para evitar nombres de variables que inicien con valores numéricos (ver 1.11).

6.2.1 Descripción de Datos

El objeto rasterStack se compone además de cada uno de los rasterLayer de una serie de atributos:

```
attributes(pop_stack)
```

```
$names
 [1] "filename" "layers"   "title"    "extent"   "rotated"  "rotation"
 [7] "ncols"    "nrows"    "crs"      "history"  "z"        "class"
```

Y dentro del atributo `layers` se encuentran almacenados cada uno de los objetos rastLayer que componen el objeto. Para extraer uno de los *layers* o capa o ráster podemos usar una de las alternativas posibles:

```
r <- pop_stack[[1]]
r <- pop_stack$X2000
r <- pop_stack@layers[[1]]
```

```
r
```

```
class      : RasterLayer
dimensions : 5874, 6418, 37699332  (nrow, ncol, ncell)
resolution : 0.0008333333, 0.0008333333  (x, y)
extent     : -58.44292, -53.09458, -34.97458, -30.07958  (xmin, xmax, ymin, ymax)
crs        : +proj=longlat +datum=WGS84 +no_defs
source     : ury_ppp_2000_UNadj.tif
names      : X2000
```

El conjunto de datos que componen el RasterStack considera datos de población de Uruguay para el período 2000-2020 ajustado por la ONU. Los datos se encuentran en formato Geotiff con una resolución de 3 arco segundo (aproximadamente 100 m en el ecuador).

La proyección es Sistema de coordenadas geográficas, WGS84.

La unidad del mapa ráster es **número de personas por píxel** (Figura 6.6, desarrollado con el paquete **tmap** (Tennekes [2018])), con los totales de los países ajustados para que coincidan con las estimaciones oficiales de población correspondientes de las Naciones Unidas que han sido preparadas por la División de Población del Departamento de Asuntos Económicos y Sociales de la Secretaría de las Naciones Unidas[2].

[2] www.un.org/development/desa/pd/

```
tm_shape(r)+
tm_raster(style= "quantile",
          n=9,
          palette= brewer.pal("BuPu", n = 9),
          title= "Pobl.Uruguay(2000)") +
  tm_layout(legend.outside = TRUE) +
  tm_credits("Fuente: Worldpop Project.",
             size=0.6,
             position=c("left","bottom"))
```

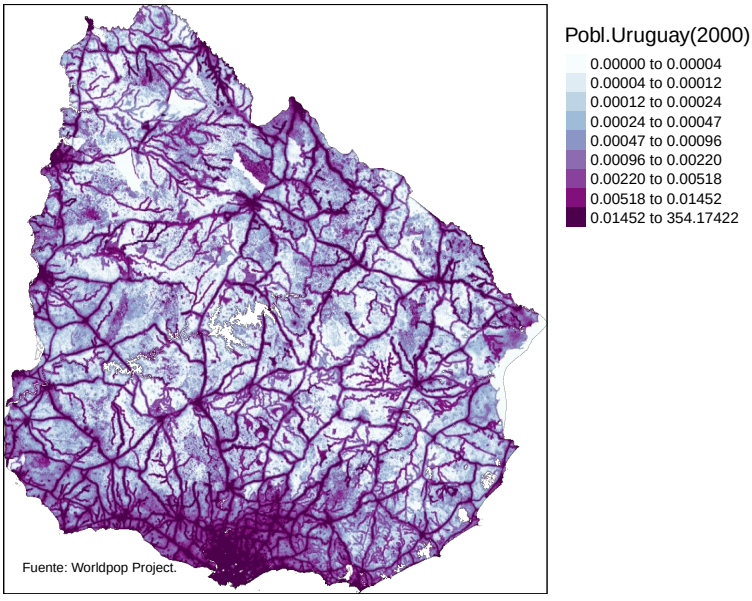


Figura 6.6: Población Uruguay año 2000. Capa Ráster.

6.3 División Política Administrativa

Tal como se describe en 3.13 descargamos la división política administrativa de primer nivel de Uruguay (llamados *distritos*).

```
uruguay <- gadm_sp_loadCountries("URY",
                                 level = 1,
                                 basefile = "RAW/")
```

A continuación, preparamos el objeto final extrayendo la data espacial geográfica, removemos columnas de data que no usaremos y renombramos la primera columna que corresponde a los distritos:

```
uru_geom <- uruguay$spdf
uru_data <- uru_geom[-c(1:3,5:13)] #- borra
names(uru_data)[1] <- "nombre" #renombrar columna

print(uru_data)
```

```
class       : SpatialPolygonsDataFrame
features    : 19
extent      : -58.44272, -53.09425, -34.97403, -30.07969  (xmin, xmax, ymin, ymax)
crs         : +proj=longlat +datum=WGS84 +no_defs
variables   : 1
names       :         nombre
min values  :        Artigas
max values  : Treinta y Tres
```

El mapa se ha desarrollado utilizando el paquete **tmap** (Tennekes [2018]), figura 6.7.

6.4 Población Nacional

1. Primero vamos a extraer desde el stack de años el total de población para el año 2000:

```
(pop_2000 <- cellStats(pop_stack[[1]], 'sum'))
```

```
[1] 3319734
```

2. Y sobre todo el stack para obtener datos globales desde el 2000 al 2020 expresados en millones:

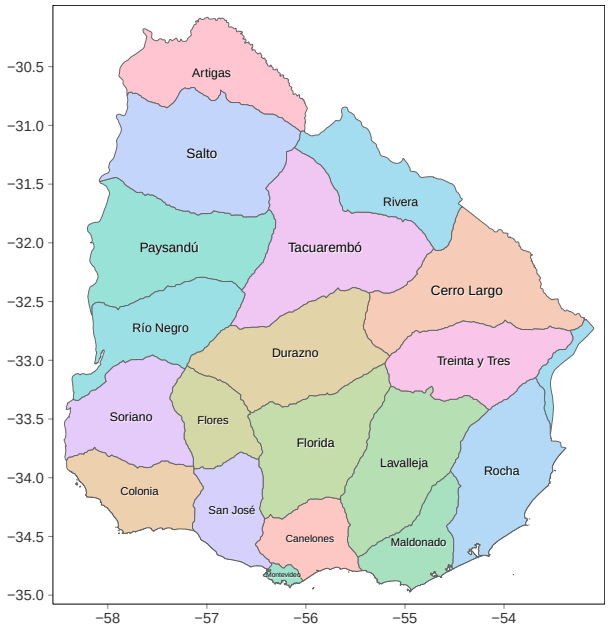


Figura 6.7: Distritos Uruguay. División Político Administrativa.

```
pop_anual_total <- cellStats(pop_stack,'sum')/1e6
```

```
   X2000    X2001    X2002    X2003    X2004    X2005    X2006
3.319734 3.325471 3.326046 3.323661 3.321486 3.321799 3.325403
   X2007    X2008    X2009    X2010    X2011    X2012    X2013
3.331753 3.340221 3.349676 3.359273 3.368926 3.378975 3.389436
   X2014    X2015    X2016    X2017    X2018    X2019    X2020
3.400439 3.412013 3.424139 3.436645 3.449290 3.461731 3.473727
```

6.4.1 Consultas locales

Para consultar directamente al stack por número de celda, coordenadas, fila y columna (ver 3.9) usamos las distintas opciones de consulta.

Por coordenada:

```
extract(pop_stack, cbind(-56.3,-34.82))
```

```
      X2000    X2001    X2002    X2003    X2004    X2005    X2006
[1,] 6.03028 5.094154 5.047142 4.146567 3.660629 3.194571 5.782102
     X2007 X2008    X2009    X2010    X2011    X2012    X2013
```

```
[1,] 4.0812 3.366 3.384968 10.21005 9.746797 9.168478 9.593966
       X2014    X2015    X2016    X2017    X2018    X2019    X2020
[1,] 9.944308 10.50004 7.896178 6.623003 6.579251 7.363348 6.563245
```

Extraer información por número de celda:

(Número calculado con: `cellFromXY(pop_stack, cbind(-56.3,-34.82))`)

```
extract(pop_stack, 36508156)
```

```
      X2000    X2001    X2002    X2003    X2004    X2005    X2006
[1,] 6.03028 5.094154 5.047142 4.146567 3.660629 3.194571 5.782102
     X2007 X2008    X2009    X2010    X2011    X2012    X2013
[1,] 4.0812 3.366 3.384968 10.21005 9.746797 9.168478 9.593966
       X2014    X2015    X2016    X2017    X2018    X2019    X2020
[1,] 9.944308 10.50004 7.896178 6.623003 6.579251 7.363348 6.563245
```

6.5 Población por distrito

Primero debemos instalar el paquete **exact_extract** (Daniel Baston [2021]) `install.packages('exactextractr')` (se aconseja realizar en la consola) y cargar en memoria con `library (exactextractr)` en nuestro archivo script.

El paquete contiene la función **exact_extract()** que extrae los valores de las celdas en un ráster que están cubiertos por polígonos en una colección SpatialPolygonDataFrame, en una fracción o totalidad de cada celda que está cubierta por el polígono. La función puede devolver estos valores directamente o devolver el resultado de una operación de resumen predefinida o una función R definida por el usuario aplicada a los valores.

Devolver extrayendo valores directamente: Si no se especifica `fun=`, *exact_extract* devolverá una lista con un *marco de datos_dataframe* para cada polígono en la colección de la capa.

Operaciones de resumen predefinidas: Los valores de píxeles individuales no siempre son necesarios y solo se requiere una o más estadísticas de resumen (por ejemplo, media, suma) para cada polígono.

Se admiten las siguientes operaciones de resumen:

- **min**: el valor mínimo definido (no NA) en cualquier celda ráster cubierta total o parcialmente por el polígono.
- **max**: el valor máximo definido (no NA) en cualquier celda

ráster cubierta total o parcialmente por el polígono.

- **count**: la suma de fracciones de celdas ráster con valores definidos que no son NA cubiertos por el polígono.
- **sum**: la suma de los valores de celda ráster definidos (no NA), multiplicada por la fracción de la celda que está cubierta por el polígono.
- **mean**: el valor medio de la celda, ponderado por la fracción de cada celda que está cubierta por el polígono.
- **mediana**: el valor de celda mediano, ponderado por la fracción de cada celda que está cubierta por el polígono.
- **quantile**: cuantiles arbitrarios de valores de celda, especificados en *quantiles*, ponderados por la fracción de cada celda que está cubierta por el polígono.
- **mode**: el valor de celda más común, ponderado por la fracción de cada celda que está cubierta por el polígono. Cuando varios valores ocupan el mismo número máximo de celdas ponderadas, se devolverá el valor más grande.
- **majority**: sinónimo de mode.
- **minority**: el valor de celda menos común, ponderado por la fracción de cada celda que está cubierta por el polígono. Cuando varios valores ocupan el mismo número mínimo de celdas ponderadas, se devolverá el valor más pequeño.
- **variety**: el número de valores distintos en las celdas que están total o parcialmente cubiertos por el polígono.
- **variance**: la varianza de la población de los valores de celda, ponderada por la fracción de cada celda que está cubierta por el polígono.
- **stdev**: la desviación estándar de la población de los valores de celda, ponderada por la fracción de cada celda que está cubierta por el polígono.
- **coefficient_of_variation**: el coeficiente de variación de población de los valores de celda, ponderado por la fracción de cada celda que está cubierta por el polígono.
- **weighted_mean**: el valor medio de la celda, ponderado por el producto de la fracción de cada celda cubierta por el polígono y el valor de un segundo ráster de ponderación proporcionado como pesos.

- **weighted_sum**: la suma de los valores de celda ráster definidos, multiplicada por la fracción de cada celda que está cubierta por el polígono y el valor de un segundo ráster de ponderación proporcionado como pesos.

En todas las operaciones de resumen, los valores de NA en el ráster primario X se ignoran (es decir, na.rm = TRUE). Si los valores de NA aparecen en el ráster de ponderación, el resultado de la operación ponderada será NA. Los valores NA tanto en X como en ráster de pesos se pueden reemplazar sobre la marcha utilizando los parámetros `default_value` y `default_weight`.

Funciones de resumen definidas por el usuario: Si ninguna operación de resumen predefinida es adecuada, se puede proporcionar una función R definida por el usuario en `fun=`. La función se llamará una vez para cada característica y debe devolver un solo valor o un marco de datos. Los resultados de la función para cada característica se combinarán y serán devueltas por *exact_extract*.

```
dis_po <- exact_extract(x = pop_stack,
                        y = uru_data,
                        fun="sum")/1e6
```

Agregamos nombre de registros para mantener ordenado nuestro objeto:

```
row.names(dis_po) <- uru_data$nombre
head(dis_po[1:5])
```

```
             sum.X2000  sum.X2001  sum.X2002  sum.X2003  sum.X2004
Artigas     0.08254298 0.08152311 0.08044468 0.07935055 0.07830984
Canelones   0.49313606 0.49803175 0.50207963 0.50574044 0.50921734
Cerro Largo 0.08747583 0.08709592 0.08668951 0.08622784 0.08578623
Colonia     0.12114566 0.12169118 0.12201077 0.12225610 0.12247498
Durazno     0.07071064 0.07039380 0.06988729 0.06931055 0.06875884
Flores      0.02472905 0.02471908 0.02472640 0.02472486 0.02467764
```

Y tenemos nuestro objeto con los distritos por nombre y el total de población por cada año del período 2000-2020.

6.5.1 Densidad por Distrito

Para el cálculo de la densidad por distrito necesitamos contar con el área de cada distrito expresado en km^2.

El proceso es directo para crear una nueva columna, nombrarla y asignarle valores:

```
uru_data$area_km2 <- round(area(uru_data)/1e6, 2)
```

Forma de estimar total área:

```
sum(uru_data$area_km2)
```

```
[1] 177465.6
```

Y el cálculo de la densidad de población es directa:

```
dis_den <- round(dis_po/uru_data$area_km2,2)
```

Y corregimos el nombre de las columnas para facilidad y comodidad de uso:

```
colnames(dis_po) <- sprintf("pob%d",2000:2020)
colnames(dis_den) <- sprintf("den%d",2000:2020)
```

Y finalmente, agregamos las columnas a nuestro objeto principal:

```
uru_data <- cbind(uru_data, dis_po, dis_den)
```

Y finalmente nuestro objeto de polígonos contiene los siguientes datos:

```
names(uru_data)
```

```
 [1] "nombre"   "area_km2" "pob2000"  "pob2001"  "pob2002"  "pob2003"
 [7] "pob2004"  "pob2005"  "pob2006"  "pob2007"  "pob2008"  "pob2009"
[13] "pob2010"  "pob2011"  "pob2012"  "pob2013"  "pob2014"  "pob2015"
[19] "pob2016"  "pob2017"  "pob2018"  "pob2019"  "pob2020"  "den2000"
[25] "den2001"  "den2002"  "den2003"  "den2004"  "den2005"  "den2006"
[31] "den2007"  "den2008"  "den2009"  "den2010"  "den2011"  "den2012"
[37] "den2013"  "den2014"  "den2015"  "den2016"  "den2017"  "den2018"
[43] "den2019"  "den2020"
```

Además de la información relativa a las coordenadas que definen los polígonos, información de la proyección asociada y otros. Cada ítem corresponde a un *slot* de datos (ver 3.5) y podemos

revisar la lista:

```
slotNames(uru_data)
```

```
[1] "data"        "polygons"    "plotOrder"   "bbox"
[5] "proj4string"
```

Y los podemos usar para acceder los datos contenidos en ellos:

```
uru_data[, 1:5]@data
```

```
              nombre area_km2    pob2000    pob2001    pob2002
URY_1         Artigas  9600.25 0.08254298 0.08152311 0.08044468
URY_12      Canelones  4852.64 0.49313606 0.49803175 0.50207963
URY_13    Cerro Largo 13186.99 0.08747583 0.08709592 0.08668951
URY_14        Colonia  6361.95 0.12114566 0.12169118 0.12201077
URY_15        Durazno 12406.89 0.07071064 0.07039380 0.06988729
URY_16         Flores  5153.86 0.02472905 0.02471908 0.02472640
URY_17        Florida 10728.93 0.07068616 0.07052128 0.07023898
URY_18      Lavalleja 11519.21 0.06852002 0.06812036 0.06760733
URY_19      Maldonado  5079.09 0.13229903 0.13523392 0.13802527
URY_2      Montevideo   474.53 1.36095737 1.36133663 1.35942287
URY_3        Paysandú 13346.56 0.10898330 0.10880420 0.10849514
URY_4       Río Negro  8812.75 0.06134086 0.06138376 0.06140478
URY_5          Rivera  9792.02 0.10810921 0.10790988 0.10750380
URY_6           Rocha 10885.87 0.07499932 0.07457375 0.07414183
URY_7           Salto 15852.28 0.12703530 0.12723262 0.12724349
URY_8        San José  5467.80 0.09925859 0.09990505 0.10030549
URY_9         Soriano  8817.19 0.08679124 0.08651615 0.08613352
URY_10     Tacuarembó 15673.51 0.07832621 0.07815319 0.07798021
URY_11 Treinta y Tres  9453.31 0.05446833 0.05424595 0.05378307
```

6.6 Tasa Crecimiento

En demografía, geografía de la población y ecología, la *tasa de crecimiento poblacional* o *tasa de crecimiento demográfico* (PGR de las siglas en inglés: Population growth rate) es la tasa que indica el crecimiento o decrecimiento de la población.

Específicamente, la tasa de crecimiento demográfico se refiere ordinariamente al cambio en la población durante un período expresado a menudo como un porcentaje del número de individuos existentes en un país o lugar a fines de un año sobre la población inicial en el mismo año (ver fórmula 6.1).

Un valor positivo de la tasa del crecimiento indica que la población está *aumentando*, mientras que un valor negativo indica la *declinación* de la población. Un valor cero indica que no existe variación (el mismo número de gente en los dos tiempos medidos).

$$PGR = \frac{x_t - x_{t-1}}{x_{t-1}} * 100 \tag{6.1}$$

Y en R utilizando el objeto dataframe creado en el punto 6.5 podemos calcular:

```
PGR <- (dis_po[,21:2]-dis_po[,20:1]) / dis_po[,20:1] * 100
```

Si deseamos revisar el resultado ordenado por nombre (Figura 6.8):

```
PGR_nombre <- PGR[order(row.names(PGR)), ]

boxplot(t(PGR_nombre),horizontal=F, las=2, cex.axis=.75)
```

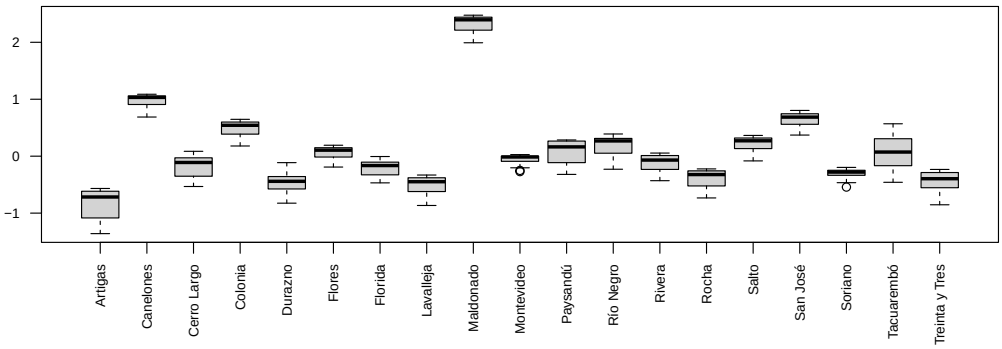


Figura 6.8: Tasa de Crecimiento período 2000-2020 por distritos Uruguay.

O deseamos revisar el resultado ordenado por la media (Figura 6.9):

```
PGR$mean <- rowMeans(PGR[,1:20]) #columna de apoyo

PGR_mean <- PGR[order(PGR$mean,decreasing = F), ]

boxplot(t(PGR_mean),horizontal=F, las=2,cra=1.2, cex.axis=0.75)
```

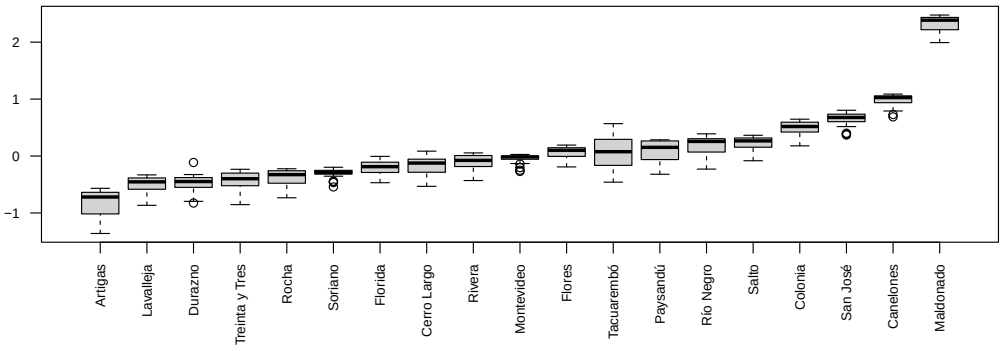


Figura 6.9: Tasa de Crecimiento período 2000-2020 por distritos Uruguay, ascendente.

6.7 *Mapeando la Tasa de Crecimiento*

La visualización en formato de mapa del PGR (6.1), la expresamos como álgebra de mapas (Figura 6.10):

```
uru_data$var_per <- (uru_data$pob2020 - uru_data$pob2000) / uru_data$pob2020 * 100
```

```
tm_shape(uru_data) +
  tm_polygons('var_per',
              palette=qualitative_hcl(20,
                                      palette = "Pastel 1",
                                      rev = F),
              n=7,
              style="jenks",
              midpoint=0,
              title= "Período 2000-2020")  +
  tm_layout(legend.outside = TRUE) +
  tm_text("nombre",
          size = "AREA",
          root =5)
```

6.8 *Pirámides de Población*

Una *pirámide de población* (Figura 6.11) es una forma de visualizar dos variables: **edad** y **sexo**. Son utilizados por demógrafos, que estudian poblaciones. Una pirámide de población es un gráfico que muestra la distribución de edades en una población dividida en el centro entre miembros masculinos y femeninos de

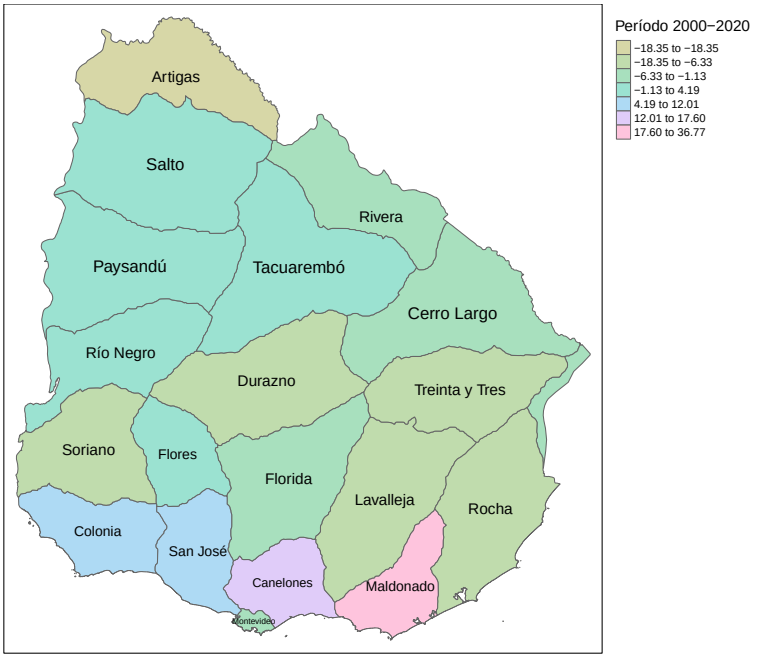


Figura 6.10: Tasa de Crecimiento Poblacional. Por Distritos, Uruguay.

la población. El gráfico comienza desde el más joven en la parte inferior hasta el más antiguo en la parte superior.

Se llama pirámide de población porque cuando la población crece (nacen más bebés que personas mueren), el gráfico tiene la forma de un *triángulo*. Se puede usar una pirámide de población para comparar las diferencias entre las poblaciones masculinas y femeninas de un área. También muestran el número de dependientes económicamente (niños y, en ocasiones, personas mayores) y la *estructura general de la población* en un momento dado.

Existen tres tendencias principales en las poblaciones que afectan la forma de una pirámide de población.

La primera es cuando las tasas tanto de fecundidad y mortalidad son altas entre los miembros más jóvenes. Este tipo de población, conocida como **expansiva**, crea una forma de triángulo agudo en el gráfico. Las pirámides expansivas significan que la población no aumenta mucho en número total y tiene muchos jóvenes.

La segunda tendencia, conocida como **constrictiva**, es cuando hay una tasa de mortalidad más baja y la tasa de fecundidad se mantiene constante. Estas pirámides de población son más anchas en el medio del gráfico, ya que la población tiene un alto número de personas de mediana edad y de edad avanzada, pero menos jóvenes.

La tercera tendencia es **estacionaria** que es una población con baja mortalidad y bajas tasas de fecundidad. Estos gráficos tienen forma cuadrada o pilar en lugar de piramidal. Estas pirámides de población representan una población estable que no cambiará significativamente a menos que se produzcan cambios repentinos en las tasas de fertilidad o mortalidad.

Las pirámides de población son útiles para estudiar el *futuro de una región*, así como para examinar las tendencias históricas y actuales de la población. Si parte de la población se ha visto afectada por cambios repentinos, como víctimas de conflictos armados, alta mortalidad femenina en el parto o la migración de jóvenes trabajadoras de regiones más pobres, el gráfico ofrecerá una forma de visualizar cómo será afectada la población futura. También pueden ayudar a dirigir la distribución de servicios del gobierno y la industria privada para las regiones en función de las necesidades de la población.

6.8.1 Datos por Estructura de Edad y Sexo

En el mismo sitio, además se encuentra información detallada de la estructura de la población y siguiendo los mismos pasos podemos extraer la información y realizar el trazado de pirámides de población.

Los productos para descarga se componen por estimaciones del número total de personas por celda (o píxel) desglosadas por género y grupos de edad (incluidos 0-1 y por 5 años hasta 80+) por años (desde el 2000 al 2020).

En el siguiente ejercicio usaremos la información del año 2016 para Bolivia.

El conjunto de datos está disponible para descargar en formato Geotiff con una resolución de 3 grados de arco (aproximadamente

100 m en el ecuador). La proyección es Sistema de coordenadas geográficas, WGS84.

El contenido de cada celda es el número estimado de hombres/mujeres en cada grupo de edad. El método de cálculo y estimación se presentan en Pezzulo et al. [2017].

6.8.2 Área de Estudio

Descargamos el nivel 3 (Municipios) de Bolivia y creamos nuestro objeto de polígonos en sistema coordenado WGS84.

```
bolivia <- gadm_sp_loadCountries("BOL",
                                 level = 3,
                                 basefile = "RAW/")
bol_geom <- bolivia$spdf
nombres <- bol_geom$NAME_3
```

6.8.3 Descarga de Datos

Luego de revisar los link de descarga, podemos resumir el número y estructura de archivos por edades siguiendo la secuencia:

```
edades <- c(0,1, seq(5,80,by= 5))
edades
```

```
 [1]  0  1  5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80
```

Y construimos el vector con las direcciones de descarga de los archivos tif.

```
urlf <- paste0("https://data.worldpop.org/GIS/AgeSex_structures/Global_2000_2020/",
               2016,
               "/BOL/bol_f_",
               edades,
               "_",
               2016,
               ".tif")
urlm <- paste0("https://data.worldpop.org/GIS/AgeSex_structures/Global_2000_2020/",
               2016,
               "/BOL/bol_m_",
               edades,
               "_",
               2016,
```

```
                ".tif")
```

También se debe construir el nombre de archivos de salida:

```
destf <- paste0("bol_f_",
                edades,
                "_",
                2016,
                ".tif")
destm <- paste0("bol_m_",
                edades,
                "_",
                2016,
                ".tif")
```

Y descargamos utilizando la estructura de ciclo for:

```
for(i in 1:18){
  download.file(urlm[i],
                here("raw", destm[i]),
                mode="wb")
  download.file(urlf[i],
                here("raw", destf[i]),
                mode="wb")
}
```

6.8.4 RasterStack

Con el conjunto de archivos (cada uno con su correspondiente capa) construimos un objeto *RasterStack*:

```
popf_bol_2016 <- stack(here("raw", destf[1:18]))
popm_bol_2016 <- stack(here("raw", destm[1:18]))
```

Y utilizando los polígonos de los municipios extraemos la información de población por sexo y edad contenido en el stack utilizando la función *sum* para el cálculo del total:

```
f_2016 <- exact_extract(popf_bol_2016,
                        bol_geom,
                        "sum")
m_2016 <- exact_extract(popm_bol_2016,
```

```
                                    bol_geom,
                                    "sum")
```

El stack en la posición 1 y 2 contienen los datos de 0-11 meses y de 1 a 4 años, así que vamos a reorganizar para generar un solo grupo de 1 a 4 años y coincida así con los grupos superiores (5 a 9, 10 a 15, etc.):

Sumamos las capas 1 y 2 y guardamos en capa 2:

```
f_2016[2] <- f_2016[1] + f_2016[2]
m_2016[2] <- m_2016[1] + m_2016[2]
```

Se remueve la capa 1:

```
f_2016 <- f_2016[-1]
m_2016 <- m_2016[-1]
```

Y ajustamos el vector con rango de edades removiendo el grupo '0' y ajustamos el contenido del rango superior para que coincida con la data correspondiente:

```
edades <- edades[c(-1)]
edades[17] <- "80+"
```

Finalmente, tenemos una serie de vectores con la información ordenada para los 319 municipios por fila y 17 columnas con los valores de población por rango cada 5 años.

Para generar la pirámide de población del municipio "Tarvita" para el año 2016 buscamos el índice del vector de nombres y con él extraemos las poblaciones desde el objeto *f_2016* para población femenina y *m_2016* para población masculina:

```
id <- which(nombrespop=="Tarvita")
fem <- as.numeric(f_2016[id,])
mas <- as.numeric(m_2016[id,])
titulo <- paste0("Municipio de ", nombrespop[id])

total_f <- as.integer(sum(fem))
total_m <- as.integer(sum(mas))
```

Así tenemos vectores con la población por sexos y por rango etáreos en grupos de 5 años: 1 a 5, 5 a 10, 10 a 15, hasta 80 y más, para cada uno de los 319 municipios de Bolivia.

6.8.5 Paquete pyramid

Luego de instalar el paquete *pyramid* (Nakazawa [2019]) y cargar en memoria podemos trazar nuestras pirámides de población.

```
pyramids(Right = fem,
         Left = mas,
         Center = edades,
         Clab = "Edad",
         Rlab = paste0("Mujeres\n(",total_f,")"),
         Llab = paste0("Hombres\n(",total_m,")"),
         Rcol= "gold",
         Lcol= "skyblue",
         main= titulo)
```

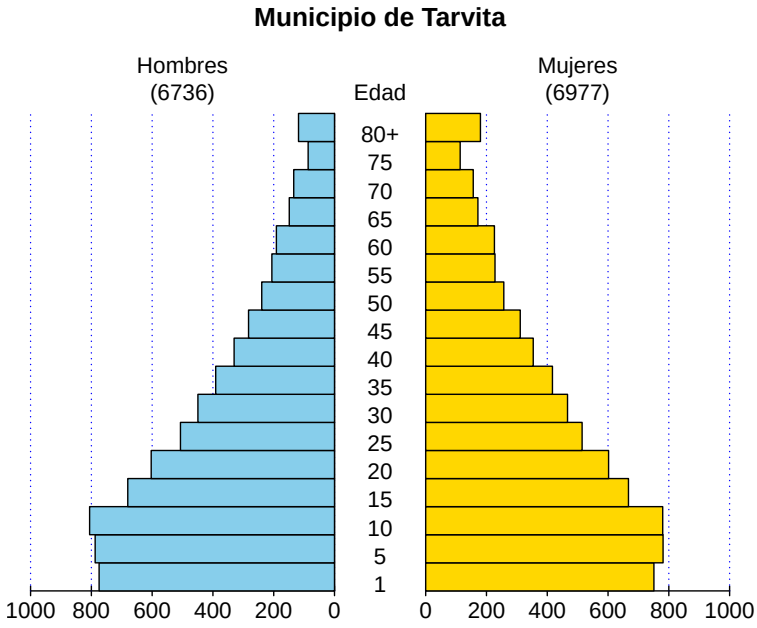


Figura 6.11: Pirámide de Población Municipio de Tarvita, Bolivia 2016.

CLIMA

6.9 El Archivo CEDA

El Archivo CEDA (por su nombre ingles *Centre for Environmental Data Analysis*) forma parte del Servicio de Datos Ambientales (EDS *Environmental Data Service*) del NERC (*Natural Environment Research Council*) y es responsable de cuidar los datos de la investigación atmosférica y de observación de la tierra.

Alberga más de 18 *Petabytes* de datos de modelos climáticos, satélites, aviones, observaciones meteorológicas y otras fuentes.

El enlace directo a la barra de búsqueda está dado por la dirección: catalogue.ceda.ac.uk.

Para acceder a la información primero debemos registrar nuestro nombre de usuario y *logearnos* al sitio y con un simple paso se registra y logea el usuario (Figuras 6.12 y 6.13) para realizar búsquedas de los datos utilizando la barra de consultas (Figura 6.14).

Las variables contenidas en CRU TS4.05 se detallan en el cuadro 6.1.

Y están disponibles para el período de enero de 1901 - diciembre de 2020.

Los datos de CRU TS4.05 se produjeron mediante interpolación de ponderación de distancia angular (ADW).

Los datos CRU TS4.05 son celdas con un valor mensual basados en datos de observación mensuales calculados a partir de datos diarios o subdiarios por los Servicios Meteorológicos Nacionales y otros agentes externos.

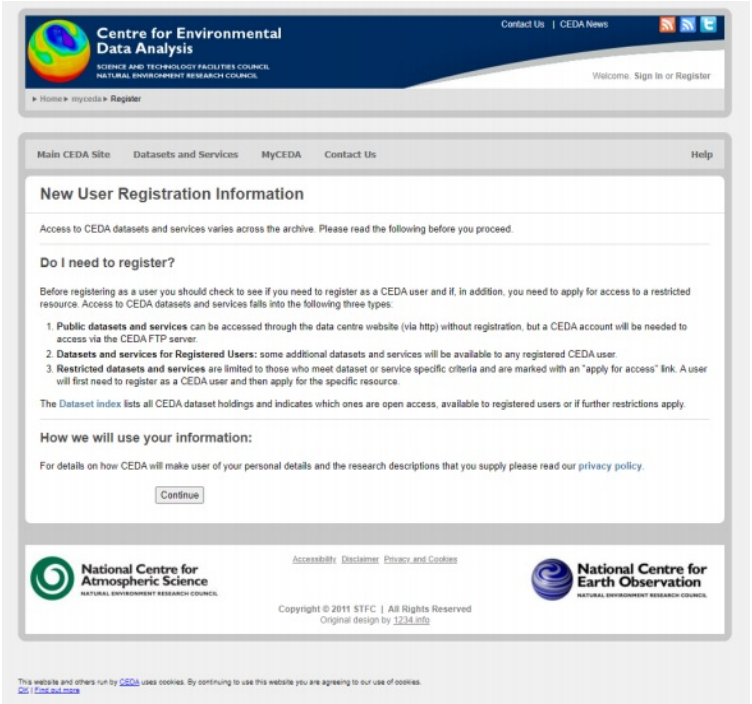


Figura 6.12: Primer paso de registro en el sitio del CEDA.

Tema	Variable
Cloud cover	cld
Diurnal temperature range	dtr
Ground frost frequency	frs
Potential evapotranspiration	pet
Precipitation	pre
Near-surface temperature minimum	tmn
Near-surface temperature	tmp
Near-surface temperature maximum	tmx
Vapour pressure	vap
Wet day frequency	wat

Cuadro 6.1: Nombre de Variables disponibles en sitio web.

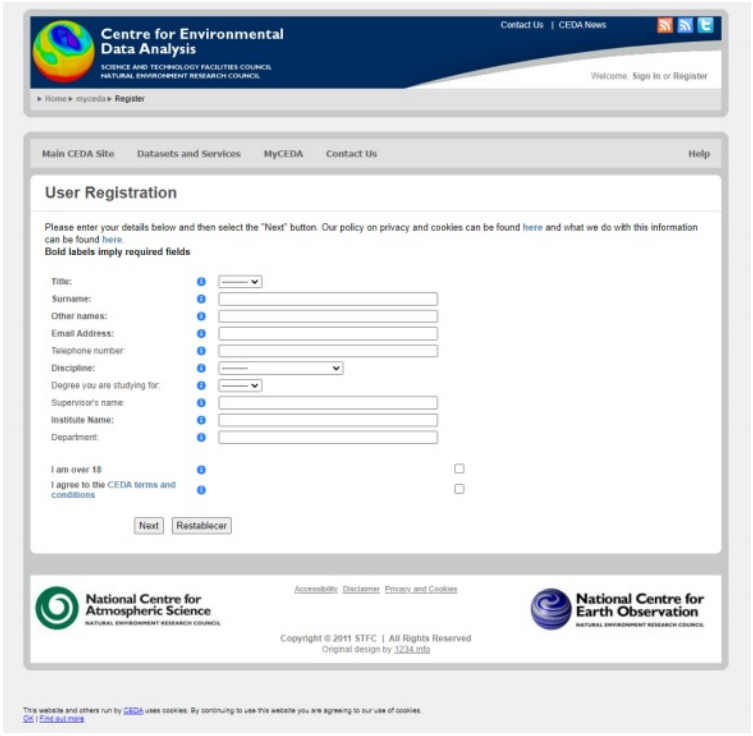


Figura 6.13: Datos que se solicitan para registrar el nombre de usuario.

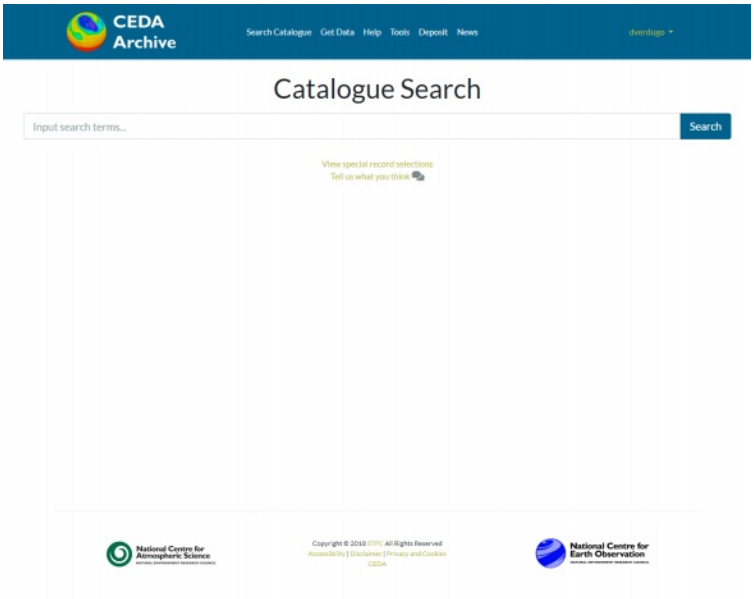


Figura 6.14: Barra de búsqueda del catálogo disponible de datos.

Los archivos de datos ASCII y NetCDF contienen valores medios mensuales para los distintos parámetros. Las versiones de NetCDF contienen una variable entera adicional, “stn”, que proporciona, para cada dato de la variable principal, un recuento (entre 0 y 8) del número de estaciones utilizadas en esa interpolación. El código de valor faltante para ‘stn’ es -999.

6.10 Obtención de Datos

6.10.1 Búsqueda y Descarga

En la barra de búsqueda comenzaremos con los términos **CRU** y **TS** de *Climate Research Unit* y *Time-Series*, los cuales acotan el universo de resultados obtenidos. De todas las versiones localizar la más nueva (en junio de 2021) es la versión CRU TS4.05 (Harris et al. [2020]), figura 6.15.

Figura 6.15: Catálogo más reciente de datos climáticos disponibles.

Hacer clic en la opción **Download** y elegir la opción **data**, elegir la variable a descargar: *pre*, *tmn* y *tmx*.

Dentro de la carpeta *pre* buscaremos el archivo comprimido (.gz) del período 1901-2020 lo mismo en la carpeta *tmn* y *tmx*.

Los archivos tendrán los siguientes nombres (julio 2021), figura 6.16:

- cru_ts4.05.1901.2020.pre.dat.nc.gz
- cru_ts4.05.1901.2020.tmn.dat.nc.gz
- cru_ts4.05.1901.2020.tmx.dat.nc.gz

En esta ocasión hemos descargado manualmente los archivos ya que se requiere previamente estar *logeado* en el sitio y no acepta llamadas directas vía web.

Primero vamos a buscar vía R los archivos con extensión .gz que ubicamos en la carpeta de data cruda *raw*.

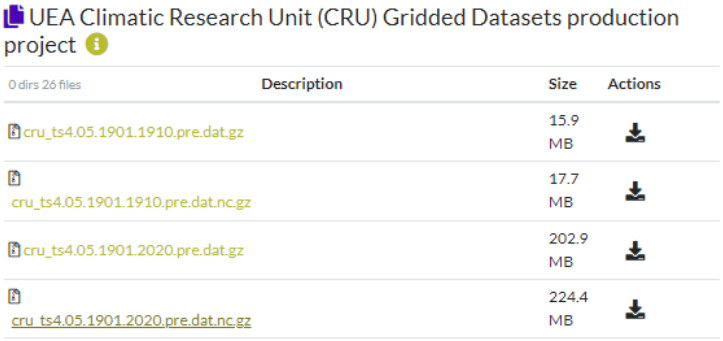


Figura 6.16: Detalle de archivo a descargar (ejemplo de precipitaciones).

```
lst <- list.files(here("raw"),pattern = ".gz")
lst
```

```
character(0)
```

Utilizando el paquete R.utils (Bengtsson [2020]) (luego de instalado y cargado en memoria) podemos usar la función `gunzip()` que comprime y descomprime archivos con formato *gzip* y *bzip2*. El atributo por defecto `remove=T` indica que el archivo ingresado será eliminado.

```
gunzip(here("RAW",lst), remove=T)
```

La función acepta solo un nombre de archivo a la vez, pero tenemos una lista con nuestros nombres y direcciones completas, para estos casos, existe una familia de comandos que permite aplicar una función sobre una lista o vector.

- **lapply**: retorna una lista de la misma longitud que el parámetro de entrada.
- **vapply**: retorna un valor.
- **sapply**: es una versión simplificada y más universal.

La sintaxis básica es `sapply(x, fun, ...)` dónde `x` generalmente es un vector y `fun` la función a ser aplicada a cada elemento de x. La versión *sapply* sería:

```
sapply(lst, function(x) gunzip(here("raw", x)))
```

6.11 Formato NetCDF

Los archivos resultantes se encuentran en el formato **NetCDF** de su sigla en inglés *Network Common Data Form*, "se designa tanto un formato de archivo para el intercambio de datos científicos en arreglos (tablas) como al conjunto de software de código abierto que permite la creación y acceso a los datos. Se trata de un formato para datos binarios, que, por medio de la incorporación de metadatos en el archivo mismo, permite la independencia con respecto al sistema operativo o máquina utilizada para su uso. NetCDF es utilizado para el almacenamiento, proceso y evaluación de grandes cantidades de datos, especialmente en meteorología y ciencias de la tierra." (Wikipedia).

El paquete **ncdf4** (Pierce [2019]) proporciona una interfaz R para los archivos de datos escritos con la biblioteca netCDF de Unidata [3] (versión 4 o anterior), que son archivos de datos binarios que son portátiles entre plataformas e incluyen información de metadatos además de los conjuntos de datos. Con este paquete, los archivos netCDF (ya sea la versión 4 o la versión "*clásica*" 3) se pueden abrir y los conjuntos de datos se pueden leer fácilmente. También es fácil crear nuevas dimensiones, variables y archivos netCDF, en formato de versión 3 o 4, y manipular archivos netCDF existentes.

[3] **UNIDATA** es una comunidad de instituciones de educación e investigación con el objetivo de compartir datos de geociencias, herramientas para acceder y visualizar esos datos. Financiado por National Science Foundation (NSF). www.unidata.ucar.edu.

Que nos permite abrir, examinar y procesar los datos contenidos en ellos:

```
nc_pre <- nc_open(here("RAW",
                       "cru_ts4.05.1901.2020.pre.dat.nc"))
nc_min <- nc_open(here("RAW",
                       "cru_ts4.05.1901.2020.tmn.dat.nc"))
nc_max <- nc_open(here("RAW",
                       "cru_ts4.05.1901.2020.tmx.dat.nc"))
```

No olvidar que luego de usado los archivos se debe cerrar la conexión.

```
nc_close(nc_pre)
nc_close(nc_min)
nc_close(nc_max)
```

También podemos utilizar los mismos comandos (que son modificados en el paquete correspondiente y aquí otra ventaja de R, la capacidad para simplificar el acceso a diversos formatos de datos vía las mismas interfaces o funciones) para leer directamente los datos a un objeto RasterBrick.

```
pre <- brick(here("RAW","cru_ts4.05.1901.2020.pre.dat.nc"),
             varname="pre")

t_min <- brick(here("RAW","cru_ts4.05.1901.2020.tmn.dat.nc"),
             varname="tmn")

t_max <- brick(here("RAW","cru_ts4.05.1901.2020.tmx.dat.nc"),
             varname="tmx")
```

*6.11.1 brick **netCFD**, Estructura Interna*

El conjunto de slots que componen la estructura del archivo es:

```
slotNames(pre)
```

```
 [1] "file"     "data"     "legend"   "title"    "extent"   "rotated"
 [7] "rotation" "ncols"    "nrows"    "crs"      "history"  "z"
```

Para chequear las fechas extremas revisamos el primer y último valor del slot *z*:

```
pre[[1]]@z
```

```
[[1]]
[1] "1901-01-16"
```

```
pre[[nlayers(pre)]]@z
```

```
[[1]]
[1] "2020-12-16"
```

Cada slot tiene su propio contenido estructurado en variables:

```
slotNames(pre@data)
```

```
 [1] "values"     "offset"     "gain"       "inmemory"   "fromdisk"
 [6] "nlayers"    "dropped"    "isfactor"   "attributes" "haveminmax"
[11] "min"        "max"        "unit"       "names"
```

El contenido de *unit* almacena la unidad de los datos:

```
pre@data@unit
```

```
[1] "mm/month"
```

```
t_min@data@unit
```

```
[1] "degrees Celsius"
```

```
t_max@data@unit
```

```
[1] "degrees Celsius"
```

Como primer paso vamos a corregir el nombre de las capas o layer para facilitar los posteriores trabajos de consulta y búsqueda, organizando por *variable_año_mes* (en español).

```
years <- 1901:2020
meses <- c("Enero", "Febrero", "Marzo", "Abril",
           "Mayo", "Junio", "Julio", "Agosto",
           "Septiembre", "Octubre", "Noviembre",
           "Diciembre")
names(pre) <- paste("pre",rep(years, each=12),
                    rep(meses, times=119), sep="_")
names(t_min) <- paste("min",rep(years, each=12),
                      rep(meses, times=119), sep="_")
names(t_max) <- paste("max",rep(years, each=12),
                      rep(meses, times=119), sep="_")
```

Con la data estructurada en el objeto stack podemos realizar fácilmente distintas estadísticas de la data:

Por ejemplo:

1. Precipitación Media Anual 1901 (Figura 6.17).

```
mean(pre[[1:12]])
```

2. Precipitación Acumulada Anual 1901 (Figura 6.18).

```
sum(pre[[1:12]])
```

3. Temperatura Máxima período Enero Marzo 1901 (Figura 6.19).

```
max(t_max[[1:3]])
```

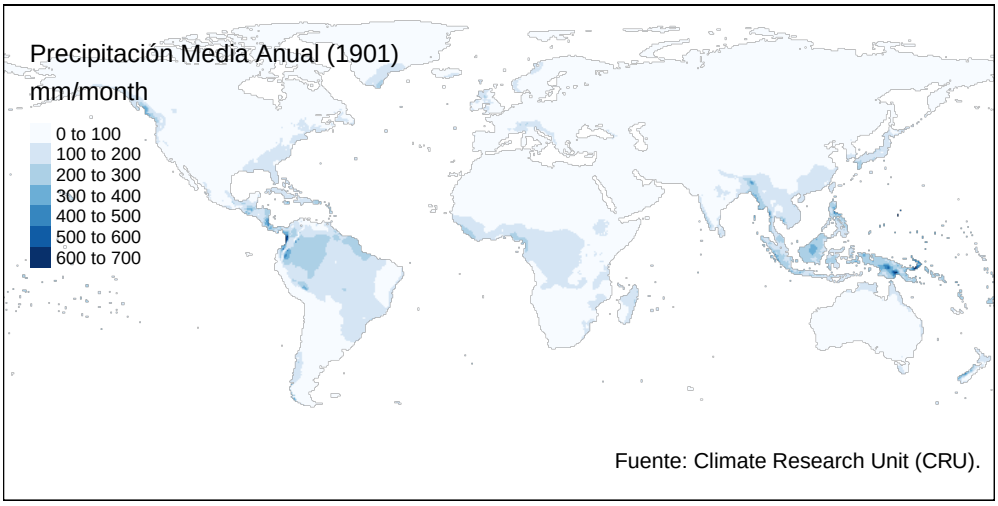


Figura 6.17: Precipitación Media Anual

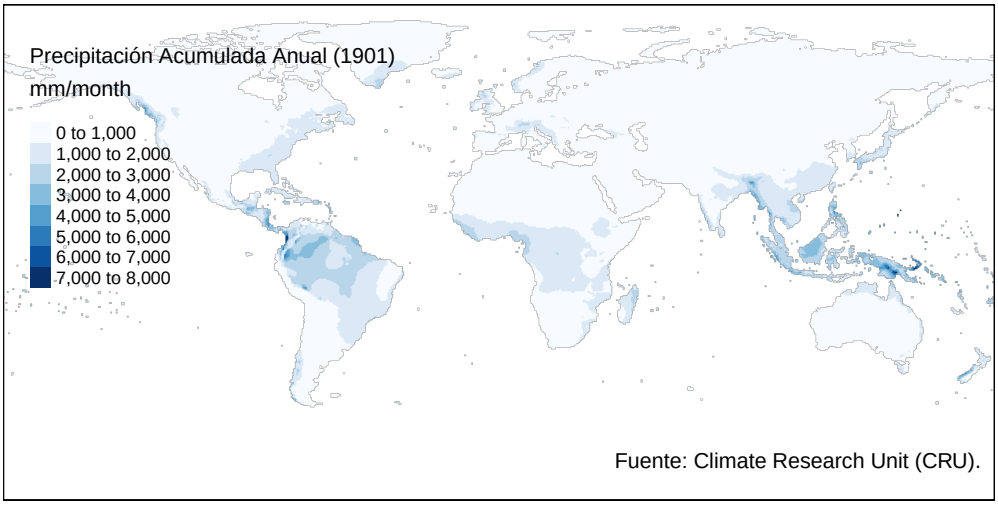


Figura 6.18: Precipitación Acumulada año 1901.

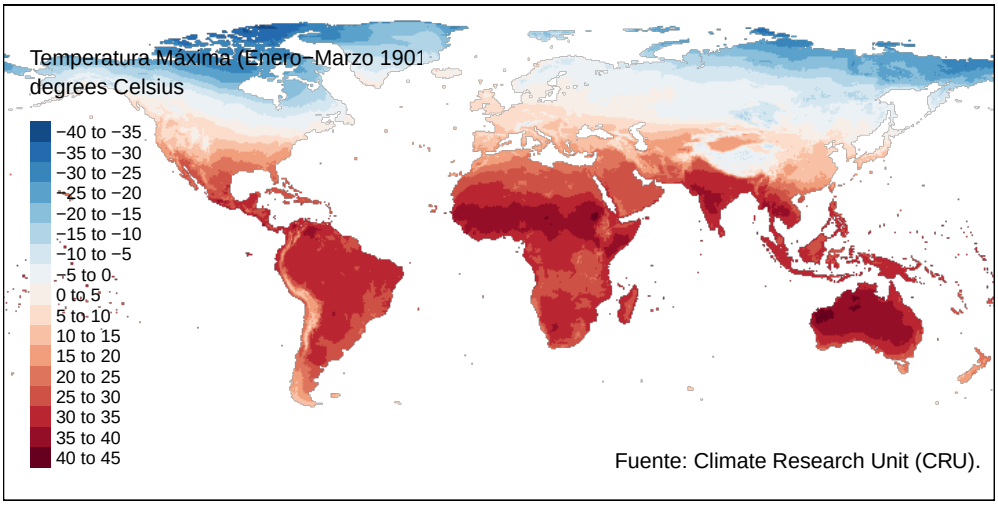


Figura 6.19: Temperatura Máxima Período Enero-Marzo 1901.

6.12 *Extracción de Datos Climáticos*

Primero vamos a definir el área de estudio utilizando información geográfica provista por R y vamos a utilizar un nuevo método a los revisados en 3.10.3.

6.12.1 *Paquete* rnaturalearth

El paquete *rnaturalearth* South [2017] provee una serie de información geográfica provista por el proyecto voluntario *Natural Earth* apoyado por la *Sociedad de Información Cartográfica Norte Americana* (NACIS)[4] compuesta por coordenadas y datos asociados de dominio público disponible a diferentes escalas.

[4] www.naturalearthdata.com

El listado de países se puede extraer con la función `ne_countries()` y el parámetro `returnclass='sp'` define el tipo de objeto devuelto (compatible con el paquete *raster* y sus operaciones gráficas).

```
paises <- ne_countries(returnclass = "sp")
```

Cada polígono tiene los siguientes atributos:

```
names(paises)
```

```
 [1] "scalerank"  "featurecla" "labelrank"  "sovereignt" "sov_a3"
 [6] "adm0_dif"   "level"      "type"       "admin"      "adm0_a3"
[11] "geou_dif"   "geounit"    "gu_a3"      "su_dif"     "subunit"
```

```
[16] "su_a3"      "brk_diff"   "name"       "name_long"  "brk_a3"
[21] "brk_name"   "brk_group"  "abbrev"     "postal"     "formal_en"
[26] "formal_fr"  "note_adm0"  "note_brk"   "name_sort"  "name_alt"
[31] "mapcolor7"  "mapcolor8"  "mapcolor9"  "mapcolor13" "pop_est"
[36] "gdp_md_est" "pop_year"   "lastcensus" "gdp_year"   "economy"
[41] "income_grp" "wikipedia"  "fips_10"    "iso_a2"     "iso_a3"
[46] "iso_n3"     "un_a3"      "wb_a2"      "wb_a3"      "woe_id"
[51] "adm0_a3_is" "adm0_a3_us" "adm0_a3_un" "adm0_a3_wb" "continent"
[56] "region_un"  "subregion"  "region_wb"  "name_len"   "long_len"
[61] "abbrev_len" "tiny"       "homepart"
```

Para definir nuestra área de trabajo como el continente sudamericano vamos a realizar un subconjunto utilizando el campo *continent* (Figura 6.20).

```
continente <- ne_countries(continent = "South America")
```

Figura 6.20: Precipitación Media Anual 1901.

Con el área definida vamos a recortar los modelos de precipitación y temperaturas mínimas y máximas (Figura 6.21).

```
prec_latam <- crop(pre, continente)
tmin_latam <- crop(t_min, continente)
tmax_latam <- crop(t_max, continente)
```

6.12.2 Respaldo en disco (tif)

Usando la misma función para exportar nuestros ráster exportamos a nuestra carpeta *raw* en formato *tif* nuestros RasterStack®:

```
writeRaster(x= prec_latam,
            filename= here("raw","prec_latam.tif"),
            option= c('COMPRESS=LWZ'),
            overwrite=T)
writeRaster(x= tmin_latam,
            filename= here("raw","tmin_latam.tif"),
            option= c('COMPRESS=LWZ'),
            overwrite=T)
writeRaster(x= tmax_latam,
            filename= here("raw","tmax_latam.tif"),
            option= c('COMPRESS=LWZ'),
            overwrite=T)
```

Hay que destacar que la data asociada no se almacena en el archivo tif, por eso un consejo importante es respaldar las fechas de cada uno de los layers o capas del objeto para su uso posterior con el nombre de *fechas.rds*®.

```
saveRDS(prec_latam@z$time,
        here("raw", "fechas.rds"))
```

6.12.3 Estadísticos Básicos

Ahora que tenemos los datos climáticos crudos cargados en R, se abren las posibilidades de múltiples cosas por hacer. Por ejemplo, calcular niveles de precipitación anual o calcular temperaturas medias anuales. También observar tendencias en los datos, como las condiciones climáticas promedio para cada década o zonas geográficas y visualizar estos cambios.

El objeto RasterStack contiene:

```
nlayers(prec_latam)
```

```
[1] 1440
```

Capas o layers que se extienden entre los años 1901 al 2020 y generamos un vector de fechas para usar en el eje x de nuestros gráficos extrayendo la información en formato de *date* (ver 1.31) vía los slots:

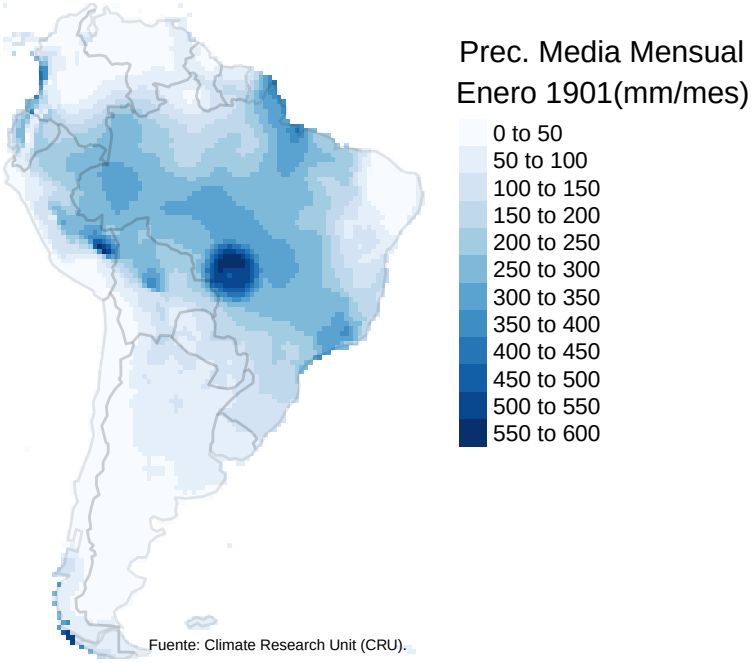


Figura 6.21: Precipitación Media Mensual Enero 1901.

```
fechas <- tmax_latam@z$time
```

Ahora, podemos explorar los valores globales con las funciones que conocemos para los modelos ráster:

1. *Temperatura Máxima Mensual*: Por cada modelo ráster (temperatura media mensual) se extrae el valor máximo contenido:

```
temp_maxima <- cellStats(tmax_latam, 'max')
```

Y graficamos estos datos contra las fechas y agregamos una línea de tendencias suavizada con `lowess()`, función que realiza los cálculos para el suavizador **LOWESS** (Cleveland [1979], Cleveland [1981]) que utiliza regresión polinomial ponderada localmente (Figura 6.22).

```
#suavización
lo <- lowess(x = fechas, y = temp_maxima, f = 0.15)
#gráfico de tipo línea
plot(x = fechas, y = temp_maxima, type="l",
     xlab= "Años",
```

```
     ylab= "Temperatura Máxima Mensual (°C)")
#agregar línea suavizada
lines(x = lo, col="red", lwd=2)
```

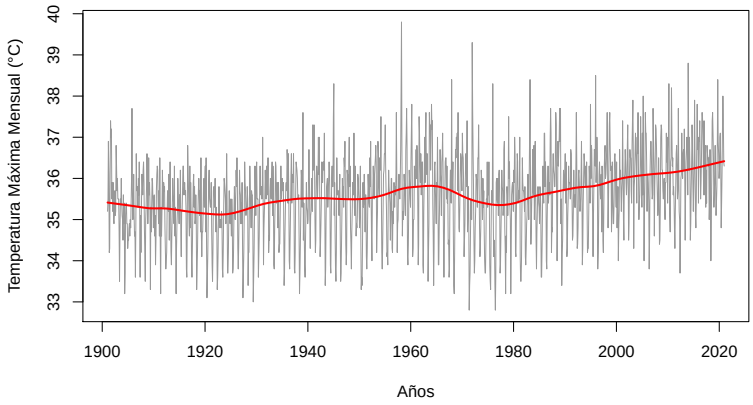


Figura 6.22: Temperatura Máxima Mensual, período Enero 1901 - Diciembre 2020. Suavizamiento Lowess (rojo).

2. *Precipitación Media Mensual*: El objeto *prec_latam* creado anteriormente contiene la precipitación media mensual para el continente, ahora extraemos el valor medio para *toda* la extensión:

```
prec_mean <- cellStats(prec_latam, 'mean')
```

Y graficamos (Figura 6.23):

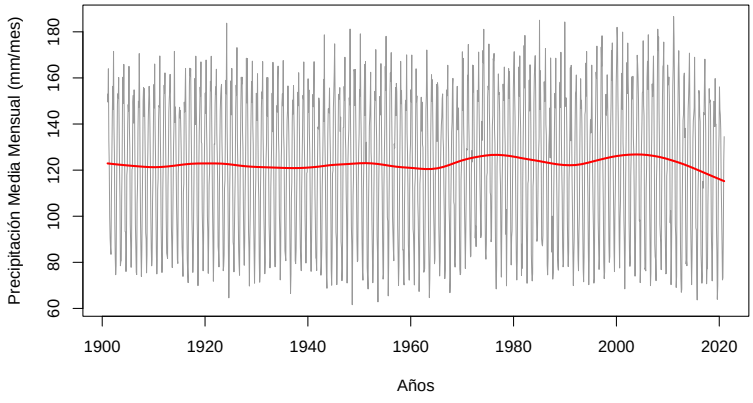


Figura 6.23: Precipitación Media Mensual, período Enero 1901 - Diciembre 2020. Suavizamiento Lowess (rojo).

3. *Temperatura Media Mínima*: Y la temperatura media mínima mensual (Figura 6.24):

```
tmin_mean <- cellStats(tmin_latam, 'mean')
```

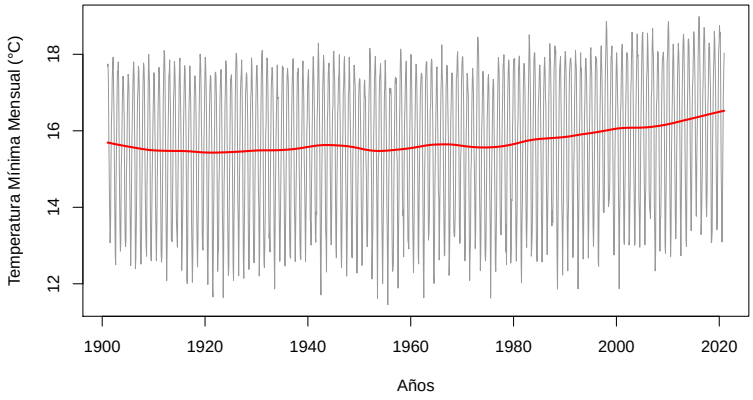


Figura 6.24: Temperatura Mínima Mensual, período Enero 1901 - Diciembre 2020. Suavizamiento Lowess (rojo).

Si necesitamos extraer el valor extremo (mínimo o máximo) de toda la muestra podemos usar la combinación:

```
min(minValue(tmin_latam))
```

```
[1] -15.2
```

También:

```
min(cellStats(tmin_latam, 'min'))
```

```
[1] -15.2
```

6.12.4 Buscar Celdas con Valores Extremos

Una pregunta distinta es ¿qué celdas tienen el valor mínimo o máximo (para un RasterLayer), o qué capa tiene el valor mínimo o máximo (para un RasterStack o RasterBrick)? `which.min()` y `which.max()` devuelven el **índice** de la primera capa que tiene el valor mínimo o máximo para cada una de las celdas (ver 1.15.3).

Por ejemplo, ¿dónde o cómo se distribuyen las temperaturas máximas? (Figura 6.25):

```
maximos <- which.max(tmax_latam)
```

Una forma de buscar índices por valores contenidos en el nombre de las capas o layers es `which()`, por ejemplo deseamos encontrar los índices que definen las capas del verano de 1948:

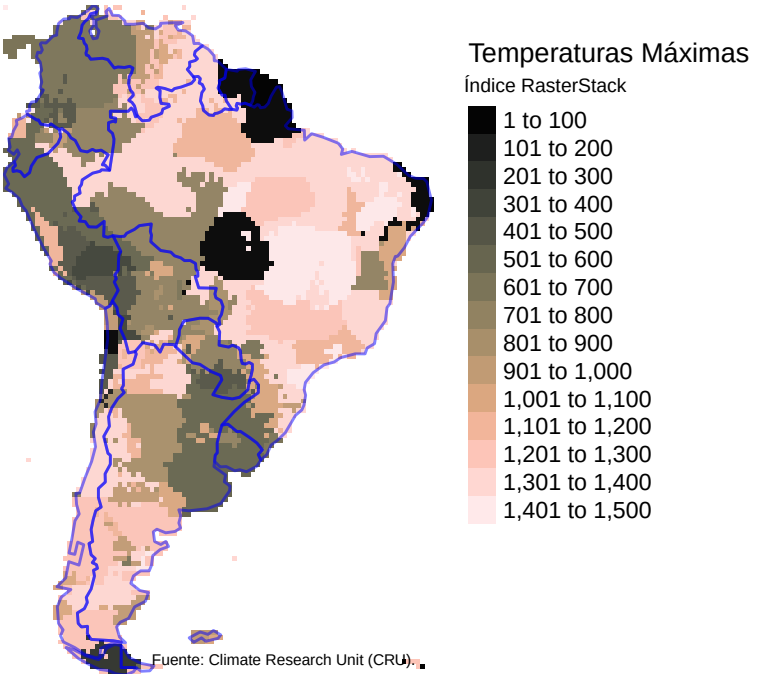


Figura 6.25: Índice de la capa donde cada celda contiene la Temperatura Máxima.

```
verano48_indice <- which(names(prec_latam) =="pre_1947_Diciembre")
verano48_indice_fin <- which(names(prec_latam) =="pre_1948_Febrero")
```

Y acumulamos las precipitaciones para el período:

```
p_acum_ver_48 <- sum(prec_latam[[verano48_indice:verano48_indice_fin]])
```

Mientras el verano 2018:

```
ver18_indice <- which(names(prec_latam) =="pre_2017_Diciembre")
ver18_indice_fin <- which(names(prec_latam) =="pre_2018_Febrero")
```

Y la precipitación acumulada:

```
p_acum_ver_18 <- sum(prec_latam[[ver18_indice:ver18_indice_fin]])
```

Y la diferencia entre ambos veranos la podemos obtener simplemente por su diferencia (Figura 6.26):

```
delta <- p_acum_ver_18 - p_acum_ver_48
```

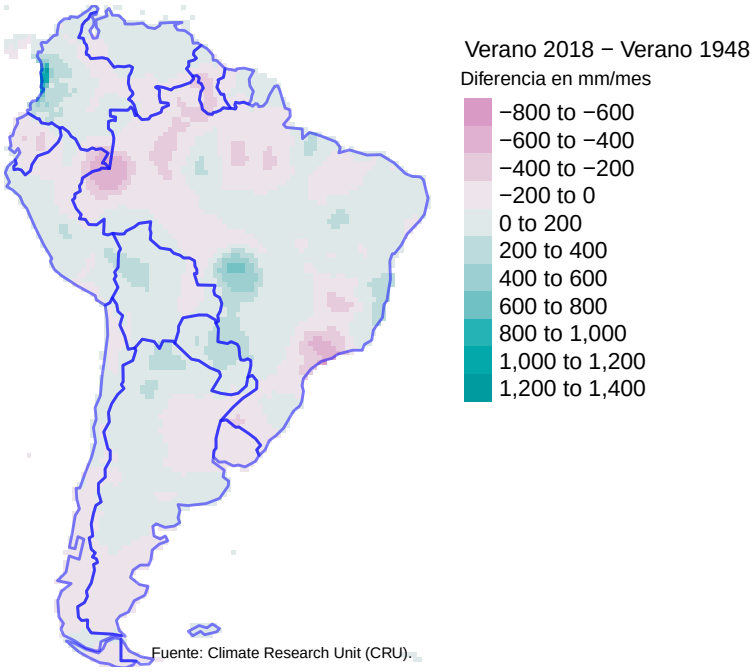


Figura 6.26: Diferencia Precipitación Acumulada Verano 2018 vs Verano 1948.

6.12.5 Función en subconjuntos de Capas

La función `stackApply(x, fun, indices)` aplica una función `fun` a un subconjunto de un rasterStack o rasterBrick `x` combinando las capas como se indice en `indices`.

Las capas por combinar se indican con los índices vectoriales. La función utilizada debe devolver un valor único y el número de capas en el ráster * de salida es igual al número de valores únicos en los índices. Por ejemplo, si tiene un RasterStack con 6 capas, puede usar índices = c (1,1,1,2,2,2) y fun = sum.

Esto devolverá un RasterBrick con dos capas. La primera capa es la suma de las tres primeras capas del RasterStack de entrada y la segunda capa es la suma de las tres últimas capas del RasterStack de entrada.

Los índices se reciclan de manera que los índices = c(1,2) también devolverían un RasterBrick con dos capas (una basada en las capas impares (1,3,5), la otra basada en las capas pares (2,4,6)). Como ejemplo aplicamos el promedio a los índices 1 al 12 (Figura

6.27).

```
stacked <- stackApply(tmax_latam, fun='mean', indices=c(1:12))
names(stacked) <- month.abb
```

```
class       : RasterBrick
dimensions : 136, 94, 12784, 12  (nrow, ncol, ncell, nlayers)
resolution : 0.5, 0.5  (x, y)
extent      : -81.5, -34.5, -55.5, 12.5  (xmin, xmax, ymin, ymax)
crs         : +proj=longlat +datum=WGS84 +no_defs
source      : memory
names       :  Jan,  Feb,  Mar,  Apr,  May,  Jun,  Jul,  Aug,  Sep,  Oct,  Nov,  Dec
min values :  2.9,  3.0,  2.1,  0.0, -2.3, -3.5, -3.9, -3.3, -1.8,  0.0,  1.1,  1.6
max values : 35.5, 35.6, 36.4, 35.5, 33.9, 33.9, 34.5, 35.5, 36.3, 36.0, 35.6, 35.5
```

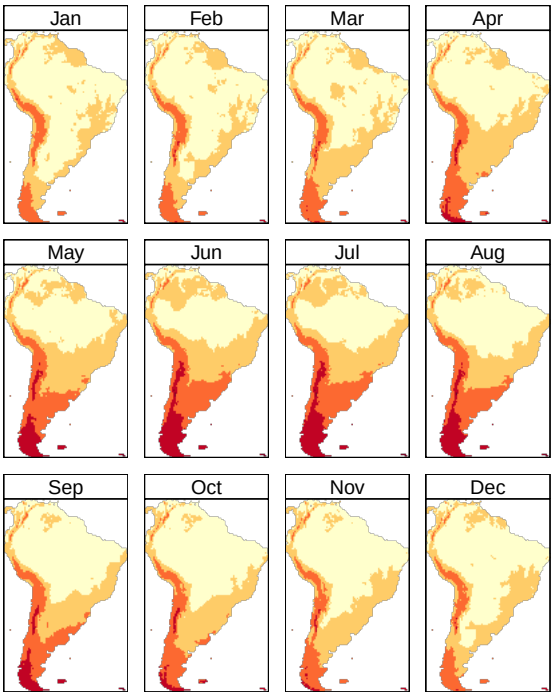


Figura 6.27: Temperatura máxima, promedio mensual (1901-2020).

6.13 Muestreos

Si necesitamos extraer datos de distintos sitios, necesitamos conseguir sus coordenadas y con ellas construir un objeto espacial *SpatialPointsDataFrame* (ver 5.2.2) o un *dataframe* (ver 1.23.3) con un **nombre** o **id**, **latitud** y **longitud**.

6.13.1 Muestreo por Coordenadas: Capitales

El primer método de muestreo está basado en coordenadas de puntos específicos o conocidos, por ejemplo, capitales nacionales. Para ello vamos a instalar y cargar el paquete *maps* (Becker et al. [2018]) que provee una extensa serie de datos geográficos junto con herramientas de trazado de mapas.

La data incorporada en el paquete `world.cities` contiene localidades pobladas por más de 40 mil habitantes, capitales nacionales de cualquier población y muchas pequeñas localidades. Debes instalar y cargar el paquete para tener la data disponible:

1. Primero vamos a filtrar los registros clasificados como capitales (1) y revisar los atributos asociados:

```
capital <- world.cities[world.cities$capital == 1,]
names(capital)
```

```
[1] "name"        "country.etc" "pop"         "lat"
[5] "long"        "capital"
```

2. Convertir el dataframe a un objeto espacial asociando las columnas de coordenadas *lat*, *long* del tipo numérico a objetos geográficos con:

```
coordinates(capital) <- c("long","lat")
```

3. Remover y renombrar columnas no útiles para el ejercicio:

```
capital <- capital[-c(3,4)]
names(capital) <- c("nombre","pais")
```

4. Interceptar los puntos con el área de estudio (Figura 6.28):

```
cap_latam <- intersect(capital, continente)
```

5. Si deseamos ajustar una coordenada manual (por ejemplo, por ajuste de decimal o recorte de la superficie ráster) podemos acceder al slot de coordenadas *coords* vía condición lógica de subconjunto:

```
cap_latam@coords[cap_latam$nombre=="Lima",]
```

```
  long    lat
-77.05 -12.07
```

Y tenemos acceso de dos formas:

```
cap_latam@coords[cap_latam$nombre=="Lima",][1] <- -77
cap_latam@coords[cap_latam$nombre=="Lima",][2] <- -12.04
```

O con vector:

```
cap_latam@coords[cap_latam$nombre=="Lima",] <- c(-77.0, -12.04)
```

Figura 6.28: Mapa de capitales de sur América.

6. Extraer valores en la ubicación definida por coordenadas geográficas:

```
prec_site <- extract(x = prec_latam,
                     y = cap_latam,
                     na.rm=T,
                     df=T)
```

7. Ahora podemos asignar los nombres de cada capital a nombre de filas, agregar como atributos las coordenadas y remover las columnas que no serán utilizadas:

```
rownames(prec_site) <- cap_latam$nombre
prec_site <- cbind(prec_site, cap_latam@coords[,1:2])
prec_site <- prec_site[,-1]
```

Para revisar el contenido y estructura extraemos una pequeña muestra de los datos:

```
prec_site[1:3,1439:1442]
```

```
         pre_2020_Noviembre pre_2020_Diciembre   long    lat
Asuncion              160.2              159.3 -57.63 -25.30
Bogota                127.6               93.4 -74.09   4.63
Brasilia              174.0              181.6 -47.91 -15.78
```

Y así obtenemos para cada una de las capitales un objeto *dataframe con una muestra de 1440 registros* con la precipitación media mensual expresadas en mm/mes más dos columnas con sus coordenadas *long* y *lat*.

De el mismo modo obtenemos la temperatura máxima mensual reemplazando el RasterStack desde donde extraemos la información:

```
tmax_site <- extract(tmax_latam,
                     cap_latam,
                     na.rm=T,
                     df=T)

rownames(tmax_site) <- cap_latam$nombre
tmax_site <- cbind(tmax_site, cap_latam@coords[,1:2])
tmax_site <- tmax_site[,-1]
tmax_site[4:6,c(1,1440:1442)]
```

```
             max_1901_Enero max_2020_Diciembre   long    lat
```

```
Buenos Aires           27.3                28.3 -58.37 -34.61
Caracas                24.1                24.3 -66.93  10.54
Georgetown             29.6                29.9 -58.16   6.79
```

Y extraemos la temperatura mínima:

```
tmin_site <- extract(tmin_latam,
                     cap_latam,
                     na.rm=T,
                     df=T)

rownames(tmin_site) <- cap_latam$nombre
tmin_site <- cbind(tmin_site, cap_latam@coords[,1:2])
tmin_site <- tmin_site[,-1]
tmin_site[9:11,c(1,1440:1442)]
```

```
         min_1901_Enero min_2020_Diciembre   long    lat
Quito               4.6                 4.5 -78.50  -0.19
Santiago           12.0                12.2 -70.64 -33.46
Sucre              10.6                10.4 -65.26 -19.06
```

6.13.2 Respaldo en disco (.csv)

R provee una serie de funciones que permiten exportar fácilmente objetos a distintos formatos de salida. Los datos generados para las capitales las vamos a exportar a nuestra carpeta *raw* en formato *csv* o separado por comas:

```
write.csv(prec_site, file=here("raw","precipitacion_data.csv"))
write.csv(tmin_site, file=here("raw","temp_min_data.csv"))
write.csv(tmax_site, file=here("raw","temp_max_data.csv"))
```

6.13.3 Muestreo Regular

El paquete *raster* incluye métodos para tomar muestras vía coordenadas usando las técnicas de muestreo (*sampling*).

El primero es **muestreo regular** que sistemáticamente toma las muestras y solo requiere el número máximo de muestras (aunque si se remueven los *NA* podría ser menor) y el límite geográfico (Figura 6.29, izquierda):

```
regp <- sampleRegular(prec_latam,
                      size= 100,
                      na.rm= T,
                      sp=T,
                      ext=continente)
```

Que devuelve un objeto espacial *SpatialPointsDataFrame*. Para convertir a un objeto dataframe y remover aquellos puntos que no contienen datos (caen fuera del modelo ráster):

```
regd <-na.omit(data.frame(regp))
```

```
regd[1:5,1440:1442]
```

```
   pre_2020_Diciembre      x    y
2                20.9 -72.75 9.75
3                23.0 -67.25 9.75
4                76.3 -61.25 9.75
10               78.7 -72.75 4.25
11               97.0 -67.25 4.25
```

Y un poco de cosmética para igualar el formato a los obtenidos en el punto *capitales*.

```
regd <- regd[c(1:1440,1442,1441)]
rownames(regd) <- paste0("sitio_", rownames(regd))
names(regd)[c(1441,1442)] <- c("long","lat")
```

Y el formato final es:

```
regd[1:3,1439:1442]
```

```
        pre_2020_Noviembre pre_2020_Diciembre long    lat
sitio_2              230.1                20.9 9.75 -72.75
sitio_3               59.0                23.0 9.75 -67.25
sitio_4              124.8                76.3 9.75 -61.25
```

6.13.4 Muestreo Aleatorio

El segundo método incluido es el **muestreo aleatorio** (Figura 6.29, derecha):

```
rndp <- sampleRandom(x = prec_latam,
                     na.rm=TRUE,
                     sp=T,
                     size = 100,
                     ext= continente)
```

Y la conversión a dataframe y edición posterior para igualar formatos:

```
rndd <- na.omit(data.frame(rndp))
rndd <- rndd[c(1:1440,1442,1441)]
rownames(rndd) <- paste0("sitio_",rownames(rndd))
names(rndd)[c(1441,1442)] <- c("long","lat")
rndd[1:3,1439:1442]
```

```
        pre_2020_Noviembre pre_2020_Diciembre   long    lat
sitio_1                5.3                14.1 -47.25 -68.25
sitio_2              183.8               255.0  -3.75 -68.75
sitio_3                0.0                 0.0 -25.75 -70.25
```

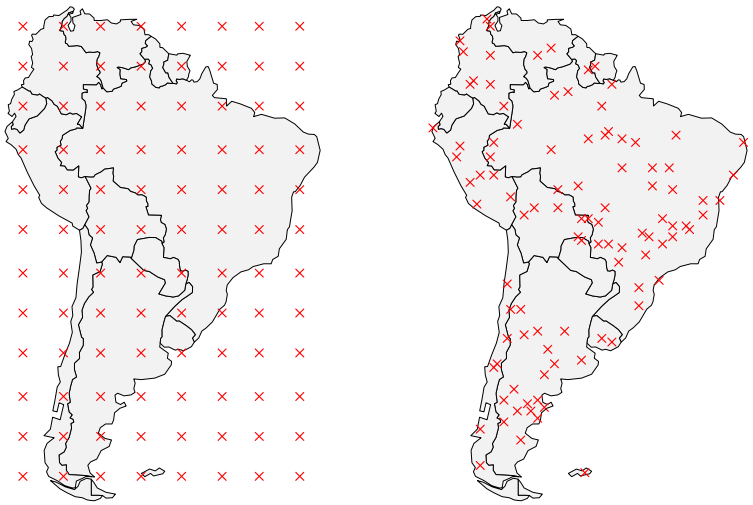


Figura 6.29: Muestreo Sistemático (izquierda) y Muestreo Aleatorio (derecha).

6.13.5 Muestreo Estratificado

Finalmente, debemos mencionar que también existe el **método estratificado** (Figura 6.30) que se encarga de tomar una

muestra aleatoria estratificada de los valores de celda de un objeto ráster (sin reemplazo). Se intenta muestrear celdas en cada estrato. Para la definición de cada estrato los valores del RasterLayer se redondean a números enteros y cada valor representa un estrato. Dentro de cada uno de ellos se extraen las muestras definidas:

```
estratos <- prec_latam[[1]] * 1/10
stratp <- sampleStratified(estratos,
                           na.rm=TRUE,
                           sp=T,
                           size = 5,
                           ext= continente)
```

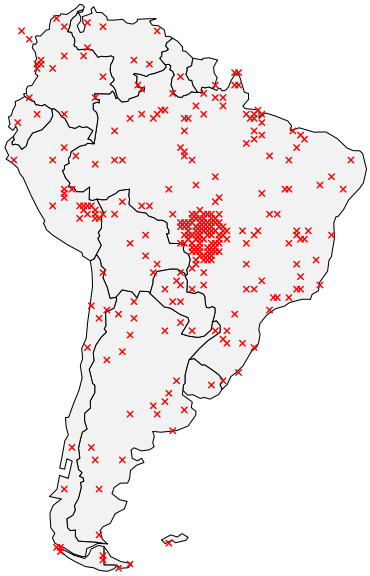


Figura 6.30: Muestreo Estratificado.

6.14 Diagramas Climáticos

Los diagramas climáticos son herramientas útiles que proporcionan un resumen visual de las condiciones climáticas medias de un lugar durante un período de tiempo específico.

Esta guía explica las diferentes partes de un diagrama climá-

tico de Walter y Lieth (Walter et al. [1975]) y mostrará cómo crearlos.

6.14.1 Paquete climatol

El paquete **climatol** (Guijarro [2019]) implementa dichos gráficos. Los datos climáticos deben pasarse como una *matriz 4x12* de datos **mensuales** (de enero a diciembre), en el siguiente orden:

- **Fila 1**: Precipitación media.
- **Fila 2**: Temperatura diaria máxima media.
- **Fila 3**: Temperatura diaria mínima media.
- **Fila 4**: Temperatura mínima absoluta mensual.

Esta última fila solo se utiliza para determinar los probables meses de heladas (cuando los mínimos mensuales absolutos son iguales o inferiores a 0°C).

Y utilizando los datos obtenidos en las capitales (ver 6.13.1) extraemos los datos para el año 1901 de la ciudad de Asunción (Figura 6.31):

```
i <- which(names(prec_latam) =="pre_1901_Enero")
ii <- which(names(prec_latam) =="pre_1901_Diciembre")
prec <- as.numeric(prec_site["Asuncion",i:ii])
tmax <- as.numeric(tmax_site["Asuncion",i:ii])
tmin <- as.numeric(tmin_site["Asuncion",i:ii])
tmina <- as.numeric(tmin_site["Asuncion",i:ii])
```

Y generamos la matriz que requerirá la función de dibujo:

```
data <- matrix(c(prec,tmax,tmin,tmina),
               nrow=4,
               ncol=12,
               dimnames=list(
                 c("prec","max","min","mina")),
                 c(1:12)
               )

round(data,0)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
```

```
prec  105  107   88  124  104   47   33   72   91  105  140   31
max    33   32   31   26   25   23   24   25   29   30   31   34
min    22   22   21   16   16   15   15   15   18   20   20   22
mina   22   22   21   16   16   15   15   15   18   20   20   22
```

Para las estaciones meteorológicas ubicadas en el **hemisferio sur** es útil configurar `shem=TRUE`, para mantener el período de verano en la zona central del gráfico (el diagrama comenzará la trama con los datos de julio).

Como describen Walter y Lieth (Walter et al. [1975]), cuando la precipitación mensual es superior a 100 mm, la escala se aumenta de 2mm/ºC a 20mm/ºC para evitar diagramas demasiado altos en lugares muy húmedos. Este cambio se indica con una línea horizontal negra y el gráfico que se encuentra encima se rellena en azul sólido.

Cuando el gráfico de precipitación se encuentra debajo del gráfico de temperatura ($P < 2T$), tenemos un período árido (lleno de líneas verticales rojas punteadas). De lo contrario, el período se considera húmedo (rellenado con líneas azules), a menos que `p3line=TRUE` que dibuje una línea negra de precipitación con una escala $P = 3T$, en este caso, el período en el que $3T > P > 2T$ se considera semiárido.

La *temperatura media diaria máxima* del mes más caluroso y la *temperatura media mínima diaria* del mes más frío se utilizan con frecuencia en los estudios de vegetación y están etiquetadas en negro en el margen izquierdo del diagrama.

```
diagwl(data,
       est="Asunción (PY)",
       mlab="es",
       shem=T,
       per="1901")
```

Otro ejemplo, Santiago 2010 (Figura 6.32).

```
diagwl(data,
       est="Santiago (CL)",
       mlab="es",
       shem=T,
       per="2010")
```

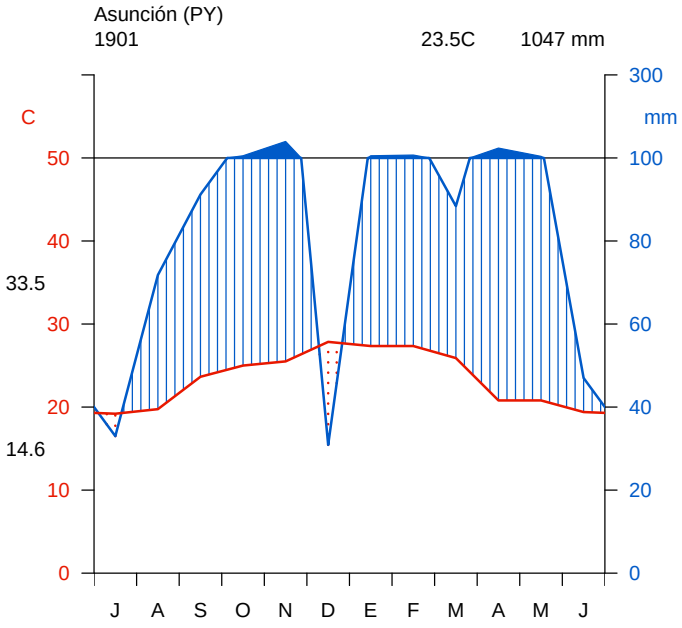


Figura 6.31: Diagrama Climático Walter-Lieth, Asunción 1901.

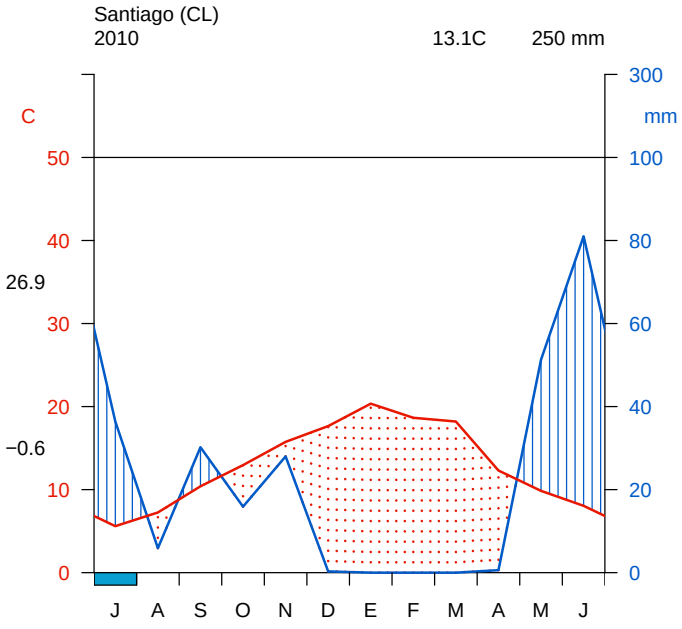


Figura 6.32: Precipitación Media Mensual Enero 1901.

6.15 Serie Temporal

Cualquier métrica que se mida en intervalos de tiempo regulares constituye una **serie temporal**. Los datos de series de tiempo están en todas partes, ya que el tiempo es un componente de todo lo que es observable. Por ejemplo, datos meteorológicos, precios de acciones, estadísticas de búsquedas por internet, etc. son algunos de los más habituales.

Primero vamos a revisar conceptos y prácticas sobre series temporales medidas sobre un *punto geográfico* (coordenadas de ciudades capitales) y a continuación series temporales de *áreas geográficas* (objetos RasterBrick).

6.15.1 Continuidad de los Datos

Es muy importante detectar si la serie de datos existe para cada uno de momentos temporales. Si la serie es continua y no posee datos faltantes o vacíos. Los datos climáticos utilizados a continuación han sido corregidos y poseen para cada fecha una observación, pero no siempre será el caso, y para resolver el problema se han desarrollado diversas metodologías correctivas (ver 9.3.1).

6.15.2 Carga de Datos

En el presente capítulo vamos a utilizar la data preparada en el punto 6.13.2. Cargado los archivos *csv* para las tres variables:

1. Carga de Precipitación Media Mensual. Período Enero 1901 - Diciembre 2020.

```
prec_site <- read.csv(here("raw","precipitacion_data.csv"),
                      header=TRUE,
                      row.names=1,
                      sep=",", check.names=FALSE)
prec_site[1:5,1:3]
```

```
             pre_1901_Enero pre_1901_Febrero pre_1901_Marzo
Asuncion              105.2            106.9           88.4
Bogota                 23.5             71.0           67.9
```

Brasilia	256.0	214.0	186.0
Buenos Aires	65.7	57.4	58.0
Caracas	0.1	84.1	24.0

2. Carga de Temperatura Mínima Mensual. Período Enero 1901 - Diciembre 2020.

```
tmin_site <- read.csv(here("raw","temp_min_data.csv"),
                      header=TRUE,
                      row.names=1,
                      sep=",", check.names=FALSE)
tmin_site[1:5,1:3]
```

	min_1901_Enero	min_1901_Febrero	min_1901_Marzo
Asuncion	22.0	22.200001	20.8
Bogota	9.6	9.900001	10.6
Brasilia	17.8	17.900000	17.8
Buenos Aires	16.6	17.200001	15.9
Caracas	13.7	13.600000	14.2

3. Carga de Temperatura Máxima Mensual. Período Enero 1901 - Diciembre 2020.

```
tmax_site <- read.csv(here("raw","temp_max_data.csv"),
                      header=TRUE,
                      row.names=1,
                      sep=",", check.names=FALSE)
tmax_site[1:5,1:3]
```

	max_1901_Enero	max_1901_Febrero	max_1901_Marzo
Asuncion	32.7	32.5	31.0
Bogota	20.3	20.3	20.3
Brasilia	27.1	27.4	27.7
Buenos Aires	27.3	27.3	25.6
Caracas	24.1	24.4	25.1

6.15.3 Objeto timeserie

Para la construcción de un objeto *time-serie* definido en uno de los paquetes base *stats* (R Core Team [2021]) definimos un vector con valores (la variable a analizar) y especificar la frecuencia de

los datos (en nuestro caso *mensual*) y el año de inicio, ambos datos para internamente definir un vector de misma longitud de datos en formato *Date*.

```
i <- 3
valores <- as.numeric(tmax_site[i,c(-1441,-1442)]) #remover long,lat
ciudad <- rownames(valores)[i]
```

Y la función para construir el objeto:

```
ts_base <- ts(valores,
              frequency=12,
              start=1901)
```

Y para la exploración de los datos utilizaremos el paquete **TSstudio** (Krispin [2020]) que nos presenta interfaces muy modernas y ágiles.

El comando básico es confirmar que nuestro objeto es un objeto del tipo **ts**, sus variables y el número de observaciones:

```
ts_info(ts_base)
```

```
The ts_base series is a ts object with 1 variable and 1440 observations
Frequency: 12
Start time: 1901 1
End time: 2020 12
```

La opción básica es el trazado del perfil mediante la función `ts_plot()` en un panel interactivo (Figura 6.33):

```
ts_plot(ts_base, width = 1)
```

En el panel algunas de las funcionalidades importantes se encuentran en el borde superior derecho en forma de barra de herramientas que aparecen al mover el cursor sobre ella (Figura 6.34).

Alguna de las opciones importantes contenidas en la barra:

- Grabar un *png* de la información desplegada.
- Activar la opción de *Zoom*.
- Activar la opción de *Pan*.
- Restaurar la vista.

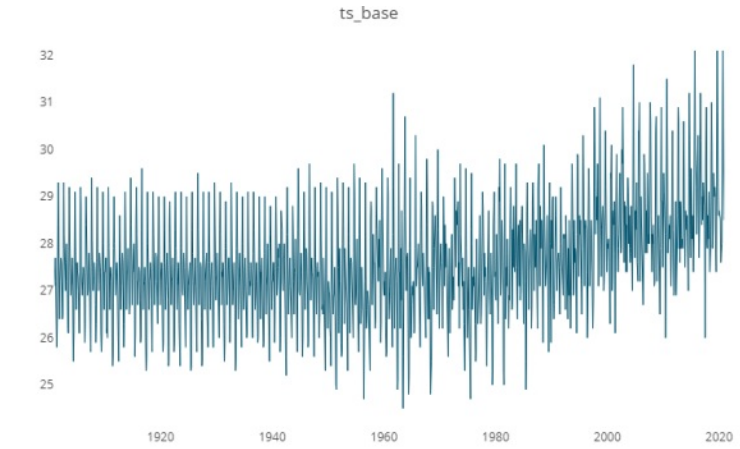


Figura 6.33: Interfaz gráfica interáctiva creada por función ts_plot().

Figura 6.34: Barra completa de Herramientas.

6.15.4 Descomposición de Series de Tiempo

Los datos de series de tiempo pueden exhibir una variedad de patrones y, a menudo, es útil dividir una serie de tiempo en varios componentes, cada uno de los cuales representa una categoría de patrón subyacente (Figura 6.35):

```
ts_decompose(ts_base)
```

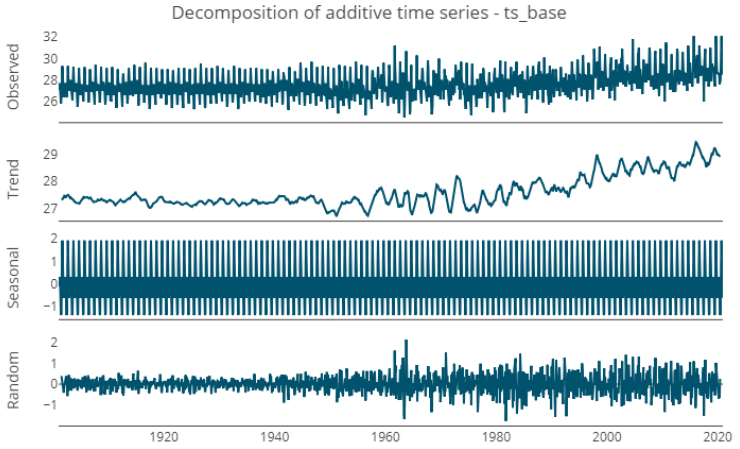


Figura 6.35: Interfaz gráfica interactiva creada por función ts_decompose() del paquete TSstudio.

La **descomposición** es una *tarea estadística* en la que los datos de la serie temporal se descomponen en varios componentes o extrayendo la estacionalidad y tendencia de los datos de una serie. Estos componentes se definen de la siguiente manera:

- **Data** *(observed)*: la serie de datos.
- **Tendencia** *(trend)*: el valor creciente o decreciente de la serie.
- **Estacionalidad** *(seasonal)*: el ciclo a corto plazo que se repite en la serie.
- **Ruido** *(random)*: la variación aleatoria en la serie.

Los datos de series de tiempo son una combinación de estos componentes. Todas las series tienen ruido. Los componentes de tendencia y estacionalidad son opcionales. En los datos de series de tiempo, estos componentes se combinan tanto de forma **aditiva** o **multiplicativa**. Revisar fundamentos en Cowpertwait and Metcalfe [2009].

Modelo aditivo: El modelo aditivo es aquel en el que la varianza de los datos no cambia en los diferentes valores de la serie de tiempo. El componente sistemático es la suma aritmética de los efectos individuales de los predictores.

El modelo aditivo es lineal y la línea de tendencia aquí es una línea recta y la estacionalidad tiene la misma frecuencia y amplitud (altura y ancho del ciclo, respectivamente).

Modelo multiplicativo: El modelo multiplicativo es aquel en el que a medida que aumentan los datos, también aumenta el patrón estacional o la varianza. Aquí, los componentes de tendencia y estacional se multiplican y luego se agregan al componente de error.

El modelo multiplicativo es no lineal, como cuadrático o exponencial y la tendencia es una línea curva y la estacionalidad tiene una frecuencia y amplitud crecientes o decrecientes a lo largo del tiempo.

6.15.5 Análisis de Estacionalidad

La **estacionalidad** es la repetición de determinadas variaciones en alguna variable cada cierto período, normalmente igual o menor a un año. En períodos más amplios se suele hablar de

ciclos, aunque las variaciones cíclicas no son tan frecuentes como las estacionales y se contrapone a la tendencia (comportamiento a largo plazo).

El análisis de estacionalidad es uno de los elementos centrales del proceso de análisis descriptivo de los datos de series de tiempo vía la función `ts_seasonal()`.

Este proceso generalmente se realiza con el uso de herramientas de **visualización de datos** y métodos de **estadísticas resumidas**. El paquete TSstudio proporciona un conjunto de funciones para gráficos de estacionalidad (Figura 6.36):

```
ts_seasonal(ts.obj = ts_base,
            type = "all")
```

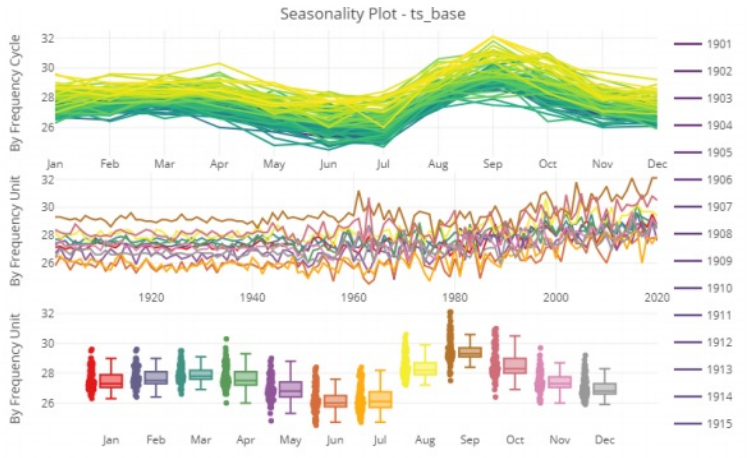


Figura 6.36: Interfaz gráfica interactiva creada por función ts_seasonal() del paquete TSstudio.

La leyenda interáctiva a la extrema derecha del panel de visualización que vía clic permite apagar dicho elemento.

6.15.6 Visualización de Lags (retrasos)

El análisis secuencial de retardos (*lags*) es un método para analizar la *dependencia secuencial* en una serie temporal que representan diferentes estados del sistema.

Si el sistema está en un estado A en un momento t, entonces es más o menos probable que el sistema esté en estado A, en los momentos $t+1, t+2, ..., t+k$.

El análisis asume que los eventos están secuenciados en el tiempo (serie de tiempo) pero no asume intervalos de tiempo iguales entre eventos.

Un **lags** es una *cantidad fija de tiempo transcurrido*; un conjunto de observaciones en una serie de tiempo se grafican contra un segundo conjunto de datos posteriores (Figura 6.37).

```
ts_lags(ts_base,lags = c(12, 24, 36, 48, 60, 72))
```

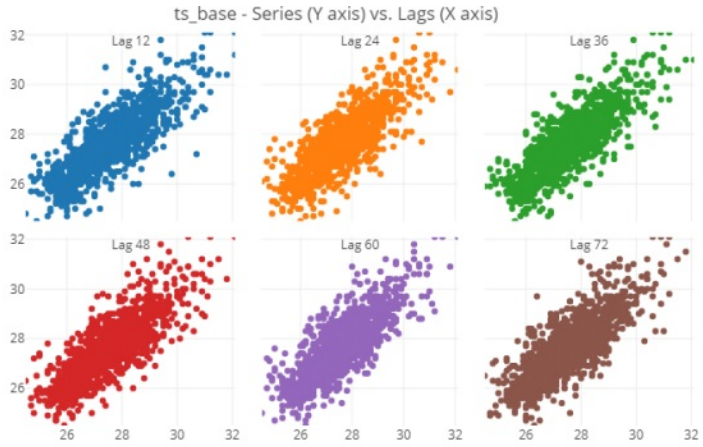


Figura 6.37: Interfaz gráfica interáctiva creada por función ts_lags() del paquete TSstudio.

6.16 Serie Temporal Ráster

Las series temporales de archivos ráster construidos y guardados en disco en 6.12.2 son cargados con el comando brick() del paquete raster:

```
prec_latam <- brick(here("raw", "prec_latam.tif"))
tmin_latam <- brick(here("raw", "tmin_latam.tif"))
tmax_latam <- brick(here("raw", "tmax_latam.tif"))
```

Como independientemente guardamos el listado de fechas en formato *data* de R, leemos como el objeto *fechas*®:

```
fechas <- readRDS(here("raw", "fechas.rds"))
```

Y asignamos al slot que define el tiempo de cada capa:

```
prec_latam@z$time <- fechas
tmin_latam@z$time <- fechas
tmax_latam@z$time <- fechas
```

También asignamos a los nombres de capas para referencia:

```
names(prec_latam) <- fechas
names(tmin_latam) <- fechas
names(tmax_latam) <- fechas
```

6.16.1 Paquete rts

El paquete *rts* (Naimi [2019]) provee las funciones básicas para la construcción, edición, manejo y análisis de series temporales ráster en R.

Lo primero es instalar (desde la consola) y cargar en memoria el paquete (desde el script si está construyendo uno):

```
library(rts)
```

Y usando la función para construir nuestra serie temporal con el objeto RasterBrick y la lista de nombres construimos un objeto *rts*.

```
prec_ts <- rts(prec_latam,fechas)
```

Se puede extraer una capa por su índice y convertirla en un objeto ráster básico:

```
r <- prec_ts[[2]]
class(r)
```

```
[1] "RasterLayer"
attr(,"package")
[1] "raster"
```

También extraer vía slot *raster*:

```
r <- prec_ts@raster$prec_latam.6
class(r)
```

```
[1] "RasterLayer"
```

```
attr(,"package")
[1] "raster"
```

6.16.2 Herramientas de Subconjuntos

El paquete *rts* posee poderosas herramientas para extraer capas por plazos de tiempo:

- Por Rango acotado:

1. *Entre Fechas concretas*: Entre dos fechas concretas siguiendo el formato YYYYMMDD/YYYYMMDD (el símbolo '/' es obligatorio).

```
prec_periodo <- subset(prec_ts, "20190505/20190903")
```

2. *Período Anual*: Solo definir año de inicio y fin:

```
prec_2013_2014 <- subset(prec_ts,"2013/2014")
```

- Por un límite definido:

1. *Fijar el inicio*: Todos los años desde (e incluyendo) el fijado hasta el final.

```
prec_2014_fin <- subset(prec_ts,"2014/")
```

2. *Fijar el fin*: Todo hasta el final del año fijado.

```
prec_ini_2014 <- subset(prec_ts,"/2014")
```

3. *Fijar el final en detalle*: Todo hasta el final del mes fijado.

```
prec_ini_jun_2014 <- subset(prec_ts,"/2014-06")
```

- Por fecha particular:

1. *Un Año*: Seleccionar todo el contenido para un año particular.

```
prec_2012 <- subset(prec_ts,"2012")
```

2. *Un mes de un año*: Seleccionar un mes de un año particular.

```
prec_jun_2016 <- subset(prec_ts,"2016-05")
```

El contenido del objeto *prec_2012* (Figura 6.38) se compone de 12 modelos ráster que forman un objeto RasterStack más algunos parámetros particulares de un objeto **rts**.

```
prec_2012
```

```
Raster Time Series with monthly periodicity from 2012-01-16 to 2012-12-16
class       : RasterStackTS
raster dimensions  : 136, 94, 12784, 12  (nrow, ncol, ncell, nlayers)
raster resolution  : 0.5, 0.5  (x, y)
raster extent      : -81.5, -34.5, -55.5, 12.5  (xmin, xmax, ymin, ymax)
coord. ref. : +proj=longlat +datum=WGS84 +no_defs
min values  : 0 0 0 0 0 0 0 0 0 0 ...
max values  : 885 802 830 709 782 513 520 656 421 690 ...
```

Y podemos usar la función *plot* para trazar un gráfico con su contenido en formato de mapa:

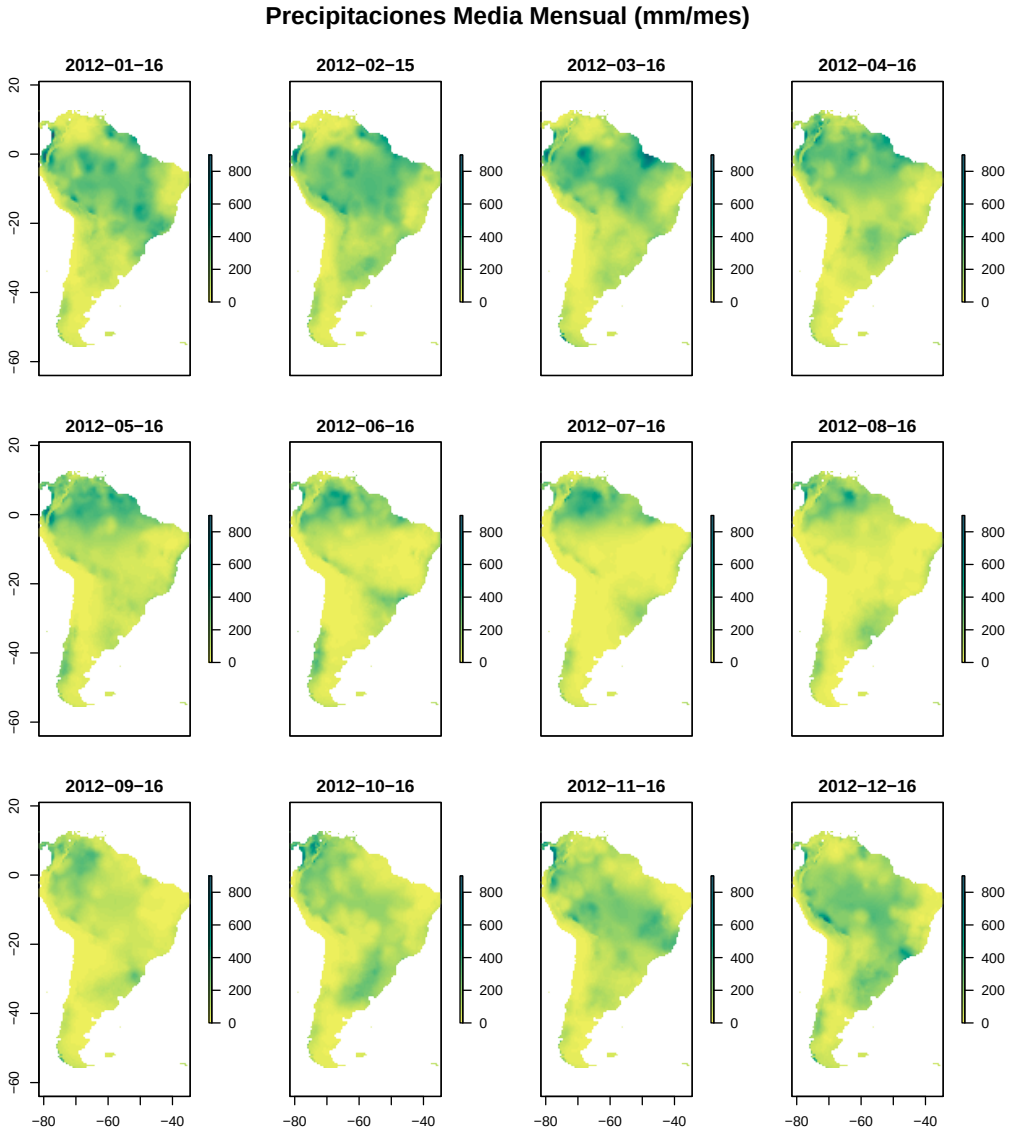


Figura 6.38: Año 2012, precipitaciones media mensual (mm/mes).

6.16.3 Operaciones Sobre un rst

El paquete provee también una serie de funciones que permiten aplicar una función sobre un período temporal dentro de la serie temporal misma. Los períodos son:

- apply.daily(x, FUN, ...)
- apply.weekly(x, FUN, ...)
- apply.monthly(x, FUN, ...)
- apply.quarterly(x, FUN, ...)
- apply.yearly(x, FUN, ...)

Y devuelven un nuevo objeto RasterStackTS:

```
prec_periodo <- subset(prec_ts,"1981/1984")
prec_anuales <- apply.yearly(prec_periodo, FUN = 'max', na.rm=TRUE)
```

```
Raster Time Series with yearly periodicity from 1981-12-16 to 1984-12-16
class       : RasterStackTS
raster dimensions  : 136, 94, 12784, 4  (nrow, ncol, ncell, nlayers)
raster resolution  : 0.5, 0.5  (x, y)
raster extent      : -81.5, -34.5, -55.5, 12.5  (xmin, xmax, ymin, ymax)
coord. ref. : +proj=longlat +datum=WGS84 +no_defs
min values  : 0 0 0 0
max values  : 1470 1436 1100 1410
```

Y el mapa resultante (Figura 6.39).

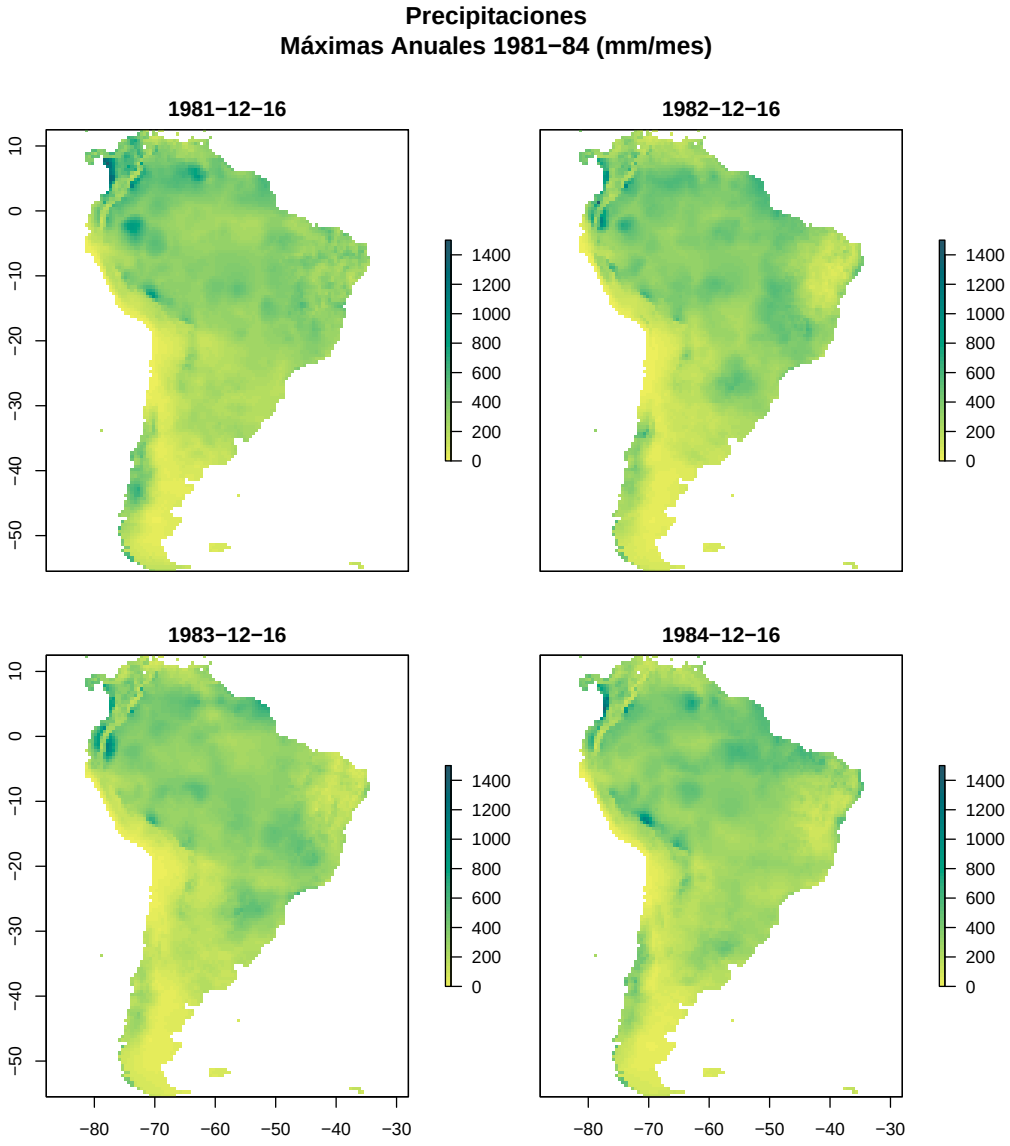


Figura 6.39: Período 1981-84, precipitaciones máximas anuales (mm/mes).

VII OBSERVACIÓN DE LA TIERRA

Introducción

En la actualidad a tomado relevancia el concepto de **Observación de la Tierra** desde el espacio (EO del inglés) que hace referencia a las actividades de captura de datos de la superficie terrestre a partir de vehículos espaciales diseñados específicamente para observar en un amplio espectro de la energía radiante de la tierra.

También al realizar la observación desde lejos (sin contacto con el objeto de estudio) se denominan ya desde principios de la década del 60 como *técnicas de teledetección*.

Las actividades de EO o teledetección incluyen una amplia gama de técnicas para la adquisición, almacenamiento y transmisión de los datos los cuales luego deben ser convertidos en información relevante, que finalmente debe llegar de manera oportuna a una amplia variedad de usuarios finales potencial para el uso de agricultores, como pescadores, periodistas, glaciólogos, ecologistas, geógrafos y un amplio etcétera.

De las diversas fuentes de datos hoy las imágenes de satélite de resolución media (el nivel de detalle observable) y de acceso libre (gratis en realidad) se han vuelto ampliamente accesibles, proporcionan cobertura mundial con una amplia variación espectral (región del espectro que pueden sensar) y temporal (período de tiempo entre las observaciones para el mismo punto geográfico), lo que les hace aptos para una amplia gama de estudios.

La señal de energía radiante que es detectada y medida por el sensor satelital es almacenada en la memoria a bordo del satélite o transmitida a un receptor en tierra para su posterior interpretación.

La teledetección y la EO también incluyen el **análisis** y la **interpretación** de los datos adquiridos.

La extracción de información relevante requiere de una buena comprensión de la base física, el proceso de adquisición y un conocimiento sólido de los algoritmos utilizados para procesar los datos originales.

7.1 Energía Electromagnética

La observación remota requiere algún tipo de interacción energética entre el objetivo y el sensor. La señal detectada por el sensor puede ser energía solar (generada en el Sol) que es reflejada por la superficie de la Tierra, como también puede ser energía emitida desde la superficie por sí misma o la generada por el mismo sensor que produce su propio pulso de energía.

Todo este proceso tiene una fuerte base física, ya que implica recopilar señales electromagnéticas (EM) provenientes de objetos con diferentes propiedades físicas y químicas. Y necesariamente debemos comenzar con una descripción básica e histórica de la física que soporta todo este desarrollo.

7.1.1 Fundamentos Históricos y Físicos

Todo comienza con dos fenómenos muy simples y observables en la vida diaria, uno el típico chispazo que sentimos en la punta de los dedos luego de caminar sobre una alfombra, o cuando se erizan los cabellos cuando se roza con un trozo de tela o los rayos que observamos en una tormenta, todos ellos productos de la **electricidad** y dos, el efecto que hace funcionar los imanes o una brújula, el **magnetismo**.

Figura 7.1: Muestra de Ámbar. (foto: De Hannes Grobe-Trabajo propio, CC BY-SA 2.5)

7.1.2 Fenómeno Eléctrico

Ya en la antigua Grecia Tales de Mileto notó que al frotar una piedra de ámbar (Figura 7.1) era capaz de atraer por el aire pequeños objetos (ámbar en griego es élektron y de allí deriva la palabra electricidad) y trató de buscar una explicación, aunque para encontrar una verdaderamente moderna del fenómeno de la

electricidad, hubo que esperar unos dos mil años por el descubrimiento del **electrón**, gracias a los trabajos de Sir J.J. Thompson (Figura 7.2).

Figura 7.2: Físico Inglés Sir J.J.Thomson.

El electrón es una partícula muy ligera y con gran movilidad que existe dentro de los átomos, una partícula que como responde a los efectos eléctricos se dice que está *cargada eléctricamente*; de hecho, hoy entendemos la electricidad como un movimiento de electrones en un conductor y esta es la base por ejemplo de toda la electrónica moderna y como electrones hay en todas partes es muy fácil generar fenómenos eléctricos.

7.1.3 Fenómeno Magnético

Figura 7.3: Los Propileos de Magnesia del Meandro.Turquía. (foto: De Vadimph-Trabajo propio,CC BY-SA 3.0)

También desde la antigüedad (otra vez los griegos y otra vez Tales de Mileto) constatan una extraña fuerza que entre ciertas piedras muy comunes en la Región de Magnesia y ahí su nombre: **magnetismo** (Figura 7.3).

La explicación moderna a este magnetismo natural se explica también por los electrones que componen la materia. Hay que imaginar los electrones como pequeños imanes dentro de los materiales normales (aunque por ejemplo en la madera estos pequeños imanes están todos desorientados de forma tal que el efecto general difiere a lo que ocurre en esas piedras especiales).

Estos "pequeños imanes" en los materiales magnéticos están ordenados según una orientación específica, lo que genera un efecto global, lo que genera un gran imán, caracterizado como ya saben por el polo norte y el polo sur, hoy a estos pequeños imanes en los electrones lo conocemos con el nombre de **spin**, una propiedad cuántica muy peculiar.

Los imanes hoy en día los encuentras por todas partes y son una pieza fundamental del desarrollo moderno.

Figura 7.4: Daguerrotipo de Hans Christian Ørsted.

7.2 La Relación Electricidad Magnetismo

Si ambos fenómenos se relacionan íntimamente con los electrones alguna conexión entre el *magnetismo* y la *electricidad* debía existir.

La primera indicación de que eran fenómenos realmente conectados la observó Hans Christian Ørsted (Figura 7.4) un físico danés en el siglo 19.

Todo fue a raíz de una observación muy importante pero que ocurrió de manera casual, durante una demostración en clases al conectar la corriente en un circuito eléctrico observó que la aguja de una brújula cercana se desviaba y tras numerosas pruebas y experimentos llegó esencialmente a la conclusión correcta: *el flujo de cargas -una corriente eléctrica- genera un fenómeno magnético alrededor del cable* y esto da lugar a un nuevo tipo de imanes, el **electroimán**.

Si haces pasar una corriente eléctrica por un cable circular (una *espira*) se crea un campo magnético como el de un imán con un polo norte y un polo sur. Este mecanismo es fundamental en la sociedad moderna dando lugar a los motores eléctricos: la transformación de electricidad en movimiento.

Figura 7.5: Grabado de André-Marie Ampère por Ambrose Tardieu.

Ørsted publica sus resultados y un joven científico francés muy talentoso, André-Marie Ampère (Figura 7.5) decide probar por su cuenta, dando con lo que se conoce como *la ley de Ampère*, relativa a las fuerzas de dos cables que conducen corriente.

Michael Faraday (Figura 7.6) un joven inglés sin ninguna formación científica oye de estos nuevos desarrollos y se pregunta si una corriente eléctrica es capaz de producir magnetismo ¿el magnetismo será capaz también de crear una corriente eléctrica? una intuición maravillosa y muy científica, luego de un arduo trabajo consiguió probarlo, no sin antes romperse la cabeza ya que esta intuición venía con una *pequeña sutileza*, no es la presencia del imán en el conductor lo que consigue generar una corriente sino su movimiento.

Figura 7.6: Retrato de Michael Faraday pintado por H.W. Pickersgill.

Es la variación del magnetismo lo que crea la electricidad, aunque la gran revolución estaba aún por llegar, en todo esto, había algo que no le encajaba al joven Faraday, unas fuerzas misteriosas la eléctrica y magnética se desplazaban por el aire sin ningún tipo de sustento, esto parecía poco físico para él y varios experimentos le hicieron apuntar en la dirección correcta: primero consiguió en un cristal ver una conexión entre la electricidad y la luz y luego tenía el famoso experimento de las limaduras de hierro

que espolvoreadas en un papel y pones debajo un imán ves que se generaban patrones que recorrían la distancia entre los polos, para Faraday tenía que existir agua invisible en el aire algo que no podemos ver pero que de alguna forma estaba guiando las limaduras (Figura 7.7) tomada de Black [1913].

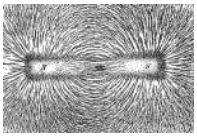


Figura 7.7: Las limaduras de hierro permiten mostrar la dirección del campo magnético de un imán permanente.

Faraday llamó a este peculiar fenómeno **campo**, este concepto resultó ser uno de los conceptos más poderosos de la historia de la ciencia: *un objeto altera las propiedades de su entorno cuando otro cuerpo se acerca a este*, es el propio espacio el que ejerce la acción sobre el cuerpo es una forma de transmitir una fuerza por el espacio, una forma que usaría el mismo Albert Einstein en su teoría de la gravedad en la *relatividad general*.

7.2.1 El Concepto de Campo

El concepto de campo fue una de las ideas más importantes de la ciencia, hoy en día todas las modernas teorías de fuerzas e interacciones están basadas en el concepto de campo, la fuerza eléctrica y magnética se transmiten por medio de campos.

Con esta revolución se está creando una nueva área de la ciencia y el puntapié inicial la da un joven escocés James Clerk Maxwell (Figura 7.8). Maxwell era un científico sobresaliente de gran intuición y una formación matemática muy sólida y ve en el concepto de campo una herramienta fundamental para llevar al próximo nivel la idea de electricidad y magnetismo apoyado en esta sorprendente simetría.

Figura 7.8: Grabado de James Clerk Maxwell por G. J. Stodart.

Con la ayuda de los desarrollos teóricos previos junto con el concepto de campo Maxwell desarrolla un todo unificado. Inicialmente 20 ecuaciones que hoy reducidas a cuatro son las famosas ecuaciones de Maxwell uno de los desarrollos más prodigiosos, potentes y revolucionarios que han surgido nunca del intelecto humano.

7.2.2 El Electromagnetismo

Cuando Maxwell se propuso resolver sus ecuaciones observó algo muy peculiar: la solución de sus ecuaciones eran una onda. Una onda de electricidad y magnetismo, una onda **electromag-**

nética, una onda que se auto sustentaba en el espacio.

Una de las propiedades de esta onda es que la velocidad a la que se desplaza es exactamente la velocidad de la luz. Ese descubrimiento le lleva a teorizar:

"Esta velocidad es tan próxima a la de la luz, que parece que tenemos sólidas razones para concluir que la propia luz (incluyendo el calor radiante, y otras radiaciones si las hay) es una perturbación electromagnética en forma de ondas que se propagan de acuerdo con las leyes del electromagnetismo".

Como hemos descubierto hasta ahora, si la variación del campo magnético es el que genera el campo eléctrico y viceversa, podemos pensar en el caso de la luz, tomamos un electrón en el vacío que de repente oscila, esto hace que se genere el magnetismo -un campo magnético- que tiene la propiedad de penetrar el vacío, luego, ese campo magnético va decreciendo y esta variación a su vez genera un campo eléctrico que se va haciendo cada vez más grande según el campo magnético va disminuyendo y en ese momento ocurre lo contrario, el campo eléctrico empieza a disminuir lo que va generando un campo magnético y así sucesivamente. Ambos de forma acoplada se van reformando y reforzando, permitiendo que la *perturbación viaje hasta el infinito.*

Así el campo eléctrico y el campo magnético en una onda electromagnética vibran perpendicularmente (90°) entre sí (Figura 7.9).

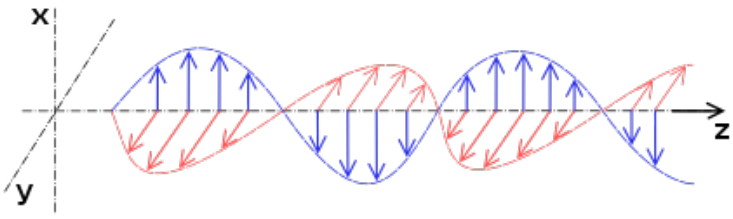


Figura 7.9: Onda electromagnética. Fuente: SuperManu, CC BY-SA 3.0.

7.3 Cómo se Produce y Transporta la Energía

7.3.1 Las Ondas

Una *onda* se puede describir usando algunos atributos básicos (Figura 7.10):

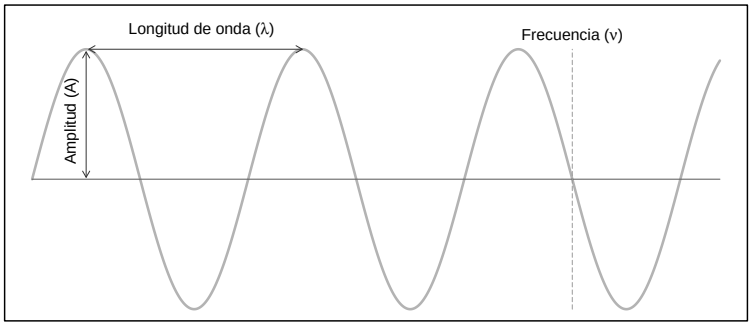


Figura 7.10: Descripción general de una onda.

- **Amplitud (A)**: Distancia entre el punto central y la *cresta* (punto de altura máxima), la unidad depende del tipo de onda (m, v/m).
- **Longitud de Onda (λ)**: Distancia entre dos crestas o *valles* (punto de altura mínima), se mide en unidad de distancia (m, mm, μm).
- **Frecuencia (ν)**: Número de crestas que pasan por un punto dado por unidad de tiempo. La unidad de medida es el hercio (HZ), que equivale a un ciclo por segundo (un ciclo es el paso de dos crestas).
- **Período (T)**: Tiempo transcurrido entre el paso de dos crestas por un punto.
- **Fase (φ)**: El *ángulo de fase* o *fase* de una onda se refiere a su desplazamiento hacia la derecha o la izquierda con respecto a una referencia. Se mide en grados (°) o radianes (rad).

Algunas relaciones que se desprenden de esta descripción son:

1. Frecuencia y Período son recíprocas entre sí:

$$\nu = \frac{1}{T} \tag{7.1}$$

2. La longitud de onda λ y frecuencia ν están inversamente relacionadas:

$$c = \lambda\nu \tag{7.2}$$

Donde c es la velocidad de la luz en el vacío, ~$3 \times 10^8 \frac{m}{s}$.

7.3.2 Cuantos y Dualidad de la Luz

De acuerdo con la física clásica, la materia estaba compuesta por partículas que tenían masa, cuya posición en el espacio podía ser conocida; por otro lado, consideraban que las ondas luminosas no tenían masa, y que su posición en el espacio no podía ser determinada. Puesto que pensaban que pertenecían a diferentes categorías, los científicos no tenían una buena comprensión de cómo interactuaban la luz con la materia. Sin embargo, esto cambió en 1900, cuando el físico alemán Max Planck (Figura 7.11) comenzó a estudiar cuerpos negros (cuerpos que se calientan hasta que empiezan a brillar).

Figura 7.11: Max Planck. Fuente: Bundesarchiv, Bild 183-R0116-504 CC-BY-SA 3.0.

Planck encontró que la radiación electromagnética emitida por un cuerpo negro no podía ser explicada por la física clásica, que postula que la materia puede absorber y emitir cualquier cantidad de radiación electromagnética. Planck observó que, de hecho, la materia absorbía o emitía energía solo en múltiplos enteros del valor $h\nu$, donde ν es la frecuencia de la luz absorbida o emitida y h es la constante de Planck, $6.626 \times 10^{-34} Js$. Este fue un descubrimiento sorprendente, pues desafió la idea de que la energía era continua y que se podía transferir en cualquier cantidad. La realidad, que descubrió Planck, es que la energía no es continua, sino que está cuantizada (es decir, que solo puede transferirse en *paquetes* individuales o -partículas- de tamaño νh). Cada uno de estos paquetes de energía es conocido como **cuanto** (cuantos, en plural).

El descubrimiento de Planck de que la radiación electromagnética está cuantizada cambió para siempre la idea de que la luz se comporta solamente como onda. En realidad, *parecía que la luz tenía tanto propiedades de onda como de partícula.*

7.3.3 El Fotón

Los descubrimientos de Planck pavimentaron el camino para el descubrimiento del fotón. El fotón es la partícula elemental, o cuanto, de la luz. Como pronto veremos, los átomos y las moléculas pueden absorber o emitir fotones. Cuando un átomo o una molécula absorbe un fotón, este le transfiere su energía. Ya que la energía está cuantizada, se transfiere toda la energía del fotón (recuerda que no puede transferirse en fracciones de cuantos, que son los "paquetes de energía" más pequeños posibles). El proceso inverso también es verdadero. Cuando un átomo o una molécula pierde energía, emite un fotón con exactamente la misma cantidad de energía que perdió. Este cambio en la energía es directamente proporcional a la frecuencia del fotón emitido o absorbido, y está dado por la famosa ecuación de Planck:

$$E = h\nu \tag{7.3}$$

donde E es la energía del fotón absorbido o emitido (dada en joules, J), ν es la frecuencia del fotón (dada en hertz, Hz) y h es la constante de Planck ($6.626 \times 10^{-34} Js$).

7.3.4 Relación entre Teoría Ondulatoria y Cuántica

Ahora estamos listos para integrar las naturalezas ondulatoria y cuántica de la luz, el movimiento de toda partícula lleva asociada una onda de longitud λ y de energía E:

De la ecuación 7.2 reorganizamos los componentes para expresar la frecuencia ν en función de la velocidad de la luz c y longitud de onda λ:

$$\nu = \frac{c}{\lambda} \tag{7.4}$$

y reemplazamos en 7.3 tenemos la energía E de la partícula expresada en términos de dos constantes (c y h) y longitud de onda λ:

$$E = h\frac{c}{\lambda} \tag{7.5}$$

Y de la atenta observación de la formula resultante encontramos que a menor longitud de onda mayor energía y viceversa.

7.4 El Espectro Electromagnético

El espectro electromagnético es el conjunto de todas las frecuencias posibles a las que se produce radiación electromagnética.

El límite teórico inferior del espectro electromagnético es 0 (ya que no existen frecuencias negativas) y el teórico superior es ∞. Con los medios técnicos actuales, se han detectado frecuencias electromagnéticas inferiores a $30Hz$ y superiores a $2,9 \times 10^{27} Hz$.

Aunque formalmente el espectro es infinito y continuo, se cree que la longitud de onda electromagnética (distancia entre dos valores de amplitud máxima de la onda) más pequeña posible es la **longitud de Planck** ($l_P \approx 1,616252 \times 10^{-35} m$), distancia o escala de longitud por debajo de la cual se espera que el espacio deje de tener una geometría clásica (medidas inferiores no pueden ser tratadas en los modelos de física actuales debido a la aparición de efectos de gravedad cuántica).

Igualmente, se piensa que el límite máximo para la longitud de una onda electromagnética sería el tamaño del universo.

En el paquete *SpecHelpers* Hanson [2017] encontramos la función `emSpectrum()` que dibuja un espectro eletromagnético:

```
SpecHelpers::emSpectrum(molecular = F, applications = T)
```

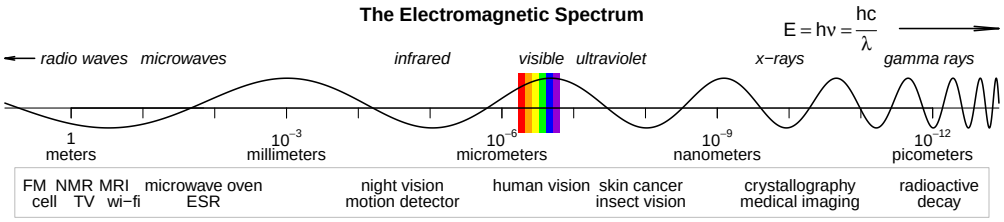


Figura 7.12: Espectro Electromagnético. Gráfico generado por el paquete SpecHelpers.

En orden decreciente de longitud de onda (y creciente de energía) encontramos las regiones llamadas *ondas de radio*, *infrarrojo*, *luz visible*, *ultravioleta*, *rayos x* y *rayos gamma* (ver figura 7.12). Algunas de estas regiones del espectro son útiles para la EO.

7.4.1 Espectro Visible (VIS)

La retina del ojo humano es sensible a las ondas electromagnéticas con frecuencias entre $0.4 - 0.7 \mu m$. Por lo tanto, esta banda

de frecuencias se llama región visible del espectro electromagnético y coincide con las longitudes de onda donde la radiación solar es máxima.

7.4.2 Infrarrojo Cercano (NIR), Próximo o Reflejado

La región del infrarrojo (IR) cercano NIR (del inglés *near* IR) $0.7-1.2\mu m$. Esta parte del espectro se encuentra un poco más allá de la capacidad de percepción del ojo humano y a veces se conoce también como el infrarrojo reflectante o infrarrojo fotográfico, porque parte de este espectro (la región $(0.7 - 0.9\mu m)$ se puede detectar con películas especiales. El NIR es de especial interés debido a su sensibilidad para determinar el estado fitosanitario. También permite diferenciar coberturas vegetales y contenidos de agua.

7.4.3 Infrarrojo Medio (MIR)

Esta región espectral se encuentra entre las regiones NIR y TIR. De $1.2 - 8\mu m$. la influencia de la energía del sol sigue siendo muy relevante, El intervalo entre $1.3 - 2.5\mu m$ es denominado Infrarrojo de Onda Corta (SWIR) y es utilizado para estimar contenidos de humedad en vegetación y suelos. Esta región proporciona las mejores estimaciones del contenido de humedad del suelo y vegetación. De $3-8\mu m$, la señal se vuelve una mezcla continua de energía reflejada por el sol y emitida por la superficie, convirtiéndose el componente emitido más relevante a medida que las longitudes de onda se hacen más largas. El intervalo de $3 - 5\mu m$ es particularmente útil para detectar fuentes de altas temperaturas, como volcanes o incendios forestales.

7.4.4 Infrarrojo Térmico (TIR)

Región $8-14\mu m$. Este es emitido por la energía de superficie de la Tierra que se usa comúnmente para mapear las temperaturas de la superficie. La región termal ha sido ampliamente utilizada para detectar evapotranspiración (ET) vegetal, propiedades del hielo y las nubes, efectos del calor urbano y para la discriminación

de rocas.

7.4.5 Microondas (MW)

Para radiaciones superiores a $1cm$. Esta región espectral es donde funcionan los sistemas de radar. Su principal ventaja es la absorción atmosférica muy baja, lo que nos permite "ver" a través de las nubes. La radiación de MW también puede penetrar en las copas de los bosques a varias profundidades y muy útil en análisis de humedad del suelo y rugosidad superficial. Requiere de la emisión de un haz energético por parte del sensor.

Las señales recibidas por un sensor en estas diferentes regiones espectrales varían con el tipo de cobertura terrestre y con las propiedades biofísicas y bioquímicas de los componentes de la superficie. Pero primero presentamos algunos de los conceptos básicos de energía y unidades de medida utilizados en teledetección para comprender mejor las propiedades características de mediciones espaciales.

7.5 Conceptos Básicos y sus Unidades

7.5.1 Ángulo Sólido

El análisis de un campo de radiación a menudo requiere la consideración de la cantidad de energía radiante confinada a un elemento de *ángulo sólido*. Un ángulo sólido se define como la relación entre el área o de una superficie esférica interceptada en el centro del cuadrado de radio r, como se indica en la figura 7.13 basada en Liou [2002]. Puede escribirse como:

$$\Omega = o/r^2 \tag{7.6}$$

Las unidades de ángulo sólido se expresan en términos de estereorradián (sr). Una esfera cuya área superficial es $4\pi r^2$ el ángulo sólido correspondiente sería de $4\pi sr$.

7.5.2 Magnitudes Radiométricas

Una observación remota requiere una fuente de energía y un sensor que puede detectar dicha energía EM emitida (o reflejada)

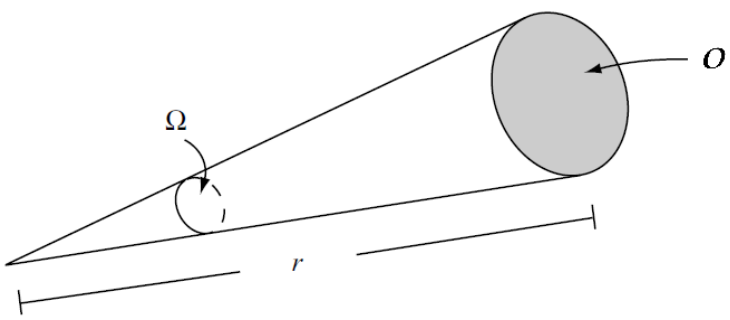


Figura 7.13: Definición de ángulo sólido donde o es el área y r la distancia.

por la superficie de la Tierra hacia el campo de visión del sensor. La energía electromagnética (EEM) de interés tiene una cierta intensidad, composición espectral, y dirección (es decir, la energía puede dirigirse hacia o lejos de la superficie).

A continuación, describimos las unidades comúnmente utilizadas en aplicaciones de EO. Las fórmulas para cada uno de estos términos energéticos se incluyen en el cuadro 7.1.

- **Energía Radiante (Q)**: Medida en julios (J), es la unidad de energía más básica y se refiere a la *energía total irradiada* en todas las direcciones hacia superficie.
- **Flujo Radiante (ϕ)**: medido en vatios (W), es la cantidad de julios por segundo ($Js^{-1} = W$) y representa la tasa de transferencia de energía en todas las direcciones por unidad de tiempo.
- **Densidad de Flujo Radiante**: es la tasa de transferencia de energía por unidad de área medida en vatios por metro cuadrado (Wm^{-2}).
- **Exitancia Radiante o Emitancia (M)**: es la densidad de flujo radiante que sale de la superficie en todas las direcciones por unidad de área y por unidad de tiempo (Wm^{-2}).
- **Irradiancia Radiante (E)**: es la densidad de flujo radiante que incide sobre la superficie por unidad de área y por unidad de tiempo (Wm^{-2}). Es el mismo concepto que la emitancia, pero en este caso se refiere a la energía que llega a la superficie en lugar de aquella que sale de la superficie.

Cuadro 7.1: Magnitudes Físicas.

Concepto	Fórmula	Unidad
Energía Radiante (Q)		julios (J)
Flujo Radiante (ϕ)	$\partial Q/\partial t$	vatios (W)
Irradiancia (E)	$\partial\phi/\partial A$	Wm^{-2}
Emitancia (M)	$\partial\phi/\partial A$	Wm^{-2}
Intensidad Radiante (I)	$\partial\phi/\partial\Omega$	Wsr^{-1}
Radiancia (L)	$\partial I/\partial A\ cos(\Theta)$	$Wm^{-2}sr^{-1}$
Radiancia Espectral (L_λ)	$\partial L/\partial_\lambda$	$Wm^{-2}sr^{-1}\mu m^{-1}$
Emisividad (ϵ)	M/M_n	-
Reflectividad (ρ)	ϕ_r/ϕ_i	-
Absortividad (α)	ϕ_a/ϕ_i	-
Transmisividad (τ)	ϕ_t/ϕ_i	-

Nota:

t: Unidad de Tiempo.

A: Unidad de Área.

Ω: Ángulo sólido.

Θ: Ángulo entre dirección del flujo enegético y la normal.

ϕ_r: Flujo Reflejado.

ϕ_a: Flujo Absorbido.

ϕ_i: Flujo Incidente.

ϕ_t: Flujo Transmitido.

- **Intensidad Radiante (I)**: es la energía total que sale de la superficie por unidad de tiempo y dentro de una unidad de ángulo sólido (Ω). Así, la intensidad radiante se mide en vatios por estereorradián (Wsr^{-1}).

- **Radiancia (L)**: es la energía total que sale en una determinada dirección por unidad de área y ángulo sólido. Es el término más fundamental en teledetección, ya que describe **exactamente lo que mide el sensor**. La radiancia es expresada en vatios por metro cuadrado por estereorradián ($Wm^{-2}sr^{-1}$).

Si expresamos la radiancia en función de las longitudes de onda a sensar podemos expresar como:

$$L\left(\lambda_1, \lambda_2\right) = \int_{\lambda_1}^{\lambda_2} L_\lambda \partial\lambda \tag{7.7}$$

Donde vemos que es la integral (sumatoria continua) de energía entre dos longitudes de onda (rango válido del sensor).

Así, los términos de energía anteriores también se pueden expresar en base a longitudes de onda y tienen el sufijo *espectral* aplicado a ellos, como *radiancia* o *irradiancia espectral*.

Por ejemplo, el término *radiación espectral*, L_λ, se refiere a la producción de energía por una unidad de área por unidad de ángulo sólido y longitud de onda o $L_\lambda = Wm^{-2}sr^{-1}\lambda^{-1}$. De manera similar, la irradiancia espectral, $E\lambda$, se refiere a la energía incidente sobre una superficie por unidad de longitud de onda.

También hay una serie de términos de energía adimensional, que varían de 0 a 1, que son ampliamente utilizados para caracterizar las propiedades espectrales de la superficie de la Tierra:

- **Emisividad (ϵ)**: es la relación entre la salida radiante de un superficie (M) relativa a la de un emisor perfecto a la misma temperatura (M_n). Un emisor perfecto también se conoce como cuerpo negro y tiene emisividad 1. Los materiales naturales, por otro lado, son emisores imperfectos con emisividad van de 0 a <1. Valores de emisividad en diferentes longitudes de onda son útiles para caracterizar materiales.

- **Reflectividad (ρ)**: es la relación entre la energía reflejada por una superficie y la energía incidente sobre esa superficie.

- **Absortividad (α)**: es la relación entre la energía absorbida por la superficie y la energía incidente sobre esa superficie.
- **Transmitividad (τ)**: es la relación entre la energía transmitida a través de una superficie y la energía incidente sobre esa superficie.

Es importante destacar que por el principio de conservación de la energía tenemos la relación:

$$\alpha + \rho + \tau = 1 \tag{7.8}$$

A estos términos sin unidades también se les puede agregar el sufijo *espectral*, como en la *reflectancia espectral*. Hay algunas relaciones útiles que se derivan de los términos energéticos mencionados anteriormente.

El término *albedo* es la relación entre toda la energía saliente y la energía incidente para una superficie determinada. Más específicamente, albedo es la relación de la *emitancia* (M) sobre la *irradiancia* (E) sobre todas las longitudes de onda reflectantes solares o de onda corta y es el equivalente de la reflectancia hemisférica, es decir, la reflectancia integrada en todas las direcciones (Ecuación 7.9).

$$ALBEDO = \rho_{hemisferico} = \frac{M}{E} \tag{7.9}$$

El albedo es una variable fundamental en estudios de balance energético, modelización climática y estudios de degradación del suelo. Albedo espectral se refiere a la salida dividida por la irradiancia para una banda espectral específica (Ecuación 7.10).

$$ALBEDO\ ESPECTRAL = \rho_{hemisferico,\lambda} = \frac{M_\lambda}{E_\lambda} \tag{7.10}$$

Es importante tener en cuenta que un sensor satelital no mide las señales hemisféricas de la energía emitidas en todas las direcciones, sino que mide la radiancia espectral ($L\lambda$) desde un campo de visión angular estrecho. La radiancia direccional espectral ($L\lambda$) está relacionada con la emitancia espectral hemisférica ($M\lambda$) de la siguiente forma (Slater [1980]):

$$M_\lambda = \pi L_\lambda \tag{7.11}$$

De manera similar, las reflectancias de la superficie derivadas de las mediciones satélitales son reflectancias direccionales, ya que se refieren a una geometría de medición específica entre sensor de satélite y el Sol, en relación con la superficie. La relación entre el valor de radiancia espectral recibidos por *un sensor de satélite* ($L_{sen(\lambda)}$) y la reflectancia espectral de superficie (ρ_λ) se convierte en:

$$\rho_\lambda = \frac{\pi L_\lambda}{E_{0,\lambda}} \tag{7.12}$$

donde $E_{0,\lambda}$ es la irradiancia solar que llega a la superficie.

7.5.3 Importancia de la Temperatura

Las temperaturas son expresadas en unidades **Kelvin** en honor al físico británico Willian Thomson primer Barón de Kelvin (Figura 7.14) quien determinó y fijó el cero absoluto en los $-273,15°C$. La escala de temperatura de Kelvin constituye la escala natural en la que se anotan las ecuaciones termodinámicas y la unidad de temperatura en el Sistema Internacional de Unidades.

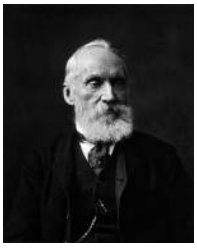


Figura 7.14: Fotografía de William Thomson, primer Barón Kelvin. Fuente: Librería Smithsoniana.

7.6 Radiación de Cuerpo Negro

Las leyes de la radiación del cuerpo negro son básicas para comprender la *absorción* y procesos de *emisión*. Un *cuerpo negro* (Figura 7.15) es un concepto básico en física y se puede visualizar considerando una cavidad con un pequeño orificio de entrada. La mayoría del flujo radiante que ingresa a este orificio desde el exterior quedará atrapada dentro de la cavidad, independientemente del material y las características superficiales de la pared. Los reflejos se repiten internamente ocurren hasta que todos los flujos son absorbidos por la pared. La probabilidad de que cualquiera de los flujos entrantes escapará a través del orificio es tan pequeño que el interior aparece oscuro. El término *cuerpo negro* se utiliza para una configuración de material donde la absorción será completa.

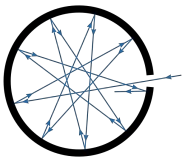


Figura 7.15: Idealización aproximada de un cuerpo negro como un pequeño agujero en un recinto aislado. Fuente:AG Caesar.Wikipedia.

La emisión de un cuerpo negro es lo opuesto a la absorción. El flujo emitido por cualquier área pequeña de la pared se refle-

ja repetidamente y en cada encuentro con la pared, el flujo es debilitado por la absorción y reforzado por una nueva emisión. Después de numerosos encuentros, la emisión y la absorción alcanzan una condición de equilibrio con respecto a la temperatura de la pared.

A continuación, presentamos cuatro leyes fundamentales que gobiernan la radiación del cuerpo negro, comenzando con la ley de Planck.

7.6.1 Ley de Planck

La cantidad de energía que contiene un flujo radiante es inversamente proporcional a su longitud de onda. La distribución espectral de dicha radiación EM emitida por un cuerpo negro (un emisor perfecto) se puede caracterizar por la ley de radiación de Planck (Figura 7.11) como sigue:

$$M_{n,\lambda} = \frac{C_1}{\lambda^5 e^{\left(\frac{c_2}{\lambda T}\right)} - 1} \tag{7.13}$$

Donde:

$M_{n,\lambda}(W\ m^{-2}\ \mu m^{-1})$ indica la emitancia espectral en una longitud de onda λ en μm.

Constantes $C_1 = 3,741 \times 10^8\ Wm^{-2}\mu m^4$ y $C_2 = 1,438 \times 10^4\ \mu m\ K$.

T temperatura en K.

Esta ecuación describe la distribución espectral de salida de un cuerpo negro a cierta temperatura como una curva suave con un único máximo (Figura 7.16). La ecuación indica que cualquier objeto más caliente que el cero absoluto $(-273,15°C)$ emite energía radiante y que la energía aumenta en proporción a su temperatura.

Veamos un caso práctico, la temperatura promedio del planeta tierra se ha fijado en $288K$ y podemos calcular usando la ley de Planck las longitudes de onda dónde se produce el máximo de energía radiante:

```
t_k <- 288
lambda <- seq(8,15,1)
```

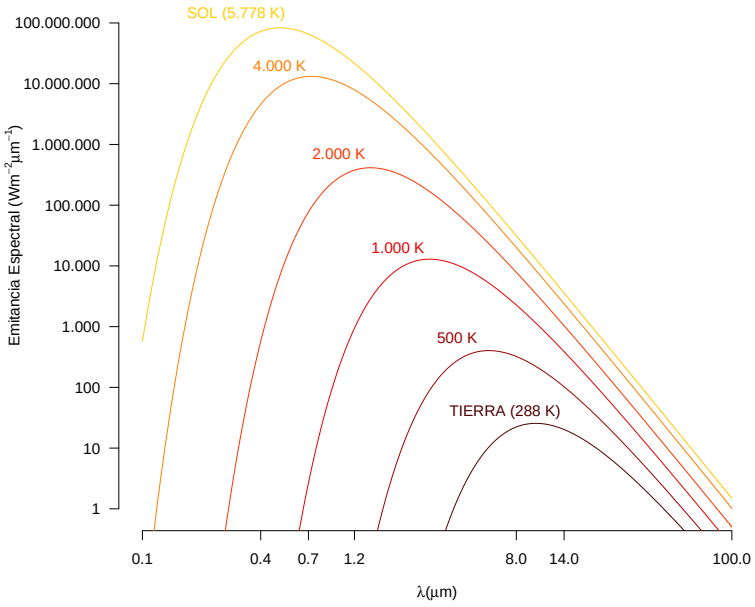


Figura 7.16: Curvas de Emitancia Espectral de Cuerpo Negro a varias temperaturas.

```
M <- 3.741e8/(lambda^5 * (exp(1.438e4/(lambda * t_k))-1))
names(M) <- lambda
round(M,3)
```

```
     8      9     10     11     12     13     14     15
22.275 24.778 25.556 25.082 23.816 22.113 20.225 18.312
```

Encontramos el máximo de energía radiante entorno a los 11 μm que cae dentro del rango del espectro que hemos denominado TIR o infrarrojo térmico (ver 7.4.4).

Y en el caso de sol cuya temperatura estimada es $5778K$:

```
t_k <- 5778
lambda <- seq(0.0,2,0.5)
M <- 3.741e8/(lambda^5 * (exp(1.438e4/(lambda * t_k))-1))
names(M) <- lambda
round(M/1e6,0)
```

```
  0 0.5   1 1.5   2
NaN  83  34  12   5
```

Encontramos los máximos de energía radiante en las longitudes de onda del rango visible (ver 7.4.1) e infrarrojo cercano (ver

7.4.2).

7.6.2 Ley de Desplazamiento de Wien

El sol como fuente primaria de energía se encuentra a una temperatura de aproximadamente 5778 K y de acuerdo a la ley de radiación del *cuerpo negro de Planck* (ver 7.6) donde se relaciona la cantidad energía que emite un cuerpo con la distribución espectral (longitudes de onda de los máximos (Figura 7.16), caen en el que llamamos espectro visible (ver 7.4.1) y por eso nuestra visión se ha adecuado precisamente a ese rango.

Figura 7.17: Wilhelm Wien, físico esloveno, fuente: nobelprize.org.

La superficie terrestre tal como se ve en la figura 7.16 tiene sus máximos alrededor de las longitudes de onda en el rango 8 – 14 μm identificado anteriormente como TIR (ver 7.4.4).

Como se aprecia en estos ejemplos, la temperatura es clave al momento de definir el máximo de emitancia de un objeto (por lo tanto, el mejor rango de longitud de onda para diseñar nuestro sensor) ya que todo objeto con temperatura superior a los 0 K emite energía, requisito básico para ser percibido por dicho sensor.

La ley de desplazamiento de Wien (Figura 7.17) establece que la longitud de onda de la intensidad máxima de la radiación del cuerpo negro λ_{max} es inversamente proporcional a la temperatura:

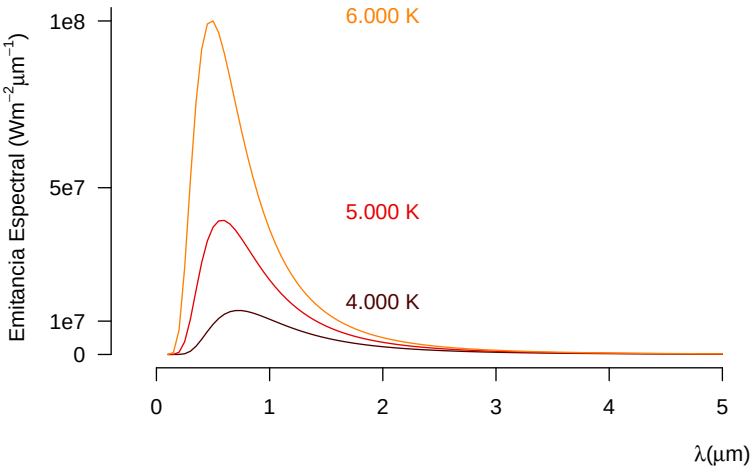


Figura 7.18: Longitud de Onda y Emitancia Espectral de Cuerpo Negro a varias temperaturas.

$$\lambda_{max} = \frac{2.897,6\ \mu m\ K}{T} \tag{7.14}$$

Donde T es la temperatura en unidades Kelvin.

La dependencia de la posición de la intensidad máxima de la temperatura es evidente a partir de las curvas de cuerpo negro que se muestran en la figura 7.18.

Si la temperatura del sol la estimamos en 6000 K (la fotosfera) encontramos el pico de longitud de onda λ_{max} es:

```
t_sol <- 6000
l_max <- 2898 / t_sol
l_max
```

```
[1] 0.483
```

Valor expresado en μm (Lo ubica dentro del rango visible del espectro).

7.6.3 Ley de Stefan-Boltzmann

Figura 7.19: Jožef Stefan, físico esloveno, fuente: Grabado por K. Schönbauer.

Para conocer el total de la energía radiante espectral de un cuerpo negro se debe integrar la curva de Planck sobre el dominio de las longitudes de onda de 0 a ∞.

$$M(T) = \sigma\ T^4 \tag{7.15}$$

Donde σ es la constante de Stefan-Boltzmann:

$5,67 \times 10^{-8}\ W\ m^{-2}\ K^{-4}$ y la temperatura T en Kelvin.

La ley fue deducida en 1879 por el físico austriaco Jožef Stefan (Figura 7.19) y la ley fue derivada en 1884 a partir de consideraciones teóricas por Ludwig Boltzmann (Figura 7.20).

Figura 7.20: Ludwig Eduard Boltzmann, físico austríaco, fuente: Universität Wien.

La emitancia total de un objeto es función de su temperatura. Pequeños cambios en la temperatura suponen cambios notables en su emitancia radiante. Debido a esta dependencia en la cuarta potencia, la emisión de radiación por los cuerpos terrestres cambia considerablemente durante el día o a través de las estaciones.

7.6.4 Ley de Kirchhoff

Las tres leyes fundamentales anteriores se refieren a la intensidad radiante emitida por un cuerpo negro, que depende de la

longitud de onda emisora y la temperatura del medio. Un medio puede absorber radiación de una longitud de onda particular, y al mismo tiempo también emiten radiación de la misma longitud de onda. La razón a la cual la emisión toma lugar es función de la temperatura y la longitud de onda. Esta es la propiedad fundamental de un medio en condiciones de *equilibrio termodinámico*. La declaración física en cuanto a absorción y emisión fue propuesta por primera vez por el físico Alemán Gustav Kirchhoff (Figura 7.21).

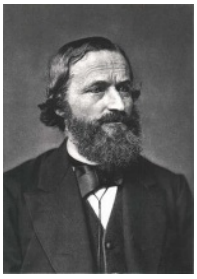


Figura 7.21: Gustav Robert Kirchhoff fuente: Librería Smithsoniana.

Para comprender el significado físico de la ley de Kirchhoff, consideremos un recinto aislado con paredes negras. Suponga que este sistema ha alcanzado el estado de equilibrio termodinámico caracterizado por una temperatura uniforme y una radiación isotrópica. Debido a que las paredes son negras, la radiación emitida por el sistema es absorbido por las paredes. Además, debido a que existe un equilibrio, la misma cantidad de radiación absorbida por las paredes también es emitida. Dado que el cuerpo negro absorbe el máximo posible radiación, tiene que emitir la misma cantidad de radiación. Si emitiera más, el equilibrio no sería posible y esto violaría la segunda ley de la termodinámica.

La radiación dentro del sistema se conoce como radiación de cuerpo negro como ya se ha mencionado, y la cantidad de intensidad radiante es una función de la temperatura y longitud de onda.

Sobre la base de la discusión anterior, la emisividad de una longitud de onda dada, ϵ_λ, de un medio es igual a la absortividad, α_λ ", de ese medio en equilibrio termodinámico. Por lo tanto, podemos escribir:

$$\epsilon_\lambda = \alpha_\lambda \tag{7.16}$$

Un medio con una absortividad α_λ absorbe solo α_λ veces la intensidad radiante del cuerpo negro, por lo tanto, emite ϵ_λ veces la intensidad radiante del cuerpo negro.

Si en la ecuación 7.8 en su versión espectral, hacemos la transmitividad $\tau_\lambda = 0$ (suponiendo un objeto opaco) y usando $\epsilon_\lambda = \alpha_\lambda$ reemplazamos resulta:

$$\rho_\lambda + \epsilon_\lambda = 1 \tag{7.17}$$

Entonces, para mantener el equilibrio térmico, el exceso de energía absorbida debe ser emitido:

- Las superficies con alta reflectividad espectral (por ejemplo, nieve) son muy poco emisivas.
- Las superficies con baja reflectividad espectral (agua) son muy emisivas.

EEM

7.7 Flujo de la EEM

En la teledetección como en la EO, la energía que emana de la superficie de la tierra se mide utilizando un sensor montado en una plataforma de aeronave o nave espacial. Esa medida se usa para construir una imagen del paisaje debajo de la plataforma.

En principio, cualquier energía procedente de la superficie terrestre puede utilizarse para formar una imagen. La mayoría de las veces se refleja la luz del sol, por lo que la imagen grabada es, en muchos sentidos, similar a la vista que tendríamos de la superficie de la tierra desde un avión, aunque las longitudes de onda utilizadas en la teledetección suelen estar fuera del alcance de la visión humana.

La energía recibida también podría provenir de la propia tierra que actúa como un radiador debido a su propia temperatura. Alternativamente, podría ser energía que se dispersa hasta el sensor al haber sido irradiado a la superficie por una fuente artificial, como un láser o un radar.

Siempre que se disponga de una fuente de energía, se podría utilizar casi cualquier longitud de onda para imaginen las características de la superficie terrestre. Sin embargo, existe una limitación, particularmente cuando se toman imágenes de altitudes de las naves espaciales. La atmósfera terrestre no permite el paso de radiación en todas las longitudes de onda. Energía en algunas longitudes de onda son absorbidas por los constituyentes moleculares de la atmósfera.

Las longitudes de onda para las que hay poca o ninguna absor-

ción atmosférica forman las que son llamadas *ventanas atmosféricas* (ver 7.8.2) y encontramos cierto número de dichas ventanas en las regiones visible e infrarroja del espectro. Las longitudes de onda utilizadas para la obtención de imágenes en la teledetección están claramente restringidas a estas ventanas atmosféricas. Incluyen las llamadas longitudes de onda *ópticas* que cubren el visible y el infrarrojo, las longitudes de onda térmicas y las longitudes de onda de radio que se utilizan en imágenes de radar y microondas pasivas de la superficie terrestre.

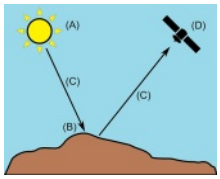
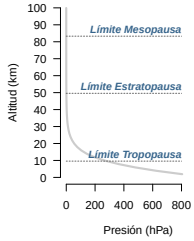


Figura 7.22: Esquema básico del flujo de la EEM en la EO. Basado en Olaya [2020]

Cualquiera que sea el rango de longitud de onda que se utilice para obtener imágenes de la superficie terrestre, el sistema es muy complejo, basado en Olaya [2020] podemos distinguir a los menos tres elementos principales en el ciclo o flujo de la EEM.

Siguiendo la figura 7.22 encontramos:

Una **Fuente de radiación (A)**, que puede ser de origen natural o artificial. La radiación emitida por dicha fuente viaja a través de un medio gaseoso, **la atmósfera (C)** y llega a la **superficie terrestre (B)** y sufre perturbaciones causadas por los elementos presentes en ella. Los *propios objetos pueden ser también* emisores de radiación que nuevamente viajan por la atmósfera (C) hasta **el receptor (D)**.



7.8 Interacción EEM con la Atmósfera

7.8.1 La Atmósfera

El 97% de la masa atmosférica se encuentra entre la superficie terrestre y los 25 km de altura. La densidad varía a medida que ascendemos producto de la gravedad, menor presión a medida que ascendemos (Figura 7.23, superior).

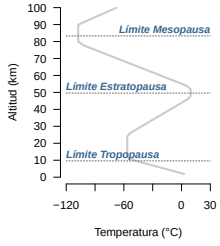


Figura 7.23: Modelo Atmósfera Estándar, Presión (superior) y Temperatura (inferior) versus Altitud.

La temperatura varía con la altitud debido a la presión, en la troposfera la temperatura disminuye con la altitud, en la estratosfera la temperatura permanece constante para luego aumentar con la altitud, en la mesósfera disminuye, en la termósfera aumenta con la altitud (Figura 7.23, inferior).

La atmósfera se compone principalmente de gases como el N_2 (78%), O_2 (21%), Ar (0,9%), CO_2 (0,03%), algunas tra-

zas de gases nobles y Ozono (O_3), también vapor de agua (H_2O) y Aerosoles: partículas sólidas como humo o polvo o partículas líquidas dispersas en el medio gaseoso como gotas de agua.

Y todo el conjunto reduce el total de energía incidente de $\sim$ 340 Wm^{-2} a sólo $\sim$ 173 Wm^{-2} que llegan a la superficie terrestre.

En conjunto todos estos fenómenos deben ser considerados al momento de interpretar las señales de respuestas de los objetos y la superficie terrestre capturados en los sensores correspondientes. Vamos a clasificar en tres los efectos de la atmósfera:

- **Absorción** de EEM en determinadas bandas del espectro (ver figura 7.24).
- **Dispersión** de determinadas magnitudes de longitud de onda causadas por los aerosoles y nubes.
- **Emisión** de EEM por cualquier cuerpo sobre el cero absoluto adicionales al flujo radiante del sol.

7.8.2 Absorción

La *absorción* implica que algunas moléculas presentes en la atmósfera absorben algunas longitudes de onda, quedando el resto intactas. Las tres moléculas presentes en la atmósfera que contribuyen en mayor medida a la absorción de radiación son el *ozono* (O_3), el *dióxido de carbono* (CO_2)y el *vapor de agua* (H_20).

El **ozono** absorbe radiación en el rango del *ultravioleta*. Este tipo de radiación resulta perjudicial para los seres humanos, por lo que la desaparición del ozono causaría que la luz del sol produjese quemaduras en nuestra piel. El **dióxido de carbono** absorbe radiación en el rango del *infrarrojo*. Este tipo de radiación está asociada a fenómenos térmicos, por lo que el gas contribuye a que cierta cantidad de calor quede atrapado en la atmósfera. El **vapor de agua** absorbe principalmente *microondas*, y su distribución varía enormemente a lo largo del planeta. En las capas de aire por encima de un desierto, por ejemplo, hay una cantidad muy pequeña de vapor de agua, mientras que sobre los trópicos esta cantidad es mucho más grande.

La existencia de moléculas en la atmósfera capaces de absor-

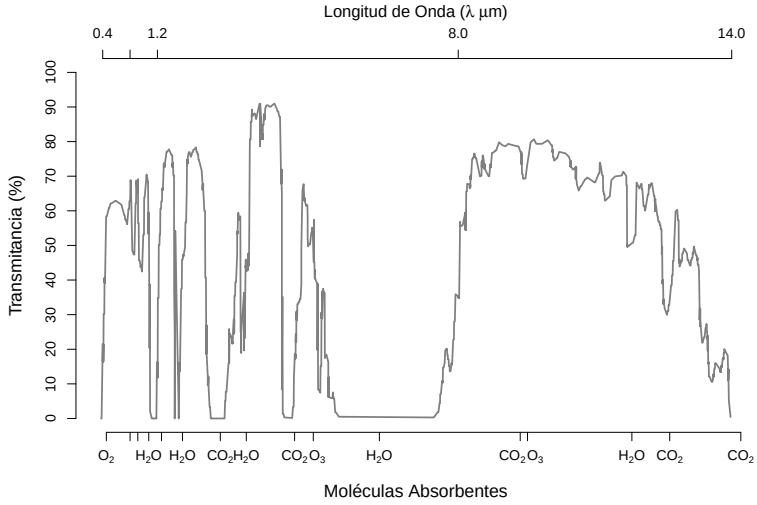


Figura 7.24: Transmitancia de la atmósfera terrestre y su relación con las moléculas absorbentes.

ber radiación electromagnética limita el rango de longitudes de onda que pueden aplicarse a la medición remota de propiedades atmosféricas. Las zonas del espectro en las que no se produce absorción se suelen denominar **ventanas atmosféricas** (Figura 7.24). En dichas ventanas la *transmitancia* de la atmósfera terrestre es máxima.

En la parte inferior de la figura se indican las moléculas de gas presentes en la atmósfera y su ubicación relativa de aquellas longitudes de onda que son absorbidas. Mientras que en la parte superior se representan las longitudes de onda límite de las zonas que hemos estudiado.

7.8.3 Dispersión

La dispersión de la EEM es causada por el reflejo de la radiación solar entrante por gases, aerosoles y vapor de agua presentes en la atmósfera. Como resultado, la radiación detectada por el sensor es una mezcla de energía reflejada tanto por la superficie como por la atmósfera, por lo tanto, contiene un ruido indeseable que se debe eliminar durante la recuperación de las propiedades superficiales.

En la superficie, la *irradiancia* directa se reduce, mientras que la difusa aumenta el resplandor (procedente de otros objetos).

Los **aerosoles** se originan tanto por causas naturales como humanas. Pueden ser oceánicos, causado por el movimiento del agua, o continental, como en polvo en suspensión o partículas emitidas por combustión. Debido a que estas partículas atmosféricas son muy variables en el tiempo y espacio, la corrección de la dispersión atmosférica es bastante problemática. Sin embargo, este factor se debe tener en cuenta siempre que se realice una estimación precisa de la radiancia del suelo.

Se requiere corregir, por ejemplo, cuando las medidas desde el satélite se comparan con las mediciones radiométricas terrestres o cuando intentamos detectar cambios entre dos imágenes adquiridas con varios años de diferencia. Normalmente, las mediciones atmosféricas *in situ* no están disponibles en tiempos de adquisición de satélites y, por lo tanto, la corrección de la atmósfera a menudo debe depender de datos extraídos de la propia imagen. Algunos métodos pueden implicar cambios en la *reflectancia*.

Según su origen y las características de los aerosoles, varían mucho en tamaño, lo que causa diferentes tipos de dispersión, ya que este proceso es altamente dependiente del diámetro de las partículas dispersas (Figura 7.25) basada en Richards [2013]).

Los tres tipos principales de dispersión atmosférica son:

- **Rayleigh** que es causada por partículas más pequeñas que la longitud de onda de la radiación.
- **Mie** cuando la partícula es de tamaño similar a la longitud de onda.
- **No Selectiva** cuando el tamaño de la partícula es mayor que la longitud de onda.

La dispersión de Rayleigh depende en gran medida de la longitud de onda y afecta principalmente a las bandas más cortas. Provoca, por ejemplo, el color azul del cielo porque el azul es la longitud de onda más baja del espectro VIS.

La dispersión de Mie también depende de la longitud de onda de la radiación entrante, aunque en menor grado que la dispersión de Rayleigh. Aerosoles y polvo atmosférico son la principal causa de este tipo de dispersión, aunque también es causada por los incendios forestales o neblina costera.

Finalmente, la dispersión no selectiva afecta por igual a varias

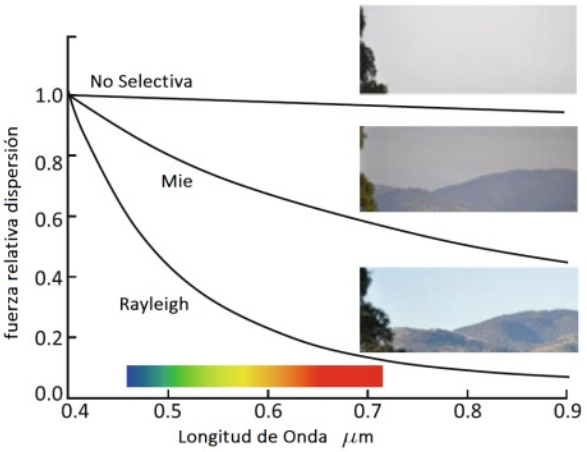


Figura 7.25: Dispersión atmosférica y de partículas. Modificado de Richards(2013)

longitudes de onda. Es decir, dispersan las diferentes bandas del espectro VIS de forma similar por ejemplo, las nubes y la niebla tienden a aparecer en tonos grises ya dispersan todos los colores.

7.8.4 Emisión

También hay influencias de las emisiones atmosféricas que son importantes principalmente en el rango termal (TIR). Esta debe corregirse para obtener con precisión la temperatura de la superficie a partir de imágenes adquiridas por satélite. Todos los materiales con una temperatura superior al cero absoluto, incluyendo la atmósfera emiten energía radiante que viajará en la dirección del sensor y será detectado. Así, al igual que con la dispersión atmosférica, la señal de la emisión de la atmósfera debe ser separada de la señal térmica que emite la superficie (Figura 7.26).

7.9 Últimos pasos

Luego de este recorrido de la EEM desde el emisor hasta la superficie terrestre y de regreso al sensor montado en un vehículo espacial cruzando la atmósfera y sufriendo todos los efectos que hemos descrito la EEM impacta en un sensor electrónico y almacena esa magnitud convertida en un valor numérico.

Figura 7.26: Emisión TIR, adicional a la emisión general. Imagen Sentinel-2 sobre el cráter Pu'u 'O'o' del complejo volcánico Kilauea (23 mayo 2018), fuente: ESA.

Finalmente, el receptor retransmite la información a la tierra y la data capturada va a generar como producto final una imagen (un objeto ráster), en cuyas celdas o píxeles va a contener un valor que representan la **intensidad de la radiación**.

El almacenamiento de datos en unidades de radiancia es difícil y, por lo tanto, los sensores traducen la radiancia medida en *números o niveles digitales* (DN). Por lo tanto, el rango de energía medido por el sensor se divide en distintas unidades (los números o niveles digitales).

La calibración específica del sensor determina la cantidad mínima y máxima de radiación que se puede medir. Los DN expresan la cantidad de radiación en relación con estos coeficientes de calibración específicos del sensor. El número total de DN posibles es lo que denominamos **resolución radiométrica** (ver 7.12) del sensor.

7.9.1 DN a TOA

Como los DN dependen de la calibración del sensor, a mediciones idénticas producen diferentes DN entre los sensores. *Los DN no son físicamente significativos*. En lugar de DN, a menudo nos interesa la **reflectancia**.

La reflectancia expresa la fracción de radiación reflejada en

relación con la energía entrante total (es decir, del sol) y se escala entre 0 y 1. Muchas aplicaciones requieren datos convertidos a reflectancia y / o corregidos por la influencia de la atmósfera (Figura 7.27).

Los siguientes pasos son necesarios para convertir DN a (corregido atmosféricamente) reflectancia del fondo de la atmósfera (BOA):

- Calibración del sensor (DN a radiancia).
- Conversión a reflectancia en la parte superior de la atmósfera (radiancia a TOA).
- Corrección atmosférica que produce reflectancia del fondo de la atmósfera (TOA a BOA).

Se puede realizar fácilmente una conversión de DN en luminosidad. Teniendo en cuenta las geometrías de los sensores solares y la irradiancia solar, podemos inferir fácilmente la reflectancia de la parte superior de la atmósfera (TOA) a partir de la radiación. Esto facilita, por ejemplo, una comparación de medidas de diferentes sensores.

En la práctica, esto implica un escalado lineal de los valores DN que utiliza dos factores de cambio de escala específicos de la banda. Uno es multiplicativo, uno es aditivo (ver un caso práctico en las imágenes Landsat en 8.11.1). Con los datos de Landsat Colección 2 Nivel 2 (ver 8.5) (¡este no es el caso para todos los sensores!), Podemos usar un solo conjunto de coeficientes para convertir DN directamente a reflectancia TOA.

7.9.2 TOA a BOA

La **corrección atmosférica** es el paso final hacia *mediciones comparables* de la superficie de la Tierra. Al hacer la corrección atmosférica, (teóricamente) eliminamos la influencia de la atmósfera. Por lo tanto, llamamos al producto *reflectancia del fondo de la atmósfera* (BOA). En teoría, ***el BOA medido por un sensor espacial debería ser comparable a las mediciones terrestres***. En el caso práctico de las imágenes Landsat Colección 2 Nivel 2 (ver 8.5) los pasos descritos ya han sido realizados y toda la data disponible está procesada a nivel de BOA.

Figura 7.27: Comparación BOA-TOA. Imagen Landsat 8 RGB, TOA (izquierda). BOA (derecha). Área en Nepal mayo3, 2013 (fuente: Michelle Bouchard, USGS)

Resoluciones de Imagen

Las especificaciones técnicas de la plataforma espacial y las del sensor mismo determina las *resoluciones*, espacial, espectral, radiométrica, angular y temporal (Lia [2020]) de los datos sensados.

Los distintos tipos de resolución están estrechamente interconectados. La adquisición de imágenes de alta resolución espacial suele implicar reducir la resolución temporal y/o resolución espectral. Las limitaciones en la transmisión de datos o grabación a bordo, un sensor será capaz de optimizar solo uno o dos de los tipos de resolución. Incrementar cualquiera de los cinco tipos de resoluciones implica una considerable cantidad de transmisión y procesamiento de datos adicionales, tanto para el sensor como las instalaciones terrestres de recepción.

La misión de un sistema de teledetección intenta enfatizar la resolución particular que mejor se adapta a sus requisitos. Si el sistema está diseñado para detectar *fenómenos dinámicos*, una buena cobertura temporal es fundamental, incluso sacrificando la resolución espacial (satélites meteorológicos). Si la misión está orientada a la exploración minera, los detalles espectrales y espaciales son más importantes, mientras que la cobertura temporal se vuelve menos significativa.

Finalmente, identificar la calidad de un sensor basado en su resolución espacial es muy simplista. Se requiere una alta resolución espacial para ciertas aplicaciones (por ejemplo, cartografía urbana o agrícola), pero puede ser secundaria para otras aplicaciones, como detección de incendios o estudios geológicos, evaluación de riesgos naturales, por ejemplo, requieren una alta resolución tem-

poral, al igual que el pronóstico del clima. Otras aplicaciones de teledetección deben optimizar la resolución espectral, como en la estimación de la productividad de los cultivos o la demanda de agua. En consecuencia, el tipo de resolución prioritaria es fundamental para seleccionar el sensor más adecuado para una aplicación particular.

7.10 Resolución Espacial

La resolución espacial es la medida más pequeña de un objeto que puede ser resuelto (capacidad de separar objetos cercanos y que aparezcan independientes y aislados) por el sensor, o el área del suelo en el campo instantáneo de vista o (*instantaneous field of view, IFOV*)[1] del sensor, o la dimensión lineal en el suelo representado por cada píxel.

[1] El **IFOV** Se mide en metros en el suelo, y depende de la distancia focal de la cámara y la altura de la cámara sobre el suelo. Se define como la sección angular observada por el sensor en un momento dado en el tiempo.

La resolución espacial de los sensores electrónicos ópticos depende de la *altura orbital*, *número de detectores*, *longitud focal* y *configuración del sistema*. Los sensores de observación de la Tierra en funcionamiento cubren una amplia gama de resoluciones espaciales (ver cuadro 7.2).

- ***Alta*** con un tamaño de píxel de $0.5 - 4.0\ m$.
- ***Media*** con un tamaño de píxel de $10 - 60\ m$.
- ***Gruesa*** con un tamaño de píxel de $100 - 1000\ m$.
- ***Muy Gruesa*** con un tamaño de píxel de $10 - 50\ km$.

Sensores orientados a *aplicaciones globales* tienen un tamaño de píxel entre $200-1000\ m$ y sensores orientado al *estudio atmosférico* suelen tener una resolución espacial muy gruesa $10-50\ km$.

La *resolución espacial* juega un papel importante en la interpretación de imágenes porque afecta el nivel de detalle alcanzado. Solo puede identificar objetos varias veces más grande que el tamaño del píxel, aunque se pueden detectar características más pequeñas cuando existe suficiente contraste radiométrico entre el objeto y el fondo (por ejemplo, un incendio de alta temperatura puede detectarse incluso si su tamaño es sólo un pequeño proporción del área total de píxeles).

La selección de la resolución espacial más conveniente está estrechamente relacionada con la escala adecuada para el problema

particular en estudio. Además, resolución espacial afecta la **pureza de señal de píxel**. Cuanto menor sea el tamaño del píxel, menor será la probabilidad de que el píxel sea una *combinación de dos o más tipos de cobertura*. Un píxel mixto probablemente tendrá una señal promedio de las coberturas terrestres presentes dentro del IFOV. Como resultado, ese píxel puede no parecerse a ninguna de las categorías que componen la mezcla, lo que hará su identificación más difícil.

Cuadro 7.2: Resoluciones de las misiones y sensores cuyos productos serán descritos y procesados con R en el texto.

Nivel	Satélite	Sensor	Bits	Días	Período	Espacial	Bandas
Gruesa	Terra/Aqua	MODIS	12	1	1999-	250	B1-B2
						500	B3-B7
						1000	B8-B36
	Sentinel-3	SLSTR	16		2016-	300	LST
	Sentinel-5p	TROPOMI			2018-	1100	VARIOS
Media	Landsat 9	OLI-2 TIR-2	14	16	2021-	30	B1-B9,B10-B11
	Landsat 8	OLI TIRS	16		2013-		B1-B7,B10
	Landsat 7	ETM+			1999-		B1-B6
	Landsat 5	TM			1984-2012		B1-B6
	Landsat 4				1982-1993		B1-B6
	Sentinel 2/2A	MSI	12	5	2015-	60	B1,B9,B10
						10	B2-B4,B8
						20	B5-B7,B8a,B11-B12

Nota:
Los productos Landsat (1-8) hacen referencia al nuevo set distribuido 'Level-2 Science Products'.

7.11 Resolución Espectral

La resolución espectral se refiere al número de bandas proporcionadas por el sensor y sus anchos de banda espectrales. En términos generales, un sensor proporcionará una mejor capacidad de discriminación a medida que se adquieren más bandas. Idealmente, esas bandas espectrales deberían ser lo suficiente estrechas como para identificar características de absorción específicas que de otra manera podrían verse borrosas.

Entre los sistemas de satélites EO, la peor resolución espectral se encuentra en las imágenes denominadas **pancromáticas**

(una sola banda, escala de grises) o imágenes en color RGB (tres bandas, Red, Green, Blue).

Los sensores remotos espaciales son generalmente:

- **pancromáticas** una banda (generalmente de mayor resolución espacial.
- **multiespectral** capturan algunas pocas bandas, entre $4 - 7$.
- **hiperespectrales** decenas o centenas de bandas.
- **ultraespectrales** con miles de bandas. En la actualidad se trabaja en el diseño y pruebas de sensores de este tipo.

Entra las mayores resoluciones entre los sensores de satélites EO se encuentra el *Hyperion* a bordo del satélite EO-1 con 220 bandas que cubren de 0.4 a 2.5 μm. El espectrómetro de imágenes Visible/infrarrojo aerotransportado de la NASA *AVIRIS* tiene 224 bandas con una cobertura espectral similar.

La selección del número de bandas, ancho y rango espectral medido por el sensor está relacionada con los objetivos que se espera alcanzar. Mientras aplicaciones para la minería requieren múltiples bandas en los rangos del infrarrojo visible, cercano y medio, sensor satelital meteorológico puede requerir solo una o dos bandas en el rango visible y varias docenas en los canales infrarrojos medios para estimar la composición del gas atmosférico.

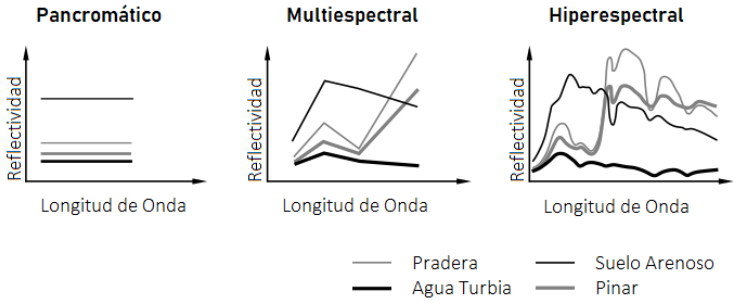


Figura 7.28: Firmas Espectrales en Imágen Pancromática, Multi e Hiperespectral para cuatro coberturas de suelo.

7.11.1 Firma Espectral

En la figura 7.28 vemos la importancia de una alta resolución espectral **Cada elemento sobre la superficie terrestre tiene**

una respuesta única a las distintas longitudes de onda, esa forma específica es una firma del material y física del componente, es lo que denominamos **firma espectral**. Entonces, para una imagen con una mayor cantidad y más finas bandas podemos revisar más fielmente esas formas y compararlas con extensas librerías e identificar los objetos de estudio.

Figura 7.29: Sitio Web del Proyecto Librería Espectral ECOSTRESS de la NASA.

Los científicos han creado extensas librerías con datos espectrales para distintas coberturas o grupos de materiales para ser usados como referencias, entre ellas vamos a destacar la Biblioteca Espectral ASTER Versión 2, (speclib.jpl.nasa.gov), proyecto desarrollado por Meerdink et al. [2019] y Baldridge et al. [2009] (Figura 7.29).

La biblioteca espectral ASTER proporciona una colección completa de miles de espectros compuesto por una amplia variedad de materiales naturales y artificiales que cubren la longitud de onda rango $0,4\mu m - 15,4\mu m$.

La versión 2.0 de la biblioteca espectral ASTER se lanzó el 3 de diciembre de 2008. La biblioteca espectral ASTER incluye datos de otras tres bibliotecas espectrales:

- Biblioteca espectral de la Universidad Johns Hopkins (JHU).
- Laboratorio de propulsión a chorro (JPL).
- Biblioteca espectral del Servicio Geológico de los Estados Unidos (USGS - Reston).

La biblioteca de espectros se proporciona con más de seis mil espectros individuales.

Para su descarga debemos seguir algunos pasos preliminares:

1. Visitar el sitio web de la librería speclib.jpl.nasa.gov. (Figura 7.29).

2. Visitar la ficha **Download** para explorar las alternativas disponibles de descarga (Figura 7.30).

3. Previo a la descarga se debe llenar un formulario de contacto y solicitar los archivos a descargar, y en el correo electrónico entregado se recibirá el enlace correspondiente. (Figura 7.31).

4. En cuestión de minutos, se recibe el correo de acuso recibo con el resumen de la orden y aviso de que se trabaja en ello.

Figura 7.30: Ficha de descarga de archivos de librerías.

Figura 7.31: Formulario de contacto para informar los archivos solicitados.

5. Y finalmente se recibe el correo con el enlace de descarga (Figura 7.32).

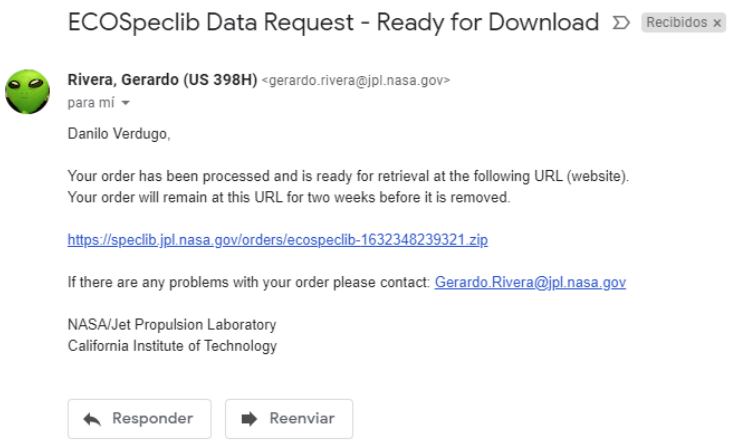


Figura 7.32: Correo con link de descarga.

7.11.2 Import Dataset

Es importante destacar que los archivos de las firmas espectrales se componen de información general respecto a los datos, o técnicos de la captura entre otros. A continuación, un número variable de registros en formato de dos columnas, en la primera *longitud de onda* (en μm) y un valor de *reflectancia* (en %), separados por espacio usando símbolo decimal el punto (.).

Es un formato complejo para una carga directa ya que debemos separar la cabecera de los datos y luego cada registro separar en dos valores numéricos con valores decimales.

Para resolver estos casos, RStudio ofrece una forma interactiva para la carga de datos: **Import Dataset**, opción en la barra de herramientas de la ficha *Environment* (Figura 7.33).

De las alternativas disponibles utilizaremos una función más completa que la disponible en los paquetes *base* de R mediante el paquete **readr**.

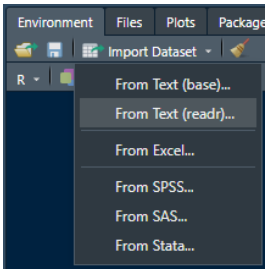


Figura 7.33: Detalle de opción 'Importar Datos Externos'.

Activando la opción se presenta un cuadro de dialogo con varios grupos de opciones para seleccionar, configurar, cargar directo a un objeto R a la sesión o generar el código R para pegar en un archivo script.

El primer paso es seleccionar el archivo de texto buscando el archivo con el botón **Browse...** (Figura 7.34).

Figura 7.34: Detalle de opciones para buscar o escribir el nombre del archivo a leer.

Una vez seleccionado, el sistema puede tardar unos segundos en responder y nos entrega una vista preliminar del contenido en un panel dedicado (Figura 7.35), que responde dinámicamente a los cambios en los parámetros de lectura disponibles en la parte inferior del cuadro de dialogo.

Los parámetros más importantes para la lectura son:

- **Name**: Modificar el nombre del objeto por uno compatible con los nombres R.
- **Skip**: Número de líneas de texto al inicio que no son datos. Ellas son ignoradas.
- **Delimiter**: Carácter o símbolo que se usa en el archivo para separar los datos en columnas.

Los parámetros ideales para la lectura de los archivos de firmas espectrales descargados del sitio de la NASA se pueden revisar en la figura 7.36.

Finalmente, existen dos formas de rescatar los datos (Figura 7.37):

- **Importar el Objeto**: a la sesión actual como un dataframe mediante el botón **Import**.

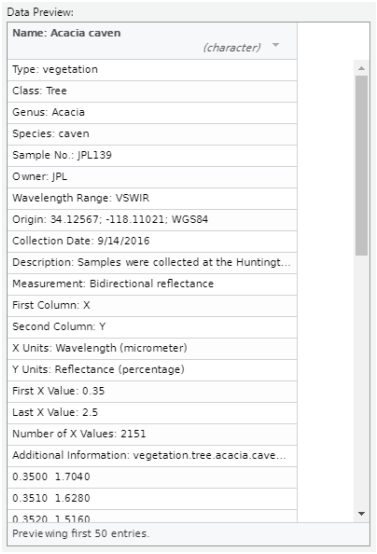


Figura 7.35: Detalle de opción 'Importar Datos Externos'.

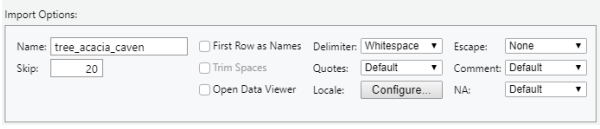


Figura 7.36: Detalle de opción 'Importar Datos Externos' con los parámetros usados para cargar nuestro archivo de texto.

- **Copiar el código**: Mediante el ícono  se copia al *clipboard* y usando la combinación CNTRL+V para pegar en el script o consola.

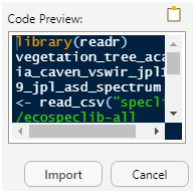


Figura 7.37: Detalle de las opciones para importar los datos a nuestra sesión R.

7.11.3 Procesando en R

El código generado con la ayuda del cuadro de dialogo se copia en nuestro script y se puede modificar ligeramente para refinar el objeto y mejorar la usabilidad posterior.

Primero se debe cargar el paquete que contiene la función:

```
library(readr)
```

Pasamos el nombre del archivo a leer a una variable para facilidad de reusabilidad.

```
nombre_file <- "vegetation.tree.acacia.caven.vswir.jpl139.jpl.asd.spectrum.txt"
```

También asignamos los nombres de las columnas.

```
aromo <- read_table(here("speclib","ecospeclib-all", nombre_file),
                    col_names = c("wavelength","reflectance"),
                    col_types = cols(wavelength = col_double(),
                                     reflectance = col_double()),
                    skip = 20)
```

Expresamos en formato gráfico para revisar la *firma espectral* propiamente tal (Figura 7.38).

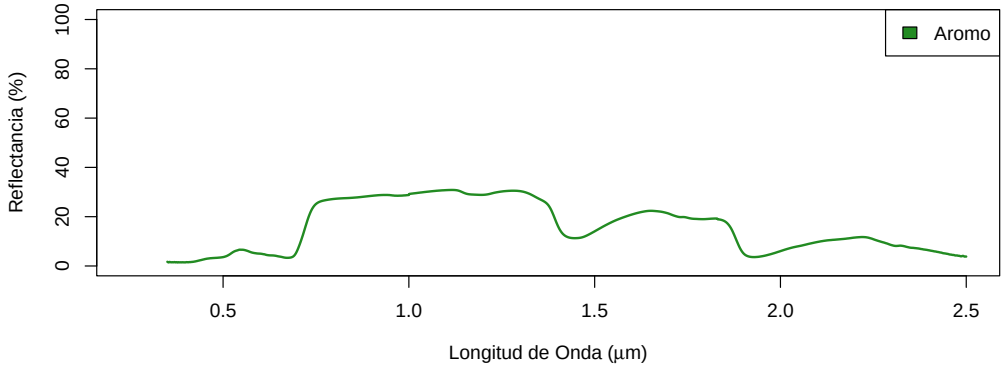


Figura 7.38: Firma Espectral del Árbol Acacia Caven (Aromo), librería ASTER v2.

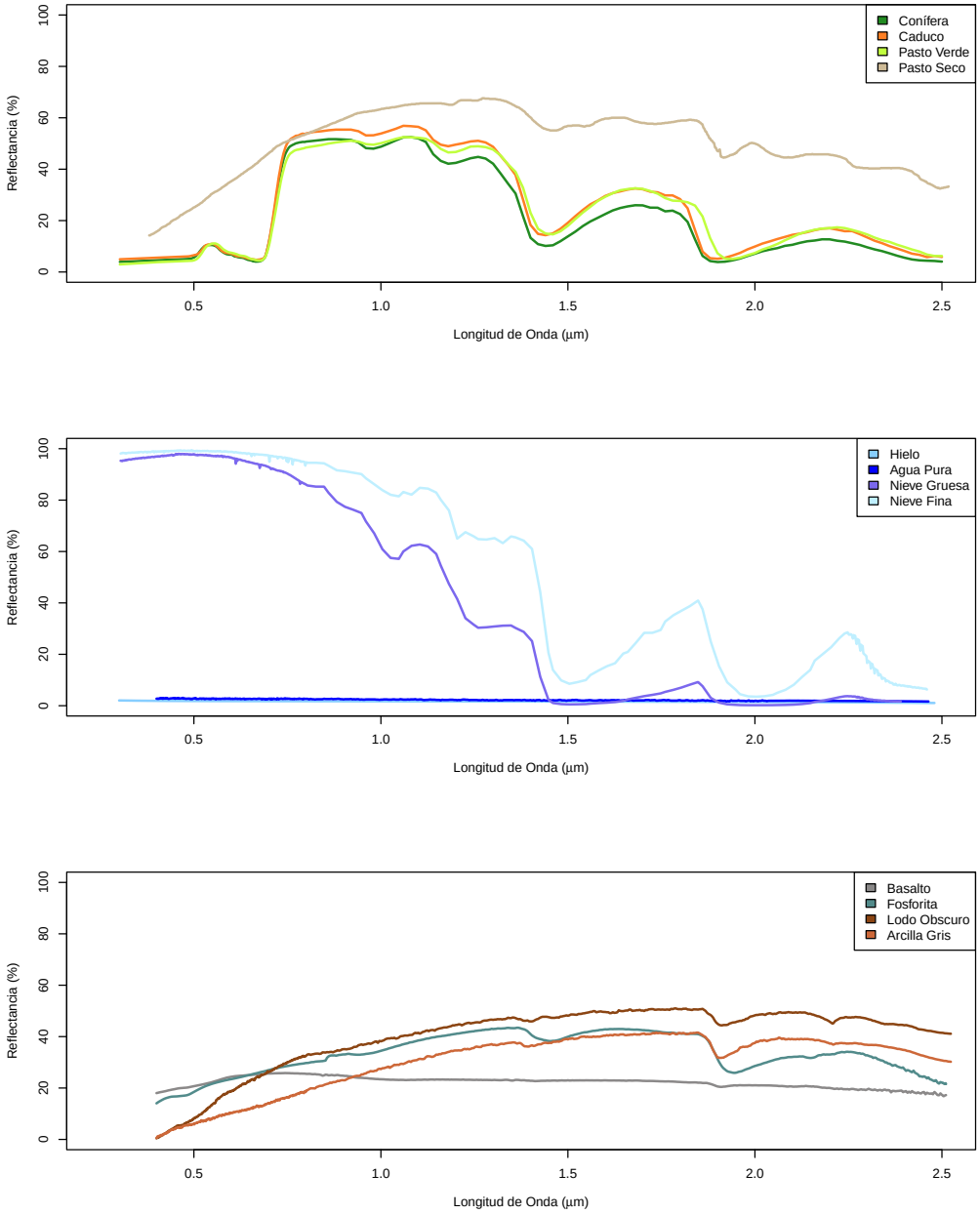


Figura 7.39: Ejemplos de firmas espectrales organizadas por grupos de tipos de cobertura.

7.12 Resolución Radiométrica

La resolución radiométrica denota la *sensibilidad del sensor*, es decir, su capacidad para discriminar pequeñas variaciones en la radiación espectral registrada. En los sensores electrónicos ópticos, la imagen es adquirida digitalmente y su resolución radiométrica se expresa comúnmente como el rango de valores utilizados para codificar la radiancia de entrada o, más precisamente, como el **número de bits** utilizados para almacenar la señal de entrada. Un sistema de sensor de 8 bits puede discriminar 256 radiancias de entrada diferentes por píxel ($2^8 = 256$), por lo tanto, un rango de **números digitales (DN)** que va de 0 a 255 (Figura 7.40).

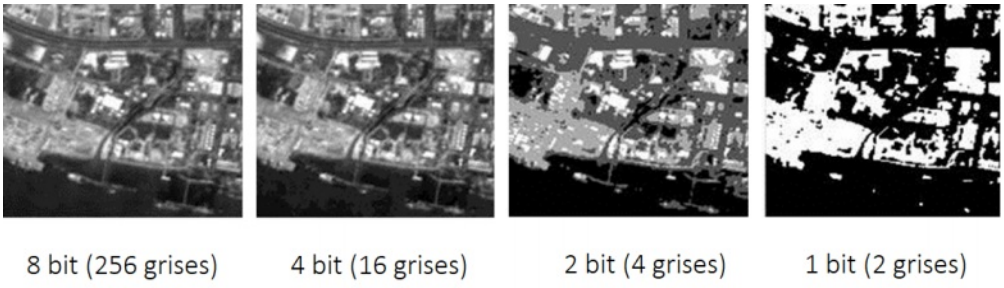


Figura 7.40: Comparación resoluciones radiométricas.

debido a la mayor velocidad de la transmisión de datos y su almacenamiento, la resolución radiométrica de los sensores está aumentando. Anteriormente, la mayoría de los sensores proporcionaban Codificación de 8 bits, mientras que hoy en día varios tienen entre 11 y 16 bits por píxel (¡uno de 65536 valores!), por ejemplo, GeoEye, WorldView, MODIS o Landsat 8 OLI.

La resolución radiométrica es más crítica en el análisis digital que en el visual. El número de niveles de gris que el ojo humano es capaz de discernir no supera los 64, y no puede distinguir más de 200.000 colores. Parece redundante tener incluso 256 valores digitales por banda (16.8 millones de valores en una imagen en color de tres bandas). Sin embargo, cuando la interpretación es digital, las computadoras aprovechan todo el rango, en cuyo caso una alta resolución radiométrica es importante para discriminar objetos con firmas espectrales similares, lo que no sería posible con sensores menos sensibles.

7.13 Resolución Temporal

La resolución temporal se refiere a la *frecuencia de observación* (período de revisión) proporcionada por el sensor. Este ciclo es función de las características orbitales del satélite (altura, velocidad y declinación), así como del FOV[2] del sensor.

[2] El **campo de visión total** (*field of view*, FOV) es el área observada por el satélite en una sola imagen.

Sensores con *alta resolución temporal* tienen una *resolución espacial gruesa*, ya que podrán observar una mayor área en cada adquisición de imágenes. En consecuencia, también tienen un campo de visión amplio, que implica problemas geométricos, ya que la misma área se puede observar desde muy diferentes ángulos en un breve tiempo diario, ver 7.14.

Si el sensor observa un área más pequeña (campo de visión más estrecho), necesitará más órbitas para repetir la misma área observada y así, las capturas tendrán mayores períodos de tiempo. Para aliviar este problema, algunos sensores pueden hacer observaciones fuera del nadir, detectando áreas adyacentes a la órbita (las capacidades de apuntar generalmente van de 0° a 20°). Este es el caso de las cámaras a bordo de los satélites SPOT o WorldView. Una alternativa a los sensores que miran el nadir es tener varios satélites con las mismas características. Este es el caso de la constelación RapidEye (propiedad de una empresa privada), que proporciona una frecuencia de adquisición global diaria con una resolución espacial de 6.5 m utilizando cinco satélites.

La resolución temporal de los sistemas de sensores EO varía según los objetivos de la misión. Los satélites meteorológicos observan fenómenos muy dinámicos y, por lo tanto, proporcionan la resolución temporal más alta: los satélites geoestacionarios pueden adquirir imágenes cada 15 o 30 minutos, y los satélites en órbita polar, como la NOAA (Estados Unidos), METOP (Europa) o Fengyun (China), proporcionan al mundo cobertura cada 12 h. Normalmente, los sensores de mayor resolución espacial ofrecen menor tiempo resoluciones, de 16 días (como el caso de la serie Landsat) a 28 días (para Radares ERS). Sin embargo, existe una tendencia creciente a tener una mayor resolución incluso para satélites de resolución espacial media o alta. Por ejemplo, el satélite Sentinel-2 de la Agencia Espacial Europea (ESA) proporcionará

entre 10 y 60 m imágenes de resolución para 13 canales espectrales cada 2 a 5 días cuando los dos satélites de la constelación están activos.

7.14 Resolución angular

El concepto de resolución angular es muy reciente y se refiere a la capacidad del sensor para hacer observaciones de la misma área, desde diferentes ángulos de visión (Diner et al. [1999]). Las variaciones en reflectancia con ángulos de visión e iluminación variables se denominan *función de distribución de reflectancia bidireccional* (BRDF). Y para modelar estos efectos es necesario observar la superficie desde diferentes direcciones. También, las observaciones multiangulares son de gran interés en estimar algunas propiedades atmosféricas como el espesor de la capa de aerosoles o la altura de las nubes.

VIII MISIONES ESPACIALES

Imágenes Satelitales Libres

8.1 Plataforma Búsqueda y Descarga

El sitio web que centraliza el acceso a datos de la NASA y otros productos espaciales es: earthexplorer.usgs.gov.

Pasos principales para la descarga de imágenes:

1. El sitio se compone de dos unidades principales, la zona de configuración de la búsqueda (panel izquierdo) y el mapa interactivo para definir el área geográfica de estudio (Figura 8.1).

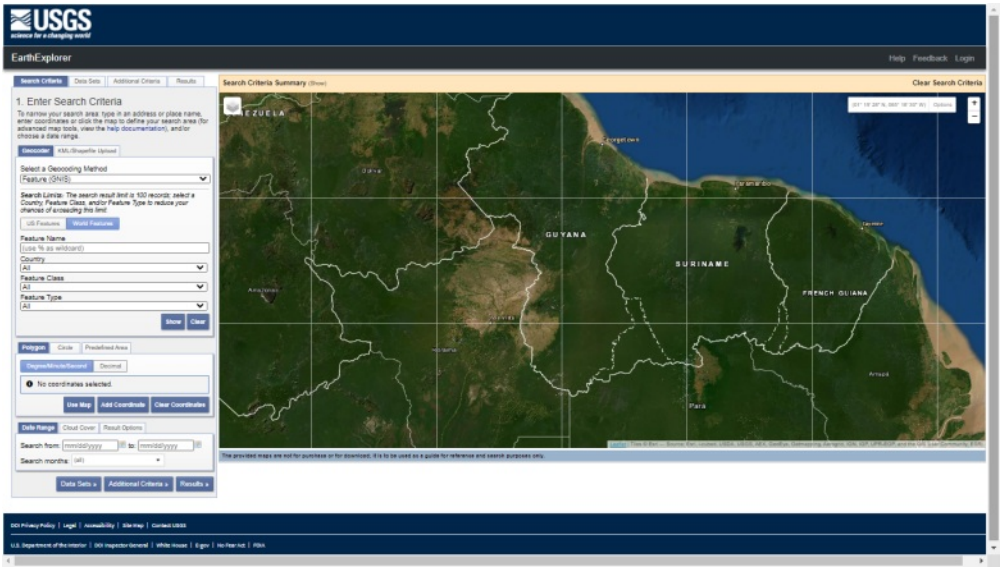


Figura 8.1: Vista principal earthexplorer.usgs.gov

2. Si no se ha realizado el registro previo, debe realizar el proceso paso a paso (con la ayuda del mismo sitio) o *logearse* median-

te el botón apropiado ubicado en la esquina superior derecha (Figura 8.2).

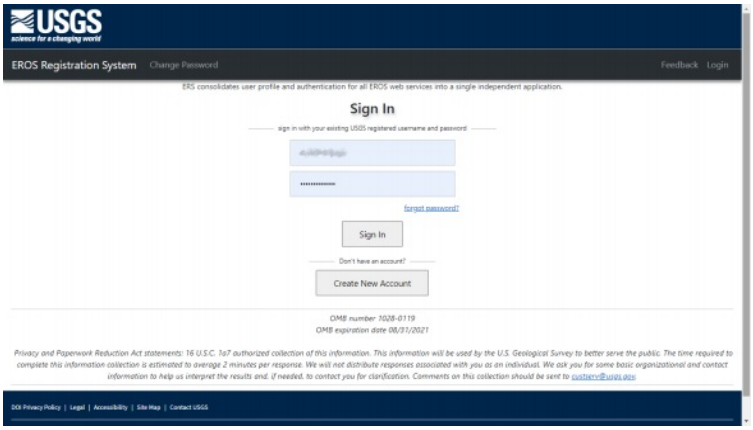


Figura 8.2: Ventana de Login o de Registro, paso obligatorio.

3. Configurar la búsqueda espacial (Figura 8.3 zona A y B):

- A) Usar servicios de geocodificación (para EE.UU.) convirtiendo una dirección postal en coordenadas geográficas o para el resto del mundo buscar por nombres propios o tipo de elemento geográfico. También se puede utilizar archivos *kml* generado en la aplicación Google Earth Pro o archivos *shapefile* estándar en las aplicaciones genéricamente denominadas SIG.

- B) Definir la zona geográfica por coordenadas y configurar el método de obtención de coordenadas o el formato de ingreso de los textos. En la ficha **Polygon** usar la opción **Use Map** y automáticamente la vista actual del mapa (sector derecho) se convierte en la zona geográfica de búsqueda. El listado de vértices puede ser editado, corregido, agregar o eliminar puntos mediante la interfaz o modificando directamente las marcas en el mapa arrastrándolas a su nueva ubicación. El siguiente método disponible es agregar el centro y radio de un círculo que delimita el espacio geográfico.

Figura 8.3: Parámetros para crear y refinar una búsqueda, detalles en texto.

Además, se pueden agregar parámetros no espaciales (Figura 8.3 zona C):

- C) **Rango Temporal** mediante el uso de calendario de inicio y fin (Figura 8.4).

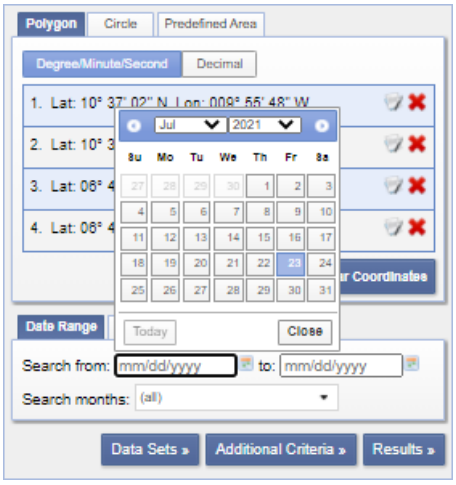


Figura 8.4: Opciones para refinar la búsqueda, fecha inicial y final.

También puede filtrar los meses específicos para realizar la búsqueda (Figura 8.5).

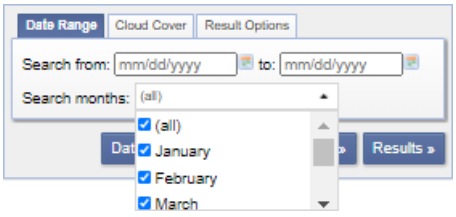


Figura 8.5: Lista para seleccionar meses particulares.

Otro parámetro muy importante es fijar el porcentaje de *cobertura nubosa* que se define como el porcentaje del área que cubre la imagen se encuentra obstruida por nubes (Figura 8.6).

Figura 8.6: Porcentaje de cobertura nubosa aceptable para las imágenes seleccionadas.

El siguiente paso es la definición del tipo de datos. En la plataforma existe un gran número de misiones, sensores y tipos de

datos, figura 8.7 MODIS (izquierda), Landsat (derecha). La plataforma permite selección múltiple.

En el listado vamos a ubicar el grupo **NASA LPDAAC Collections** y dentro de las opciones, **MODIS Land Surface Reflectance- V6**.

En las alternativas disponibles para este grupo de productos encontramos el grupo que corresponde al producto descrito en 8.3, MOD09A1 v6 del sensor MODIS.

También encontramos los productos de la misión Landsat 4-5, 7 y 8 (Figura 8.7, derecha) que revisaremos en detalle en el punto 8.7.

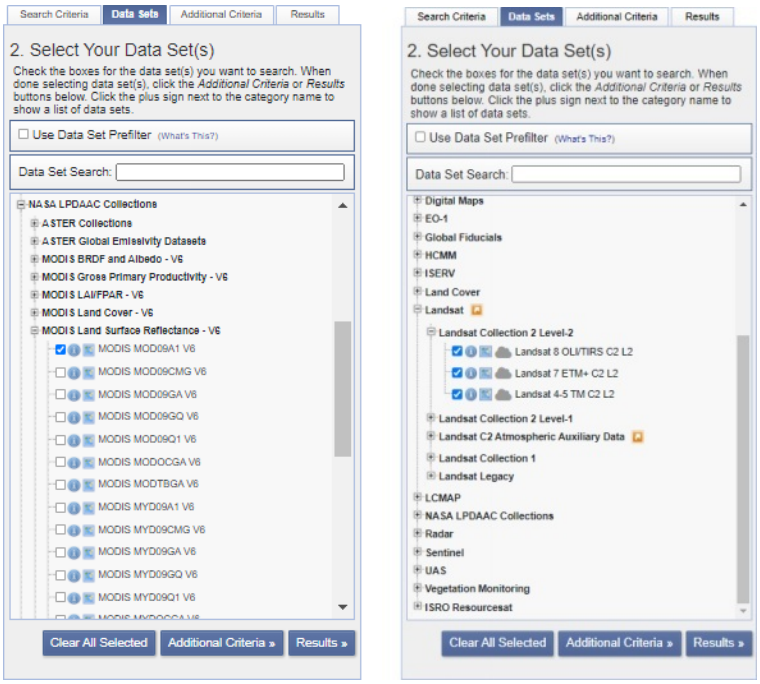


Figura 8.7: Lista de tipos de datos disponibles para descarga de la misión MODIS (izquierda) y Landsat (derecha).

Figura 8.8: Mensaje que alerta de la realización de la consulta y botón para cancelar.

Finalmente, para realizar la búsqueda se debe presionar el botón **Results »**, y mientras se despliega el mensaje de búsqueda se podrá cancelar para redefinir o corregir algún parámetro (Figura 8.8).

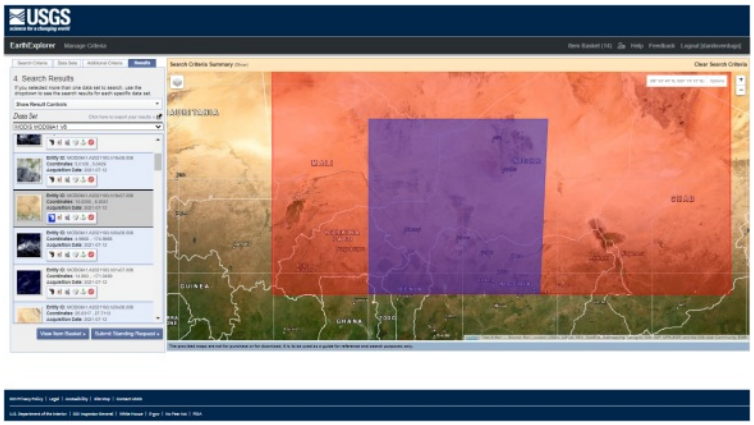


Figura 8.9: Resultado de la búsqueda. Detalle de imágenes que cumplen los criterios definidos.

8.1.1 Descarga de Datos

El resultado final es una lista interactiva con todos los productos que cumplen los parámetros de búsqueda. Para cada uno de ellos se disponen de varias opciones (Figura 8.9).

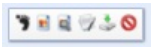


Figura 8.10: Opciones Disponibles en archivo seleccionados.

De izquierda a derecha, las opciones son (Figura 8.10):

- **Silueta Gráfica** de la extensión espacial.
- **Visualización Baja Resolución** en su ubicación exacta.
- **Imagen para comparación** entre productos seleccionados en nueva ventana.
- **Visualizar Metadatos** y detalles técnicos de la imagen.
- **Descarga Directa** (Figura 8.11) del archivo a disco local.
- **Excluir de Descarga**

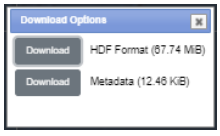


Figura 8.11: Opciones de descarga (depende del producto o tipo de datos a descargar, en el ejemplo MODIS).

El primer sensor que vamos a estudiar es el sensor MODIS y las imágenes del set MOD09A1. Descargamos de la lista de opciones las cuatro imágenes más actualizadas que cubran el país de Niger con una baja cobertura nubosa (Figura 8.17):

MOD09A1.A2021185.h18v07.061.2021194052227.hdf
MOD09A1.A2021193.h18v06.061.2021202214452.hdf
MOD09A1.A2021193.h19v06.061.2021202214223.hdf
MOD09A1.A2021193.h19v07.061.2021202220423.hdf

Misiones MODIS

8.2 Terra y Aqua

MODIS o espectrorradiómetro de imágenes de resolución moderada es un instrumento clave a bordo de los satélites **Terra** (originalmente conocido como EOS AM-1) y **Aqua** (originalmente conocido como EOS PM-1). La órbita de Terra alrededor de la Tierra está programada para que pase de norte a sur a través del ecuador por la mañana, mientras que Aqua pasa de sur a norte sobre el ecuador por la tarde. Terra MODIS y Aqua MODIS están viendo toda la superficie de la Tierra cada 1 o 2 días, adquiriendo datos en 36 bandas espectrales o grupos de longitudes de onda (consulte las Especificaciones técnicas de MODIS). Estos datos mejorarán nuestra comprensión de la dinámica y los procesos globales que ocurren en la tierra, en los océanos y en la atmósfera inferior.

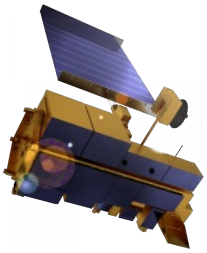


Figura 8.12: Satélite Terra. NASA.

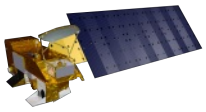


Figura 8.13: Satélite Aqua. NASA.

Las plataformas *Terra* y *Aqua* son parte del Sistema de Observación de la Tierra (EOS) de la NASA, **Terra** se lanzó con éxito en diciembre de 1999 y se encuentra en una órbita casi polar, sincrónica con el Sol, cruzando el ecuador alrededor de las 10:30 a.m. y p.m. La altura orbital es de 705 km, con un período de 98,88 min y un ciclo de repetición de 16 días.

Lleva cinco sensores (MODIS, CERES, MISR, MOPITT y ASTER) diseñado para observaciones globales de tierras, océanos y variables atmosféricas.

Aqua se lanzó en mayo de 2002 y tiene características orbitales similares a Terra, pero con un retraso de 3 horas, por tanto, cruza el ecuador a la 1:30 y a las 13:30. Este satélite lleva seis

sensores (AIRS, AMSR-E, CERES, HSB y MODIS), con una configuración de instrumentos orientado a estudios oceanográficos.

El sensor principal a bordo de las dos plataformas es MODIS. Adquiere datos diarios de todo el mundo en 36 bandas espectrales, a diferentes resoluciones, 250 a 1000 m. Las primeras dos bandas tienen una resolución espacial más fina (250 m) e incluyen el rojo y el espectro NIR. Longitudes de onda para monitorear la actividad de la vegetación.

Se adquieren otras cinco bandas a 500 m de resolución que cubren el visible y SWIR y están destinados principalmente a la captura de propiedades de vegetación, nieve y suelo.

El resto de las bandas tienen 1000 m de resolución y cubren longitudes de onda adicionales en el visible, MIR y TIR. La franja cubierta por MODIS es de 2300 km, y la frecuencia de observación es diaria (dos veces cuando se contabilizan los dos satélites). El programa MODIS se construyó para obtener radiancias (ρ) calibradas y otros productos finales.

Estos productos se agrupan en tres dominios:

- **Productos Atmosféricos**, incluyendo aerosoles, agua total precipitable, nubes y perfiles atmosféricos.
- **Productos de Tierra**, incluida la reflectancia de la superficie, temperatura de la superficie, la cobertura del suelo, índices de vegetación, incendios activos, radiación incidente, evapotranspiración, productividad de la planta, albedo, áreas quemadas, manto de nieve y hielo marino.
- **Productos Oceánicos**, que incluyen temperatura del mar, concentración de clorofila-a, partículas de carbono, fluorescencia y radiación fotosintética incidente.

Existe una amplia serie de productos disponibles, todos ellos se encuentran detallados en el sitio de la NASA dedicado al sensor MODIS:

modis.gsfc.nasa.gov/data/dataprod/index.php#atmosphere.

A continuación, vamos a trabajar con el producto **MOD09**, productos **MODIS Surface Reflectance** proporcionan una *estimación de la reflectancia espectral de la superficie* tal como se mediría a nivel del suelo en **ausencia de dispersión o absorción atmosférica**.

En el producto de 8 días, cada píxel de reflectancia de la superficie contiene la mejor observación posible durante un período de 8 días, según se selecciona en función de la alta cobertura de observación, el ángulo de visión bajo, la ausencia de nubes o sombras de nubes y la carga de aerosoles.

8.3 MOD09A1

Los productos **MOD09A1**, *versión 6* (Vermote [2015]) que proporciona una estimación de la reflectancia espectral superficial de las bandas Terra de 1 a 7 corregidas para condiciones atmosféricas como gases, aerosoles y dispersión de Rayleigh (Figura 8.14 y el cuadro 8.1).

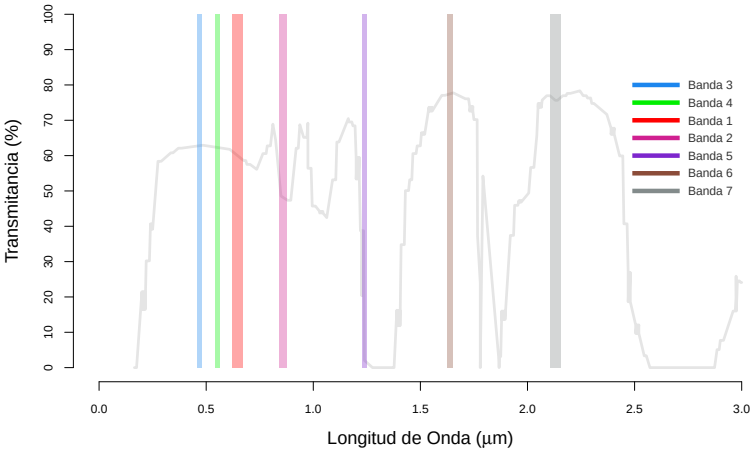


Figura 8.14: Distribución y espesor de bandas en sensor MODIS.

Junto a las siete bandas de reflectancia de 500 m hay dos capas de calidad y cuatro bandas que describen el momento de la observación. Para cada píxel se selecciona un valor de todas las capturas dentro del período compuesto de **8 días**.

Los criterios para la elección de píxeles incluyen la nube y el cenit solar. Cuando varias adquisiciones cumplen los criterios, se utiliza el píxel con el valor mínimo del canal 3 (azul).

La colección de imágenes se extiende desde el 2000-02-24 hasta el presente con cobertura global y una frecuencia de 8 días.

El píxel tiene un tamaño de 500 m en imágenes de 2400 $\times$ 2400 filas y columnas en el sistema de coordenadas **Sinusoidal**,

Cuadro 8.1: Resumen de Bandas MODIS.

Nombre Archivo	Bandas (ρ nm)	Nombre	Escala
sur_refl_b01	B1(620 – 670)	Red	0.0001
sur_refl_b02	B2(841 – 876)	NIR	0.0001
sur_refl_b03	B3(459 – 479)	Blue	0.0001
sur_refl_b04	B4(545 – 565)	Green	0.0001
sur_refl_b05	B5(1230 – 1250)	SWIR1	0.0001
sur_refl_b06	B6(1628 – 1652)	SWIR2	0.0001
sur_refl_b07	B7(2105 – 2155)	SWIR3	0.0001
sur_refl_qc_500m	QA		N/A
sur_refl_szen	$\zeta\ Solar$		0.01
sur_refl_vzen	$\zeta\ Vista$		0.01
sur_refl_raz	Az. relativo		0.01
sur_refl_state_500m	Estados		N/A
sur_refl_day_of_year	DOY píxel		N/A

Nota:

ρ: Reflectancia Superficial.

ζ: ángulo Cenital.

DOY: Día del Año. $DayOfYear$

(EPSG: 6974).

Código proyección: Proj4js.defs[“SR-ORG:6974”] = “+proj=sinu +lon_0=0 +x_0=0 +y_0=0 +ellps=WGS84 +datum=WGS84 +units=m +no_defs”

8.4 MODIS en R

Una vez descargadas las imágenes vamos a utilizar los paquetes *rgdal* (Bivand et al. [2021]) que agrega la capacidad de leer un archivo en formato _*.hdf_ [1], y el paquete *MODIS* (Mattiuzzi and Detsch [2021]) que nos da acceso a la metadata, maneja los formatos de nombre de imágenes entre otros detalles prácticos.

Luego de instalar ambos paquetes desde la consola, debe cargarlo en memoria (no olvidar agregar al inicio de su script):

```
library(rgdal)
library(MODIS)
```

Construimos un vector con los archivos que cumplan con el formato *‘hdf’* de nuestra carpeta de datos:

```
archivos <- list.files(path = here("raw","modis"),
                       pattern = "*.hdf",
                       full.names = T)
```

Usando la función `extractDate()` del paquete MODIS podemos extraer la fecha de captura de los datos a partir del nombre de archivo.

```
(datos <- extractDate(archivos[1], asDate = T))
```

```
$inputLayerDates
[1] "2021-07-04"

$pos1
[1] 10

$pos2
[1] 16
```

[1] El formato de datos jerárquico (*Hierarchical Data Format*, HDF) es un conjunto de formatos de archivo (HDF4, HDF5) diseñado para almacenar y organizar grandes cantidades de datos. Desarrollado originalmente en el **Centro Nacional de Aplicaciones de Supercomputación** y cuenta con el apoyo de *The HDF Group*, una corporación sin fines de lucro cuya misión es garantizar el desarrollo continuo de las tecnologías HDF5 y la accesibilidad continua de los datos almacenados en HDF (Wikipedia).

```
$asDate
[1] TRUE

$format
[1] "%Y%j"
```

Cada archivo *hdf* contiene al menos 13 capas de información (ver cuadro 8.1) y usando la función `getSds()` obtenemos una lista con los *nombres propios* de cada capa y la *ruta completa* para llegar al archivo hdf. Por definición la función procesa un hdf en cada llamado, así que debemos envolver con el comando *lapply* (ver detalles en 6.10.1) nuestra lista de archivos y la función a aplicar:

```
imgs <- lapply(archivos, getSds)
```

Ahora tenemos estructurado por cada archivo las capas que contiene siguiendo el formato:

```
imgs[[1]]$SDS4gdal[1]
```

```
[1] "HDF4_EOS:EOS_GRID:"C:/Users/danilo.verdugo/Documents/
rasterEdu/raw/modis/MOD09A1.A2021185.h18v07.061.2021194052227.hdf
":MOD_Grid_500m_Surface_Reflectance:sur_refl_b01"
```

Los archivos son leídos directamente con la función `readGDAL()` y convertidos a formato ráster directamente:

```
r <- readGDAL(imgs[[1]]$SDS4gdal[1])
raster(r)
```

Ahora podemos reescribir este comando R para cada imagen por cada cada capa (4×8) veces o escribir una función que facilite la tarea.

Detalle de los componentes principales para definir nuestra función:

- **`leer <-`**: Es el nombre de la función y nombre que se utiliza para llamarla.

- **`function(img)`**: Es una función que recibe un identificador de la imagen `id` de la lista **imgs**.

- `f <- 0.0001`: Para convertir los DN a ρ utilizamos el factor de escala presentado en el cuadro 8.1. Así el valor almacenado en cada ráster es la representación fiel de la energía recibida en cada píxel (ver 7.5.2).

- `variable <- raster(readGDAL(imgs[[img]]$SDS4gdal[ID], silent = T))`: Siguiendo el orden de capas del cuadro 8.1 cargamos en una variable. Se agrega el parámetro **silent = T** que nos ahorra ver un mensaje en pantalla por cada lectura realizada.

- `s <- stack(r,g,b,nir,swir1,swir2,swir3,qa)`: Convertimos todas las capas en un modelo *rasterStack*.

- `names(s) <- c('r','g','b','nir','swir1','swir2','swir3','qa')`: Asignamos nombres propios a cada capa para fácil referencia.

- `s`: Al ser la última línea, es el objeto que regresa la función.

Y todo integrado:

```
leer <- function(img){
  f <- 0.0001
  r <- raster(readGDAL(imgs[[img]]$SDS4gdal[1], silent = T)) * f
  g <- raster(readGDAL(imgs[[img]]$SDS4gdal[4], silent = T)) * f
  b <- raster(readGDAL(imgs[[img]]$SDS4gdal[3], silent = T)) * f
  nir <- raster(readGDAL(imgs[[img]]$SDS4gdal[2], silent = T)) * f
  swir1 <- raster(readGDAL(imgs[[img]]$SDS4gdal[5], silent = T)) * f
  swir2 <- raster(readGDAL(imgs[[img]]$SDS4gdal[6], silent = T)) * f
  swir3 <- raster(readGDAL(imgs[[img]]$SDS4gdal[7], silent = T)) * f
  qa <- raster(readGDAL(imgs[[img]]$SDS4gdal[12], silent = T))
  s <- stack(r, g, b, nir, swir1, swir2, swir3, qa)
  names(s) <- c("r","g","b","nir","swir1","swir2","swir3","qa")
  s
}
```

Y aplicamos nuestra función para leer los datos:

```
imagen1 <- leer(img = 1)
imagen2 <- leer(img = 2)
imagen3 <- leer(img = 3)
imagen4 <- leer(img = 4)
```

8.4.1 Mapeo RGB

Hasta ahora hemos trabajado principalmente con objetos rasterLayer y para generar un mapa usamos el comando *plot()*, se define la paleta de colores, algunos otros parámetros y se genera la gráfica.

Para los objetos *rasterStack* o *rasterBrick* existe la versión especializada `plotRGB()` que permite realizar un gráfico *RGB* basado en tres capas. Esas tres capas (que también podemos denominar *bandas* porque pueden representar diferentes anchos de banda en el espectro electromagnético) se combinan de manera que representan el canal rojo, verde y azul para formar imágenes en color.

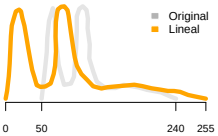


Figura 8.15: Descripción Gráfica del método stretch Lineal.

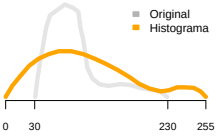


Figura 8.16: Descripción Gráfica del método stretch Histograma

`r`: Índice o nombre de la capa que se usará para definir el canal *rojo* de la imagen.

`g`: Índice o nombre de la capa que se usará para definir el canal *verde* de la imagen.

`b`: Índice o nombre de la capa que se usará para definir el canal *azul* de la imagen.

`stretch`: Se encarga de realizar ajustes de **visualización** (no modifica los valores internos del objeto) para aprovechar toda la escala de tonos de color en la imagen. Los valores permitidos son **lin**: *método lineal*, donde el rango de ND (niveles digitales) se ajusta para ocupar todo el rango disponible (Figura 8.15). Por ejemplo, el rango $50 - 240$ se extiende al rango $0 - 255$ (ejemplo con 8 bit). El segundo, **hist**: *método histograma*, donde los valores además de extenderlos al rango completo de valores ($0 - 255$ a 8 bit) **proporcionalmente** a su frecuencia, a mayor cantidad del valor digital, mayor rango de ND asignados (ver figura 8.16).

La sintaxis básica para crear una imagen en color verdadero es asignar por cada canal el color correspondiente:

```
plotRGB(imagen1, r=1, g=2, b=3, stretch="lin")
```

O usar los nombres de capas para no tener que recordar el valor numérico de índice dentro del objeto Stack, por ejemplo:

```
plotRGB(imagen1, r="r", g="g", b="b", stretch="lin")
```

La figura 8.17 muestra el resultado de la gráfica en *color ver-*

dadero, en el cañon **r** la banda *1* o *'r'*, en el cañon **g** la banda *2* o *'g'* y en el cañon **b** la banda *3* o *'b'*.

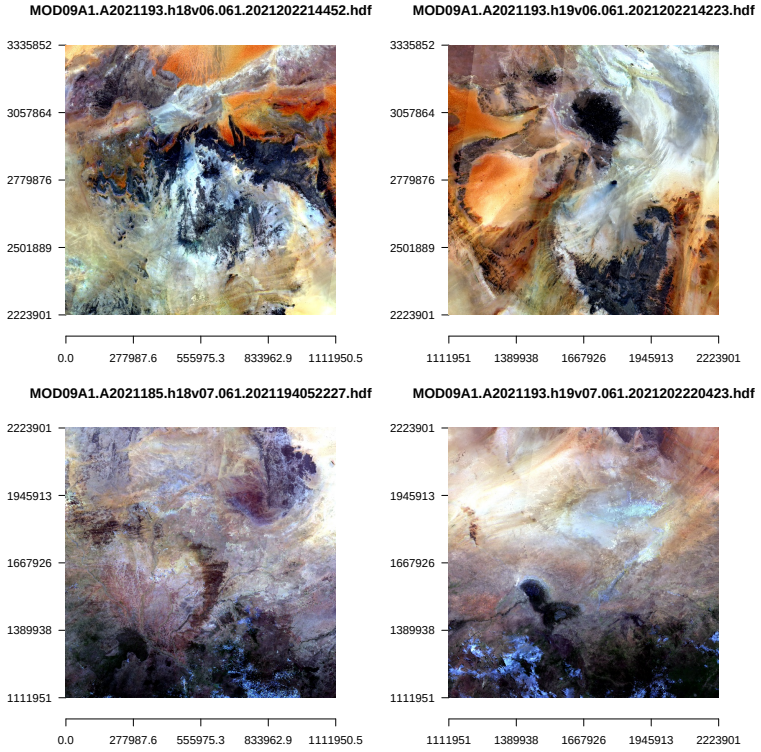


Figura 8.17: Ploteo de las imágenes utilizadas en el ejercicio usando las bandas RGB.

Guardamos en disco las imágenes descargadas y procesadas para el posterior uso en el tema de la combinación de bandas (ver 9.5).

```
writeRaster(imagen1,here("raw","mosaico-test11.grd"),overwrite=T)
writeRaster(imagen2,here("raw","mosaico-test22.grd"),overwrite=T)
writeRaster(imagen3,here("raw","mosaico-test33.grd"),overwrite=T)
writeRaster(imagen4,here("raw","mosaico-test44.grd"),overwrite=T)
```

Misiones Landsat

El Departamento del Interior, la NASA y el Departamento de Agricultura se embarcaron en un ambicioso esfuerzo para desarrollar y lanzar el primer satélite civil de observación de la Tierra. Su objetivo se logró el 23 de julio de 1972, con el lanzamiento del Satélite de Tecnología de Recursos Terrestres (ERTS-1), que luego pasó a llamarse **Landsat 1**. Los lanzamientos de **Landsat 2**, **Landsat 3** (Figura 8.18) y **Landsat 4** siguieron en 1975, 1978 y 1982, respectivamente. Landsat 4 y 5 son satélites idénticos (Figura 8.19).

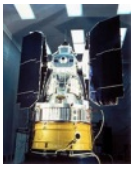


Figura 8.18: Satélite Landsat 1, 2 y 3. NASA.

Figura 8.19: Satélite Landsat 4, 5.

Cuando **Landsat 5** se lanzó en 1984, nadie podría haber predicho que el satélite continuaría entregando datos globales de alta calidad de las superficies terrestres de la Tierra durante 28 años y 10 meses, estableciendo oficialmente un récord mundial Guinness para el *satélite de observación de la Tierra con el funcionamiento más largo*.

Landsat 6 no logró alcanzar la órbita en 1993 (Figura 8.20).

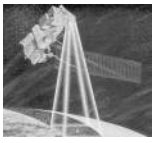


Figura 8.20: Representación Artística de Satélite Landsat 6.

Landsat 7 se lanzó con éxito en 1999 (Figura 8.21), **Landsat 8** en 2013, y ambos satélites continúan adquiriendo datos.

Figura 8.21: Satélite Landsat 7.

El satélite **Landsat 9**, casi idéntico a Landsat 8 (Figura 8.22) ha sido lanzado con éxito el día 27 de septiembre de 2021 (¡mientras escribo estas palabras!) y planea comenzar a entregar imágenes públicas y gratis a los 100 días del lanzamiento (Figura 8.23).

Figura 8.22: Satélite Landsat 8 y 9.

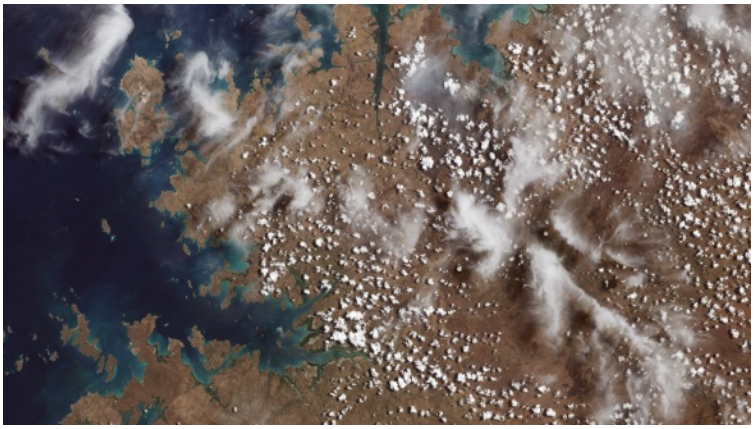


Figura 8.23: Costa Kimberley, Australia. Primera imagen capturada por Landsat 9. 5 noviembre 2021.

8.5 Niveles de Procesamiento

En 2016, el USGS reorganizó el archivo histórico de Landsat en una estructura de colecciones por niveles titulada *Landsat Collection 1*. Esta estructura asegura que todos los productos Landsat (Nivel-1) proporcionen un archivo consistente de calidad de datos conocida, mientras se controla la mejora continua del archivo y el acceso a todos los datos a medida que se adquieren.

La Colección 1 contiene todos los datos de Nivel 1 adquiridos desde 1972 hasta el presente (Landsat 1 al 8).

Landsat Collection 2 marca el segundo evento importante de reprocesamiento del archivo Landsat Nivel-1 de USGS, lo que da como resultado varias mejoras en los productos de datos que aprovechan los avances recientes en el procesamiento de datos, el desarrollo de algoritmos y las capacidades de acceso y distribución de datos.

Una característica principal de la Colección 2 es la mejora sustancial en la precisión de **geolocalización absoluta** del conjunto de datos de referencia terrestre global utilizado en el flujo de procesamiento de datos Landsat Nivel-1. Además, la Colección 2 incluye fuentes de modelado de elevación digital global actualizadas y actualizaciones de **calibración** y **validación**, así

como productos basados en escenas de *reflectancia de superficie* y *temperatura de superficie global* de Nivel 2 **desde 1982 hasta el presente**. Donde todas las bandas han sido llevadas a una resolución espacial de 30 m.

En el cuadro 8.2 se resumen las propiedades importantes para comprender los datos almacenados en las imágenes *tif* Nivel 2.

Cuadro 8.2: Resumen de Propiedades Genéricas para la misión Landsat 4 al 8.

Propiedad	Reflectancia	Temperatura
Valor de Relleno	0	0
Factor de Escala	0.0000275	0.00341802
Factor Aditivo	-0.2	+149.0
Tipo de Datos	Int16U	Int16U
Rango Válido	1 - 65455	1 - 65535

Nota:
Int16U: Entero de 16 bits sin signo.

8.6 Landsat 9

Landsat 9 se ubica en la órbita del Landsat 7, que posee combustible suficiente para operar hasta el año 2021, y consecuentemente será descontinuado. Landsat 9 captura imágenes con una revisita de 16 días en un desplazamiento de 8 días con las imágenes Landsat 8. Planea capturar 750 escenas por día.

8.6.1 Sensores OLI-2 y TIRS-2

Los sensores abordo del Landsat 9 son replicas mejoradas de aquellos abordo del Lansat 8.

El sensor **OLI-2**, Operational Land Imager 2 mejora la resolución radiométrica a 14 bits. Captura datos en la región visible, infrarrojo cercano e infrarrojo de onda corta. La resolución espectral es idéntica a la provista por el sensor OLI. La resolución espacial es de 30 m.

TIRS-2 encargado de la medición en el rango termal, con una resolución espacial de 100 m. Y proveerá dos bandas térmicas.

Las bandas se describen en el cuadro 8.3 y figura 8.24.

Cuadro 8.3: Bandas Landsat 9 (OLI-2/TIRS-2).

Número de Banda	Descripción	Rango (ρ)
1	Aerosoles costeros	$435 - 451nm$
2	Azul	$452 - 512nm$
3	Verde	$533 - 590nm$
4	Rojo	$636 - 673nm$
5	NIR	$851 - 879nm$
6	SWIR1	$1566 - 1651nm$
7	SWIR2	$2107 - 2294nm$
9	Cirrus	$1360 - 1380nm$
10	Térmico 1	$10600 - 11190nm$
11	Térmico	$11500 - 12510nm$

Nota:
ρ: Reflectancia Superficial.

8.7 Landsat 8

El satélite *Landsat 8* orbita la Tierra en una órbita casi polar sincrónica con el sol, a una altitud de 705 km (438 millas), inclinada a 98.2 grados y completa una órbita terrestre cada 99 $minutos$. El satélite tiene una resolución temporal 16 días con un tiempo de cruce ecuatorial: $10:00\ am \pm 15\ minutos$

Landsat 8 adquiere alrededor de 740 escenas por día, el tamaño de cada una escena de ellas es de $185\ km \times 180\ km$.

La data disponible incluye el período abril 2013 al presente.

8.7.1 Sensores OLI y TIRS

El sensor **OLI**, Operational Land Imager construido por la *Ball Aerospace & Technologies Corporation.*

OLI captura datos con precisión radiométrica mejorada en un rango dinámico de 12 bits, lo que mejora la relación general entre señal y ruido. Esto se traduce en 4096 niveles de gris potenciales, en comparación con solo 256 niveles de gris en los instrumentos Landsat 1-7 de 8 bits. El rendimiento mejorado de señal a ruido

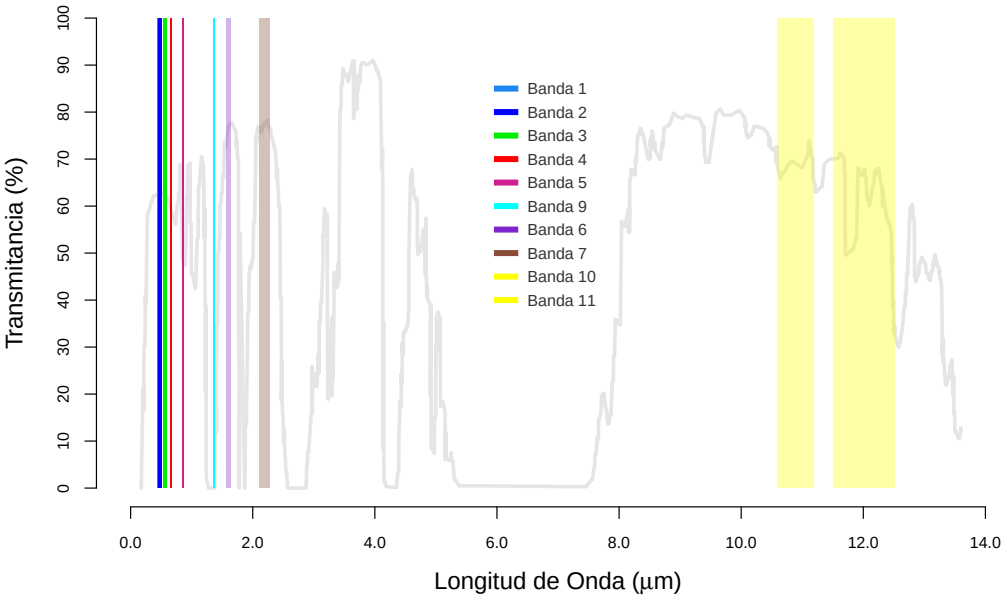


Figura 8.24: Distribución y espesor de bandas en sensor OLI-2 (1 a 9) y TIRS-2 (10-11) Landsat 9.

permite una mejor caracterización del estado y la condición de la cubierta terrestre.

Los datos de 12 bits se escalan a números enteros de 16 bits y se entregan en los productos de datos de Nivel 1. Los productos se escalan a 55,000 niveles de gris y se pueden escalar a la reflectancia y/o resplandor de la parte superior de la atmósfera (TOA) utilizando los coeficientes de cambio de escala radiométricos proporcionados en el archivo de metadatos del producto (archivo MTL).

El sensor de infrarrojos térmico **TIRS** mide la temperatura de la superficie terrestre en dos bandas térmicas con una nueva tecnología que aplica la física cuántica para detectar el calor. TIRS utiliza fotodetectores infrarrojos de pozo cuántico (QWIP) para detectar longitudes de onda largas de luz emitida por la Tierra cuya intensidad depende de la temperatura de la superficie. Estas longitudes de onda, llamadas infrarrojas térmicas, están mucho más allá del alcance de la visión humana. Los QWIP son una alternativa nueva y de menor costo a la tecnología infrarroja convencional y fueron desarrollados en el Goddard Space Flight

Center de la NASA en Greenbelt, Maryland.

Los QWIP que utiliza TIRS son sensibles a dos bandas de longitud de onda infrarroja térmica, lo que ayuda a separar la temperatura de la superficie de la Tierra de la de la atmósfera. Su diseño opera sobre los complejos principios de la mecánica cuántica. Los chips semiconductores de arseniuro de galio atrapan electrones en un estado de energía *bien*[2] hasta que los electrones se elevan a un estado superior por la luz infrarroja térmica de una determinada longitud de onda. Los electrones elevados crean una señal eléctrica que se puede leer y grabar para crear una imagen digital.

[2] En física, una energía *'bien' ('well')* describe un "equilibrio estable" que no alcanza su menor energía posible.

Las bandas se describen en el cuadro 8.4 y figura 8.25.

Cuadro 8.4: Bandas Landsat 8 (OLI/TIRS).

Número de Banda	Descripción	Rango (ρ)
1	Aerosoles costeros	$435 - 451nm$
2	Azul	$452 - 512nm$
3	Verde	$533 - 590nm$
4	Rojo	$636 - 673nm$
5	NIR	$851 - 879nm$
6	SWIR1	$1566 - 1651nm$
7	SWIR2	$2107 - 2294nm$
10	Térmico	$10600 - 11190nm$

Nota:

ρ: Reflectancia Superficial.

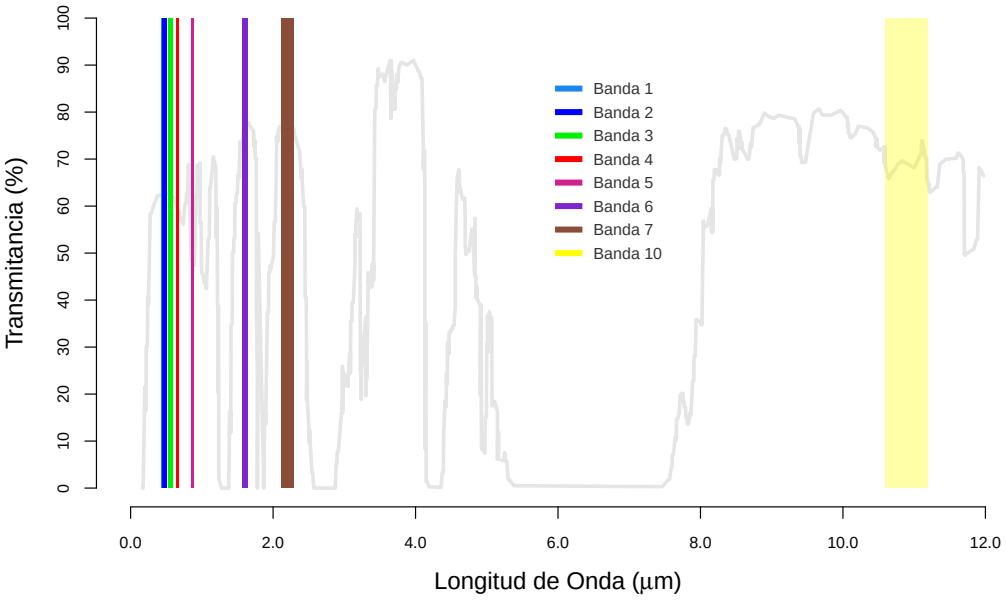


Figura 8.25: Distribución y espesor de bandas en sensor OLI (1 a 7) y TIRS 1(10) Landsat 8.

8.8 Landsat 7

El satélite *Landsat 7* orbita la Tierra en una órbita casi polar sincrónica con el sol, a una altitud de 705 km (438 millas), inclinada a 98.2 grados y completa una órbita terrestre cada 99 *minutos*. El satélite tiene una resolución temporal 16 días con un tiempo de cruce ecuatorial: $10:00\ am \pm 15\ minutos$

Landsat 7 adquiere hasta 532 escenas por día, el tamaño de cada una escena de ellas es de $183\ km \times 170\ km$.

La data disponible incluye el período julio 1999 al presente.

El sensor a bordo es el *Mapeador Temático Mejorado Plus (ETM+)* (Enhanced Thematic Mapper Plus)

Las bandas se describen en el cuadro 8.5 y figura 8.26.

8.8.1 Falla del Corrector de Línea de Escaneo

Seis semanas después de sufrir la pérdida de su *corrector de línea de escaneo (SLC)*, el sensor ETM+ reanudó su misión de estudio global de la tierra, lo que resultó en una breve suspensión de la captura de imágenes para archivo. Sin embargo, el mal

Cuadro 8.5: Bandas Landsat 7 (ETM+).

Número de Banda	Descripción	Rango (ρ)
1	Azul	$450 - 520nm$
2	Verde	$520 - 600nm$
3	Rojo	$630 - 690nm$
4	NIR	$760 - 900nm$
5	SWIR1	$1550 - 1750nm$
6	Térmico	$10420 - 12500nm$
7	SWIR2	$2080 - 2350nm$

Nota:
ρ: Reflectancia Superficial.

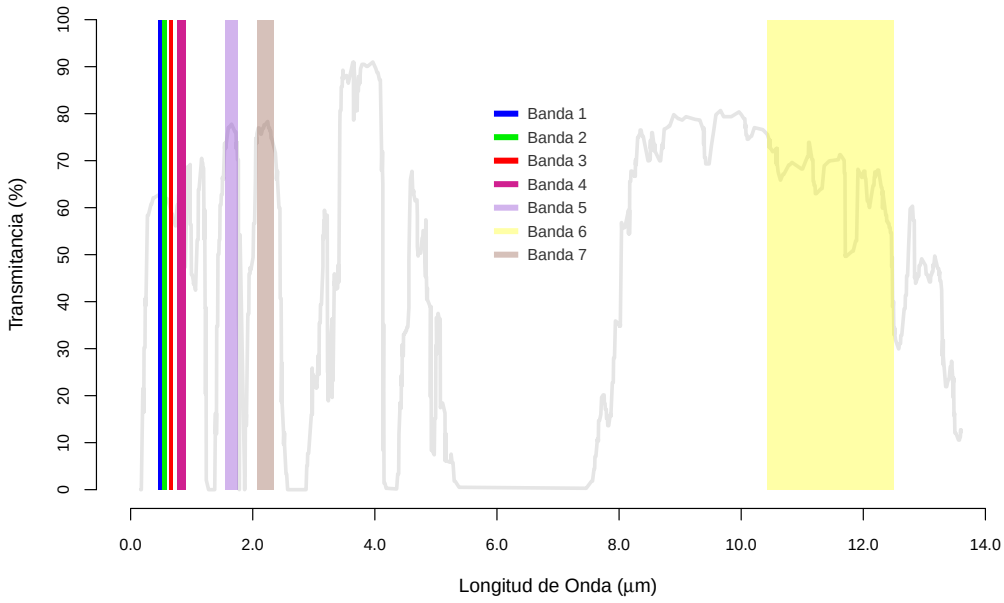


Figura 8.26: Distribución y espesor de bandas en sensor ETM+ de Landsat 7.

funcionamiento ha afectado las imágenes de Landsat 7.

Específicamente, la óptica ETM+ contiene el conjunto *Scan Mirror* y *Scan Line Corrector*, entre otros componentes. El espejo de exploración proporciona el movimiento a lo largo de la vía para la imagen, mientras que la velocidad de avance de la nave espacial proporciona el movimiento a lo largo de la vía. El conjunto Scan Line Corrector (SLC) se utiliza para **eliminar el movimiento en *'zigzag'* del campo de visión de la imagen** producto por la combinación del movimiento a lo largo y a través de la trayectoria. Sin un SLC operativo, la línea de visión del ETM+ ahora traza un patrón en zigzag a través de la trayectoria terrestre del satélite.

En este modo de SLC apagado, el ETM+ aún adquiere **aproximadamente el 75% de los datos para cualquier escena dada**. Los espacios en los datos forman cuñas alternas que aumentan de ancho desde el centro hasta el borde de una escena.

El resto del sensor ETM+, incluido el espejo primario, continúa funcionando, radiométrica y geométricamente, con el mismo alto nivel de exactitud y precisión que tenía antes de la anomalía; por lo tanto, los píxeles de la imagen aún están geolocalizados y calibrados con precisión.

8.9 Landsat 4-5

Los satélites *Landsat 4 y 5* siempre se mencionan juntos ya que fueron naves gemelas (salvo pequeñas modificaciones luego de estudiados algunos errores en Landsat 4), ambos orbitan la Tierra en una órbita casi polar sincrónica con el sol, a una altitud de 705 km (438 millas), inclinada a 98.2 grados y completan una órbita terrestre cada 99 $minutos$. Tienen una resolución temporal de 16 días con un tiempo de cruce ecuatorial: $09:45\ am \pm 15\ minutos$. Landsat 4 lanzado en julio de 1982 y dado de baja en junio de 2001, Landsat 5 fue lanzado en marzo de 1984 y dado de baja en enero de 2013.

El sensor a bordo es el *Mapeador Temático (TM)* (Thematic Mapper).

Las bandas se describen en el cuadro 8.6 y figura 8.27.

Cuadro 8.6: Bandas Landsat 4-5 (TM).

Número de Banda	Descripción	Rango (ρ)
1	Azul	$450 - 520nm$
2	Verde	$520 - 600nm$
3	Rojo	$630 - 690nm$
4	NIR	$760 - 900nm$
5	SWIR1	$1550 - 1750nm$
6	Térmico	$10410 - 12500nm$
7	SWIR2	$2080 - 2350nm$

Nota:

ρ: Reflectancia Superficial.

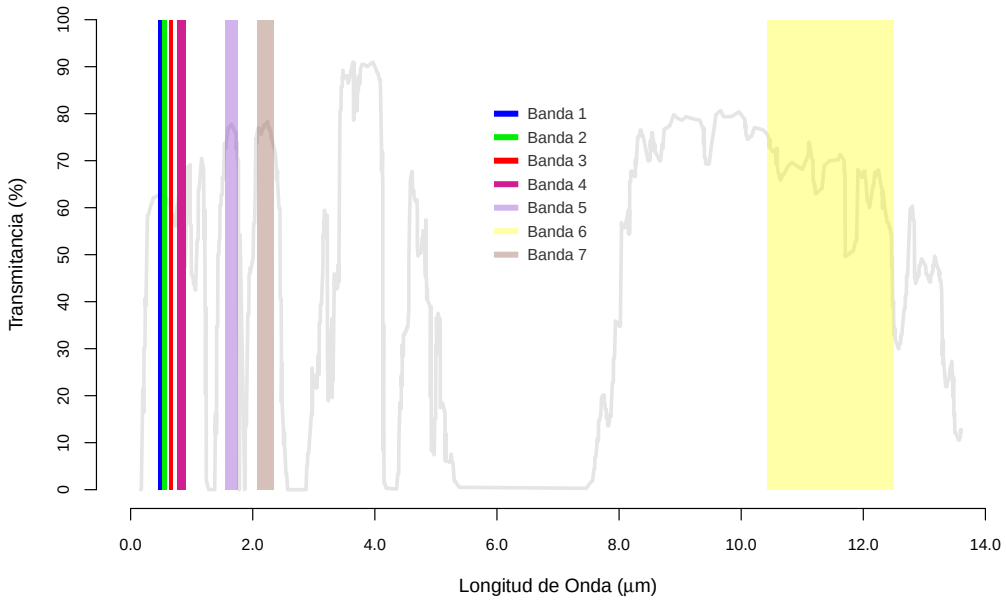


Figura 8.27: Distribución y espesor de bandas en sensor TM de Landsat 4-5.

8.10 Descarga de Datos para Landsat

Siguiendo la misma metodología de definición de parámetros de búsqueda visto para el sensor MODIS, (8.1) solo debemos seleccionar la lista **Landsat Collection 2 Level-2** (Figura 8.28) y seleccionar uno, dos o tres de las opciones disponibles.

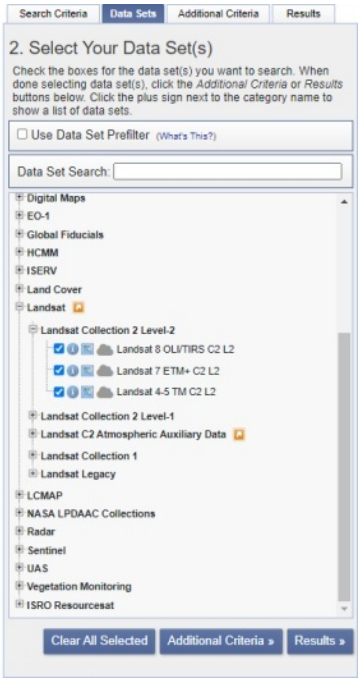


Figura 8.28: Lista de tipos de datos disponibles para descarga de productos de la misión Landsat.

Para el ejercicio práctico vamos a utilizar una imagen *Landsat 8* que cubre la zona costera de la Región Libertador General Bernardo O'Higgins de Chile (Figura 8.29). La cual es descargada dentro de nuestra carpeta de proyecto **raw** en la carpeta **sats**.

8.11 Landsat 8 en R

Las imágenes descargadas con el proceso descrito en 8.10 están comprimidas en formato *tar*. Usamos la función *dir* para crear la lista de archivos:

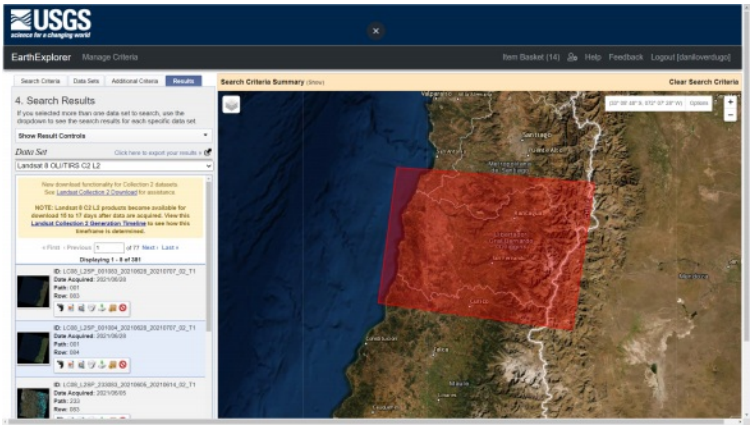


Figura 8.29: Área para descargar imágenes Landsat 8, Región Libertador General Bernado O'Higgins, Chile.

```
tar <- dir(path = here("raw","sats"), pattern ="tar")
```

A partir de los nombres de los archivos descargados, removemos la extensión *.tar* y construimos la lista con nombre de la carpeta donde extraer el contenido.

```
carpeta <- sub(".tar","", tar)
```

Y asociamos la dirección completa:

```
origen <- paste(here("raw","sats"), tar, sep = "/")
destino <- paste(here("raw","sats"), carpeta, sep = "/")
```

En un ciclo *for* extraemos los archivos:

```
for (i in 1:length(origen)){
  untar(tarfile = origen[i], exdir = destino[i])
}
```

El contenido de la carpeta se compone de tres tipos de archivos:

- **txt**: Metadatos
- **xml**: Metadatos en formato xml
- **TIF**: Archivos de imágenes

Podemos revisar el contenido total:

```
dir(path = destino[1])
```

```
[1] "LC08_L2SP_001084_20210628_20210707_02_T1_ANG.txt"
[2] "LC08_L2SP_001084_20210628_20210707_02_T1_MD5.txt"
```

```
 [3] "LC08_L2SP_001084_20210628_20210707_02_T1_MTL.txt"
 [4] "LC08_L2SP_001084_20210628_20210707_02_T1_MTL.xml"
 [5] "LC08_L2SP_001084_20210628_20210707_02_T1_QA_PIXEL.TIF"
 [6] "LC08_L2SP_001084_20210628_20210707_02_T1_QA_RADSAT.TIF"
 [7] "LC08_L2SP_001084_20210628_20210707_02_T1_SR_B1.TIF"
 [8] "LC08_L2SP_001084_20210628_20210707_02_T1_SR_B2.TIF"
 [9] "LC08_L2SP_001084_20210628_20210707_02_T1_SR_B3.TIF"
[10] "LC08_L2SP_001084_20210628_20210707_02_T1_SR_B4.TIF"
[11] "LC08_L2SP_001084_20210628_20210707_02_T1_SR_B5.TIF"
[12] "LC08_L2SP_001084_20210628_20210707_02_T1_SR_B6.TIF"
[13] "LC08_L2SP_001084_20210628_20210707_02_T1_SR_B7.TIF"
[14] "LC08_L2SP_001084_20210628_20210707_02_T1_SR_QA_AEROSOL.TIF"
[15] "LC08_L2SP_001084_20210628_20210707_02_T1_ST_ATRAN.TIF"
[16] "LC08_L2SP_001084_20210628_20210707_02_T1_ST_B10.TIF"
[17] "LC08_L2SP_001084_20210628_20210707_02_T1_ST_CDIST.TIF"
[18] "LC08_L2SP_001084_20210628_20210707_02_T1_ST_DRAD.TIF"
[19] "LC08_L2SP_001084_20210628_20210707_02_T1_ST_EMIS.TIF"
[20] "LC08_L2SP_001084_20210628_20210707_02_T1_ST_EMSD.TIF"
[21] "LC08_L2SP_001084_20210628_20210707_02_T1_ST_QA.TIF"
[22] "LC08_L2SP_001084_20210628_20210707_02_T1_ST_TRAD.TIF"
[23] "LC08_L2SP_001084_20210628_20210707_02_T1_ST_URAD.TIF"
```

También podemos crear listas por tipo de archivo:

```
tifs <- dir(path = destino[1], pattern ="TIF")
txts <- dir(path = destino[1], pattern ="txt")
xmls <- dir(path = destino[1], pattern ="xml")
```

8.11.1 Metadatos

Es importante tener acceso a la metadata de cada imagen para extraer información detallada de la escena, el formato son columnas separadas por coma. R cuenta con una familia de funciones para la carga de archivos ASCII, con una serie de parámetros que permite adaptarse a una amplia gama de combinaciones.

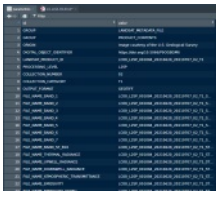


Figura 8.30: Visualización de Metadados en panel 'View'.

La siguiente configuración nos permite cargar los datos:

```
parametros <- read.csv(file = file.path(destino[1],
                                        txts[3]
                                        ),
                       header = FALSE,
                       sep = "=",
```

```
                        stringsAsFactors = FALSE,
                        strip.white = TRUE)
names(parametros) <- c('id','valor')
```

Para visualizar el contenido podemos utilizar el mismo entorno abriendo un nuevo panel y desplegar los datos en forma de planilla de datos (Figura 8.30).

```
View(parametros)
```

También existen varias alternativas de visualización o edición de datos:

```
fix(parametros)
edit(parametros)
data.entry(parametros)
```

Del enorme conjunto de datos disponibles, siempre conviene tener claro al menos los datos de:

- Porcentaje de cobertura Nubosa de la imagen:

```
cloud <- parametros[which(parametros$id == "CLOUD_COVER"), 2]
```

- Porcentaje de cobertura nubosa sobre superficie terrestre:

```
cloud_land <- parametros[which(parametros$id == "CLOUD_COVER_LAND"), 2]
```

- Fecha de toma de la imagen:

```
fecha <- parametros[which(parametros$id == "DATE_ACQUIRED"), 2]
```

- Nombre oficial de la imagen; se usa el índice *[1]* ya que existe varias veces en el archivo:

```
id <- parametros[which(parametros$id == "LANDSAT_PRODUCT_ID"), 2][1]
```

8.12 Procesando las Bandas en R

8.12.1 QA_pixel

El primer archivo *tif* de nuestro vector de imágenes es la imagen de control de calidad de los píxeles, que se basa en la infor-

mación de la banda de *Evaluación de Calidad de Nivel 1*, específicamente las marcas de confianza para Nubes, Sombra de Nubes y Nieve/Hielo extraídas del algoritmo **CFMask**.

Para respaldar los productos de datos científicos que utilizan el Nivel 2 como entrada, los valores del agua se vuelven a calcular y los píxeles de nubes de alta confianza se dilatan.

```
qa_px <- file.path(destino[1], tifs[1])
qa_px_r <- raster(qa_px)
```

El sistema coordenado de los datos es:

```
crs(qa_px_r)
```

Pero cabe mencionar que está basado en huso 18N (EPSG:32618) y necesitamos la versión sur, o EPSG:32718:

```
utm18s  <- CRS("+init=epsg:32718")
```

Reproyectamos usando el método del vecino más cercano (no queremos cambiar los valores de las marcas o banderas de calidad):

```
qa_px_18s <- projectRaster(from = qa_px_r,
                           crs = utm18s,
                           method = "ngb")
```

Y obtenemos nuestro nuevo rasterLayer:

```
qa_px_18s
```

```
class      : RasterLayer
dimensions : 7941, 7891, 62662431  (nrow, ncol, ncell)
resolution : 30, 30  (x, y)
extent     : 595635, 832365, 6048235, 6286465  (xmin, xmax, ymin, ymax)
crs        : +proj=utm +zone=18 +south +datum=WGS84 +units=m +no_defs
source     : qa_px_18s.grd
names      : LC08_L2SP_001084_20210628_20210707_02_T1_QA_PIXEL
values     : 1, 62820  (min, max)
```

Comparar las coordenadas en los slots *extent*.

En relación al contenido de la banda podemos revisar los va-

lores contenidos y revisar su significado en el cuadro 8.7.

```
unique(qa_px_18s)
```

```
 [1]     1 21762 21824 21890 21952 22018 22080 22146 22208 22280 23826
[12] 23888 24082 24144 29986 30048 30242 30304 54534 54596 54662 54724
[23] 54790 54852 54918 54980 55052 56598 56660 56854 56916 62820
```

Y los datos los podemos usar como máscara para futuros procesos (Figura 8.31):

```
mascara <- qa_px_18s
mascara[mascara != 21824] <- NA
```

Figura 8.31: Ejemplo de Máscara de QA píxel 'Claro'.

8.12.2 QA_RADSAT

El segundo archivo tif listado en el vector *tifs* es la banda de QA_RADSAT o banda de *Evaluación de la Calidad de Satura-*

Cuadro 8.7: Descripción de Valores Banda QA Pixel más importantes.

Valor	Descripción
1	Relleno
21824	Claro
21826	Nube sobre superficie
21888	Agua
21890	Nube sobre Agua
22080	Nube Confianza Media
22144	Nube Sobre Agua Confianza Media
22280	Nube
23888	Sombra Nubes
23952	Agua y Sombra Nube
24088	Nube y Sombra Confianza Media
24216	Nube y Sombra sobre el Agua Confianza Media
24344	Nube y Sombra Confianza Alta
24472	Nube y Sombra sobre Agua Alta Confianza
30048	Nieve o Hielo Alta Confianza
54596	Cirrus Alta Confianza
54852	Cirrus, Nube Media
55052	Cirrus Nubes Altas
56856	Cirrus y Sombra Nubes Confianza Media
56984	Cirrus, Sombra Nubes Confianza Media sobre Agua
57240	Cirrus y Sombra de Nubes Alta Confianza

ción Radiométrica (QA_RADSAT), que es una representación empaquetada de bits de qué bandas de sensores se saturaron durante la captura de datos, lo que arroja datos inutilizables. La saturación en Landsat 8 no es común. Cuando ocurre la saturación, ocurre sobre superficies reflectantes en las bandas ópticas, o volcanes e incendios forestales en el infrarrojo de onda corta (SWIR) y bandas térmicas.

La saturación se puede encontrar de dos formas: una, los píxeles saturados pueden mostrarse como el valor máximo de 16 bits: 65535; o dos, los valores de los píxeles pueden "pasar" al extremo inferior del rango válido (no necesariamente solo un valor de 0), lo que se denomina sobresaturación. La sobresaturación solo ocurre con OLI y no ocurrirá con las bandas térmicas TIRS. La banda L8 QA_RADSAT marcará solo los casos de saturación.

8.12.3 Bandas OLI

Los siguientes archivos en el vector corresponden a las bandas del sensor OLI que capturan la *reflectancia superficial* (ρ).

Los productos de procesamiento Nivel-2 o (productos científicos -SP) Landsat 8 categoría *Surface Reflectance (SR)* se generan a partir de un software especializado llamado *Land Surface Reflectance Code (LaSRC)*.

El algoritmo LaSRC original fue desarrollado por el Dr. Eric Vermote, Centro de Vuelo Espacial Goddard (GSFC) de la Administración Nacional de Aeronáutica y del Espacio (NASA) y fue modificado por el personal del Centro de Observación y Ciencia de Recursos Terrestres (EROS) del USGS.

LaSRC genera reflectancia de la parte superior de la atmósfera (TOA) y temperatura de brillo (BT) utilizando los parámetros de calibración de los metadatos. Las rutinas de corrección atmosférica se aplican luego a los datos de reflectancia TOA de Landsat 8, utilizando datos de entrada auxiliares como vapor de agua, ozono y espesor óptico de aerosoles (AOT) recuperados del espectrorradiómetro de imágenes de resolución moderada (MODIS), y elevación digital derivada de GTOPO5 para generar la reflectancia superficial.

```
bandas <- file.path(destino[1], tifs[c(3:9)])
nombres <- c("AC","B","G","R","NIR","SWIR1","SWIR2")
imagen1 <- stack(bandas)
names(imagen1) <- nombres
```

Con los parámetros contenidos en 8.2 se transforman los valores enteros a valores de reflectancia reales.

```
imagen <- (imagen1 * 0.0000275) + -0.2
imagen1_corr <- projectRaster(from = imagen,
                              crs = utm18s,
                              method = "bilinear")
```

Realizaremos un recorte en el sector de la desembocadura del *Río Mataquito*, Región Libertador General Bernardo O´Higgins en Chile para estudiar la relación entre las propiedades de la respuesta espectral y las coberturas naturales (Figura 8.32). La

zona comparte áreas boscosas, de cultivos y suelo desnudo.

Diametralmente diferente es la zona desértica y rocosa de la provincia de Tamanrasset, Algeria región que cubre la imagen MODIS procesada en 8.4.1, que aconsejo usar como patrón de comparación de los valores en cada banda (Figura 8.32).

```
recorte <- crop(x = imagen1_corr,
                y = extent(753411.8, 770155.5,
                           6114039, 6140000))
```

8.12.4 Imágenes en Color Real

En el punto 9.5 estudiamos en detalle el trabajo con la combinación de bandas para formar imágenes denominadas *falso color*, entre las más conocidas son:

- Color Verdadero o RGB. Figura 8.33.

```
plotRGB(recorte,
        r="R",
        g="G",
        b="B",
        stretch="lin",
        axes=F)
```

- Falso Color o NIR/R/G. Figura 8.34.

El MDR lo guardamos en disco, y en el punto 9.7 es usado en la construcción de *gráficos de dispersión*®.

```
writeRaster(recorte,
            here("raw","recorte_1.grd"),
            overwrite=T)
```

8.12.5 TIRS

El último archivo que vamos a describir es el archivo número 12 que corresponde a la *banda termal* o banda TIRS.

El producto *Landsat Surface Temperature (ST)* se genera a partir de las bandas infrarrojas térmicas de *Landsat Collection 2 Level 1* utilizando la reflectancia de la parte superior de la

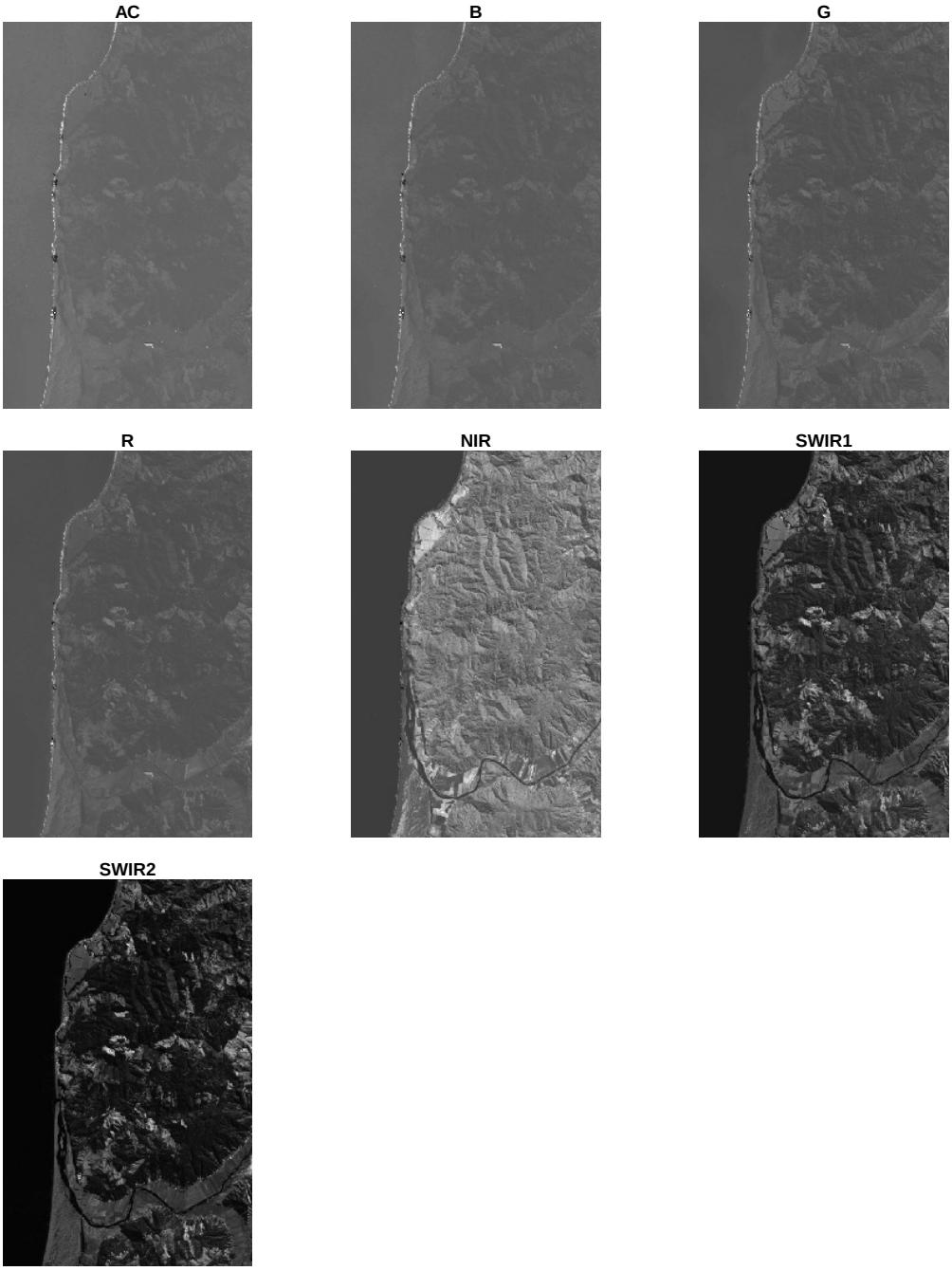


Figura 8.32: Bandas Imagen Landsat 8 OLI. Sector Desembocadura Río Mataquito, Región Libertador General Bernardo O´Higgins, Chile.

Figura 8.33: Composición Color Verdadero (RGB) Imagen Landsat 8 OLI. Sector Desembocadura Río Mataquito, Región Libertador General Bernardo O´Higgins, Chile

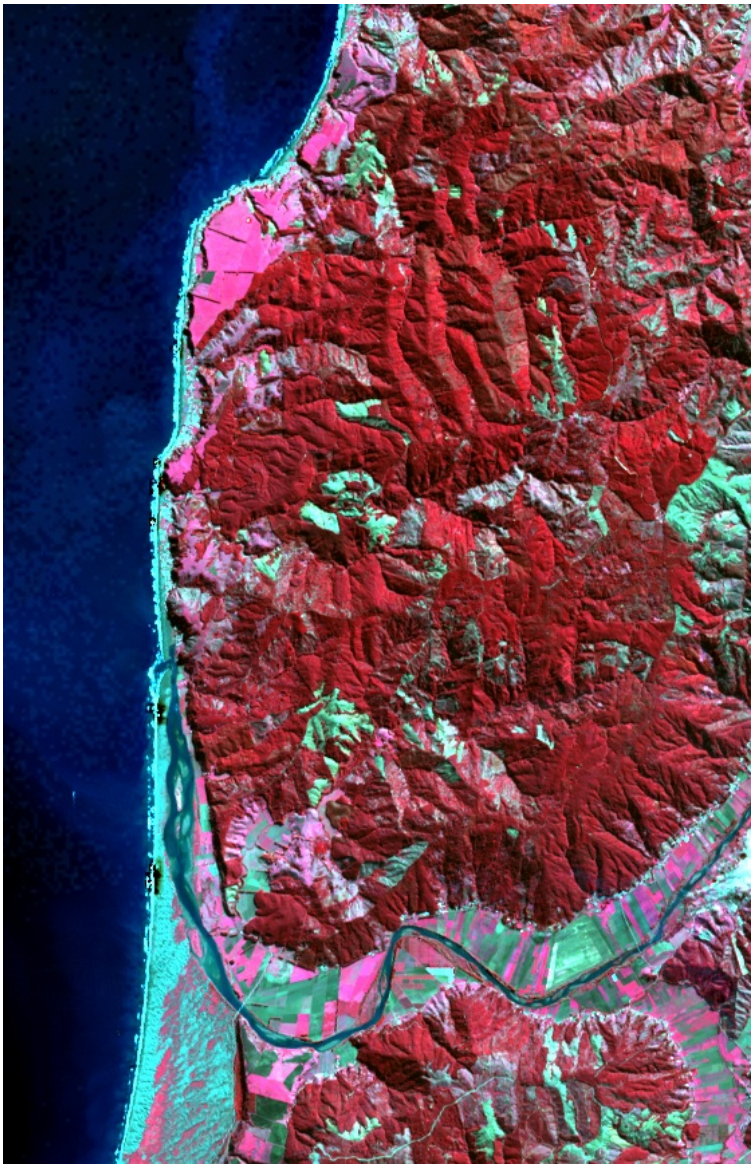


Figura 8.34: Composición Falso Color (NIR,R,G) Imagen Landsat 8 OLI. Sector Desembocadura Río Mataquito, Región Libertador General Bernardo O´Higgins, Chile

atmósfera (TOA), la temperatura de brillo (BT), el radiómetro de emisión y reflexión térmica avanzada del espacio (ASTER), la base de datos de emisividad global (GED), datos del índice de vegetación de diferencia normalizada (NDVI) de ASTER y perfiles atmosféricos de altura geopotencial, humedad específica y temperatura del aire extraídos de los datos de reanálisis.

Con los parámetros contenidos en el cuadro 8.2 se transforman los valores enteros a valores de temperatura superficial en grados celsius.

```
b10 <- raster(file.path(destino[1], tifs[12]))
termal <- (b10 * 0.00341802 + 149) - 273.15
termal_corr <- projectRaster(from = termal,
                             crs = utm18s,
                             method = "bilinear")

termal_recorte <- crop(x = termal_corr,
                       y = extent(753411.8, 770155.5,
                                  6114039, 6140000))
```

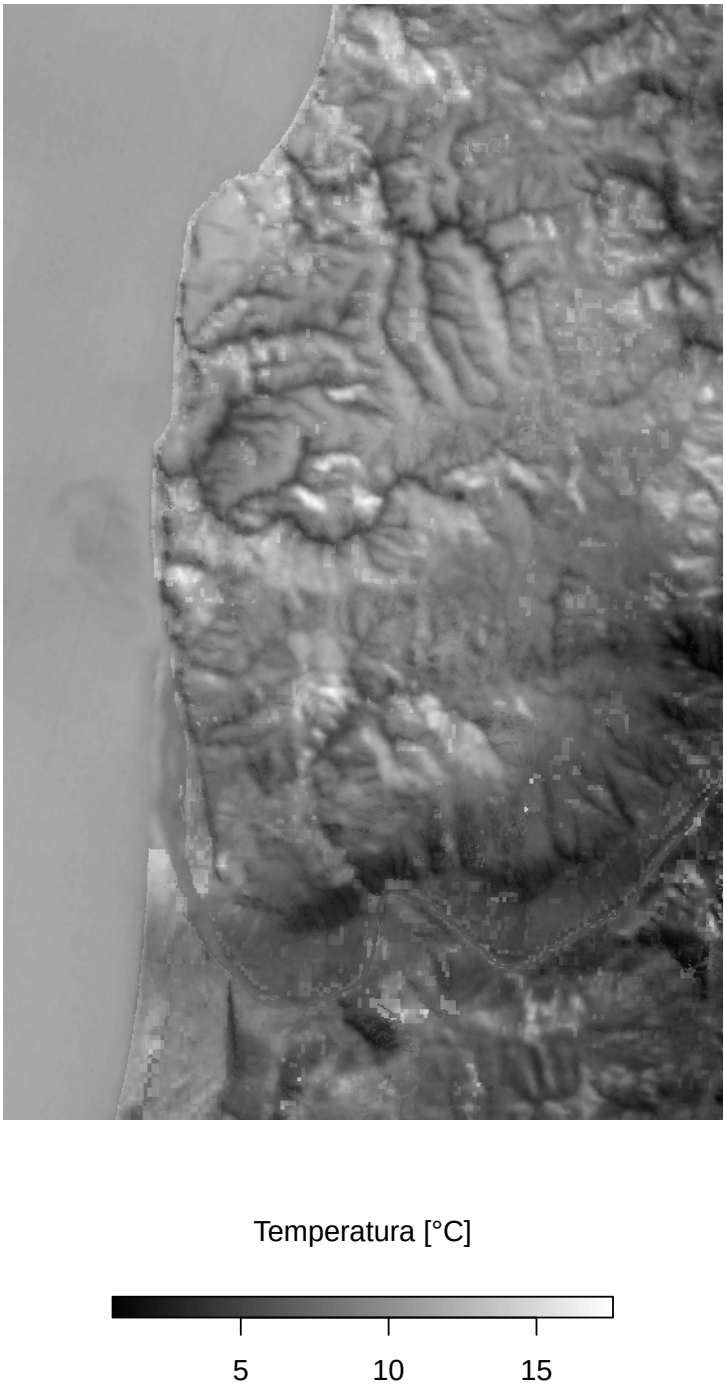


Figura 8.35: Banda Termal (B10), temperatura superficial en grados celsius. Imagen Landsat 8 TIRS. Sector Desembocadura Río Mataquito, Región Libertador General Bernardo O´Higgins, Chile

Misiones Sentinel

La **Agencia Espacial Europea (ESA)**[3] es la puerta de entrada de Europa al espacio. Su misión es dar forma al desarrollo de la capacidad espacial de Europa y garantizar que la inversión en el espacio continúe aportando beneficios a los ciudadanos de Europa y del mundo.

[3] Link www.esa.int

ESA desarrolló una serie de misiones espaciales de observación de la Tierra de próxima generación, en nombre de la iniciativa conjunta de la ESA y la Comisión Europea Copernicus.

El objetivo del programa Sentinel es reemplazar las misiones de observación de la Tierra más antiguas que se han retirado, como las misiones *ERS* y *Envisat*, o que actualmente se acercan al final de su vida útil operativa. Esto garantizará la continuidad de los datos para que no existan lagunas en los estudios en curso.

Cada misión se centra en un aspecto diferente de la observación de la Tierra; Monitoreo atmosférico, oceánico y terrestre, y los datos son útiles en muchas aplicaciones.

Sentinel-1: Su objetivo de monitoreo terrestre y oceánico, Sentinel-1 estará compuesto por dos satélites en órbita polar que operan día y noche, y realizará **imágenes de radar**, lo que les permitirá adquirir imágenes independientemente del clima. El primer satélite Sentinel-1 se lanzó en abril de 2014 (Figura 8.36).

Figura 8.36: Satélite Sentinel 1

Sentinel-2: El objetivo de Sentinel-2 es el monitoreo terrestre, y la misión estará compuesta por dos satélites en órbita polar que proporcionarán imágenes ópticas de alta resolución. La vegetación, el suelo y las zonas costeras se encuentran entre los objetivos de seguimiento. El primer satélite Sentinel-2 se lanzó en junio de 2015 (Figura 8.37).

Figura 8.37: Satélite Sentinel 2

Sentinel-3: El objetivo principal de Sentinel-3 es la observación marina, y estudiará la topografía de la superficie del mar, la temperatura de la superficie del mar y la tierra, el color del océano y la tierra. Compuesto por tres satélites, el instrumento principal de la misión es un altímetro de radar, pero los satélites en órbita polar llevarán múltiples instrumentos, incluidos los generadores de imágenes ópticas (Figura 8.38).

Figura 8.38: Satélite Sentinel 3

Sentinel-4: Se dedica al monitoreo de la calidad del aire. El instrumento Sentinel-4 UVN es un espectrómetro que se lleva a bordo de los satélites Meteosat de tercera generación, operado por EUMETSAT. La misión tiene como objetivo proporcionar un seguimiento continuo de la composición de la atmósfera de la Tierra con una alta resolución temporal y espacial y los datos se utilizarán para respaldar el seguimiento y la predicción en Europa (Figura 8.39).

Figura 8.39: Satélite Sentinel 4

Sentinel-5: Se dedica al monitoreo de la calidad del aire. El instrumento Sentinel-5 UVNS es un espectrómetro que se lleva a bordo de los satélites MetOp de segunda generación. La misión tiene como objetivo proporcionar un seguimiento continuo de la composición de la atmósfera terrestre. Proporciona datos de cobertura global de amplio espectro para monitorear la calidad del aire en todo el mundo (Figura 8.40).

Figura 8.40: Satélite Sentinel 5

Sentinel-5P: Fue una misión satelital precursora, tiene como objetivo llenar el vacío de datos y proporcionar continuidad de datos entre el retiro del satélite Envisat y la misión Aura de la NASA y el lanzamiento de Sentinel-5. El objetivo principal de la misión Sentinel-5P es realizar mediciones atmosféricas, con alta resolución espaciotemporal, relacionadas con la calidad del aire, forzamiento climático, ozono y radiación UV. El satélite se lanzó con éxito el 13 de octubre de 2017 desde el cosmódromo de Plesetsk en Rusia (Figura 8.41).

Figura 8.41: Satélite Sentinel 5p

8.13 Sentinel-2

La misión Copernicus Sentinel-2 comprende una constelación de dos satélites en órbita polar colocados en la misma órbita sincrónica del sol. El tiempo de revisita de 10 días en el ecuador

con un satélite y 5 días con 2 satélites en condiciones sin nubes, lo que da como resultado 2-3 días en latitudes medias.

Cada satélite SENTINEL-2 pesa aproximadamente 1.2 toneladas. **SENTINEL-2A** y **SENTINEL-2B** se han lanzado con el lanzador europeo VEGA.

La vida útil del satélite es de 7.25 años, que incluye una fase de puesta en servicio en órbita de 3 meses. Se han proporcionado baterías y propulsores para dar cabida a 12 años de operaciones, incluidas las maniobras de abandono de órbita al final de su vida útil.

Dos satélites SENTINEL-2 idénticos operan simultáneamente, en fases a 180° entre sí, en una órbita sincrónica con el sol a una altitud media de 786 km. La posición de cada satélite SENTINEL-2 en su órbita se mide mediante un receptor del Sistema de navegación por satélite global (GNSS) de doble frecuencia. La precisión orbital se mantiene mediante un sistema de propulsión dedicado.

El sistema de satélite SENTINEL-2 fue desarrollado por un consorcio industrial liderado por Astrium GmbH (Alemania). Astrium SAS (Francia) es responsable del *Instrumento multiespectral (MSI).*

8.13.1 Sensor MSI

El MSI funciona de forma pasiva, al recoger la luz solar reflejada desde la Tierra. Se adquieren nuevos datos en el instrumento a medida que el satélite se mueve a lo largo de su trayectoria orbital. El haz de luz entrante se divide en un filtro y se enfoca en dos conjuntos de plano focal separados dentro del instrumento; uno para las bandas Visible e Infrarrojo cercano (VNIR) y otro para las bandas de infrarrojos de onda corta (SWIR). La separación espectral de cada banda en longitudes de onda individuales se logra mediante filtros de bandas montados en la parte superior de los detectores.

El diseño óptico del telescopio MSI permite un campo de visión (FOV) de 290 km.

Un mecanismo de obturación evita que el instrumento sea ilu-

minado directamente por el sol en órbita y evita la contaminación durante el lanzamiento. El mismo mecanismo funciona como un dispositivo de calibración al recoger la luz solar después de la reflexión de un difusor.

- **Resolución Espacial**: Existen tres posibles resoluciones: 10, 20 y 60 m.
- **Resolución Espectral**: El detalle de la posición y ancho de bandas se detallan en el cuadro 8.8 y la figura 8.42.
- **Resolución Radiométrica**: Los datos son almacenados en un formato de 12 bits. Aunque los productos de segundo nivel de proceso son almacenados a 16 bits.
- **Resolución Temporal**: Los satélites de la constelación Sentinel-2 proporcionarán un tiempo de revisión de cinco días en el ecuador sin nubes.

Cuadro 8.8: Bandas MSI, S2A [S2B].

Banda	Nombre	Rango (ρ nm) S2A [S2B]
B1	Aerosoles	$432.2 - 453.2$ [$431.7 - 452.7$]
B2	Azul	$459.4 - 525.4$ [$459.1 - 525.1$]
B3	Verde	$541.8 - 577.8$ [$541.0 - 577.0$]
B4	Rojo	$649.1 - 680.1$ [$649.4 - 680.4$]
B5	Red Edge 1	$696.6 - 711.6$ [$695.8 - 711.8$]
B6	Red Edge 2	$733.0 - 748.0$ [$731.6 - 746.6$]
B7	Red Edge 3	$772.8 - 792.8$ [$769.7 - 789.7$]
B8	NIR	$779.8 - 885.8$ [$779.9 - 885.9$]
B8A	Red Edge 4	$854.2 - 875.2$ [$853.0 - 875.0$]
B9	Vapor Agua	$935.1 - 955.1$ [$932.7 - 953.7$]
B11	SWIR1	$1568.2 - 1659.2$ [$1563.4 - 1657.4$]
B12	SWIR2	$2114.9 - 2289.9$ [$2093.2 - 2278.2$]

Nota:
ρ: Reflectancia Superficial.

8.13.2 Niveles de Procesamiento

La **corrección atmosférica** es el proceso de eliminar los efectos de la atmósfera en los valores de reflectancia de la parte *supe-*

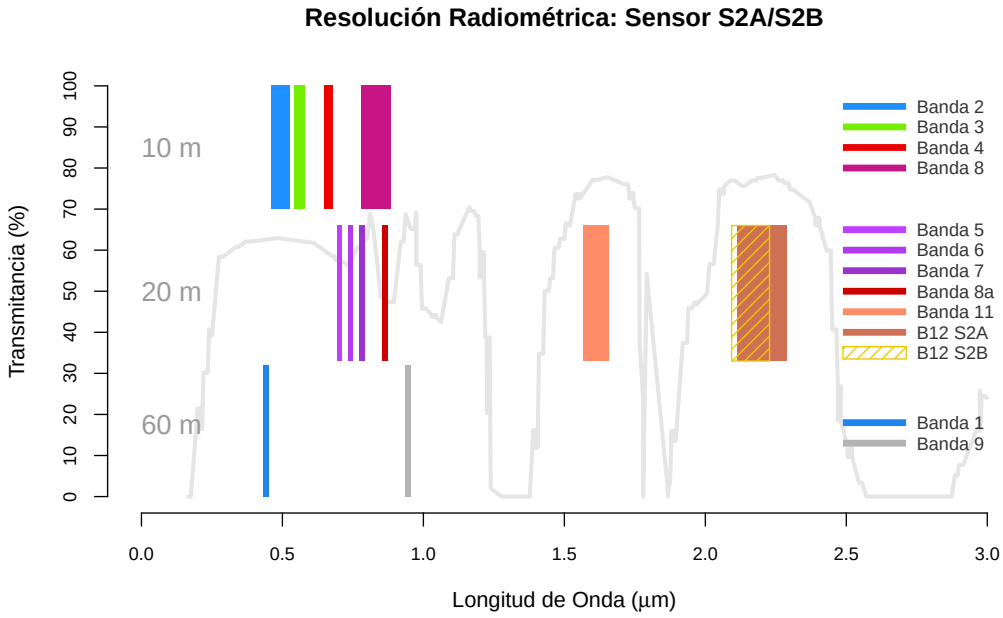


Figura 8.42: Distribución y espesor de bandas en MSI.

rior de la atmósfera (TOA) de las imágenes de satélite originales; La reflectancia TOA es una medida sin unidades que proporciona la relación entre la radiación reflejada y la radiación solar incidente en una superficie determinada. La reflectancia del fondo de la atmósfera (BOA), por otro lado, se define como la fracción de la radiación solar entrante que se refleja desde la superficie de la Tierra para el punto de visualización.

Sen2Cor es un algoritmo cuyo propósito fundamental es corregir una imagen Sentinel-2 producto de nivel 1C de los efectos de la atmósfera y resultan en una serie de productos junto al de reflectancia superficial de nivel 2A. Los valores de reflectancia superficial se encuentran generalmente entre 0,0 y 1,0. Los efectos especulares en la superficie o las nubes pueden dar lugar a valores superiores a 1,0. Para convertir los valores o niveles digitales almacenados en las imágenes a reflectancia superficial o BOA se debe aplicar la fórmula 8.1.

$$SR = \frac{ND}{10.000} \tag{8.1}$$

Un tercer nivel existió hasta marzo de 2018: **L2Ap**, o *pro-*

ductos pilotos generados por la ESA. Después de dicha fecha son productos operativos (L2A).

8.13.3 Productos Adicionales Especiales

El procesamiento vía sen2Cor además de generar las salidas en nivel 2A (L2A) incorporan una serie de productos adicionales (Detalles Adicionales en (Cuadro 8.10):

- **Clasificación de escena (SCL)**: Once clases de cobertura o uso de suelo (ver detalle en cuadro 8.9).
- **Indicadores de calidad (Probabilidad Nubes y Nieve)**. Mapas de probabilidad (%) de presencia de dichos elementos.
- **Espesor Óptico de Aerosoles (AOT)**: Es el grado en que los aerosoles impiden la transmisión de luz en la atmósfera. Un espesor óptico de aerosol de menos de 0,1 indica un cielo cristalino con máxima visibilidad, mientras que un valor de 4 indica la presencia de aerosoles tan densos que las personas tendrían dificultades para ver el Sol, incluso a mediodía.
- **Vapor de Agua (WVP)**: Rangos típicos de columnas de vapor de agua son (desde el nivel del mar hasta el espacio):
  - Condiciones tropicales: **wvp = 3-5 cm**
  - Verano de latitud media: **wvp = 2-3 cm**
  - Verano seco, primavera, otoño: **wvp = 1-1.5 cm**
  - Desierto seco o invierno: **wvp = 0.3-0.8 cm**
- **True Color Image (TCI)**: Imagen ajustada a la percepción visual humana en formato JPEG2000. El TCI es una imagen RGB construida a partir de las bandas B2 (azul), B3 (verde) y B4 (rojo). Las reflectancias están codificadas entre 1 y 255, reservando el 0 para 'Sin datos'. El nivel de saturación de 255 a un nivel 2000 en productos L2A ($\rho = 0.2$ en valor de reflectancia).

Cuadro 8.9: Códigos Tabla de Clases SCL.

Valor	SCL
1	Saturado o defectuoso
2	Área Obscura
3	Sombra de Nubes
4	Vegetación
5	Suelo Desnudo
6	Agua
7	Nubes Baja Probabilidad
8	Nubes Media Probabilidad
9	Nubes Alta Probabilidad
10	Cirros
11	Nieve/Hielo

Cuadro 8.10: Productos Adicionales Especiales MSI.

Nombre	Pixel	Descripción	Unidad	Escala
AOT	10	Espesor Aerosoles	-	0.001
WVP	10	Presión Vapor Agua	cm	0.001
SCL	20	Mapa Clasificación	-	-
MSK_CLD	20	Probabilidad Nubes	%	-
MSK_SNW	10	Probabilidad Nieve	%	-
TCI	10	Color Verdadero	-	-

8.14 Descarga de Imágenes Sentinel-2

El sitio web que centraliza el acceso al *Copernicus Open Access Hub* es: scihub.copernicus.eu.

Pasos principales para la descarga de imágenes:

1. El sitio se compone de una serie de enlaces a la data de las misiones Sentinel, novedades y estadísticas globales interesantes para el usuario. El acceso específico al panel de configuración de la búsqueda de productos Sentinel 1, 2 y 3 es mediante el sitio **Open Hub**, (Figura 8.43, área destacada en rojo).

2. Open Hub ofrece un mapa web para definir el área geográfica mediante el dibujo de un polígono y una serie de diálogos para especificar y refinar la búsqueda de productos (Figura 8.44).

3. Antes de realizar cualquier operación debes logearte o registrarte para acceder a los servicios (Figura 8.45).

4. El mapa web posee dos modalidades, *navegación* y *trazado*. En la primera debes realizar los paneos y zooms necesarios para buscar el área de estudio y en la segunda, dibujar el polígono que define dicha área, (Figura 8.46).

Figura 8.43: Vista inicial del sitio scihub.copernicus.eu. El link destacado en rojo da acceso a la data disponible de las misiones Sentinel-1,2 y 3.

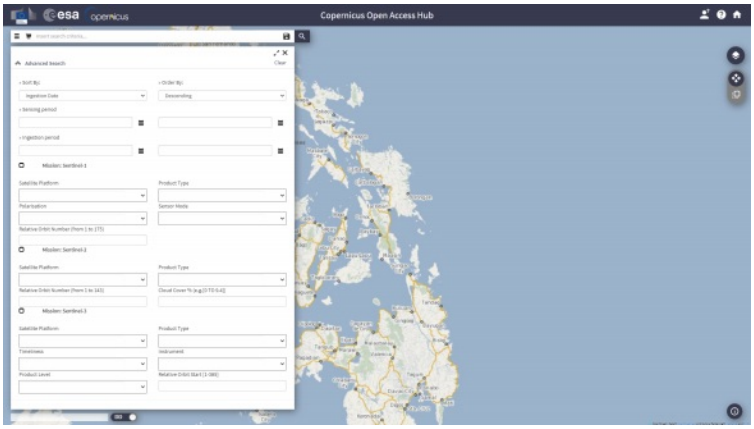


Figura 8.44: Panel de configuración de búsquedas en sitio 'scihub.copernicus.eu/dhus'

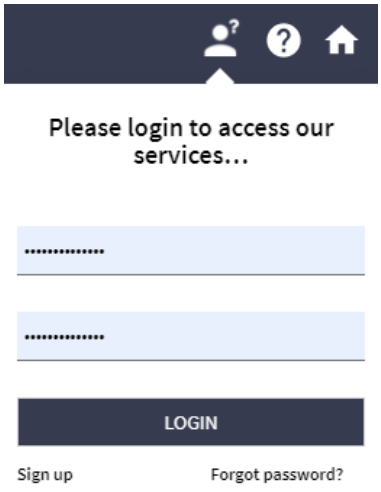


Figura 8.45: Cuadro de dialogo para logear o crear una cuenta.

Figura 8.46: Modos del mapa web. (Izquierda) Modo navegación para buscar y ajustar vista y (Derecha) Dibujar el polígono que define el área de estudio.

5. El trazado de polígono en el mapa web se puede realizar en dos modos:

- **Área Rectangular**: Luego de realizar el primer clic y *sin levantar* el botón del ratón arrastrar a la esquina opuesta se define el área de estudio rectangular.

- **Polígonal**: Una serie de clic para crear un polígono irregular para definir el área, hasta realizar un *doble clic* para finalizar el dibujo.

El resultado se usa para restringir los productos Sentinel que cubren o al menos interceptan el área (Figura 8.47).

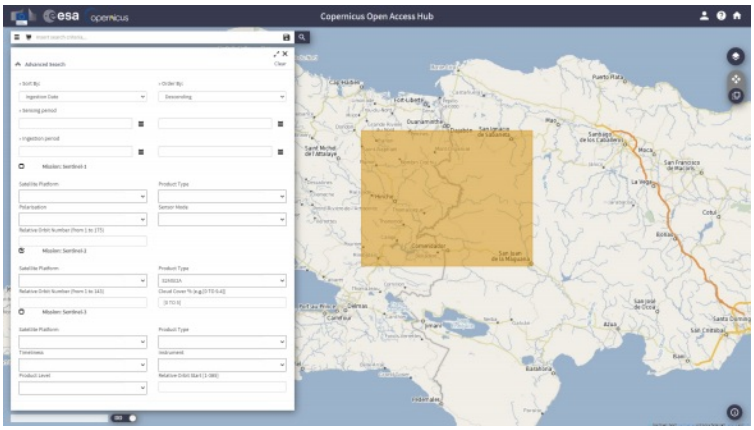


Figura 8.47: Vista principal del sitio 'earthexplorer.usgs.gov'

6. Los parámetros avanzados se agrupan en el cuadro de dialogo de configuración (Figura 8.48) con controles para configurar los parámetros de:

- **Propiedad de Ordenamiento**: Por fecha de procesamiento, captura o id de la escena.
- **Método de Ordenamiento**: Ascendente o descendente.
- **Fecha de Inicio y Término de Sensado**.
- **Fecha de Inicio y Término de Proceso**.

Las siguientes opciones del cuadro de dialogo agrupan los parámetros disponibles para las distintas misiones disponibles.

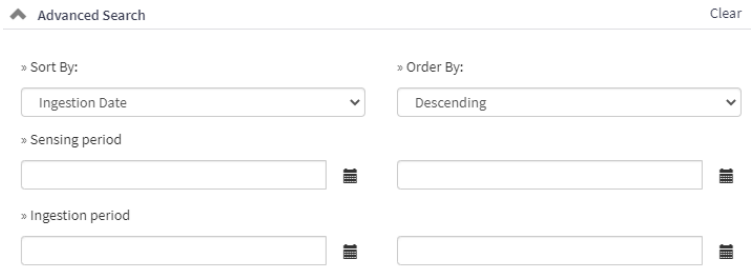
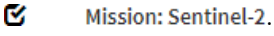


Figura 8.48: Cuadro de dialogo con opciones avanzadas de búsqueda.

7. Activar el check para descarga de misión Sentinel-2:



- **Selección del Satélite**: Al dejar la opción en blanco buscará por ambos vehículos. Seleccionando el nombre *S2A* o *S2B* se restringe solo a dicha opción (Figura 8.49).

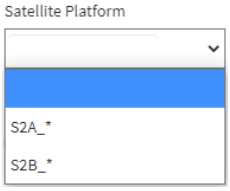


Figura 8.49: Selección de Satélite Sentinel.

- **Tipo de Producto**: Nivel de Procesamiento de las escenas (detalles en 8.13.2, figura 8.50). Si la opción se deja en blanco buscará en todos los niveles o tipos de procesamiento.

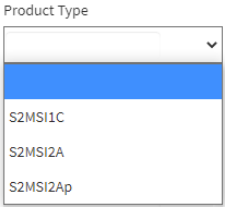


Figura 8.50: Opciones de procesamiento disponibles para escenas Sentinel-2.

- **Cobertura Nubosa**: Valor en porcentaje de la cobertura nubosa permitida. Idealmente ingresar un rango de valores en el formato señalado en el texto [inicio TO fin] expresado en %, (Figura 8.51).

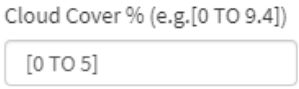


Figura 8.51: Caja de Texto para ingresar el rango de nubosidad aceptable en la búsqueda (nótese el formato específico).

Definidos los parámetros se pueden almacenar las opciones para futuras búsquedas o realizarla efectivamente ( ).

Por cada imagen que cumpla los criterios se presenta una vista de detalle (Figura 8.52), que forman una lista (Figura 8.53).

Cada elemento presenta cuatro opciones mediante los siguientes íconos:

- Centrar la vista en la imagen.
- Ver los detalles de la imagen.
- Agregar la imagen al carro de productos para su posterior descarga global.
- Descarga directa de la imagen.

Figura 8.52: Vista de detalle de la imagen seleccionada.

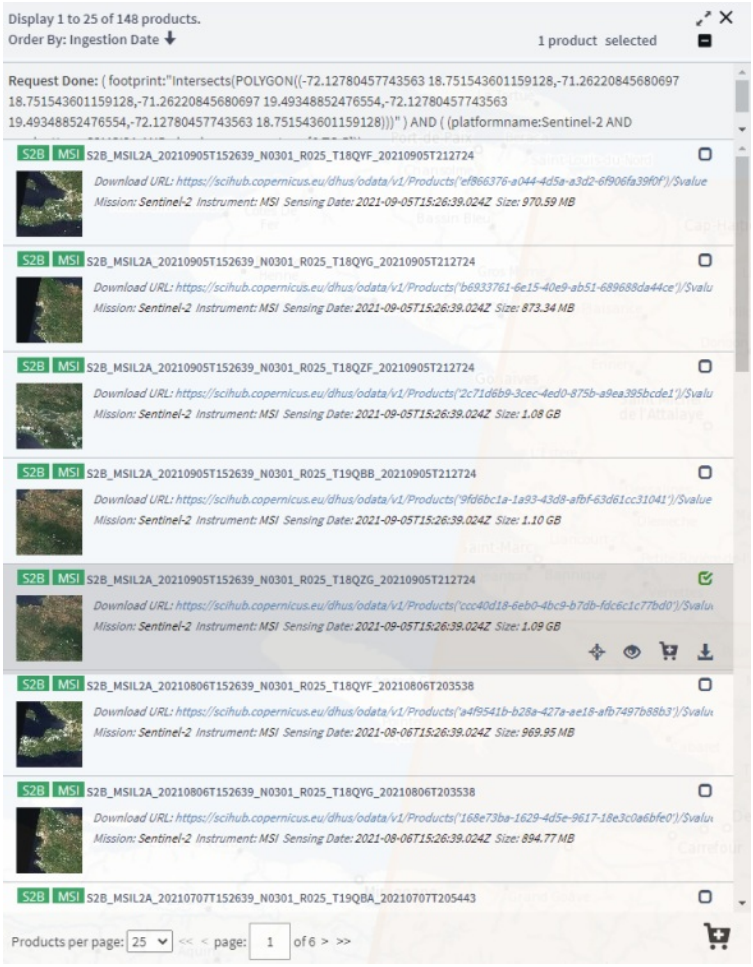


Figura 8.53: Lista de las escenas que cumplen con los criterios de búsqueda.

8.15 Imágenes Sentinel-2 en R

Las imágenes Sentinel-2 descargadas se encuentran en formato *.zip*, lo primero es construir un vector de texto con aquellos archivos que cumplan la extensión en la carpeta donde fueron descargados. En la carpeta que hemos ido desarrollando nuestros ejercicios la ruta es: *(carpeta de proyecto)*, *raw*, *sen*.

```
zipes <- dir(path = here("raw","sen"),
             pattern =".zip$")
```

Usando la función `sub(pattern, replacement, x)` donde en el objeto `x` se busca el texto `pattern` y es reemplazado por `replacement`, vamos a convertir la parte del texto en el nombre de las carpetas, *.zip* por un carácter vacío.

```
(carpetas <- sub(pattern = ".zip",
                 replacement = "",
                 x = zipes))
```

```
[1] "S2A_MSIL2A_20190131T134211_N0211_R124_T21HWB_20190131T160605"
[2] "S2B_MSIL2A_20190516T134219_N0212_R124_T21HWB_20190516T174745"
[3] "S2B_MSIL2A_20210905T152639_N0301_R025_T18QZG_20210905T212724"
```

Es importante destacar la longitud del nombre restante. Para Windows las rutas y nombres tan largos podría generar problemas por lo que vamos a extraer **un segmento de caracteres** que identifica y acorta los nombres de las carpetas que almacenarán los archivos por extraer.

La función `substr(x, start, stop)` toma el objeto `x` y selecciona el texto comenzando de la posición `start` hasta la posición `stop` y devuelve ese segmento.

```
(nombre <- substr(x = carpetas,
                  start = 1,
                  stop = 26))
```

```
[1] "S2A_MSIL2A_20190131T134211" "S2B_MSIL2A_20190516T134219"
[3] "S2B_MSIL2A_20210905T152639"
```

Ahora con toda la información en vectores podemos usar la función `unzip()` del paquete (que debemos instalar previamente)

zip (Csárdi et al. [2021]).

Los parámetros mínimos que debemos definir son `zipfile` que define la ruta y el nombre del o los archivos zip, `exdir` la ruta y nombre de la carpeta de salida y `junkpaths` que define si se deben ignorar todas las rutas de directorio al crear archivos. Si es TRUE, todos los archivos se crearán en *exdir*.

```
zip::unzip(zipfile = here("raw","sen", zipes),
           exdir = here("raw","sen",nombre),
           junkpaths=T)
```

Ahora por cada imagen descargada *zip* tenemos una carpeta con imágenes y archivos asociados que forman una escena Sentinel-2.

Para explorar y revisar el contenido detallamos una carpeta, en la configuración de mi proyecto es la tercera (*nombre[3]* aunque puede variar en el caso del lector) y configuramos con el parámetro `path`, también la extensión de los archivos a listar se define vía parámetro `pattern` (el uso del símbolo **$** especifica que la extensión *.jp2* DEBE estar al final del texto). Los restantes parámetros indican que debe buscar hacia dentro de cada carpeta y devolver los nombres y rutas:

```
jps <- list.files(path = here("raw","sen", nombre[3]),
                  pattern = ".jp2$",
                  recursive = T,
                  full.names = T)
```

Para revisar el contenido vamos a extraer los nombres de los archivos :

```
basename(jps)
```

```
 [1] "MSK_CLDPRB_20m.jp2"
 [2] "MSK_CLDPRB_60m.jp2"
 [3] "MSK_SNWPRB_20m.jp2"
 [4] "MSK_SNWPRB_60m.jp2"
 [5] "T18QZG_20210905T152639_AOT_10m.jp2"
 [6] "T18QZG_20210905T152639_AOT_20m.jp2"
 [7] "T18QZG_20210905T152639_AOT_60m.jp2"
 [8] "T18QZG_20210905T152639_B01_60m.jp2"
 [9] "T18QZG_20210905T152639_B02_10m.jp2"
[10] "T18QZG_20210905T152639_B02_20m.jp2"
[11] "T18QZG_20210905T152639_B02_60m.jp2"
[12] "T18QZG_20210905T152639_B03_10m.jp2"
```

```
[13] "T18QZG_20210905T152639_B03_20m.jp2"
[14] "T18QZG_20210905T152639_B03_60m.jp2"
[15] "T18QZG_20210905T152639_B04_10m.jp2"
[16] "T18QZG_20210905T152639_B04_20m.jp2"
[17] "T18QZG_20210905T152639_B04_60m.jp2"
[18] "T18QZG_20210905T152639_B05_20m.jp2"
[19] "T18QZG_20210905T152639_B05_60m.jp2"
[20] "T18QZG_20210905T152639_B06_20m.jp2"
[21] "T18QZG_20210905T152639_B06_60m.jp2"
[22] "T18QZG_20210905T152639_B07_20m.jp2"
[23] "T18QZG_20210905T152639_B07_60m.jp2"
[24] "T18QZG_20210905T152639_B08_10m.jp2"
[25] "T18QZG_20210905T152639_B09_60m.jp2"
[26] "T18QZG_20210905T152639_B11_20m.jp2"
[27] "T18QZG_20210905T152639_B11_60m.jp2"
[28] "T18QZG_20210905T152639_B12_20m.jp2"
[29] "T18QZG_20210905T152639_B12_60m.jp2"
[30] "T18QZG_20210905T152639_B8A_20m.jp2"
[31] "T18QZG_20210905T152639_B8A_60m.jp2"
[32] "T18QZG_20210905T152639_PVI.jp2"
[33] "T18QZG_20210905T152639_SCL_20m.jp2"
[34] "T18QZG_20210905T152639_SCL_60m.jp2"
[35] "T18QZG_20210905T152639_TCI_10m.jp2"
[36] "T18QZG_20210905T152639_TCI_20m.jp2"
[37] "T18QZG_20210905T152639_TCI_60m.jp2"
[38] "T18QZG_20210905T152639_WVP_10m.jp2"
[39] "T18QZG_20210905T152639_WVP_20m.jp2"
[40] "T18QZG_20210905T152639_WVP_60m.jp2"
```

Nos encontramos con 40 archivos jp2. Algunos corresponden a las bandas espectrales y otros a los productos adicionales. También se separan por resolución espacial mediante los sufijos *10m*, *20m* y *60m*.

8.15.1 Set de Imágenes a 10m

Vamos a extraer solo los nombres que contienen el sufijo *10m* mediante la función `str_detect(string, pattern)` que devuelve un vector lógico de la misma longitud que el vector que contiene los textos a chequear `string` donde el valor TRUE indica la existencia del texto definido con el parámetro `r, "\u0060pattern\u0060"`.

```
id <- str_detect(string = jps, pattern = "_10m")
```

Y usando el vector lógico podemos crear un subconjunto con los elementos buscados:

```
sen2_10m_img <- jps[id]
```

La imagen del tipo *TCI* (ver 8.16.1) no será incluida en la construcción de nuestro objeto RasterStack. Por lo que se remueve el índice que corresponde a esa imagen.

```
basename(sen2_10m_img[-6])
```

```
[1] "T18QZG_20210905T152639_AOT_10m.jp2"
[2] "T18QZG_20210905T152639_B02_10m.jp2"
[3] "T18QZG_20210905T152639_B03_10m.jp2"
[4] "T18QZG_20210905T152639_B04_10m.jp2"
[5] "T18QZG_20210905T152639_B08_10m.jp2"
[6] "T18QZG_20210905T152639_WVP_10m.jp2"
```

Lo que solo nos deja seis archivos jp2 en el siguiente orden:

```
nombres_10m <- c("AOT","B2","B3","B4","B8","WVP")
```

Y construimos nuestro producto final con el uso de la función stack y renombramos las capas que lo componen:

```
sen2_10m <- stack(sen2_10m_img[-6])
names(sen2_10m) <- nombres_10m
```

Si inspeccionamos los valores máximos y mínimos podemos observar que se encuentran en ND, con una resolución radiométrica de 16 bits; se aplica la fórmula 8.1 para llevar los datos a reflectancia superficial o BOA (no aplicamos a las capas AOT y WVP).

```
sen2_10m_sr <- sen2_10m[[2:4]] /10000
```

Y ajustamos los valores al rango válido mediante la función `clamp(x, lower, upper)` que restringe los valores de `x` al valor mínimo `lower` y valor máximo `upper`.

```
sen2_10m_s <- clamp(x = sen2_10m_sr,
                    lower= 0, upper= 1)
```

Finalmente, se deben agregar las capas adicionales que fueron removidas en el paso anterior:

```
sen2_10m_img <- addLayer(sen2_10m_s,
                         sen2_10m$AOT,
                         sen2_10m$WVP)
```

Y para generar la figura 8.54 usamos el código:

```
plotRGB(sen2_10m_img, r="B4", g="B3", b="B2", stretch="lin", ext=ext)
```

Figura 8.54: Detalle Fort Liberté Haití, isla La Española. Imagen Sentinel-2 a 10 m, composición RGB. Captura 5 de septiembre del 2021.

8.15.2 Set de Imágenes a 20 m

Repetimos los pasos para las imágenes con resolución de 20 m, debemos también ajustar los nombres de las bandas.

```
id <- str_detect(jps, "_20m")
sen2_20m_img <- jps[id]
```

```
basename(sen2_20m_img[-14])
```

```
 [1] "MSK_CLDPRB_20m.jp2"
 [2] "MSK_SNWPRB_20m.jp2"
 [3] "T18QZG_20210905T152639_AOT_20m.jp2"
 [4] "T18QZG_20210905T152639_B02_20m.jp2"
 [5] "T18QZG_20210905T152639_B03_20m.jp2"
 [6] "T18QZG_20210905T152639_B04_20m.jp2"
 [7] "T18QZG_20210905T152639_B05_20m.jp2"
 [8] "T18QZG_20210905T152639_B06_20m.jp2"
 [9] "T18QZG_20210905T152639_B07_20m.jp2"
[10] "T18QZG_20210905T152639_B11_20m.jp2"
[11] "T18QZG_20210905T152639_B12_20m.jp2"
[12] "T18QZG_20210905T152639_B8A_20m.jp2"
[13] "T18QZG_20210905T152639_SCL_20m.jp2"
[14] "T18QZG_20210905T152639_WVP_20m.jp2"
```

```
nombres_20m <- c("MSK_CLD","MSK_SNW","AOT",
                 "B2","B3","B4","B5","B6",
                 "B7","B11","B12","B8A","SCL",
                 "WVP")
```

```
sen2_20m <- stack(sen2_20m_img[-14])
names(sen2_20m) <- nombres_20m
sen2_20m_sr <- sen2_20m[[4:12]] /10000
sen2_20m_sr <- clamp(x = sen2_20m_sr, lower= 0, upper= 1)
sen2_20m_sr <- addLayer(sen2_20m_sr, sen2_20m$MSK_CLD,
                        sen2_20m$MSK_SNW, sen2_20m$AOT,
                        sen2_20m$WVP, sen2_20m$SCL)
```

Y su imagen en RGB a 20 m (Figura 8.55).

Figura 8.55: Detalle Fort Liberté Haití, isla La Española. Imagen Sentinel-2 a 20 m, composición RGB. Captura 5 de septiembre del 2021.

8.15.3 Set de Imágenes de 60 m

Y lo mismo realizamos para la resolución de imágenes de 60 m.

```
id <- str_detect(jps, "_60m")
sen2_60m_img <- jps[id]
```

```
basename(sen2_60m_img[-16])
```

```
 [1] "MSK_CLDPRB_60m.jp2"
 [2] "MSK_SNWPRB_60m.jp2"
 [3] "T18QZG_20210905T152639_AOT_60m.jp2"
 [4] "T18QZG_20210905T152639_B01_60m.jp2"
 [5] "T18QZG_20210905T152639_B02_60m.jp2"
 [6] "T18QZG_20210905T152639_B03_60m.jp2"
 [7] "T18QZG_20210905T152639_B04_60m.jp2"
 [8] "T18QZG_20210905T152639_B05_60m.jp2"
 [9] "T18QZG_20210905T152639_B06_60m.jp2"
[10] "T18QZG_20210905T152639_B07_60m.jp2"
[11] "T18QZG_20210905T152639_B09_60m.jp2"
[12] "T18QZG_20210905T152639_B11_60m.jp2"
[13] "T18QZG_20210905T152639_B12_60m.jp2"
[14] "T18QZG_20210905T152639_B8A_60m.jp2"
[15] "T18QZG_20210905T152639_SCL_60m.jp2"
[16] "T18QZG_20210905T152639_WVP_60m.jp2"
```

```
nombres_60m <- c("MSK_CLD","MSK_SNW","AOT",
                 "B1","B2","B3","B4","B5",
                 "B6","B7","B9","B11","B12",
                 "B8A","SCL","WVP")
sen2_60m <- stack(sen2_60m_img[-16])
names(sen2_60m) <- nombres_60m
sen2_60m_sr <- sen2_60m[[4:14]] /10000
sen2_60m_sr <- clamp(x = sen2_60m_sr, lower= 0, upper= 1)
sen2_60m_sr <- addLayer(sen2_60m_sr, sen2_60m$MSK_CLD,
                        sen2_60m$MSK_SNW, sen2_60m$AOT,
                        sen2_60m$WVP, sen2_60m$SCL)

#writeRaster(sen2_60m_sr, here("raw","sen2_60m_sr.grd"), overwrite=T)
```

Y su figura a 60 m de resolución es 8.56.

Figura 8.56: Detalle Fort Liberté Haití, isla La Española. Imagen Sentinel-2 a 60 m, composición RGB. Captura 5 de septiembre del 2021.

8.16 Productos Adicionales Especiales

Como se señala en 8.13.3 existe un juego de imágenes adicionales que componen las escenas Sentinel-2. Cada una puede requerir el uso de funciones adicionales o procesamientos diferentes, a continuación, vamos a procesar cada una de ellas.

8.16.1 TCI

La imagen de color verdadero o TCI es una composición RGB en el mismo formato *.jp2* y hasta ahora la hemos excluido de los procesos.

Mediante el paquete *magick* (Ooms [2021]), agregamos la capacidad de leer variados formatos (png, jpeg, tiff, pdf, etc) y modificarlos (rotar, escalar, recortar, aplicar efectos, etc). La función `image_read(x)` regresa un objeto *'magick'*,

```
tci_60m <- image_read(sen2_60m_img[16])
```

```
tci_60m
```

```
  format width height colorspace matte filesize density
  JP2  1830   1830       sRGB FALSE  3727816   72x72
```

Que permite manejar de manera similar una fotografía a un objeto rasterLayer (Figura 8.57). La salida al comando `print(tci_60m)` se realiza en el panel **Viewer** a escala verdadera y el sistema provee dos barras para deslizar y poder explorar la imagen.

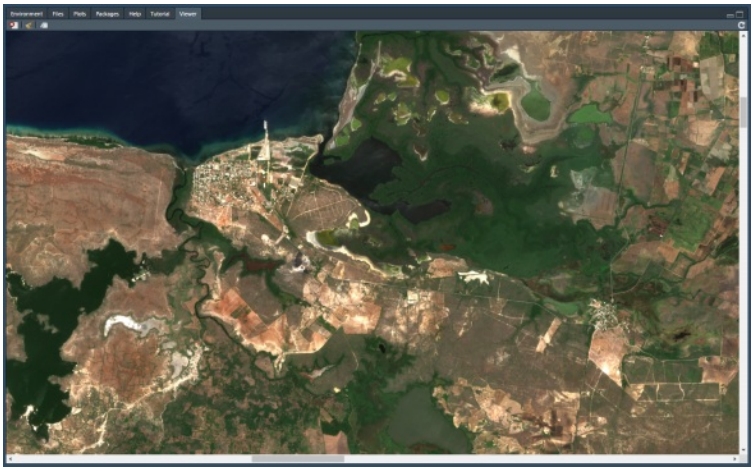


Figura 8.57: Visualización de la imagen en el panel 'Viewer'.

8.16.2 AOT

El **Espesor Óptico del Aerosol o AOT** (Figura 8.58) es el grado en que los aerosoles impiden la transmisión de luz.

AOT también conocido como **profundidad óptica del aerosol** o **espesor óptico (τ)** se define como el coeficiente de extinción integrado sobre una columna vertical de sección transversal unitaria.

El coeficiente de extinción es el agotamiento fraccional de la radiancia por unidad de longitud del camino (también llamado atenuación, especialmente en referencia a las frecuencias de radar). El grosor óptico a lo largo de la dirección vertical también se denomina *grosor óptico normal* (en comparación con el *grosor óptico a lo largo de la trayectoria inclinada*).

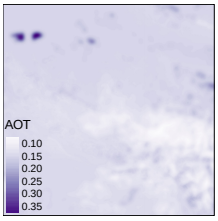


Figura 8.58: Producto AOT, Imagen Sentinel-2 a 60 m. Captura 5 de septiembre del 2021.

Aplicaciones:

1. Corrección atmosférica de elementos sensados.
2. Monitoreo de fuentes y sumideros de aerosoles.
3. Seguimiento de erupciones volcánicas e incendios forestales.
4. Modelo de transferencia radiativa.
5. Calidad del aire.
6. Salud y Medio Ambiente.
7. Cambio climático.

Para la conversión a magnitud real se escala por 1/1000.

```
aot_60m <- raster(sen2_60m_img[3])/1000
```

8.16.3 WVP

La capa denominada WVP o Columna de Vapor de Agua, se expresa en *cm* y se debe escalar usando el factor 1/1000 (Figura 8.59).

```
wvp_60m <- raster(sen2_60m_img[17])/1000
```

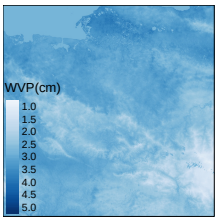


Figura 8.59: Producto WVP, Imagen Sentinel-2 a 60 m. Captura 5 de septiembre del 2021.

8.16.4 SCL

El producto Clasificación de Escena o SCL (8.9) se compone de 11 categorías las cuales se recomiendan tratar cono *factores*. Primero la convertimos en objeto rasterLayer estándar:

```
scl_60m <- raster(sen2_60m_img[15])
```

Y mediante la función `ratify()` se convierte en un ráster categórico.

```
scl_fac <- ratify(scl_60m, count=T)

levels(scl_fac)[[1]] <- cbind(levels(scl_fac)[[1]],
                              clases= c("área obscura","Sombra Nubes",
                                        "Vegetación", "Suelo Desnudo",
                                        "Agua","Nube baja","Nube Media",
                                        "Nube Alta","Nieve/Hielo"))
```

Y revisamos la distribución por categoría:

```
levels(scl_fac)
```

```
[[1]]
  ID   COUNT        clases
1  2   10593  área obscura
2  3    4013  Sombra Nubes
3  4 2388123    Vegetación
4  5  706104 Suelo Desnudo
5  6  221791         Agua
6  7   12844     Nube baja
7  8    2061    Nube Media
8  9    3336     Nube Alta
9 11      35   Nieve/Hielo
```

El producto SCL de la escena Sentinel-2 del ejercicio se puede revisar en su leyenda. (Figura 8.60).

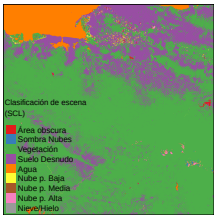


Figura 8.60: Producto SCL, Imagen Sentinel-2 a 60 m. Captura 5 de septiembre del 2021.

8.16.5 Máscaras de Probabilidad

Las máscaras de Probabilidad de Nubes y Nieves clasifican cada píxel de la escena con un valor porcentual que define que tan probable o esperado sea uno u otro elemento.

El valor digital de 0 no es nube ni nieve, 255 valor saturado. La figura 8.61 muestra la máscara de nubes.

```
msk_cld <- raster(sen2_60m_img[1])
msk_snw <- raster(sen2_60m_img[2])
```

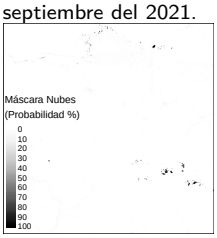


Figura 8.61: Producto Probabilidad de Nubes, Imagen Sentinel-2 a 60 m. Captura 5 de septiembre del 2021.

8.17 Sentinel-3

Sentinel-3 es una plataforma de múltiples instrumentos que se centra en la *topografía de la superficie del océano*, así como en la *temperatura de la superficie terrestre y marina.*

La plataforma lleva el radiómetro de temperatura de la superficie del mar y la tierra (SLSTR), el instrumento de color oceánico y terrestre (OLCI), así como un radar de apertura sintética (SAR) y un radiómetro de microondas (MWR).

8.17.1 Sensor SLSTR

El instrumento SLSTR se desarrolló para continuar y ampliar las mediciones realizadas por los instrumentos del radiómetro de barrido Along Track (ATSR) a bordo de los satélites 1 y 2 de detección remota europea (ERS) y el ATSR avanzado en Envisat.

El SLSTR tiene 9 bandas que cubren las áreas visibles (VIS), infrarroja de onda corta (SWIR) e infrarroja térmica (TIR) del espectro.

Con un tiempo medio de revisita de cobertura global en el ecuador de $1,9$ *das* (una nave espacial) o $0,9$ *das* (dos naves espaciales).

Ancho de franja de vista única ampliada de 1470 *km* que pro-

porciona un tiempo medio de revisión de cobertura global en el ecuador de 1 *da* (una nave espacial) o medio día (dos naves espaciales).

Una resolución en tierra de $0,5\ km$ en el nadir (en lugar de 1 km) para todos los canales VIS y SWIR. Las mediciones de resplandor de estos canales se utilizan para observaciones diurnas tanto terrestres como de nubes.

Dos canales agregados (en longitudes de onda de $2,25-1,375\ \mu m$) en la banda SWIR para permitir una mejor detección de nubes y aerosoles para brindar captura de SST / LST más precisas.

Dos canales dedicados para el monitoreo de incendios y eventos de alta temperatura con una resolución de 1 km (al extender el rango dinámico del canal de $3,7\ \mu m$ e incluir detectores dedicados a $10,8\ \mu m$ que son capaces de detectar incendios a $\sim 650\ K$ sin saturación).

Una vida útil de diseño de la misión de 7,5 años, superior a la de los instrumentos anteriores.

8.17.2 Temperatura Superficie Mar (SST)

La medición de la temperatura de la superficie del mar (SST) se obtiene mediante una calibración de alta precisión de los tres canales infrarrojos a $3.74\ , 10.85\ , 12.0\ \mu m$ (S7-S8-S9 respectivamente) utilizados para la corrección de la absorción atmosférica del vapor de agua (ventana dividida durante el día y ventana triple durante la noche) y observación del mismo píxel en el suelo mediante dos vistas de trayectoria atmosférica para la corrección de los efectos de los aerosoles.

La SST varía entre $-1.8°C$, la temperatura a la que se congela el agua de mar, y $+30°C$ cerca o debajo del Ecuador.

Es importante tener en cuenta que SLSTR devuelve mediciones de SST para la *piel* del océano. Debido a la *penetración limitada de la radiación térmica infrarroja a través de la columna de agua*, la temperatura radiométrica infrarroja es **sólo la de las primeras decenas de micrómetros**. La temperatura de la superficie de la piel del mar suele ser unas décimas de grado más fría que la temperatura unos centímetros por debajo.

La SST entendida oceanográficamente es una medida de la temperatura en los 10 *cm* superiores.

La información SST no se encuentra libremente disponible.

8.17.3 Temperatura Superficie Terrestre (LST)

La temperatura de la superficie terrestre (LST) es la temperatura radiante de la piel de la tierra derivada de la radiación infrarroja. En el proyecto SLSTR, la temperatura de "piel" se refiere a la temperatura de la superficie superior cuando está en condiciones de suelo desnudo, ya la temperatura de emisión efectiva de las "copas de los árboles" de vegetación, determinada desde una vista de la parte superior de una copa.

Una definición simplificada sería qué tan caliente se sentiría la "superficie" de la Tierra al tacto en un lugar en particular. Desde el punto de vista de un satélite, la "superficie" es lo que ve cuando mira a través de la atmósfera hacia el suelo. Podría ser nieve y hielo, la hierba en el césped, el techo de un edificio o las hojas en el dosel de un bosque. LST no es lo mismo que la temperatura del aire que se incluye en el informe meteorológico diario.

LST es un determinante básico del comportamiento térmico terrestre, ya que controla la temperatura de radiación efectiva de la superficie de la Tierra. Sin embargo, debido a la extrema heterogeneidad de la mayor parte de la superficie terrestre natural, este parámetro es difícil de estimar y validar. Varios factores pueden influir fundamentalmente en la derivación de LST, incluidos:

- Variaciones de temperatura con ángulos de visión.
- Homogeneidades de subpíxeles en temperatura y cobertura.
- Emisividad espectral de superficie en las longitudes de onda del canal.
- Variaciones de temperatura y humedad atmosféricas.
- Nubes y partículas grandes de aerosol como polvo.

Las bandas térmicas SLSTR utilizadas para la recuperación de SST, los tres canales infrarrojos 3.74 , 10.85 , 12.0 μm (S7-S8-S9

respectivamente) también se utilizan para recuperar LST en los productos SLSTR Nivel-2.

Los algoritmos para derivar LST usando radiancias están lo suficientemente avanzados como para que sea posible una precisión de $1K$, especialmente por la noche cuando no hay calentamiento diferencial de la superficie.

SLSTR también incluye dos canales IR (fuego y alta temperatura) de baja ganancia y amplio rango dinámico (F1 y F2) diseñados para entregar los datos radiométricos necesarios para la generación de productos cuantitativos contra incendios activos (productos FRP). Esto evita la saturación de los canales térmicos y se aplica a objetivos con un límite superior de $500°C$.

8.18 Descarga de Datos

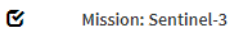
Cada producto consta de un archivo de manifiesto XML (xfdumanifest.xml) y un conjunto de once medidas y archivos de datos de anotación en formato NetCDF-4. Estos incluyen datos tanto en formato de grilla (archivos * _i? .Nc) como en cuadrícula de puntos (archivos * _t? .nc).

Se proporciona información geofísica sobre la geolocalización tanto en forma cartesiana (cartesian_*.nc), geodésica (geodetic_*.nc), ángulos cenital, solar y acimutal (geometry_tn.nc).

Escaneo a nivel de instrumento, orígenes de píxeles y detectores (indices_in.nc), tiempo de adquisición (time_in.nc). Condiciones meteorológicas (met_tx.nc); enmascaramiento de nubes (flags_in.nc); y el LST (LST_in.nc y LST_ancillary_ds.nc).

Para la descarga se deben seguir los mismos pasos descritos en 8.15 pasos 1 al 6, y a continuación seguir los siguientes pasos:

1. Activar check para descarga de misión Sentinel-3:



2. Seleccionar por instrumento correspondiente a la data deseada (Figura 8.62):

- OLCI: es un instrumento óptico utilizado para proporcionar continuidad de datos para MERIS de ENVISAT. OLCI es un

Figura 8.62: Seleccionar el instrumento disponible.

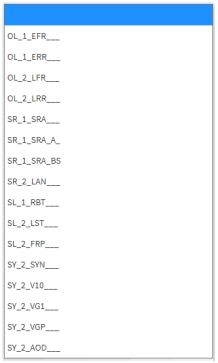


Figura 8.63: Productos disponibles publicamente para los instrumentos Sentinel-3.

espectrómetro de imágenes que mide la radiación solar reflejada por la Tierra, a una resolución espacial terrestre de 300 m, en 21 bandas espectrales.

- SRAL: misión topográfica para el estudio de la topografía oceánica, incluido el nivel medio del mar, la altura de las olas, la velocidad del viento sobre la superficie, el hielo marino, las corrientes oceánicas, las olas de Kelvin y Rossby, los remolinos y las mareas.
- SLSTR: es un radiómetro de imágenes de escaneo cónico que emplea la técnica de vista dual de escaneo a lo largo de la trayectoria para proporcionar una corrección atmosférica sólida en una franja de vista dual. El instrumento incluye canales en el espectro infrarrojo visible (VIS), térmico (TIR) y de onda corta (SWIR).
- SYNERGY: Metodología diseñada para proporcionar continuidad con SPOT VEGETATION. El objetivo principal de los productos SYN es el seguimiento del uso de la tierra. SYN también proporciona información relacionada con la seguridad alimentaria mundial y contribuye al estudio del clima.

3. Seleccionar por producto específico (Figura 8.63).

Visite el punto 8.26.4 para procesar data termal de Sentinel-3.

8.19 Sentinel-5p

La misión Copernicus Sentinel-5 Precursor es la primera misión de Copernicus dedicada a monitorear nuestra atmósfera. Copernicus Sentinel-5P es el resultado de una estrecha colaboración entre la ESA, la Comisión Europea, la Oficina Espacial de los Países Bajos, la industria, los usuarios de datos y los científicos. La misión consiste en un satélite que lleva el instrumento TROPOspheric Monitoring Instrument (TROPOMI). El instrumento TROPOMI fue cofinanciado por la ESA y los Países Bajos.

El objetivo principal de la misión Copernicus Sentinel-5P es realizar mediciones atmosféricas con alta resolución espaciotem-

poral, para ser utilizadas para la calidad del aire, ozono y radiación UV, y monitoreo y pronóstico del clima.

El satélite se lanzó con éxito el 13 de octubre de 2017 desde el cosmódromo de Plesetsk en Rusia.

Sentinel-5P es un satélite de órbita terrestre baja compatible con lanzadores pequeños, incluidos VEGA y ROCKOT, siendo ROCKOT el vehículo de lanzamiento seleccionado. Sentinel-5P está diseñado para una vida útil de siete años con 80 kg de propulsor de hidracina.

La hora local del satélite de cruce de nodo ascendente es a las 13:30 hrs. y se ha elegido para facilitar la operación en formación con la nave espacial Suomi-NPP de la NASA. Así es posible la utilización conjunta de los datos de *máscara de nube de alta resolución* proporcionados por el instrumento VIIRS (Visible Infrared Imaging Radiometer Suite) a bordo del Suomi-NPP durante los procesamientos de rutina del producto TROPOMI de metano.

La misión Copernicus Sentinel-5 Precursor reduce las brechas en la disponibilidad de productos de datos atmosféricos globales entre SCIAMACHY / Envisat (que finalizó en abril de 2012), la misión OMI / AURA y las futuras misiones Copernicus Sentinel-4 y Sentinel-5.

8.20 Sensor TROPOMI

El instrumento TROPOMI es un espectrómetro de imágenes de visualización del nadir que se encuentra en el espacio y que cubre bandas de longitud de onda entre el *ultravioleta* y el *infrarrojo de onda corta*.

El instrumento, la única carga útil de la nave espacial Sentinel-5P, utiliza técnicas de detección remota pasiva para medir en la parte superior de la atmósfera (TOA), la radiación solar reflejada e irradiada desde la tierra.

El instrumento opera en una configuración de barrido (sin escaneo), con un ancho de franja de $\sim$ 2600 km en la superficie de la Tierra. El tamaño de píxel típico (cerca del nadir) será de $7 \times 3,5\ km$ para todas las bandas espectrales, con la excepción

de la banda UV1 ($7 \times 28\ km$) y las bandas SWIR ($7 \times 7\ km$). Aunque la resolución espacial (Nivel L2) es remuestrada hasta $5,5 \times 3,5\ km$. Los datos se proporcionan a una resolución temporal menor a un día y cobertura mundial.

La información se encuentra en formato *NetCDF4*.

8.21 Niveles de Procesamiento: L1, L2

La información capturada por el sensor se dispone en dos niveles de procesamiento, **NIVEL 1 (L1)** o **Data cruda**, que no se encuentra disponible para descarga y un **NIVEL 2 (L2)** o **Productos**, que son el resultado de procesamientos geofísicos para estimar una serie de variables o componentes químicos.

8.22 Productos L2

8.22.1 CO. Columna Total de Monóxido de Carbono

Producto en mol/m^2 y valores típicos $0 - 0,1$, en ciertos eventos (incendios forestales) pueden hacer que se excedan estos límites.

El monóxido de carbono (CO) es un importante gas atmosférico para nuestra comprensión de la química troposférica. En ciertas áreas urbanas, es un contaminante atmosférico importante.

Las principales fuentes de CO son la combustión de combustibles fósiles, la quema de biomasa y la oxidación atmosférica del metano y otros hidrocarburos.

Mientras que la combustión de combustibles fósiles es la principal fuente de CO en las latitudes medias del norte, la oxidación del isopreno y la quema de biomasa juegan un papel importante en los trópicos.

TROPOMI observa la abundancia global de CO aprovechando las mediciones de radiancia terrestre en cielo despejado y cielo nublado en el rango espectral de $2,3\ \mu m$ de la parte infrarroja de onda corta (SWIR) del espectro solar. Las observaciones de cielo despejado de TROPOMI proporcionan el total de columna de

CO a la capa límite de la troposfera. Para atmósferas nubladas, la sensibilidad de la columna cambia según la trayectoria de la luz.

8.22.2 CH_2O. Columna vertical troposférica de Formaldehído

Producto en mol/m^2) y valores típicos $0 - 0,001$, en ciertos eventos (incendios forestales) pueden hacer que se excedan estos límites.

Las observaciones satelitales a largo plazo del formaldehído troposférico CH_2O son esenciales para respaldar los estudios relacionados con la calidad del aire y la química y el clima desde la escala regional hasta la global. El formaldehído es un gas intermedio en casi todas las cadenas de oxidación de *compuestos orgánicos volátiles distintos del metano (COVNM)*, que eventualmente conduce a CO_2.

La principal fuente de CH_2O en la atmósfera remota es la oxidación de CH_4. En los continentes, la oxidación de COVNM más elevados emitidos por la vegetación, los incendios, el tráfico y las fuentes industriales da como resultado mejoras importantes y localizadas de los niveles de CH_2O. Las variaciones estacionales e interanuales de la distribución del formaldehído están relacionadas principalmente con los cambios de temperatura y los incendios, pero también con los cambios en las actividades antropogénicas.

Su vida útil es del orden de unas pocas horas, las concentraciones de CH_2O en la capa límite pueden estar directamente relacionadas con la liberación de hidrocarburos de vida corta, que en su mayoría no se pueden observar directamente desde el espacio.

8.22.3 NO_2. Columna troposférica de Dióxido de Nitrógeno

Producto en mol/m^2, con valores típicos entre $0 - 0,0003$; valores máximos de las ciudades contaminadas pueden llegar a dos o tres veces el valor superior.

El Dióxido de Nitrógeno (NO_2) y el Óxido de Nitrógeno (NO)

juntos generalmente se denominan *óxidos de nitrógeno*. Son importantes trazas de gases en la atmósfera terrestre, presentes tanto en la troposfera como en la estratosfera.

Entran en la atmósfera como resultado de actividades antropogénicas (en particular, combustión de combustibles fósiles y quema de biomasa) y procesos naturales (como procesos microbiológicos en suelos, incendios forestales y rayos). Durante el día, es decir, en presencia de luz solar, un ciclo fotoquímico que involucra ozono (O_3) convierte NO en NO_2 (y viceversa) en una escala de tiempo de minutos, por lo que el NO_2 es una medida robusta para las concentraciones de óxidos de nitrógeno.

8.22.4 O_3. Columna total de Ozono

Producto expresado en mol/m^2), sus valores típicos $0 - 0,36$.

El Ozono (O_3) es de crucial importancia para el equilibrio de la atmósfera terrestre. En la estratosfera, la capa de ozono protege a la biosfera de la peligrosa radiación solar ultravioleta.

En la troposfera, actúa como un agente limpiador eficaz, pero en concentraciones elevadas también se vuelve perjudicial para la salud de los seres humanos, los animales y la vegetación. El ozono también es un importante contribuyente de gases de efecto invernadero al cambio climático en curso.

Desde el descubrimiento del agujero de ozono en la Antártida en la década de 1980 y el subsiguiente Protocolo de Montreal que regula la producción de sustancias que agotan la capa de ozono que contienen cloro, el ozono se ha monitoreado de forma rutinaria desde el suelo y desde el espacio.

8.22.5 SO_2. Columna total de Dióxido de Azufre

Expresado en mol/m^2, con valores típicos $0 - 0,01$, las erupciones volcánicas explosivas pueden superar los $0,35$ mol/m^2 y el ruido instrumental puede producir valores negativos.

El Dióxido de Azufre (SO_2) ingresa a la atmósfera terrestre a través de procesos tanto naturales como antropogénicos. Desempeña un papel en la química a escala local y global y su impacto varía desde la contaminación a corto plazo hasta los efectos so-

bre el clima. Solo alrededor del 30 del SO_2 emitido proviene de fuentes naturales; la mayoría es de origen antropogénico.

Las emisiones de SO_2 afectan negativamente a la salud humana y la calidad del aire. El SO_2 tiene un efecto sobre el clima a través del *forzamiento radiativo*[4], a través de la formación de aerosoles de sulfato. Las Emisiones de SO_2 volcánico también pueden representar una amenaza para la aviación, junto con las cenizas volcánicas.

S5P/TROPOMI muestrea la superficie de la Tierra con un tiempo de revisión de un día con una resolución espacial de $3,5 \times 5,5$ km que permite la resolución de detalles finos, incluida la detección de plumas de SO_2 mucho más pequeñas.

[4] El forzamiento radiativo o forzamiento climático es la diferencia entre la insolación (luz solar) absorbida por la Tierra y la energía irradiada de vuelta al espacio. Si es positivo, significa que la Tierra recibe más energía de la luz solar que la que irradia al espacio.

8.22.6 CH_4. Proporción de mezcla aire seco promediada de la columna de Metano

Unidad en partes por mil millones, con valores típicos $1,600 - 2,000$.

El Metano (CH_4) es, después del dióxido de carbono (CO_2), el contribuyente más importante al efecto invernadero aumentado antropogénicamente. Aproximadamente las tres cuartas partes de las emisiones de metano son antropogénicas y, como tal, es importante continuar con el registro de las mediciones basadas en satélites.

TROPOMI tiene como objetivo proporcionar concentraciones de columna de CH_4 con alta sensibilidad a la superficie de la Tierra, buena cobertura espaciotemporal y suficiente precisión para facilitar el modelado inverso de fuentes y sumideros.

TROPOMI utiliza información de absorción de la banda de oxígeno A (760 nm) y el rango espectral SWIR para monitorear la abundancia de CH_4 en la atmósfera de la Tierra.

Los datos del instrumento satelital GOSAT y los instrumentos TCCON basados en tierra se utilizan para la verificación y validación.

Actualmente, el metano es un producto de datos solo terrestre disponible en el flujo de datos de offline (OFFL) (ver 8.23).

8.22.7 UVAI. Índice de Aerosoles

calculado en base a longitudes de onda de 340 nm y 380 nm.

El índice de aerosoles (IA) es un producto de datos bien establecido que se ha calculado para varios instrumentos satelitales diferentes que abarcan un período de más de 40 años.

El índice de aerosol S5P/TROPOMI se conoce como *índice de aerosol ultravioleta (UVAI)*. El cálculo relativamente simple del índice de aerosol se basa en cambios dependientes de la longitud de onda en la dispersión de Rayleigh en el rango espectral UV donde la absorción de ozono es muy pequeña.

La UVAI también se puede calcular en presencia de nubes para que sea posible una *cobertura global diaria*.

La UVAI es ideal para rastrear la evolución de las columnas de aerosoles episódicos a partir de los brotes de polvo, las cenizas volcánicas y la quema de biomasa.

8.23 Acceso a los Datos

Los productos de datos del precursor Sentinel-5 están disponibles de forma sistemática y continua durante todo el período operativo vida útil (7 años previstos). Todos los datos serán gratuitos a los usuarios finales, incluido el público en general, científicos y entidades comerciales.

Los productos Sentinel-5P Nivel-2 están disponibles en:

- **NRT (Near Real Time)**, proporcionado al usuario dentro de las tres horas después de la detección (un día para el ozono troposférico total).
- **OFFL (Fuera de línea)** proporcionado al usuario dentro de las dos semanas posteriores a la detección.

Los productos Nivel 2 descritos en 8.22 tienen su correspondiente nombre de producto digital para descarga desde el sitio Copernicus y se resumen en el cuadro 8.11.

Cuadro 8.11: Nombre de los Productos TROPOMI Nivel 2.

Producto	Parámetro
L2___O3_____	Ozono (O3) total
L2___O3_TCL	Ozono (O3) tropo
L2___O3___PR	Ozono (O3) perfil
L2___NO2____	Dióxido Nitrógeno (NO2) tot/tropo
L2___SO2____	Dióxido Azufre (SO2) total
L2___CO_____	Monóxido Carbono (CO) total
L2___CH4____	Metano (CH4) total
L2___HCHO___	Formaldehído (HCHO) total
L2___AER_AI	Índice Aerosol UV
L2___AER_LH	Altura capa Aerosol (Presión nivel-med)

Nota:
tot=total, tropo= troposférico

8.24 Descarga de Imágenes Sentinel-5p

El sitio web que centraliza el acceso al *Copernicus Open Access Hub* es: scihub.copernicus.eu.

Pasos principales para la descarga de imágenes:

1. El sitio se compone de una serie de enlaces a la data de las misiones Sentinel, novedades y estadísticas globales interesantes para el usuario. El acceso específico al panel de configuración de la búsqueda de productos Sentinel-5p es mediante el sitio **Open Hub**, (Figura 8.64, área destacada en rojo).

2. Open Hub ofrece un mapa web para definir el área geográfica mediante el dibujo de un polígono y una serie de diálogos para especificar y refinar la búsqueda de productos (Figura 8.65).

3. Antes de realizar cualquier operación debes logearte en modo de invitado para acceder a los servicios (Figura 8.66).

4. El mapa web posee dos modalidades, *navegación* y *trazado*. En la primera debes realizar los paneos y zooms necesarios para buscar el área de estudio y en la segunda, dibujar el polígono que define dicha área, (Figura 8.67).

Figura 8.64: Vista inicial del sitio scihub.copernicus.eu. El link destacado en rojo da acceso a la data disponible de las misiones Sentinel-5p.

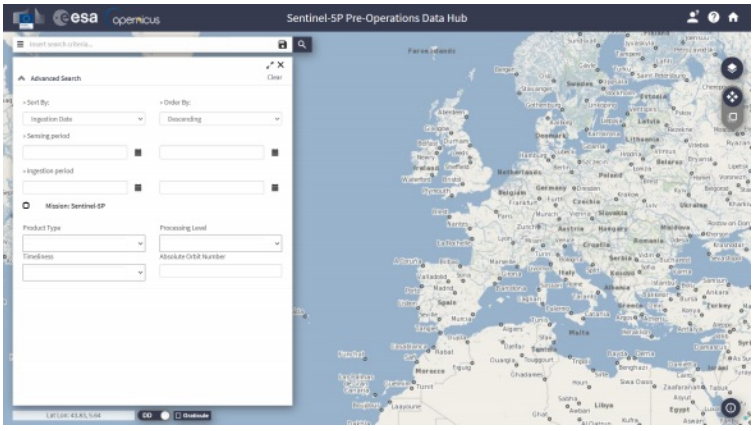


Figura 8.65: Panel de configuración de búsquedas en sitio 'scihub.copernicus.eu/dhus'

5. El trazado de polígono en el mapa web se puede realizar en dos modos:

- **Área Rectangular**: Luego de realizar el primer clic y *sin levantar* el botón del ratón arrastrar a la esquina opuesta se define el área de estudio rectangular.

- **Poligonal**: Una serie de clic para crear un polígono irregular para definir el área, hasta realizar un *doble clic* para finalizar el dibujo.

Figura 8.66: Cuadro de dialogo para logear con los datos de invitado.

El resultado se usa para restringir los productos Sentinel que cubren o al menos interseptan el área (Figura 8.68).

6. Los parámetros avanzados se agrupan en el cuadro de dialogo de configuración (Figura 8.69) con controles para configurar los parámetros de:

Figura 8.67: Modos del mapa web. (Izquierda) Modo navegación para buscar y ajustar vista y (Derecha) Dibujar el polígono que define el área de estudio.

- **Propiedad de Ordenamiento**: Por fecha de procesamiento, captura o id de la escena.
- **Método de Ordenamiento**: Ascendente o descendente.
- **Fecha de Inicio y Término de Sensado**.
- **Fecha de Inicio y Término de Proceso**.

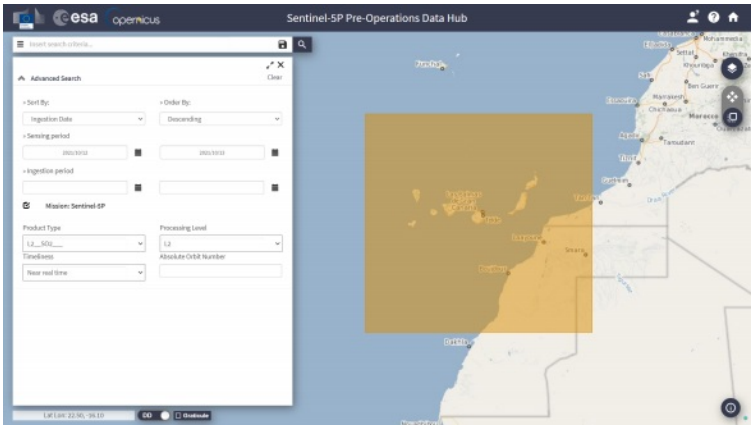


Figura 8.68: Vista general de aplicación web para descarga de datos Sentinel-5p.

Las siguientes opciones del cuadro de dialogo agrupan los parámetros disponibles.

7. Activar el check para descarga de misión Sentinel-5p.

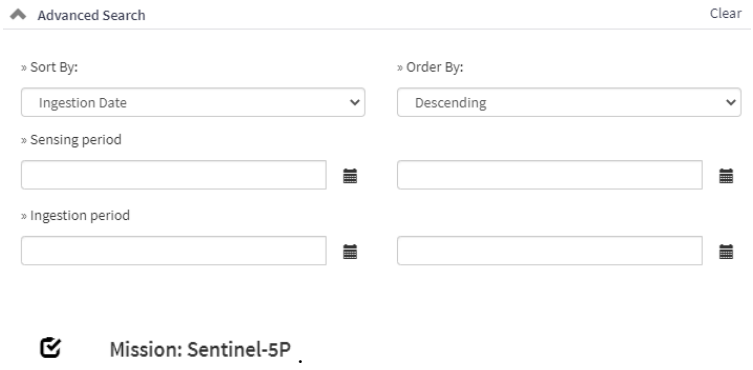


Figura 8.69: Cuadro de dialogo con opciones avanzadas de búsqueda.

- **Tipo de Producto**: Solo productos nivel 2, disponibles (detalles en 8.13.2, figura 8.70. Si la opción se deja en blanco buscará en todos los niveles o tipos de procesamiento.

- **Nivel de Procesamiento**: Nivel de Procesamiento de las escenas (detalles en 8.13.2, figura 8.71. Si la opción se deja en blanco buscará en todos los niveles o tipos de procesamiento.

  A pesar de ofrecer las alternativas en la lista de productos y en el nivel de procesamiento, todos los productos Nivel 1 (L1B) no se encuentran disponibles al público.

- **Timeliness**: Temporalidad de los productos (8.23), (Figura 8.72).

Definidos los parámetros se pueden almacenar las opciones para futuras búsquedas ( ).

Por cada imagen que cumpla los criterios se presenta una vista de detalle (Figura 8.73), que forman una lista.

Cada elemento presenta tres opciones mediante los siguientes íconos:

- Centrar la vista en la imagen.
- Ver los detalles de la imagen.
- Descarga directa de la imagen.

Figura 8.70: Lista de Productos Nivel 2 Disponibles.

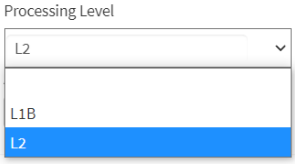


Figura 8.71: Nivel de Procesamiento de los Productos Disponibles.

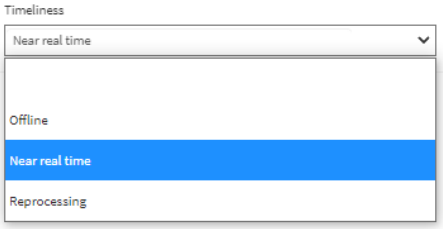


Figura 8.72: Categoría de Temporalidad Disponibles.

Figura 8.73: Vista de detalle de la imagen seleccionada.

8.25 Procesando en R

8.25.1 Gestión del Contenido NetCDF

Los archivos descargados han sido descargados en la carpeta que vamos a describir usando la función *here* (siempre debe considerar que ud. tendrá que dirigir a su propia carpeta), y poseen la extensión **nc**:

```
nc <- dir(path = here("raw", "sen", "sen5p"),
          pattern =".nc$")
```

```
[1] "S5P_NRTI_L2__CO_____20211012T135836_20211012T140336_20716_02_020200_20211012T143327.nc"
[2] "S5P_NRTI_L2__HCHO___20211012T135836_20211012T140336_20716_02_020201_20211012T144121.nc"
[3] "S5P_NRTI_L2__NO2____20211012T135836_20211012T140336_20716_02_020200_20211012T144122.nc"
[4] "S5P_NRTI_L2__O3_____20211012T135836_20211012T140336_20716_02_020201_20211012T144120.nc"
[5] "S5P_NRTI_L2__SO2____20211013T133836_20211013T134336_20730_02_020201_20211013T142447.nc"
[6] "S5P_OFFL_L2__CH4____20211011T131933_20211011T150102_20702_02_020200_20211013T051955.nc"
```

Vamos a comenzar a trabajar con el archivo de índice 1, que contiene en su nombre el texto ***CO*** que corresponde al producto Monóxido de Carbono Total (Cuadro 8.11).

Para explorar los contenidos cargamos el paquete a la sesión actual con función *library(ncdf4)* el paquete *netcdf* (Pierce [2019]) que utilizamos previamente en 6.11.

```
nc_CO <- nc_open(here("raw","sen", "sen5p", nc[1]))
```

Para revisar los atributos o propiedades principales:

```
attributes(nc_CO)
```

```
$names
 [1] "filename"    "writable"    "id"          "safemode"
 [5] "format"      "is_GMT"      "groups"      "fqgn2Rindex"
 [9] "ndims"       "natts"       "dim"         "unlimdimid"
[13] "nvars"       "var"

$class
[1] "ncdf4"
```

```
print(paste("El archivo tiene ", nc_CO$nvars, "variables."))
```

```
[1] "El archivo tiene  56 variables."
```

Y podemos revisar el contenido detallado de las variables nuevamente con *attributes()* (solo se muestran las 10 primeras variables con función *head()*).

```
head(attributes(nc_CO$var)$names,10)
```

```
 [1] "PRODUCT/delta_time"
 [2] "PRODUCT/time_utc"
 [3] "PRODUCT/qa_value"
 [4] "PRODUCT/latitude"
 [5] "PRODUCT/longitude"
 [6] "PRODUCT/carbonmonoxide_total_column"
 [7] "PRODUCT/carbonmonoxide_total_column_precision"
 [8] "PRODUCT/carbonmonoxide_total_column_corrected"
 [9] "GEOLOCATIONS/satellite_latitude"
[10] "GEOLOCATIONS/satellite_longitude"
```

Y podemos ver los contenidos de cada variable usando su índice, por ejemplo, los detalles de la variable que define el producto físico contenido en el archivo [6]:

"PRODUCT/carbonmonoxide_total_column"

Donde [6] es el índice o posición de la data que nos permitirá luego construir el mapa correspondiente.

O la unidad en que se encuentran los datos en el índice [1]:

"mol m-2"

Para revisar el contenido, usamos la función `ncatt_get(nc, varid)` que lee los atributos desde el archivo `nc` de la variable definida por el índice `varid`.

```
ncatt_get(nc= nc_CO,
          varid= attributes(nc_CO$var)$names[6])
```

Y también exploramos la calidad de los datos (variable en índice 3):

```
ncatt_get(nc= nc_CO,
          varid= attributes(nc_CO$var)$names[3])
```

Y al finalizar su uso, se debe cerrar la conexión al objeto.

```
nc_close(nc_CO)
```

Que entrega importante información adicional de cada capa o variable.

8.25.2 Data Espacial Sentinel-5p

El proceso de datos provenientes de la misión Sentinel-5p desde un aspecto espacial usaremos el paquete *S5Processor* (Philipp [2021a]) que convierte los archivos en NetCDF a objetos ráster.

El paquete no se encuentra disponible aún en CRAN, por lo tanto, se instala desde la página Github usando la función correspondiente:

```
devtools::install_github("MBalthasar/S5Processor")
```

Y para cargar en la sesión actual utilizamos la función *library*:

```
library(S5Processor)
```

La función `S5P_process(input, product, my_res, my_aoi, extent_only, interpol_method, apply_scale_factor)` convierte automáticamente los archivos de datos de Sentinel-5P Nivel 2 en archivos TIFF. El usuario debe definir el nombre y path del archivo NetCDF en `input`, indicar el índice dónde se encuentran los datos almacenados en la estructura del archivo (en nuestro caso es el índice 6) con `product`, definimos la resolución espacial del producto a generar con `my_res`, opcionalmente podemos definir el área de interés (un objeto espacial) con `my_aoi` y definir como verdadero el parámetro `extent_only` si deseo procesar solo el área definida. Por último, definir el método de interpolación de los nuevos valores ("ngb" para vecino más cercano o "bilinear" para interpolación bilineal, valor por defecto) en `interpol_method`.

Es importante destacar que los datos disponibles en su mayoría se encuentran en la unidad moléculas por metro cuadrado (8.2).

$$U = \frac{mol}{m^2} \tag{8.2}$$

El parámetro `apply_scale_factor= T` permite aplicar el factor de conversión incluido en el mismo archivo para convertir a moléculas por cm^2:

Figura 8.74: Retrato de Gordon Miller Bourne Dobson, Walter Stoneman. Galería Nacional de Retratos, Londres.

$$\frac{mol}{cm^2} = U \times 6.022141 \times 10^{19} \tag{8.3}$$

Otros factores de conversión a aplicar directamente al ráster sin aplicar la transformación anterior, encontramos:

- Para convertir a DU:

$$DU = U \times 2241.15 \tag{8.4}$$

La **unidad Dobson (DU)** es una unidad de medida de la cantidad de gas traza en una columna vertical a través de la atmósfera terrestre. Se originó y sigue utilizándose principalmente con respecto al ozono atmosférico, cuya cantidad de columna total, normalmente denominada "ozono total" y, a veces, "abundancia de columna", está dominada por las altas concentraciones de ozono en la capa de ozono estratosférico. La unidad Dobson se define como el espesor (en unidades de 10 μm) de esa capa de gas puro que estaría formada por la cantidad total de la columna en condiciones estándar de temperatura y presión (STP). Esto a veces se denomina *"mili-atmo-centímetro"*. Por lo tanto, una cantidad típica de columna de 300 DU de ozono atmosférico formaría una capa de 3 mm de gas puro en la superficie de la Tierra si su temperatura y presión se ajustaran a STP. La unidad Dobson lleva el nombre de Gordon Dobson (Figura 8.74), investigador de la Universidad de Oxford que en la década de 1920 construyó el primer instrumento para medir el ozono total desde el suelo (Wikipedia).

- Para convertir a radiancia en fotones por segundo por nanómetro por centímetro cuadrado por estereorradián:

$$\gamma_{s \times nm \times cm^2 \times sr} = U \times 6.02214 \times 10^{19} \tag{8.5}$$

8.25.3 Mapa de Distribución de Monóxido de Carbono

Imagen del 12 octubre, resolución espacial 20km. De acuerdo a los pasos descritos en 5.2.4, agregamos los límites en alta

resolución del área de estudio.

```
S5P_CO <- S5P_process(input = here("raw","sen", "sen5p", nc[1]),
                      product = 6,
                      my_res = 20000,
                      extent_only = T,
                      apply_scale_factor =F)

limites <- rnaturalearth::ne_countries(scale = "large")
area_interes <- crop(limites, extend(x = raster::extent(S5P_CO),
                                     y = 20))
```

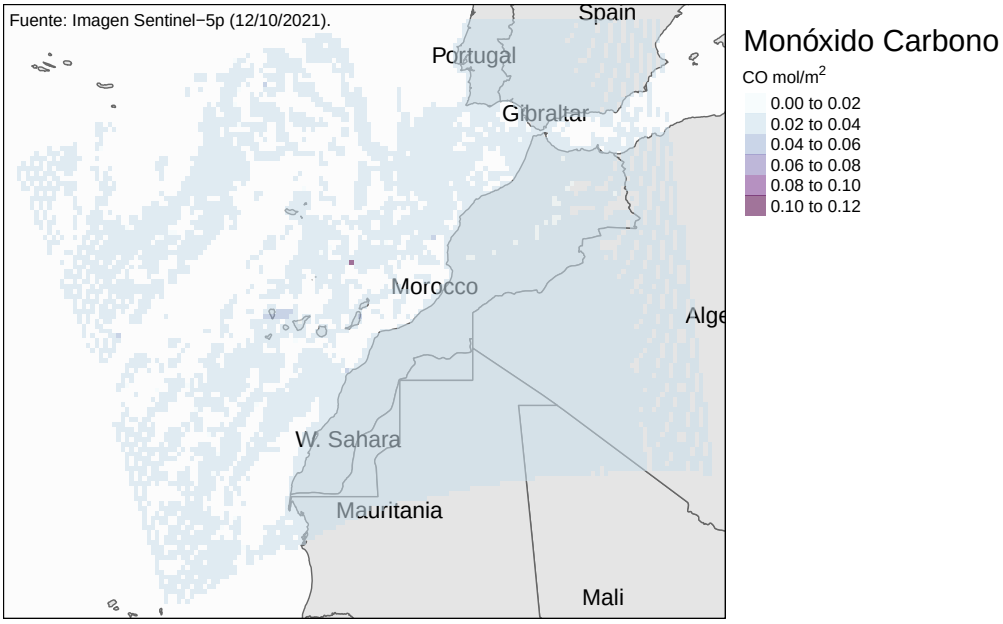


Figura 8.75: Distribución de Monóxido Carbono, 12 octubre 2021.

8.25.4 Mapa de Distribución de Formalheído

```
S5P_HCHO <- S5P_process(input = here("raw","sen", "sen5p", nc[2]),
                        product = 6,
                        my_res = 20000,
                        extent_only = T,
                        apply_scale_factor =F)
```

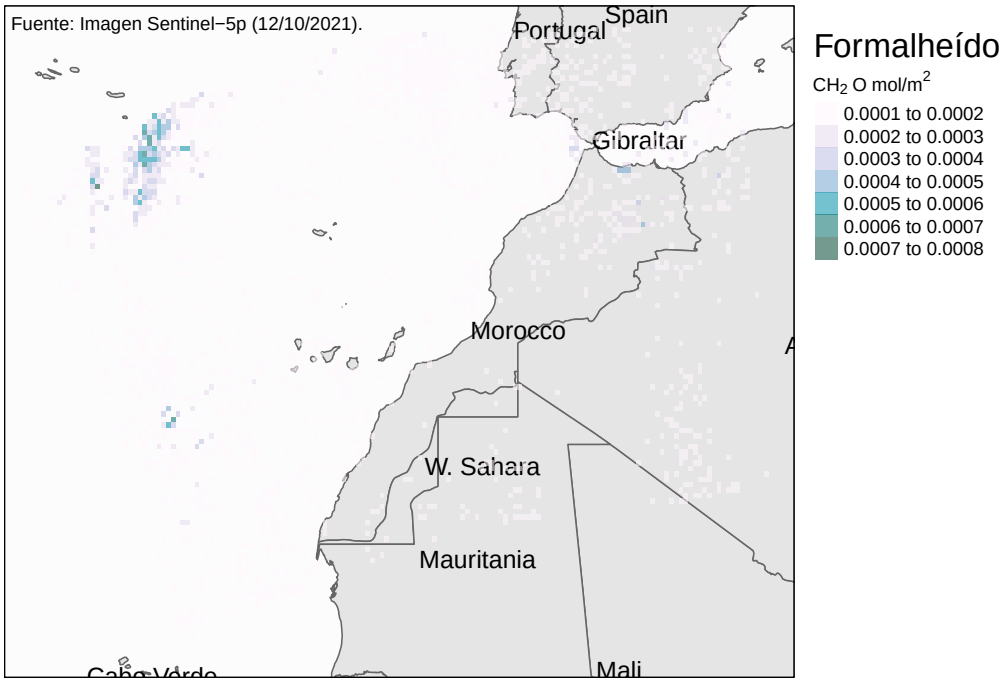


Figura 8.76: Distribución de Formalheído, 12 octubre 2021.

8.25.5 Mapa de Distribución de Monóxido de Dióxido Nitrógeno

```
S5P_NO2 <- S5P_process(input = here("raw","sen", "sen5p", nc[3]),
                       product = 6,
                       my_res = 20000,
                       extent_only = T,
                       apply_scale_factor =F)
```

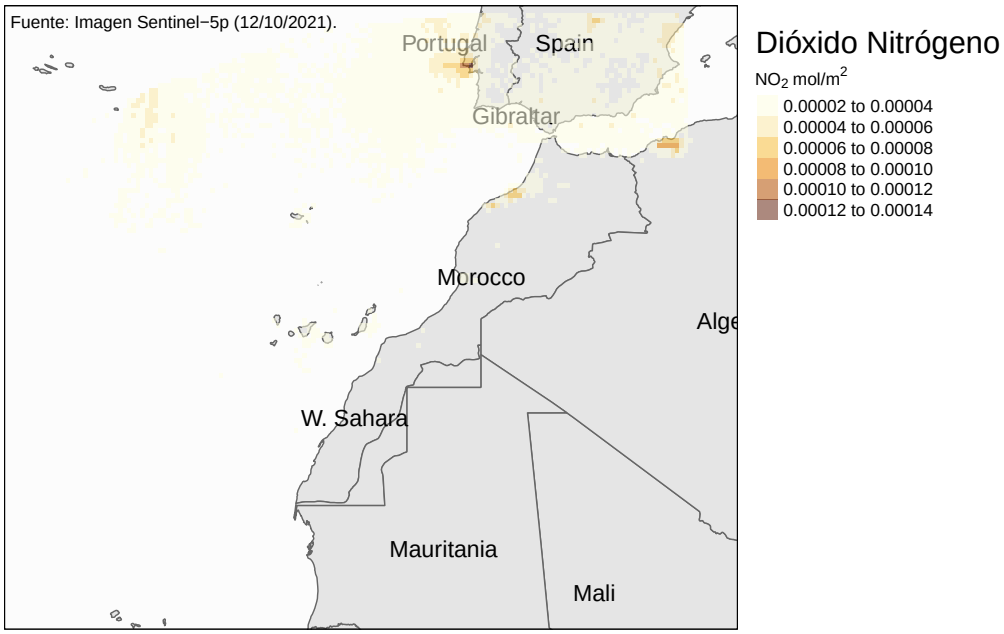


Figura 8.77: Distribución de Dióxido Nitrógeno, 12 octubre 2021.

8.25.6 Mapa de Distribución de Ozono

```
S5P_03 <- S5P_process(input = here("raw","sen", "sen5p", nc[4]),
                      product = 6,
                      my_res = 20000,
                      extent_only = T,
                      apply_scale_factor =F)
```

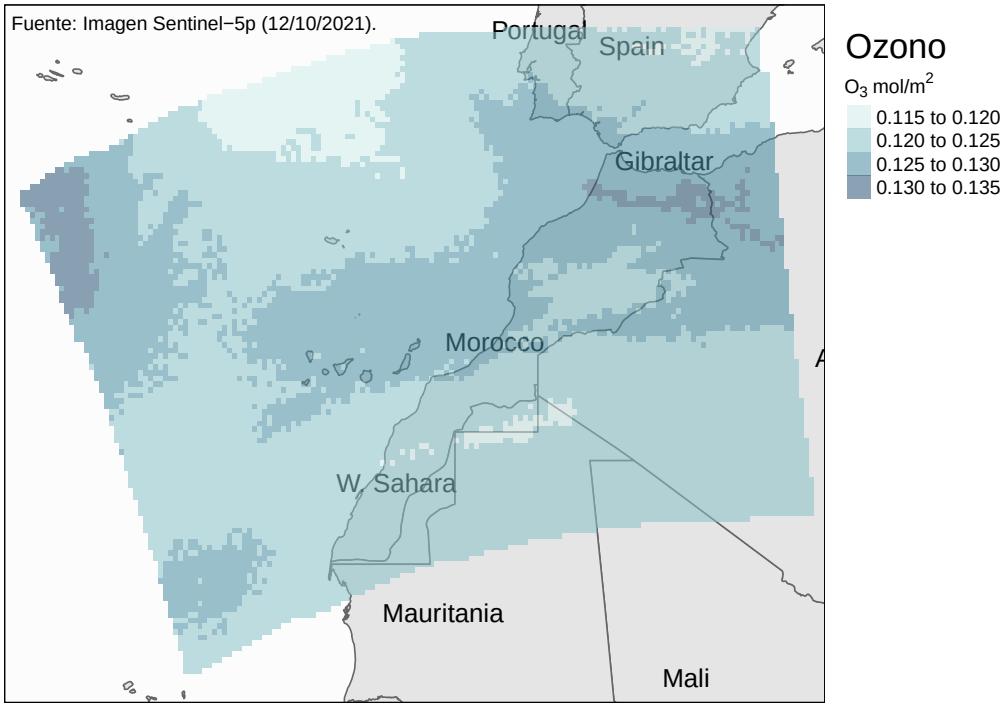


Figura 8.78: Distribución de Ozono, 12 octubre 2021.

8.25.7 Mapa de Distribución de Dióxido Azufre

```
S5P_SO2 <- S5P_process(input = here("raw","sen", "sen5p", nc[5]),
                       product = 6,
                       my_res = 20000,
                       extent_only = T,
                       apply_scale_factor =F)
```

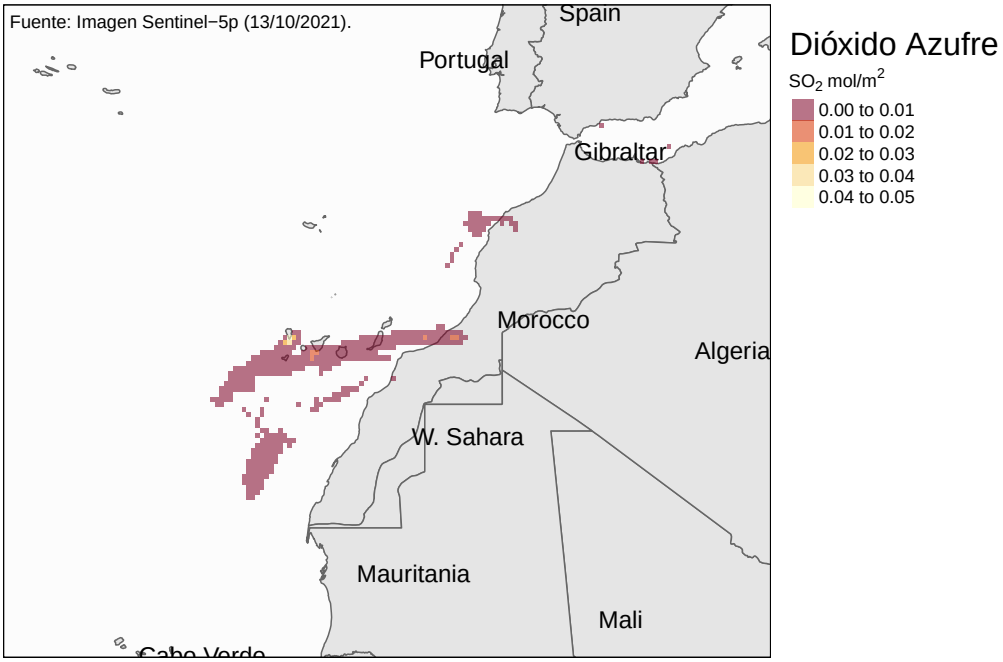


Figura 8.79: Distribución de Dióxido Azufre, 13 octubre 2021.

8.25.8 Mapa de Distribución de Monóxido de Metano

```
S5P_CH4 <- S5P_process(input = here("raw","sen", "sen5p", nc[6]),
                       product = 6,
                       my_res = 20000,
                       extent_only = T,
                       apply_scale_factor =F)
```

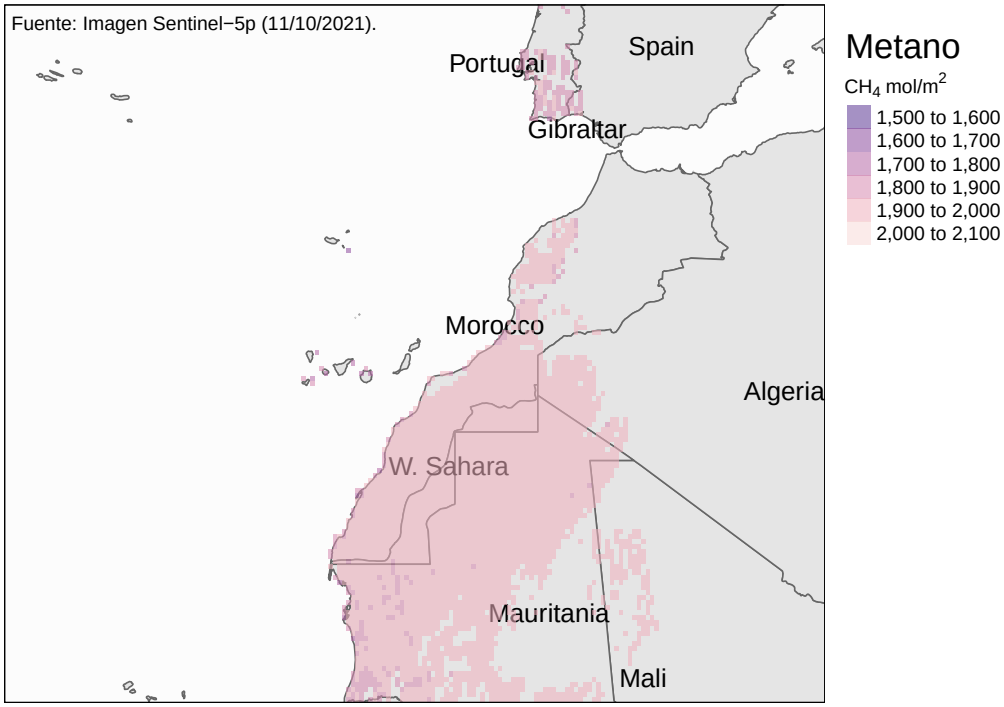


Figura 8.80: Distribución de Metano, 11 octubre 2021.

Teledetección Termal

8.26 El Descubrimiento del Infrarrojo

En 1800, Sir Williams Herschel (Figura 8.81) hizo un descubrimiento muy importante. Él estaba interesado en aprender cuánto calor atravesaba los filtros de diferentes colores que utilizaba para observar el Sol, ya que notó que filtros de diferentes colores parecían pasar distintos niveles de calor. Herschel pensó que los colores por sí mismos podrían contener diferentes niveles de calor.

Figura 8.81: William Herschel, astrónomo germano-británico. Pintura de Lemuel Francis Abbott, Galería Nacional de Retratos, Londres.

Herschel dirigió la luz del sol a través de un prisma de cristal para crear un espectro (el "arco iris" creado cuando la luz es dividida en sus colores) y midió la temperatura de cada color.

Utilizó tres termómetros con los bulbos teñidos de negro (para absorber todo el calor posible) y puso un bulbo en un color mientras los otros dos se pusieron más allá del espectro como muestras de control. Cuando midió las temperaturas de la luz de colores violeta, azul, verde, amarillo, naranja y roja, notó que todos los colores tenían la temperatura superior que las muestras de control y que la temperatura de los colores incrementaba desde el violeta a la parte roja del espectro (Figura 8.82).

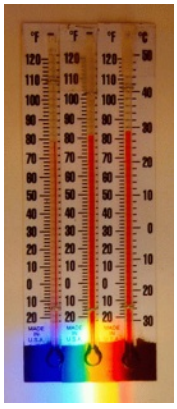


Figura 8.82: Termómetros midiendo temperaturas de diferentes colores según experimento realizado por Herschel. i-ciencias 9/ene/2016.

Después de notar este patrón, Herschel decidió medir la temperatura más allá de la porción roja del espectro en una región aparentemente desprovista de luz del sol. Para su sorpresa, encontró que esta región tenía la temperatura más alta de todas recibía lo que el llamó a los que él "rayos caloríficos". Descubrió que podían ser reflejados, refractados, absorbidos y transmitidos igual que la luz visible.

Estos "rayos caloríficos" fueron posteriormente denominados

rayos infrarrojos ("por debajo" del rojo) o radiación infrarroja y su descubrimiento permitió el desarrollo de varia (y hoy muy comunes) aplicaciones tecnológicas, como los LED infrarrojos de los controles remotos, la termografía y, por supuesto, los detectores infrarrojos para observar el cosmos.

Los cuerpos que se encuentran a baja temperatura no emiten en la región visible del espectro, sino en longitudes más largas, de menor energía. Por ejemplo, nuestro cuerpo y el de los animales emiten una radiación infrarroja (Ver 7.4.4) que no la detectamos con el ojo, pero que podemos percibir como el calor que emite el organismo. La tierra también emite ese calor y se ha desarrollado toda una tecnología para realizar su medida.

8.26.1 Temperatura de la Superficie Terrestre (LST)

La *temperatura de la superficie terrestre (LST)* es un parámetro clave en la física terrestre, impulsa los flujos de calor radiativo, latente y sensible en la interfaz superficie-atmósfera.

Es un parámetro altamente dinámico que cambia rápidamente a lo largo del tiempo debido a variaciones en la superficie de la Tierra, el clima y también variaciones en la radiación solar entrante.

El conocimiento preciso de la LST puede proporcionar información sobre el estado de equilibrio de la superficie, que es de suma importancia para una serie de campos, incluidos la hidrología, la climatología y geofísica. Los datos LST también se pueden utilizar para el estudio de la urbanización y su impacto en la superficie térmica urbana, a saber, las *islas de calor urbano superficial (SUHI)*.

En la actualidad la única forma de obtener una detallada, global, precisa ($1a2°C$) y frecuente medida de LST es mediante el uso de las bandas termales infrarrojas (TIR, ver 7.4.4) de las diversas misiones espaciales.

En condiciones de cielo despejado, la medición mediante las bandas TIR recogen al menos tres componentes principales. Estos componentes se originan tanto en la atmósfera como en la superficie:

- Radiación emitida desde la superficie.
- Radiación TIR atmosférica reflejada por la superficie en el campo de visión del sensor.
- Radiación TIR emitida en el trayecto mismo.

Para estimar LST a partir del TIR requiere desacoplar la radiación emitida de la superficie de los dos componentes atmosféricos y ajustar por la propia atenuación atmosférica e incluso la misma **emisividad de la superficie terrestre** (LSE).

El valor LSE es previamente desconocido, así el cálculo LST es matemáticamente no resoluble.

Entonces, o se debe conocer previo a la estimación o debe ser estimada simultáneamente a partir de modelos, restricciones y suposiciones.

En la actualidad las distintas misiones de EO proveen a los usuarios productos ya corregidos y modelados para dar una alta precisión de los valores estimados ($< 2°C$).

8.26.2 MODIS

Además de las imágenes espectrales analizadas previamente (8.1.1), la misión espacial MODIS, ofrece el producto **MOD11A1** V6 que proporciona valores *diarios de temperatura de la superficie terrestre (LST)* y *emisividad* en una cuadrícula de 1200×1200 *km* de *cobertura global*. El valor de temperatura se deriva del producto *MOD11_L2*.

Por encima de los 30 grados de latitud, algunos píxeles pueden tener múltiples observaciones cuando se cumplen los criterios de cielo despejado. Cuando esto ocurre, el valor de píxel es el promedio de todas las observaciones que califiquen.

Junto con las bandas de temperatura superficial tanto *diurna* como *nocturna* y sus capas indicadoras de calidad, se encuentran las bandas 31 ($10.780 - 11.280\ \mu m$) y 32 ($11.770 - 12.270\ \mu m$), seis capas de propiedades y condiciones de la observación.

El rango de datos disponibles se extiende desde 24 febrero de 2000 hasta la actualidad.

Siguiendo las instrucciones descritas en 8.1.1 pero usando ahora los productos MOD11A1 descargamos una escena de ejemplo y

vamos a procesar para crear nuestro ráster de información termal.

Buscar en la carpeta de descarga los archivos de extensión *.hdf*.

```
termico <- list.files(path = here("raw","termal"),
                      pattern = "*.hdf",
                      full.names = T)
```

Usando el paquete MODIS (8.4) extraemos un resumen de la data contenida usando la función *getSds()*.

```
ter <- MODIS::getSds(HdfName = termico)
```

Así, podemos explorar los nombres de los sets de datos, revisamos el contenido del atributo:

```
ter$SDSnames
```

```
 [1] "LST_Day_1km"      "QC_Day"           "Day_view_time"
 [4] "Day_view_angl"    "LST_Night_1km"    "QC_Night"
 [7] "Night_view_time" "Night_view_angl" "Emis_31"
[10] "Emis_32"          "Clear_day_cov"    "Clear_night_cov"
```

Y cada set corresponde a una ruta dentro de la estructura del archivo:

```
basename(ter$SDS4gdal)
```

```
 [1] "MOD11A1.A2021288.h16v06.006.2021289092620.hdf\":MODIS_Grid_Daily_1km_LST:LST_Day_1km"
 [2] "MOD11A1.A2021288.h16v06.006.2021289092620.hdf\":MODIS_Grid_Daily_1km_LST:QC_Day"
 [3] "MOD11A1.A2021288.h16v06.006.2021289092620.hdf\":MODIS_Grid_Daily_1km_LST:Day_view_time"
 [4] "MOD11A1.A2021288.h16v06.006.2021289092620.hdf\":MODIS_Grid_Daily_1km_LST:Day_view_angl"
 [5] "MOD11A1.A2021288.h16v06.006.2021289092620.hdf\":MODIS_Grid_Daily_1km_LST:LST_Night_1km"
 [6] "MOD11A1.A2021288.h16v06.006.2021289092620.hdf\":MODIS_Grid_Daily_1km_LST:QC_Night"
 [7] "MOD11A1.A2021288.h16v06.006.2021289092620.hdf\":MODIS_Grid_Daily_1km_LST:Night_view_time"
 [8] "MOD11A1.A2021288.h16v06.006.2021289092620.hdf\":MODIS_Grid_Daily_1km_LST:Night_view_angl"
 [9] "MOD11A1.A2021288.h16v06.006.2021289092620.hdf\":MODIS_Grid_Daily_1km_LST:Emis_31"
[10] "MOD11A1.A2021288.h16v06.006.2021289092620.hdf\":MODIS_Grid_Daily_1km_LST:Emis_32"
[11] "MOD11A1.A2021288.h16v06.006.2021289092620.hdf\":MODIS_Grid_Daily_1km_LST:Clear_day_cov"
[12] "MOD11A1.A2021288.h16v06.006.2021289092620.hdf\":MODIS_Grid_Daily_1km_LST:Clear_night_cov"
```

En los índices 1 y 5 encontramos los datos térmicos diurnos y nocturnos (Figura 8.83),

```
termal_diurno <- raster(x = readGDAL(fname = ter$SDS4gdal[1],
                                     silent = T))
termal_nocturno <- raster(x = readGDAL(fname = ter$SDS4gdal[5],
                                       silent = T))
```

Preparamos algunos datos adicionales para mejorar los productos cartográficos:

```
ciudades <- geonames::GNcities(north = 29,east = -12.1,
                               south = 27,west = -18,
                               lan="es", maxRows = 50)
etiquetas <- ciudades[ciudades$fcode=="PPLA3" , c(1, 9, 3, 4, 12)]
```

Como se describe en 6.12.1, el paquete describe límites políticos de resolución baja y media. El paquete rworldxtra agrega a las bases de datos estándar capas de alta resolución (South [2012]). Esta data se debe instalar como un paquete más `install.packages('rworldxtra')`.

```
latlon <- crs("+proj=longlat +datum=WGS84 +no_defs")
#Definir área de corte
ext <- extent( -18.3, -13.22, 27.605, 29.6)
#Descarga de capa ce vectores
wm <- getMap(resolution = "high")
#Recorte de la capa por la extensión
wm <- crop(wm, extent(ext))

#De los polígonos resultantes se extraen los centroides
xy <- st_centroid(st_as_sf(wm))
```

Recorte de los MDR por ext.

```
diurno <- projectRaster(from = termal_diurno,
                        crs=latlon)
diurno <- crop(x = diurno,
               y = ext)
nocturno <- projectRaster(from = termal_nocturno,
                          crs=latlon)
nocturno <- crop(x = nocturno,
                 y = ext)
```

El código para genera el mapa con las distintas capas

```
#Configuramos dos mapas verticales
par(mfrow=c(2,1))
#Mapa base MDR diurno
plot(diurno, ext=ext, asp=1)
#world map recortado al área
plot(wm, add=T)
```

```
#agregar marcas y textos de localidades
points(x=as.numeric(etiquetas$lng),
       y=as.numeric(etiquetas$lat),
       pch=17, col="red", cex=0.75)
text(x=as.numeric(etiquetas$lng),
     y=as.numeric(etiquetas$lat),
     labels=etiquetas$name,
     cex=0.75,font=3,pos=4)
#segundo plot (nótese no incluye 'add=T')
plot(nocturno,ext=ext,asp=1)
plot(wm, add=T)
points(x=as.numeric(etiquetas$lng),
       y=as.numeric(etiquetas$lat),
       pch=17, col="red", cex=.75)
text(x=as.numeric(etiquetas$lng),
     y=as.numeric(etiquetas$lat),
     labels=etiquetas$name,cex=0.75,
     font=3,pos=4)
```

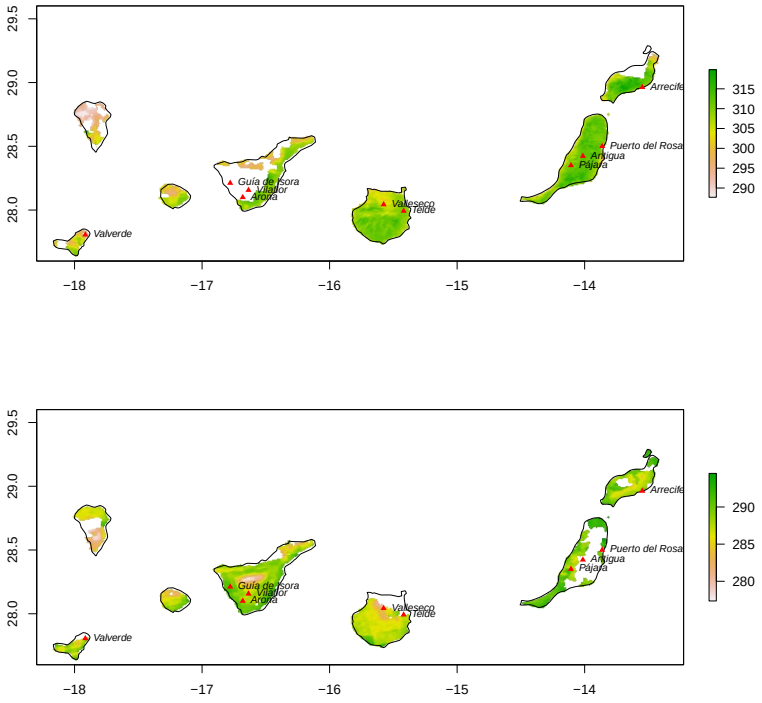


Figura 8.83: Detalle Islas Canarias, Mapa Temperaturas (en K). Escena MODIS. 28 septiembre 2021. Temperaturas Diurnas (Superior) y Nocturnas (Inferior).

8.26.3 Landsat 8

Descargamos la imagen Landsat-8 en nuestra carpeta de datos y **extraemos los archivos contenidos en el *tarfile***. Luego, creamos el vector con todas las imágenes de formato *.TIF*:

```
termico_land <- list.files(path = here("raw",
                                       "termal",
                                       "LC08_L2SP_207040_20211005_20211013_02_T1"),
                           pattern = "*.TIF",
                           full.names = T)
```

Y tal como se describe en 8.7.1 Landsat 8 posee el sensor TIRS que captura información en el rango térmico del espectro en la banda 10 y siguiendo los pasos en 8.12.5 procesamos en R:

```
termal_land8 <- raster(termico_land[12])
termal_land8 <- projectRaster(from = termal_land8,
                              crs=latlon)
```

Aplicamos los factores de conversión para llevar a grados kelvin (ver 8.2) (Figura 8.84):

```
termal_land8 <- (termal_land8 * 0.00341802) + 149

plot(termal_lan8, ext= ext,asp=1)
plot(wm, add=T)
points(x=as.numeric(etiquetas$lng),
       y=as.numeric(etiquetas$lat),
       pch=17,
       col="red",
       cex=.75)
text(x=as.numeric(etiquetas$lng),
     y=as.numeric(etiquetas$lat),
     labels=etiquetas$name,cex=0.75,
     font=3,pos=4)
```

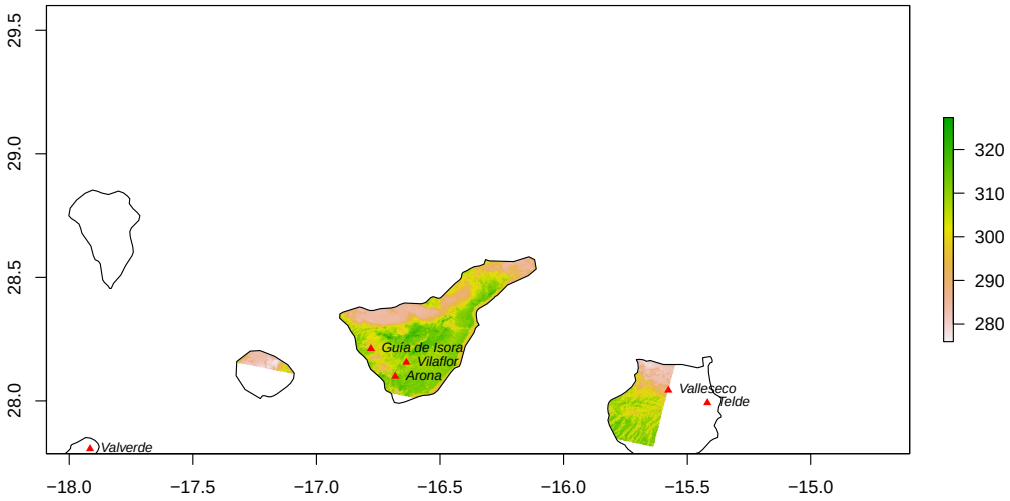


Figura 8.84: Detalle Islas Canarias, Mapa Temperaturas (en K). Escena Landsat 8. 05 octubre 2021. Temperaturas Diurnas.

8.26.4 Sentinel-3

Luego de extraer los archivos contenidos en el *zipfile* (¡y probablemente renombrar las carpetas para acortar las rutas!), encontramos los archivos descritos en 8.17.3, que existen como una serie de archivos *.nc*:

```
termico_sen <- list.files(path = here("raw",
                                      "termal",
                                      "20211018T110932",
```

```
                                "S3B_SL_2_LST____20211018T110932.SEN3"),
                    pattern = "*.nc",
                    full.names = T)
```

No olvidar cargar el paquete diseñado para el trabajo con este tipo de formatos, *library(ncdf4)* (Pierce [2019]) y extraemos la información de puntos coordenados.

El primer paso es abrir el archivo .nc del tipo *geodetic_in.nc*.

```
nc <- nc_open(filename = termico_sen[4])
names(nc$var)
```

```
[1] "elevation_in"         "elevation_orphan_in" "latitude_in"
[4] "latitude_orphan_in"   "longitude_in"        "longitude_orphan_in"
```

Y extraemos los datos correspondientes a las coordenadas geográficas:

```
lon <- ncvar_get(nc = nc,
                 varid = "longitude_in")
lat <- ncvar_get(nc = nc,
                 varid = "latitude_in")
```

Y cerramos el acceso al archivo:

```
nc_close(nc = nc)
```

Además, debemos extraer los datos de temperatura propiamente tal, contenida en el archivo *LST_in.nc*:

```
nct <- nc_open(filename = termico_sen[9])
attributes(nct$var)$names
```

```
[1] "LST"                    "LST_orphan"
[3] "LST_uncertainty"        "LST_uncertainty_orphan"
[5] "exception"              "exception_orphan"
```

En la variable **LST**:

```
lst <- ncvar_get(nc = nct,
                 varid = "LST")
```

Y cerramos el acceso.

```
nc_close(nc = nct)
```

Con los tres vectores recién construidos tenemos *longitud*, *latitud* y *temperatura* construimos un objeto matrix:

```
data_puntual <- cbind("longitud"= as.vector(lon),
                      "latitud"= as.vector(lat),
                      "lst"= as.vector(lst))
```

```
summary(data_puntual)
```

```
    longitud          latitud           lst
 Min.   :-24.450   Min.   :18.17   Min.   :226.5
 1st Qu.:-19.850   1st Qu.:22.31   1st Qu.:288.7
 Median :-16.191   Median :24.96   Median :309.8
 Mean   :-16.200   Mean   :24.96   Mean   :300.1
 3rd Qu.:-12.573   3rd Qu.:27.60   3rd Qu.:316.9
 Max.   : -7.406   Max.   :31.50   Max.   :330.9
                                   NA's   :1100870
```

Construimos un objeto *extent* que define la extensión espacial de nuestro futuro modelo ráster:

```
raster_ext <- extent(cbind(as.vector(lon), as.vector(lat)))
```

Pero necesitamos calcular en filas y columnas el tamaño del modelo ráster, el que será alimentado con los puntos coordenados con el valor LST capturado por el sensor.

Para ello vamos a medir en kilómetros la extensión en el sentido longitudial y latitudinal de nuestra área de estudio, pero los valores coordenados se encuentran en coordenadas geográficas. Para el cálculo de estas métricas se dispone del paquete *geosphere*.

El paquete *geosphere* (Hijmans [2019]) está disponible para calcular las distancias y las medidas respectivas para ubicaciones angulares (longitud / latitud). Este paquete implementa los métodos que calculan varios aspectos, *distancia*, *dirección*, *área*, entre otras para posiciones de coordenadas geográficas. El paquete se debe instalar en el espacio de trabajo usando el comando *install.packages ("geosfera")*. Y cargar en memoria las funciones disponibles con *library(geosphere)*.

La **distancia geoespacial**, también conocida como la *distancia geográfica* entre dos puntos cualesquiera en el espacio, se conoce como la distancia medida a lo largo de la superficie de la tierra. Esta distancia se mide en términos de la posición de los puntos con respecto a la latitud y la posición longitudinal. Existen diferentes aspectos y fórmulas para calcular esta distancia.

Se encuentran disponibles diferentes tipos de distancias entre los puntos teniendo en cuenta la forma de la tierra, el radio asumido de la tierra, etc. Los puntos especificados para cada uno de los métodos de cálculo de distancias pueden ser un vector de dos números que contienen las respectivas coordenadas "x" e "y", una matriz de 2 columnas, la primera es la longitud, seguida de la latitud. En caso de que los dos puntos sean iguales, la distancia se considera 0 a todos los efectos prácticos.

Construimos tres objetos para facilitar el cálculo de las distancias, representan los puntos vértices del área de interés.

```
#esquina superior izquierda
pa <- c(raster_ext@xmin,raster_ext@ymax)
#esquina superior derecha
pb <- c(raster_ext@xmax,raster_ext@ymax)
#esquina inferior izquierda
pc <- c(raster_ext@xmin,raster_ext@ymin)
```

Usando la función `distGeo(p1, p2)` para el cálculo de distancia geográfica entre los puntos p1 a p2, y se divide por 1000 para llevar los metros a kilómetros. En el sentido longitudinal:

```
#división entera a 1000m
ncolumnas <- geosphere::distGeo(p1= pa, p2=pb) %/% 1000
```

Y sentido latitudinal:

```
#división entera a 1000m
nfilas <- geosphere::distGeo(p1= pa, p2=pc) %/% 1000
```

Basado en estos valores construimos una plantilla del modelo ráster:

```
modelo_geom <- raster(x = raster_ext, ncol = ncolumnas, nrow = nfilas)
```

Al cual agregamos los valores de lst mediante la conversión de objetos de tipo espacial a las celdas del modelo ráster. Cada punto es asignado a una celda o píxel, de existir más de un valor asignado se aplica el promedio:

```
modelo_lst_ras <- rasterize(x= data_puntual[, 1:2],
                            y= modelo_geom,
                            field= data_puntual[, 3],
```

```
                                    fun = mean)
```

Y por integridad se asigna el CRS para definir su sistema coordenado:

```
crs(modelo_lst_ras) <- "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs "
```

Obtenemos un modelo ráster con todos los puntos convertidos en valor de celda que no necesariamente cubren la extensión o se distribuyen en todas las celdas disponibles (Figura 8.85, superior).

Finalmente, se estiman los valores de las celdas mediante el uso de la media de los valores vecinos (fun=mean) sin considerar los valores NA (na.rm=T) solo a las celdas vacías (NAonly=T) (Figura 8.85, inferior).

```
modelo_lst <- focal(x = modelo_lst_ras, fun=mean, na.rm=T,
                    w=matrix(1,3,3), NAonly=T)
```

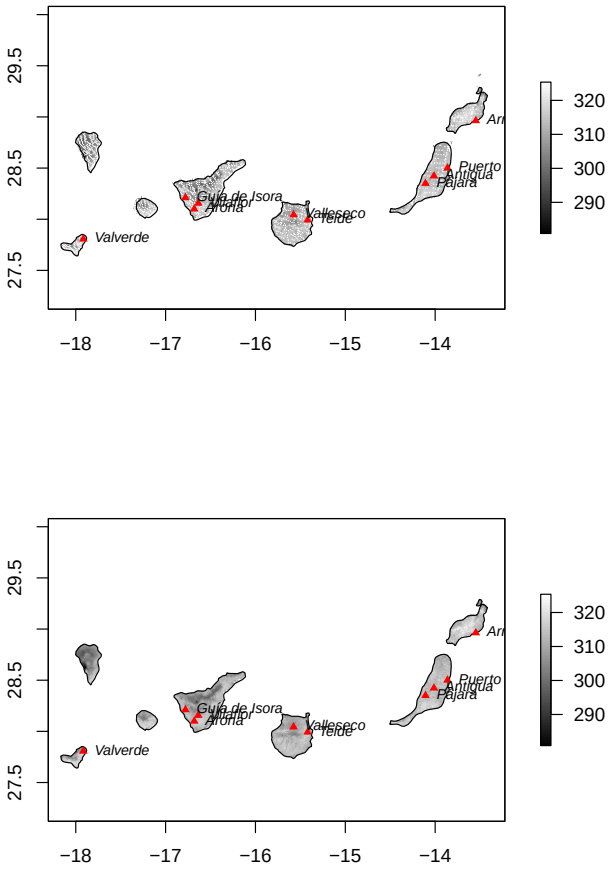


Figura 8.85: Detalle Islas Canarias, Mapa Temperaturas (en K). Escena Sentinel-3. 18 octubre 2021. Conversión de datos puntuales LST a ráster (superior). Temperaturas Diurnas (inferior).

IX APLICACIÓN

En las siguientes páginas vamos a revisar una serie de aplicaciones y técnicas de procesamiento de imágenes satelitales. Se discuten sus fundamentos teóricos y como se implementan utilizando el lenguaje R y la plataforma RStudio.

Mosaico

El mosaico de imágenes es un método para construir desde varias imágenes de la misma escena en una imagen más grande.

El producto de salida del mosaico será la unión de dos o más imágenes de entrada en una nueva imagen.

Algunas operaciones algebraicas se utilizan para obtener los valores en los píxeles en común en más de una imagen.

9.1 Creación

Usando la técnica señalada en 3.13, descargamos la data espacial vectorial del área de estudio:

```
niger <- gadm_sp_loadCountries("NER", level = 2, basefile = "RAW/")
```

Usando la transformación vectorial (3.10.3) podemos revisar la relación entre las imágenes MODIS disponibles y el área de interés, a modos de ejemplo vamos a convertir una de las cuatro imágenes del ejercicio (Figura 9.1):

```
ext <- extent(imagen1) #Crear extensión de la imagen
polygon <- as(ext, 'SpatialPolygons') #convertimos a objeto espacial
crs(polygon) <- CRS("+proj=sinu
                     +lon_0=0
                     +x_0=0
                     +y_0=0
                     +ellps=WGS84
                     +datum=WGS84
                     +units=m
                     +no_defs") #asignamos CRS de los datos
```

```
#Transformación a coordenadas geográficas, como los datos GADM
ext <- spTransform(polygon,
                   CRS("+proj=longlat +datum=WGS84 +no_defs"))
```

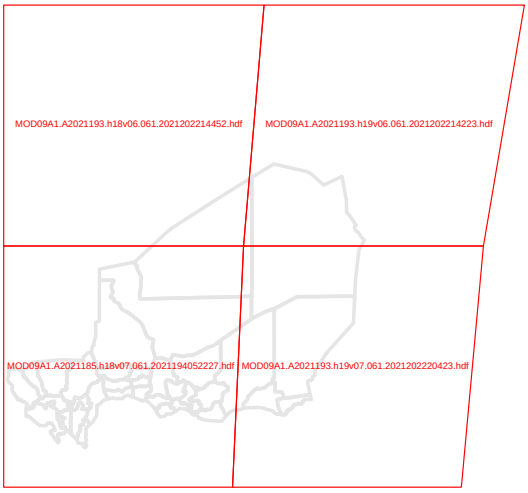


Figura 9.1: Distribución de Imágenes y polígono Límite República de Níger.

Como vemos en el diagrama las imágenes procesadas se deberán fusionar en un solo modelo para cubrir el área de estudio, para ello utilizamos el comando `mosaic()` ya que a diferencia del comando *merge* (ver 4.23) se aplica una función en los píxeles comunes, en esta ocasión aplicaremos el promedio (*mean*) para suavizar las transiciones.

```
mosaico <- mosaic(imagen1, imagen2, imagen3, imagen4, fun= mean)
```

Tenemos el mosaico en el sistema coordenado Sinusoidal y el polígono en sistema geográfico, entonces llevamos el primero al segundo o viceversa, decisión no trivial para el caso de las imágenes ya que requiere la transformación de los valores (ver 3.10.2), aunque sea pequeño es conveniente evitarlo, así que convertiremos la data vectorial al CRS de las imágenes (Figura 9.2):

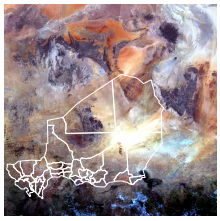


Figura 9.2: Mosaico (RGB) formado por las imágenes disponibles y el límite de la República de Níger.

```
niger_geom <- niger$spdf #data espacial
#se define el sistema de destino:
sinus <- CRS("+proj=sinu
                    +lon_0=0
                    +x_0=0
```

```
                    +y_0=0
                    +ellps=WGS84
                    +datum=WGS84
                    +units=m
                    +no_defs")
#Transformación a CRS de MODIS
niger_sin <- spTransform(niger_geom, sinus)
```

Ahora que ambos comparten CRS podemos recortar y enmascarar (Figura 9.3):

```
niger_crop <- crop(mosaico, niger_sin)
niger_msk <- mask(niger_crop, niger_sin)
```

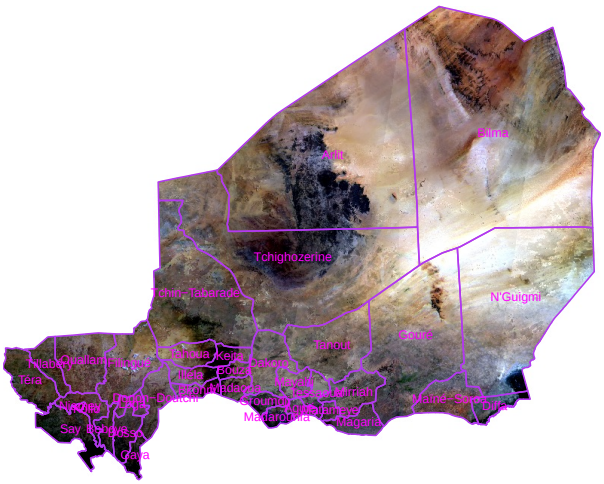


Figura 9.3: Mosaico (RGB) formado por las imágenes disponibles y el límite de la República de Níger.

9.1.1 Marcas de Estado

Tal como se describe en 8.3 además de las 7 bandas espectrales existen dos bandas de calidad, una que describe la calidad radiométrica de la medida del sensor (*sur_refl_qc_500m*) y una que describe *diferentes estados* de la atmósfera o la superficie (*sur_refl_state_500m*). Los detalles técnicos se encuentran en el documento oficial MOD09_User_Guide_V6.pdf, tabla 13.

Los códigos se encuentran comprimidos en números enteros de 16 bits. Cada píxel contiene codificado en la combinación de 1 y 0 los valores de cada bandera. En los cuadros 9.1 y 9.2 se describen los bits, la variable y los valores que pueden adquirir esas variables.

Para extraer la información de la capa *sur_refl_state_500m* debemos descomponer cada valor de píxel, encontrar los bits significativos e interpretarlos. Para ello se ofrece la función **uint2uint** que necesita definir dos variables externas: *start_index* y *stop_index* (primera columna de la tabla) y el valor entero a descomponer.

```
uint2uint <- function(int)
{
  bits <- intToBits(int)
  out <- packBits(c(bits[start_index:stop_index],
              rep(as.raw(0),32-(stop_index-start_index+1))),
              type="integer")
  return(out)
}
```

Por ejemplo, en el primer y segundo bit (de derecha a izquierda) se codifica la variable o estado ***nubosidad*** y puede tener uno de cuatro estados, Sin nubes (0), Cubierto por nubes (1), parcialmente cubierto (2) y no identificado o definido y se asigna como sin nubes (3).

El primer paso es definir las variables con la posición de los bits a leer:

```
start_index <- 1 #bit inicio
stop_index <- 2 #bit final
```

Luego, usamos la función *calc*, con la *banda* y la *función* para

Cuadro 9.1: Descripción de Estados Banda QA.

Bit	Nombre	Valor	Estado
1-2	Nubosidad	0	Claro
		1	Cubierto
		2	Parcial
		3	(No def) Claro
3	Sombra Nubes	1	Si
		0	No
4-6	Tierra/Agua	0	Océano Poco Profundo
		1	Tierra
		2	Línea de costa y lagos
		3	Agua cont. poco profunda
		4	Agua Intermitente
		5	Agua cont. profunda
		6	Océano Continental
		7	Océano Profundo
7-8	Cantidad Aerosol	0	Climatología
		1	Baja
		2	Media
		3	Alta
9-10	Cirrus Detectados	0	Ninguno
		1	Baja
		2	Mediana
		3	Alta
11	Marca Nubes	1	Nubes
		0	Despejado
12	Marca Incendio	1	Fuego

Cuadro 9.2: Continuación...

Bit	Nombre	Valor	Estado
12	Marca Incendio	1	Fuego
		0	No Fuego
13	hielo/nieve	1	Si
		0	No
14	Px Borde Nube	1	Si
		0	No
15	Salar	1	Si
16	Nieve	1	Nieve
		0	No Nieve

extraer en cada píxel el valor del estado 0, 1, 2, 3 o 4:

```
nubes <- calc(niger_msk$QA, uint2uint)
```

Es buena idea revisar los códigos que efectivamente existen en nuestra imagen y usamos *unique* (ver 3.7):

```
unique(nubes)
```

```
[1] 0 1 2
```

La capa de estados se compone de valores enteros y un número finito de valores, por definición se puede tratar como variable categórica y usamos la función `ratify()` que asocia un *RAT* o *Raster Attribute Table* y donde el valor de la celda funciona como índice a ella. La primera columna de la tabla RAT es el ID y no debe ser modificado.

Las siguientes columnas pueden ser de cualquier tipo (factor, texto, enteros o lógicos). El parámetro **count** devuelve la frecuencia o total de celdas del rango.

```
nubesf <- ratify(nubes, count=T)
```

```
nubesf
```

```
class       : RasterLayer
dimensions : 2826, 3566, 10077516  (nrow, ncol, ncell)
```

```
resolution : 463.3127, 463.3127  (x, y)
extent     : 18069.2, 1670242, 1293569, 2602891  (xmin, xmax, ymin, ymax)
crs        : +proj=sinu +lon_0=0 +x_0=0 +y_0=0 +R=6371007.181 +units=m +no_defs
source     : nubes.grd
names      : layer
values     : 0, 2  (min, max)
attributes :
 ID   COUNT
  0 9918801
  1   70917
  2   87798
```

Los datos RAT son accesibles vía la inspección de *slots* (ver 3.5):

```
nubesf@data@attributes
```

O vía función *levels* (ver 1.18):

```
levels(nubesf)[[1]]
```

```
  ID   COUNT
1  0 9918801
2  1   70917
3  2   87798
```

Ahora podemos agregar una columna con datos asociados a las categorías o índices asociando una nueva columna, por ejemplo, el uso de un texto más descriptivo para cada valor:

```
levels(nubesf)[[1]] <- cbind(levels(nubesf)[[1]],
                             estado= c("claro",
                                       "cubierto",
                                       "parcial"))
```

9.1.2 Mapa y Leyenda Categórica

Un ejemplo práctico de la utilidad de usar datos categóricos es asignar colores o leyendas para cada clase (Figura 9.4):

```
colores <- c("white", "red", "green")
plot(nubesf, legend= FALSE, col= colores) #colores a clases
plot(niger_sin, add=T) #agregar polígonos
```

```
#Crear leyenda asociando tabla RAT con colores
legend(x = "topleft",
       legend = levels(nubesf)[[1]]$estado,
       fill = colores)
```

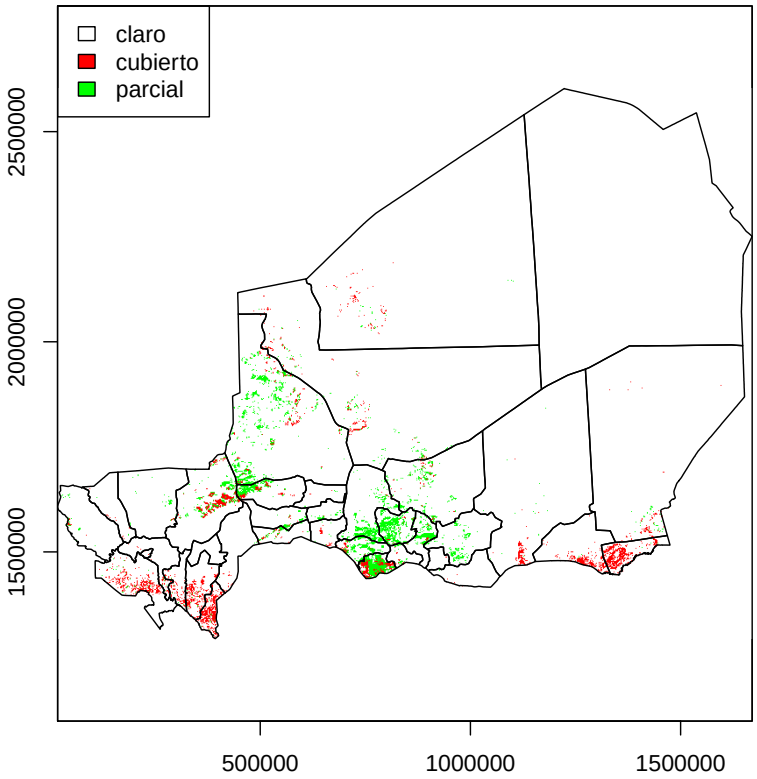


Figura 9.4: Mapa categórico de los estados relativos a la 'Nubosidad' en la banda de calidad de estados.

9.1.3 Enmascarado Categórico

El siguiente caso práctico es utilizar una categoría para *enmascarar* (borrar los píxeles que caen en la categoría). Algunos estadísticos o análisis de los datos solo son capaces de excluir automáticamente los valores *NA* y sin tener la precaución de remover valores anómalos los resultados podrían verse distorsionados.

Primero extraemos la tabla RAT con la función *levels*:

```
rat <- levels(nubesf)[[1]]
```

Luego, usamos la función `layerize(x)` que crea un RasterBrick con una capa lógica (booleana) para cada clase o categoría del rasterLayer *x*.

```
masks <- layerize(nubesf)
```

Asignamos los nombres de cada categoría a cada nueva capa o layer:

```
names(masks) <- rat$estado
```

Finalmente, los píxeles con valor cero se convierten a NA y tenemos un *rasterBrick* con una capa por categoría solo con píxeles de valor 1 (aquellos que pertenecen a dicha categoría).

```
masks[masks == 0] <- NA
```

Y para efectos de ejemplo vamos a obtener nuestras imágenes enmascaradas (Figura 9.5:

```
claro <- mask(x = niger_msk, mask = masks$claro)
cubierto <- mask(x = niger_msk, mask = masks$cubierto)
parcial <- mask(x = niger_msk, mask = masks$parcial)
```

Figura 9.5: Imagen Enmascarada por cada categoría. Se usa parámetro 'colNA' para destacar el color de los píxeles con NA para efectos de visualización.

Modelamiento Armónico

9.2 Series Temporales Incompletas

Cuando revisamos datos **históricos climáticos** (6.15) contamos con una serie **continua** de datos (6.15.1) que asegura buenos resultados de procesos estadísticos aplicados sobre ellos.

A diferencia de datos capturados en superficie e independientes de las condiciones ambientales, la *data capturada desde el espacio* por plataformas EO poseen una alta probabilidad de contar con momentos temporales sin información.

Estas pérdidas de calidad (o ausencia) son causadas principalmente por artefactos de sensores, nubes, sombras de nubes y otras condiciones climáticas.

En el contexto del análisis de series de tiempo, el **modelado armónico** es una herramienta poderosa para llenar estos vacíos y reducir el ruido suavizando la señal original.

9.3 Modelamiento Armónico

El análisis armónico (Fourier) permite que una curva compleja se exprese como la *suma de una serie de ondas en términos de coseno* y un *término aditivo*. Cada onda se define por una **amplitud** única y un **ángulo de fase** (7.3.1), donde el valor de amplitud es la mitad de la altura de una onda, y el ángulo de fase (o simplemente, fase) define el desplazamiento entre el origen y el pico de la onda en el rango 0 *a* 2π, (Figura 9.6, a). Cada término designa el número de ciclos completos completados por una onda en el intervalo definido (por ejemplo, el segundo término

completa dos ciclos) (Figura 9.6, b). Se agregan sucesivos términos armónicos para producir una curva compleja (Figura 9.6, c) y cada curva componente, o término, representa un porcentaje de la varianza total en el conjunto de datos de la serie temporal original (Jakubauskas et al. [2001]).

Así, con la sumatoria de términos se construyen curvas complejas que se utilizan para modelar la expresión de los datos capturados por el sensor, entonces, para cada dato faltante en la serie temporal, el modelo puede estimar un valor altamente confiable y terminar *rellenando* la serie (6.15.1).

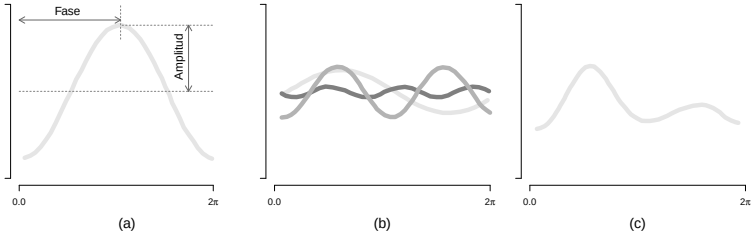


Figura 9.6: a) Curva Coseno representativa de la primera armónica. b) Curvas de armónicos en términos 1, 2 y 3. c) Curva producto de la suma de curvas en b).

9.3.1 Estimación del Modelo

Para el propósito de estimar una curva compleja y modelar, el paquete **rHarmonics** (Philipp [2021b]) permite al usuario realizar un análisis armónico en un conjunto de datos de series de tiempo dado.

Para calcular la curva armónica ajustada de una señal periódica, las regresiones de mínimos cuadrados ordinarios se calculan utilizando curvas de seno y coseno acopladas en datos de series de tiempo. El algoritmo subyacente que se basa en las ecuaciones 4.1, 4.2 de Shumway and Stoffer [2017] se puede ver a continuación:

$$X_t = \beta_0 + \beta_1 \times 2\pi t + \sum_{i=1}^{n} \left(\beta_{2i} \times cos\left(\frac{i2\pi t}{T}\right) + \beta_{3i} \times sin\left(\frac{i2\pi t}{T}\right) \right) \tag{9.1}$$

Donde:
t = Fracción de tiempo desde 01 enero 1970
T = Longitud del período temporal (un año)
n = Número total de armónicas

i = Armónica actual
β_x = Coeficientes independiente derivados de la regresión de mínimos cuadrados ordinaria (OLS)

En R procedemos instalando el paquete:

```
devtools::install_github("MBalthasar/rHarmonics")
```

Y cargamos en memoria:

```
library(rHarmonics)
```

El paquete provee una serie de datos para evaluación y estudio del funcionamiento, indicamos el nombre y dirección del paquete y archivo y leemos con la función *readRDS*.

```
ndvi_df <- base::readRDS(system.file(package = "rHarmonics",
                                     "extdata", "MODIS_NDVI_TimeSeries.rda"))
```

Los datos los podemos explorar:

```
summary(ndvi_df)
```

```
     dates                 ndvi
 Min.   :2005-01-01   Min.   :0.3800
 1st Qu.:2007-07-02   1st Qu.:0.5545
 Median :2009-12-29   Median :0.7015
 Mean   :2009-12-29   Mean   :0.6670
 3rd Qu.:2012-06-27   3rd Qu.:0.7790
 Max.   :2014-12-27   Max.   :0.8730
                      NA's   :98
```

Y encontramos que posee dos variables, *dates* (fechas) y *ndvi* para el período temporal 1 enero 2005 hasta 27 diciembre 2014 de datos MODIS para el *índice diferencia normalizada de vegetación* (9.15.2).

Una pregunta que debemos plantearnos es la resolución temporal (diferencia temporal entre un valor y su sucesivo) del set de datos.

Para encontrar esos valores utilizamos la función `diff(x)` que encuentra las diferencias de valor o fechas en un vector o matriz `x`.

```
diff(ndvi_df$dates)
```

```
Time differences in days
```

```
  [1] 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 [33] 8 8 8 8 8 8 8 8 8 8 8 8 8 5 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 [65] 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 5 8 8 8 8
 [97] 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
[129] 8 8 8 8 8 8 8 8 8 5 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
[161] 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 6 8 8 8 8 8 8 8 8
[193] 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
[225] 8 8 8 8 8 5 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
[257] 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 5 8 8 8 8 8 8 8 8 8 8 8 8 8
[289] 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
[321] 8 5 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
[353] 8 8 8 8 8 8 8 8 8 8 8 8 8 8 6 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
[385] 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 5 8 8 8
[417] 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
[449] 8 8 8 8 8 8 8 8 8 8
```

Y constatamos que el valor representativo es de 8 días. También debemos explorar el gráfico de líneas de la misma data (Figura 9.7), donde podemos apreciar en los saltos o quiebres de la línea donde no existe valor.

```
plot(x=ndvi_df$dates,
     y=ndvi_df$ndvi,
     lwd=1.5,
     type="l",
     col="gray40",
     cex=1,
     ylim=c(0.35, 0.9),
     xlab="Fechas (años)",
     ylab="NDVI")
```

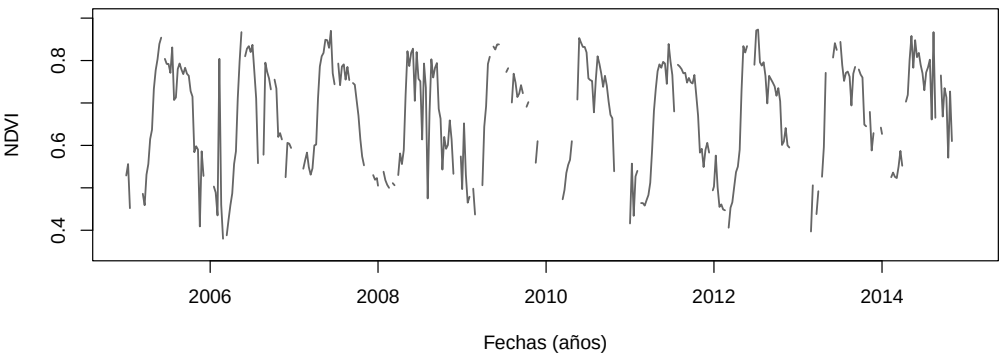


Figura 9.7: Serie Original, valores NDVI MODIS, frecuencia 8 días.

El paquete rHarmonics contiene la función `harmonics_fun(user_vals, user_dates, harmonic_deg)` que permite al usuario modelar diferentes números de ciclos por año `harmonic_deg`, para una serie de datos `user_vals` y un rango de fechas `user_dates`.

Vamos a calcular un modelo usando, uno, dos y tres términos para los datos NDVI:

```
fitted_1rd_deg <- harmonics_fun(user_vals = ndvi_df$ndvi,
                                user_dates = ndvi_df$dates,
                                harmonic_deg = 1)

fitted_2rd_deg <- harmonics_fun(user_vals = ndvi_df$ndvi,
                                user_dates = ndvi_df$dates,
                                harmonic_deg = 2)

fitted_3rd_deg <- harmonics_fun(user_vals = ndvi_df$ndvi,
                                user_dates = ndvi_df$dates,
                                harmonic_deg = 3)
```

Combinamos los valores estimados con el objeto original para efectos de dibujo y análisis:

```
combined_1df <- cbind(ndvi_df, fitted_1rd_deg)
names(combined_1df) <- c("dates", "ndvi", "fitted")

combined_2df <- cbind(ndvi_df, fitted_2rd_deg)
names(combined_2df) <- c("dates", "ndvi", "fitted")

combined_3df <- cbind(ndvi_df, fitted_3rd_deg)
names(combined_3df) <- c("dates", "ndvi", "fitted")
```

Las curvas ajustadas por un, dos y tres términos por año se puede revisar en la figura 9.8.

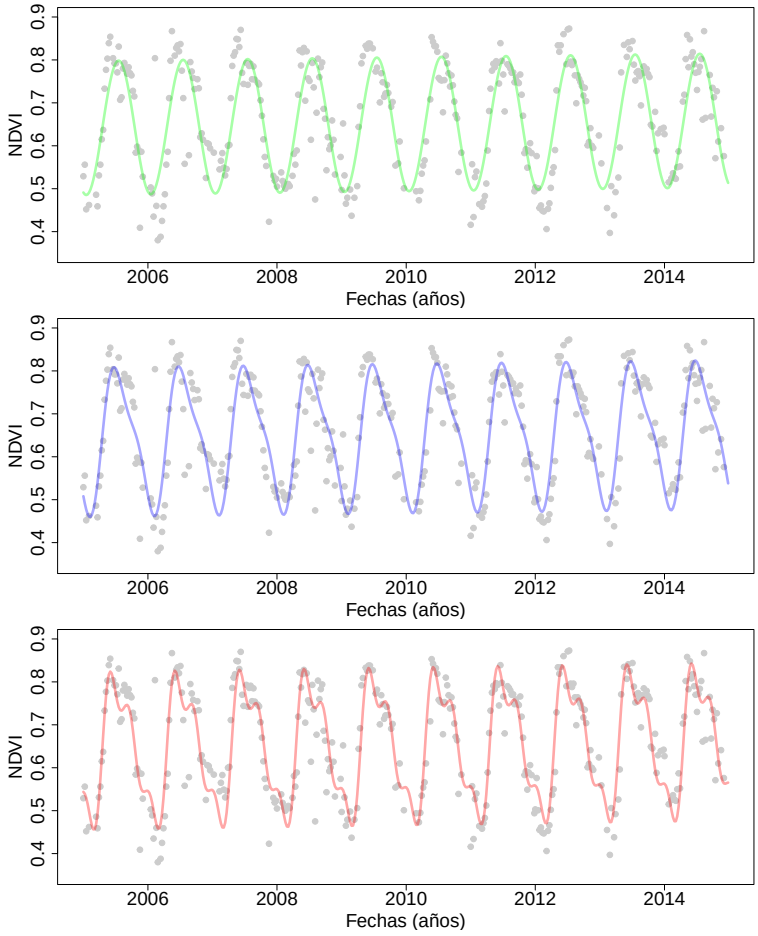


Figura 9.8: Curva NDVI ajustada por un término (superior), dos términos (centro) y tres términos (inferior).

9.3.2 Aplicación Ráster

También se puede realizar operaciones ráster sobre series temporales (6.16) y el paquete también tiene funciones para operar sobre un objeto *rasterBrick*. El mismo paquete posee un objeto para efectos de demo:

```
sample_raster <- brick(system.file(package = "rHarmonics",
                                   "extdata",
                                   "MODIS_NDVI_stack.tif"))
```

Que podemos explorar de manera interactiva en el panel *Viewer* mediante el uso del paquete (Figura 9.9):

```
library(mapview)
```

La función `mapview()` genera un mapa interactivo y le pasamos la primera capa o layer del objeto y agregamos el nombre de la capa:

```
mapview(x= sample_raster[[1]],
        layer.name="NDVI")
```

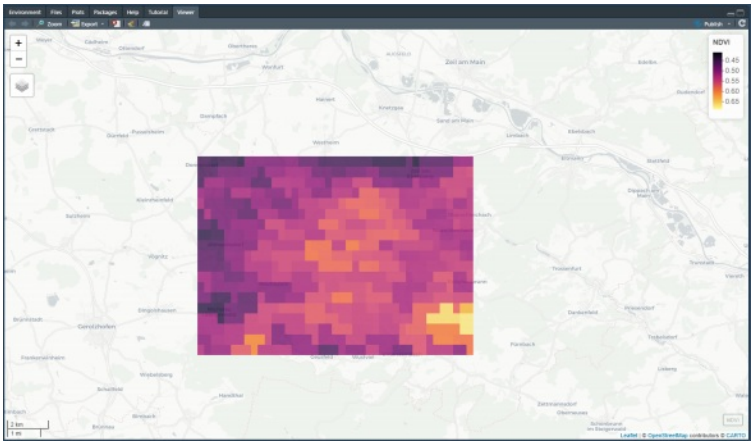


Figura 9.9: Captura de pantalla con mapa interactivo con los datos ráster NDVI contenido en paquete 'rHarmonics'.

La serie de datos posee un layer por cada fecha y contiene tanto datos válidos de NDVI como también píxeles marcados como nubes, sombras y NA (Figura 9.10).

```
r <- sample_raster[[1:12]]
plot(r,
     col=gray(1:100/100),
     asp=1,
     colNA="magenta",
     zlim=c(0,1))
```

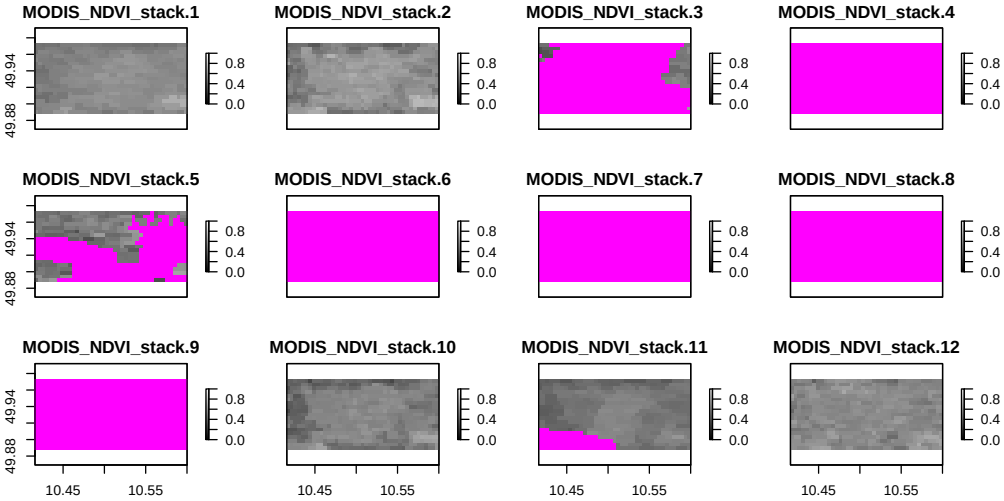


Figura 9.10: Doce primeras fechas del set NDVI, con los píxeles faltantes en color magenta.

Y para estimar cada uno de los valores perdidos se aplica de manera recursiva por cada pixel la función usada en el paso anterior (en el ejemplo vamos a utilizar tres términos) (Figura 9.11).

```
fitted_raster <- calc(sample_raster,
                      function(x){
                        harmonics_fun(x,
                                      user_dates = ndvi_df$dates,
                                      harmonic_deg = 3)
                        })
```

Y generamos un mapa para revisar los resultados de las primeras doce layer.

```
f <- fitted_raster[[1:12]]
plot(f,
     col=gray(1:100/100),
```

```
    asp=1,
    colNA="magenta",
    zlim=c(0,1))
```

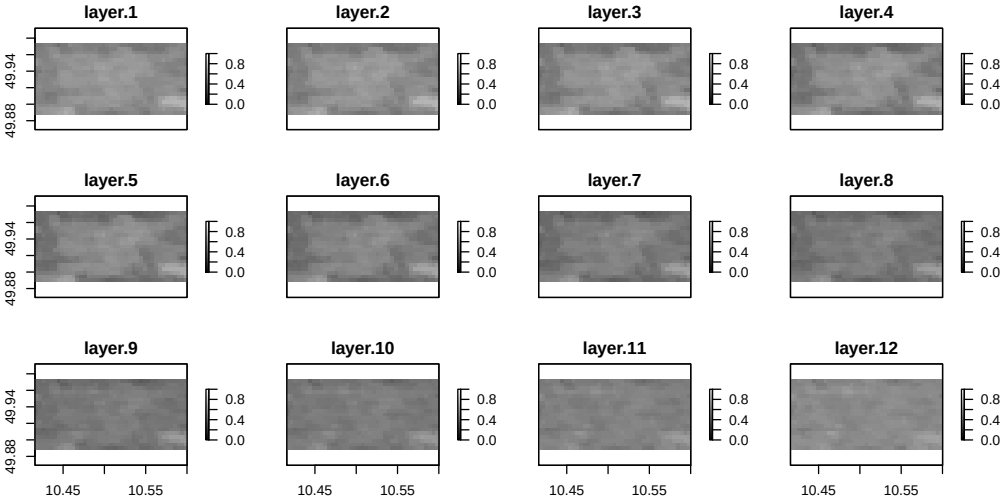


Figura 9.11: Serie temporal NDVI modelada con trés términos componentes.

9.3.3 Serie Temporal NDVI

Tal como usamos el paquete *rts* (6.16.1) con datos climáticos continuos, ahora podemos realizar las mismas operaciones sobre el objeto ráster ajustado por el modelo.

Cargar el paquete en la sesión actual:

```
library(rts)
```

Construimos el objeto *RasterBrickTS* del paquete *rts*:

```
ndvi_ts <- rts(x = fitted_raster,
               time = ndvi_df$dates)
```

Con el objeto podemos realizar las tareas de subconjunto con toda facilidad que ofrece el paquete especialmente diseñado para trabajar con fechas (Figura 9.12).

Función subset para aislar datos de un año:

```
ndvi_2006 <- subset(x= ndvi_ts,
                    subset= "2006")

plot(ndvi_2006,
     zlim=c(0,1))
```

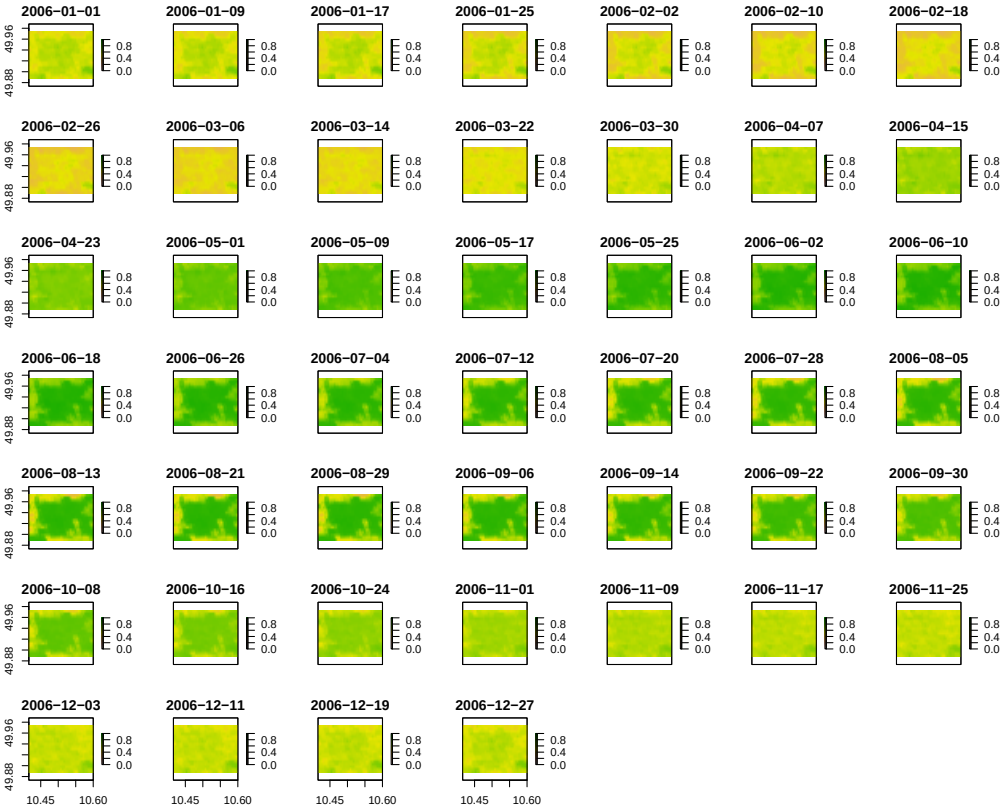


Figura 9.12: NDVI MODIS, año 2016, serie temporal corregida.

O el cálculo sobre el período mensual (Figura 9.13):

```
ndvi_mensual <- apply.monthly(x = ndvi_2006,
                              FUN = 'mean',
                              na.rm=TRUE)

plot(ndvi_mensual,
     zlim=c(0,1))
```

Finalmente, vamos a construir una serie de layer que se define por las estaciones climáticas que fechas corridas:

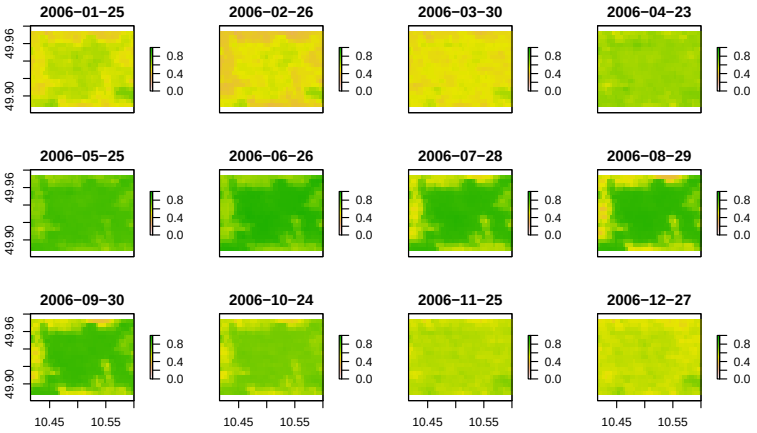


Figura 9.13: NDVI Medio Mensual.

1. Extraemos el período solsticio a solsticio (2005-2006).

```
ndvi_estac_2006 <- subset(ndvi_ts,"20051220/20061221")
```

2. Extraemos los layers disponibles.

```
nlayers(ndvi_estac_2006@raster)
```

```
[1] 46
```

3. Creamos un vector con las fechas asociadas a cada capa. Usamos la función `time()` que crea un vector de tiempos en los que se muestreó una serie de tiempo.

```
(fechas <- time(ndvi_estac_2006@time))
```

```
 [1] "2005-12-27" "2006-01-01" "2006-01-09" "2006-01-17" "2006-01-25"
 [6] "2006-02-02" "2006-02-10" "2006-02-18" "2006-02-26" "2006-03-06"
[11] "2006-03-14" "2006-03-22" "2006-03-30" "2006-04-07" "2006-04-15"
[16] "2006-04-23" "2006-05-01" "2006-05-09" "2006-05-17" "2006-05-25"
[21] "2006-06-02" "2006-06-10" "2006-06-18" "2006-06-26" "2006-07-04"
[26] "2006-07-12" "2006-07-20" "2006-07-28" "2006-08-05" "2006-08-13"
[31] "2006-08-21" "2006-08-29" "2006-09-06" "2006-09-14" "2006-09-22"
[36] "2006-09-30" "2006-10-08" "2006-10-16" "2006-10-24" "2006-11-01"
[41] "2006-11-09" "2006-11-17" "2006-11-25" "2006-12-03" "2006-12-11"
[46] "2006-12-19"
```

4. Construimos un vector con los índices que marcan las fechas de término de estación, observando las fechas del vector anterior. Aplicamos la función *mean* a las capas ráster que caen entre los límites fijados. Así obtenemos el valor NDVI por estación (Figura 9.14).

```
ndvi_est <- rts::period.apply(x = ndvi_estac_2006,
                              INDEX = c(0,11,23,34,46),
                              FUN = "mean")
names(ndvi_est@raster) <- c("Invierno","Primavera","Verano","Otoño")

plot(ndvi_est@raster,
     zlim=c(0,1))
```

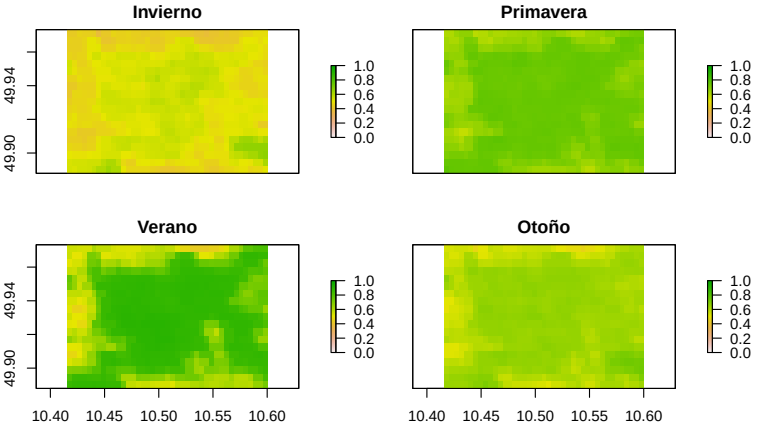


Figura 9.14: NDVI Estacional 2016.

9.4 Animación

Otra alternativa de estudiar el contenido de un rasterStack de cierto tamaño es generar un archivo gif con la animación de los datos para estudiar el desarrollo y evolución de los fenómenos representados.

El paquete *animation* (Xie [2013], Xie et al. [2018b]) está diseñado para controlar y construir un archivo gif a partir de una secuencia de ploteos o gráficos realizados con R.

Luego de instalar el paquete:

```
install.packages("animation")
```

Y cargar en la sesión actual:

```
library(animation)
```

La función `ani.options(interval)` permite configurar algunos parámetros básicos de la animación, en nuestro caso vamos

a configurar el intervalo en segundos:

```
ani.options(interval= 0.2)
```

Y preparamos un vector con las fechas de cada capa para escribir en el título del mapa:

```
fechas <- time(ndvi_ts@time)
```

La función `saveGIF(expr)` crea una serie de imágenes y luego escribe un archivo en formato *.gif*. Las imágenes se deben construir con el código R descrito en `expr`.

```
saveGIF({
  for (i in 1:nlayers(ndvi_ts@raster)){
    plot(ndvi_ts[[i]],
         legend=T,
         main = paste("Fecha:",format(fechas[i],
                                      "%d %B %Y")),
         zlim=c(0,1))
  }
})
```

Luego del proceso (que puede tardar unos minutos) se levanta automáticamente un visor con la animación (Figura 9.15).

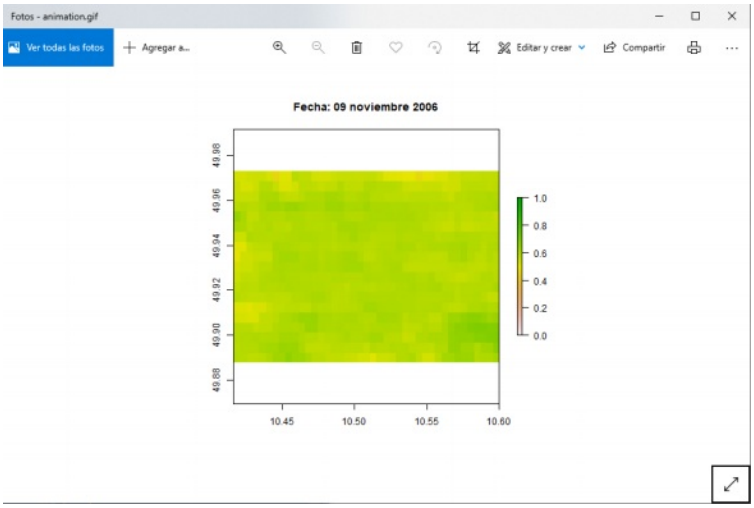


Figura 9.15: Captura de pantalla con visor de imágenes de windows con una vista de la animación GIF.

Bandas

9.5 Combinación de Bandas

El ojo humano es más sensible a las variaciones cromáticas que a las variaciones de brillo. Al mezclar tres bandas espectrales en una composición de color, hay más información disponible para la identificación de características en las imágenes. Así, el *color* se convierte en un elemento crítico en la interpretación visual de las imágenes. Nuestra percepción del color es el resultado de la reflectancia selectiva de los objetos observados a diferentes longitudes de onda. Superficies con alta reflectancia en longitudes de onda cortas (azul) y bajo en las longitudes de onda restantes aparecen azul, mientras que los objetos rojos absorben longitudes de onda más cortas y reflejan las visibles largas.

Nuestros ojos solo perciben longitudes de onda entre 400 y 700 *nm*, separando la energía recibida en tres componentes de color en nuestros propios sensores espectrales. Estos tres componentes se conocen como **colores primarios** (rojo, verde y azul o RGB), de los cuales se puede derivar cualquier otro color. Los colores que percibimos con nuestros ojos se pueden crear asignando colores primarios (RGB) a las bandas espectrales rojo, verde y azul, respectivamente (Figura 9.16). Combinación denominada **Color Verdadero**.

```
plotRGB(imagen4, r=1, g=2, b=3, stretch="lin")
```

La combinación natural puede ser modificada por el intérprete asignando en cualquier orden tres bandas espectrales. Esta nueva composición de color no corresponderá a los colores que percibimos normalmente. Los colores resultantes serán inusuales para

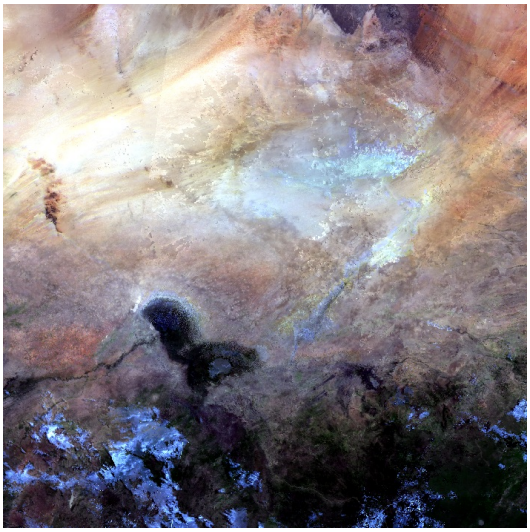


Figura 9.16: Combinación: Color Real R,G,B.

nuestros ojos y, por lo tanto, los llamamos **falsos compuestos de color** o **falso color**. Nuestros ojos no perciben los compuestos de colores falsos, pero son indicados para discriminar diferentes características del suelo y, por lo tanto, son muy común en el análisis visual de la EO.

Se puede obtener una composición de color combinando tres bandas espectrales usando dos diferentes procesos: **aditivo** y **sustractivo**. En el proceso aditivo, un color dado es obtenido mediante la adición de luz de diferentes longitudes de onda. Cuanta más reflectancia tenga un píxel, más luz (más brillante) tendrá ese color en la composición (Figura 9.17).

Figura 9.17: Modelo Aditivo.

El proceso aditivo es el que se utiliza en los sistemas de visualización electrónicos (monitores de computadora, tabletas, pantallas proyectores), y es el más utilizado para los sistemas de procesamiento de imágenes digitales. El método sustractivo se utiliza en la reproducción mecánica del color, típicamente en medios impresos (revistas, libros, etc.). Los colores impresos se obtienen mezclando tres tintas básicas: *amarillo*, *magenta* y *cian*. A diferencia del proceso aditivo, cuanta más reflectancia que tenga un píxel, menos tinta tendrá, ya que necesita ser más brillante al final de la composición.

Se han intentado establecer indicadores objetivo que ayude en la selección de la composición de color más adecuado maximizando la información no redundante. Se han utilizado diversas técnicas estadísticas, como el análisis de componentes principales o el índice de factor óptimo (OIF), ver ecuación 9.2 y punto 9.5.6.

También para el análisis visual, se han realizado algunos estudios para identificar la composición de color que ofrezcan la mejor discriminación de coberturas.

Este enfoque se basa en la selección de intérpretes con diferentes niveles de experiencia para cuantificar el número de cubiertas que se pueden discriminar adecuadamente en diversas composiciones en color.

Para los siguientes ejercicios vamos a utilizar las imágenes MODIS procesadas en el punto 8.4.

```
imagen1 <- brick(here("raw","mosaico-test11.grd"))
imagen2 <- brick(here("raw","mosaico-test22.grd"))
imagen3 <- brick(here("raw","mosaico-test33.grd"))
imagen4 <- brick(here("raw","mosaico-test44.grd"))
```

A raíz de estos resultados, el proyecto europeo de cobertura terrestre **CORINE** (land.copernicus.eu/pan-european/corine-land-cover) seleccionaron una serie de composiciones muy útiles para distintos campos del conocimiento.

9.5.1 Falso Color

Entre las múltiples combinaciones de colores utilizadas en el análisis visual, la más destacada se conoce como ***falso color***. Esta combinación se obtiene cambiando las bandas visibles hacia longitudes de onda más largas, NIR, R y G, por las RGB respectivamente (Figura 9.18). Se ha utilizado ampliamente porque la mayoría de los sensores satelitales incluyen estas tres bandas espectrales. Para facilitar la interpretación del color, es conveniente incluir una clave de color para facilitar al intérprete con este tipo de composición. Los colores más habituales corresponden a las siguientes cubiertas:

- **Rojo-magenta** representa vegetación vigorosa, como cultivos

de regadío, montañas prados, o bosques caducifolios en imágenes de verano y cultivos herbáceos de tierra de secano en imágenes de primavera. El estudio detallado de la intensidad y la saturación de rojo en estos compuestos permite identificar varias coberturas vegetales, así como para estimar sus ciclos de crecimiento y vigor.

- **Rosa** representa áreas con vegetación menos densa y / o vegetación en las primeras etapas de crecimiento y áreas suburbanas cercanas a las grandes ciudades cuando tienen jardines y árboles dispersos.

- **Blanco** representa un área con poca vegetación, nubes, arena, depósitos salinos, canteras, suelos desnudos y nieve.

- **Azul oscuro** a **negro** representa superficies total o parcialmente cubiertas por agua, ríos, lagos y presas. En zonas volcánicas, los tonos negros pueden representar lava caudales o suelos con poca cobertura vegetal.

- **Gris** a **azul metálico** representa ciudades y rocas desnudas.

- **Café** representa tierra arbustiva, variable según la densidad y el suelo material.

- **Beige** representa zonas de transición, prados secos frecuentemente asociados con exfoliación escasa.

```
plotRGB(imagen1,  r="nir", g="r", b="g", stretch="lin")
```

9.5.2 Caso 'Separación de Suelo'

Permite una diferenciación de la vegetación en tonos marrones, verdes y amarillos. Las áreas urbanas y los suelos expuestos aparecen en tonos de azul claro, mientras que las áreas inundadas y el agua aparecen en tonos azul oscuros. Esta combinación realza las diferencias de humedad en el suelo y es usada para el análisis de humedad en el suelo y vegetación (Figura 9.19).

Figura 9.18: Combinación de bandas espectrales. Falso Color(NIR,R,G)

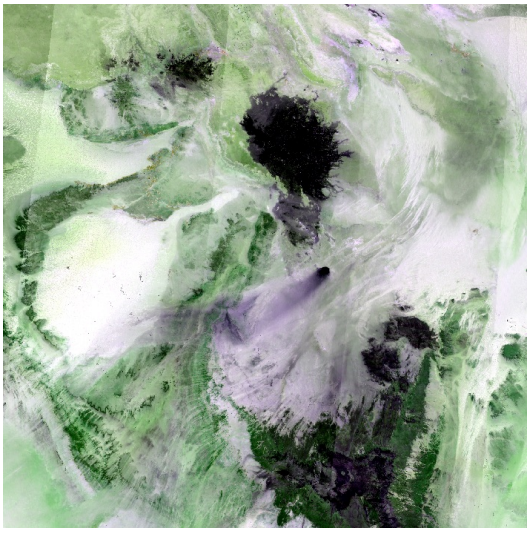


Figura 9.19: Combinación de bandas espectrales. Separación usos de suelo (NIR,SWIR1,R).

9.5.3 Caso 'Separación Zona Cultivos o Inundadas'

La vegetación sana será de color verde brillante, sin embargo, es recomendada mejor para estudios agrícolas (Figura 9.20).

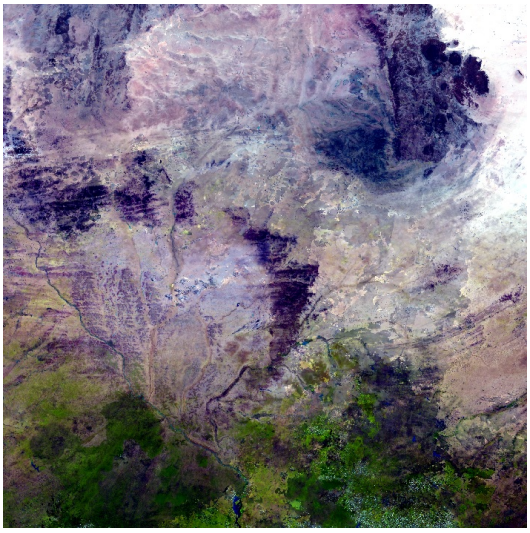


Figura 9.20: Combinación de bandas espectrales. Cultivos o zonas inundadas (SWIR1,NIR,R).

9.5.4 Caso 'Aplicaciones Geológicas 1'

Aplicaciones Geológicas, recomendada para la separación de rocas (Figura 9.21).

9.5.5 Caso 'Aplicaciones Geológicas 2'

Esta combinación implica ninguna banda visible. Proporciona la mejor penetración atmosférica. Líneas costeras, y las orillas están bien definidos. Se puede utilizar para encontrar características de textura y humedad de los suelos. La vegetación aparece en color azul. Esta combinación de bandas puede ser útil para estudios geológicos (Figura 9.22).

9.5.6 Índice de Factor Óptimo (OIF)

Cuando hay más bandas espectrales disponibles, y particularmente en el caso de sensores hiperespectrales, las posibilidades

Figura 9.21: Combinación de bandas espectrales. Aplicaciones Geológicas, separación de rocas (SWIR2,R,B).

Figura 9.22: Combinación de bandas espectrales. Aplicaciones Geológicas 2 (SWIR2,SWIR1,NIR).

de crear imágenes compuestas en color son casi ilimitadas. Lógicamente, solo algunos tendrán interés para una aplicación en particular, y algunos de ellos serán casi idénticos. Además de la percepción visual subjetiva, también se dispone de criterios cuantitativos para seleccionar la combinación de bandas óptimas. Un índice estadístico y cromático, que retiene la varianza original de los datos ha sido desarrollado, el **factor índice óptimo (OIF)** (Zheng et al. [2005]) que evalúa las varianzas de cada banda y sus correlaciones entre bandas con el objetivo de *maximizar el contenido de información* de las tres bandas seleccionadas para la composición:

$$OIF = \sum_{i=1}^{n} \left(\frac{\sigma_i}{\sum_{j=1}^{n} |r_{ij}|} \right) \tag{9.2}$$

Donde σ_i es la desviación estándar de cada una de las tres bandas incluidas en la composición, r_{ij} es el coeficiente de correlación entre cada par de las bandas seleccionadas y $n = 3$.

En R vamos a construir una función para luego poder aplicar en nuestras imágenes, vamos a considerar algunos aspectos importantes:

- Requiere tres parámetros iniciales, la imagen **img** y los índices de las tres bandas **i**, **ii** y **iii**.
- Cada banda tiene valores expresados en reflectividad (ρ) y el método estadístico está diseñado para ND de 8 bits:
  - Por cada banda se extrae el máximo: `mx <- maxValue(img)`.
  - Cada banda se divide por su máximo y se multiplica por 255 y llevamos a ND 8 bits: `r1 <- imagen1[[i]]/mx[i]*255`.
- La función requiere la desviación estándar por banda y es necesario remover sus NA.
- También se utiliza la *correlación* entre las bandas ABC; en las tres combinaciones posibles (AB,AC,BC), usando la función `cor()`.

Una de las tantas alternativas que R permite y que resume las consideraciones previas es:

```
oif <- function(img,i,ii,iii){
  mx <- maxValue(img) #extrae máximo por banda.
  r1 <- imagen1[[i]]/mx[i]*255 #llevar valores a ND
  r2 <- imagen1[[ii]]/mx[ii]*255
  r3 <- imagen1[[iii]]/mx[iii]*255

  sd1 <- cellStats(r1, stat = sd, na.rm=T) #Extraer SD
  sd2 <- cellStats(r2, stat = sd, na.rm=T)
  sd3 <- cellStats(r3, stat = sd, na.rm=T)

  cor1 <- cor(as.vector(r1),as.vector(r2)) #Correlación entre bandas
  cor2 <- cor(as.vector(r1),as.vector(r3))
  cor3 <- cor(as.vector(r2),as.vector(r3))
  sumcor <- sum(cor1,cor2,cor3) #Sumatoria

  sum(sd1/sumcor,sd2/sumcor,sd3/sumcor) #cálculo de OIF
}
```

Y el cálculo del OIF:

```
#Los índices 1(R), 2(G) y 3(B).
(valor_indice <- oif(imagen1,1,2,3))
```

```
[1] 25.80834
```

Ahora tenemos la función que entrega un valor del OIF usando tres índices, y necesitamos revisar las combinaciones posibles de las siete bandas que tenemos en nuestras imágenes y revisar aquellas combinaciones con los valores OIF mayores.

En R podemos realizar fácilmente la tarea utilizando el paquete *combinat* (Chasalow [2012]) que aporta una serie de utilidades de combinatoria.

La función `mx <- combn(x, m, simplify)` genera todas las combinaciones de x elementos tomados de m elementos a la vez. El parámetro *simplify* retorna una lista (T) o un vector(F).

Por ejemplo, tengo 4 elementos tomados de a pares nos da las siguientes combinaciones:

```
combn(1:4,m=2,simplify=T)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
```

```
[1,]    1    1    1    2    2    3
[2,]    2    3    4    3    4    4
```

Entonces, tomamos nuestros siete índices y los tomamos de a tríos y generamos las combinaciones (35 casos):

```
lst <- combn(1:7,m = 3, simplify=T)
ncol(lst)
```

```
[1] 35
```

```
lst[,1:5] #cinco primeras combinaciones
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    1    1    1    1
[2,]    2    2    2    2    2
[3,]    3    4    5    6    7
```

Ahora tomamos la imagen y la i^{sima} combinación y calculamos su valor OIF:

```
i <- 1 #creo variable i, para facilitar pruebas
oif(imagen1,lst[1,i],lst[2,i],lst[3,i])
```

```
[1] 25.80834
```

Ahora podemos llevar a una estructura de ciclo para repetir el código cierto número de veces utilizando la función `for (número de repetición) {código}`.

```
oifv <- vector() #creo objeto vacío
#Creo estructura for para repetir en número de combinaciones
for (i in 1:ncol(lst)) {
  v <- oif(imagen1, lst[1,i], lst[2,i], lst[3,i]) #cálculo de oif
  #voy agregando al objeto para tener una lista de resultados
  oifv <- rbind(oifv, v)
}
#creo lista de nombres de filas
nombres <- as.character(paste(lst[1,], lst[2,], lst[3,], sep = ""))
#asociar los nombres a los valores oif
rownames(oifv) <- nombres
```

Y obtenemos un vector con el valor del índice y el nombre de la combinación usado en su cálculo:

```
head(oifv, n=4)
```

```
        [,1]
123 25.80834
124 30.78376
125 31.73920
126 33.19833
```

Para revisar en detalle los resultados nos vamos a concentrar en los resultados de mayor valor de OIF, para ello ordenamos nuestro vector, pero necesitamos mantener los nombres de cada fila para revisar la combinación de bandas.

Para ello utilizamos un truco, no usaremos la función *sort()* sino *order()* que retorna una lista de índices ordenados:

```
ind <- order(oifv, decreasing = T)
ind
```

```
 [1] 15 12 14 34 35 13 11 33  5 25 31  9 24 22 30 10  4 28  8 32  3 23
[23]  7 21  2 29 27  6 19 20 26 18  1 17 16
```

Y usamos esos indices para revisar nuestro vector (por ejemplo, los mayores):

```
oifv[ind[1:9],]
```

```
     167      147      157      467      567      156      146
39.50488 38.85906 38.78856 37.81389 37.24589 35.99998 35.96169
     457      127
35.45019 35.35547
```

Mientras que los menores calculados son:

```
oifv[ind[27:35],]
```

```
     346      134      237      245      345      236      123
30.07406 29.74194 29.02487 28.05230 27.11844 26.76002 25.80834
     235      234
24.88766 23.48171
```

El resultado arroja las combinaciones 167 con el máximo y 234 menor valor OIF (Figura 9.23).

La semejanza en el valor del estadístico es reflejada en la semejanza de la representación visual (Figura 9.24).

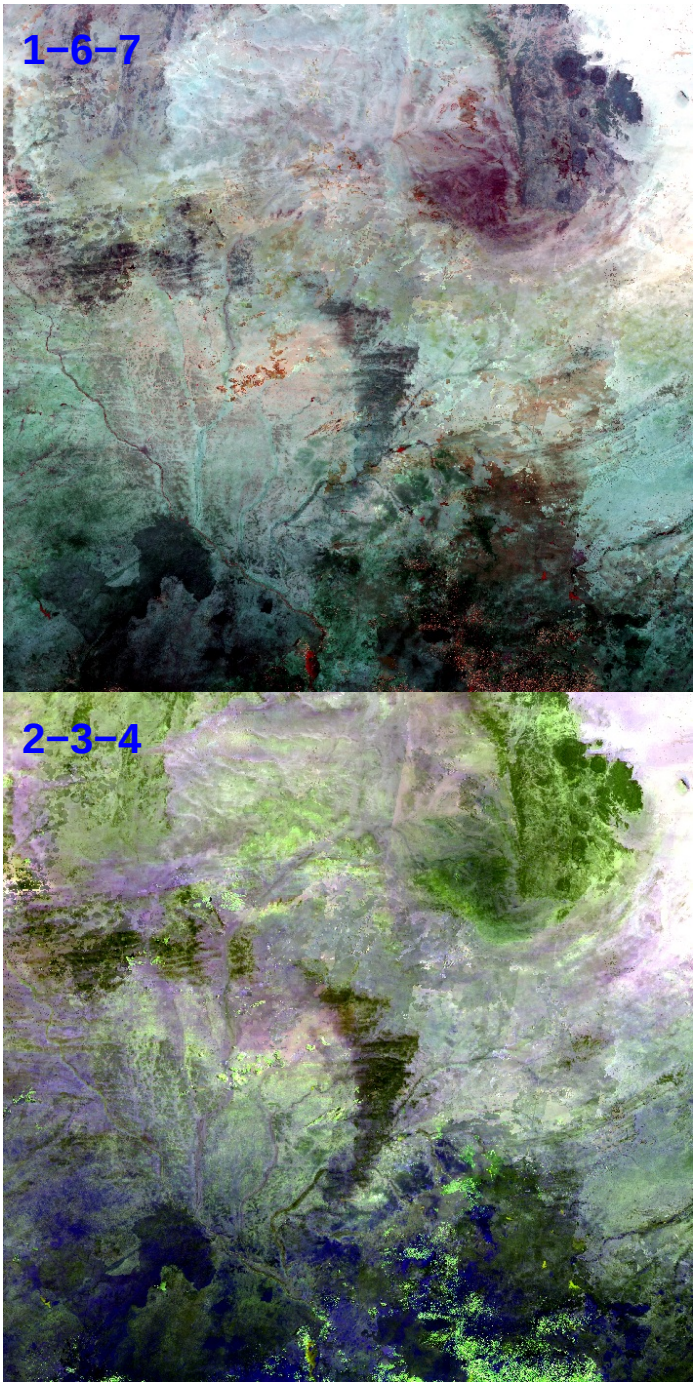


Figura 9.23: Comparación de combinación de bandas espectrales con el mayor (superior) y menor (inferior) valor de OIF.

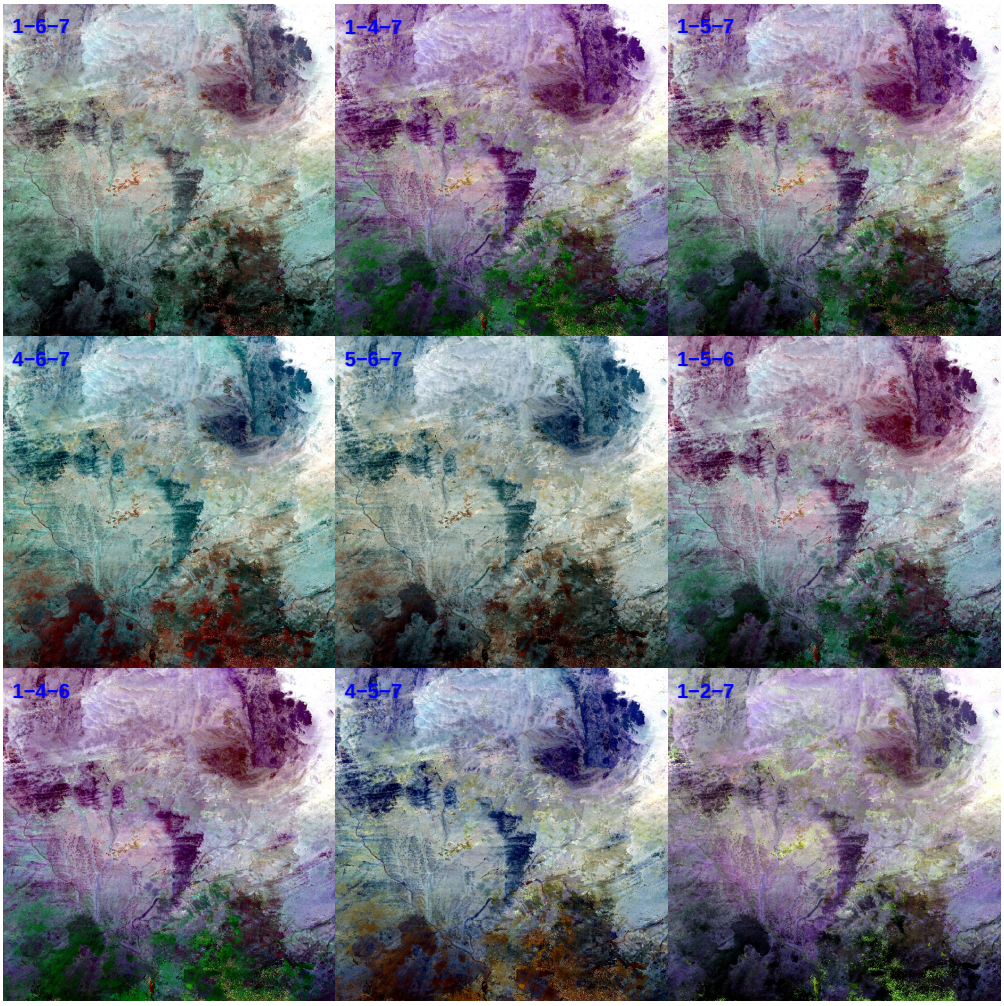


Figura 9.24: Combinación de bandas espectrales con los nueve mayores valores OIF.

En el extremo menor de los valores OIF se revisan en la figura 9.25.

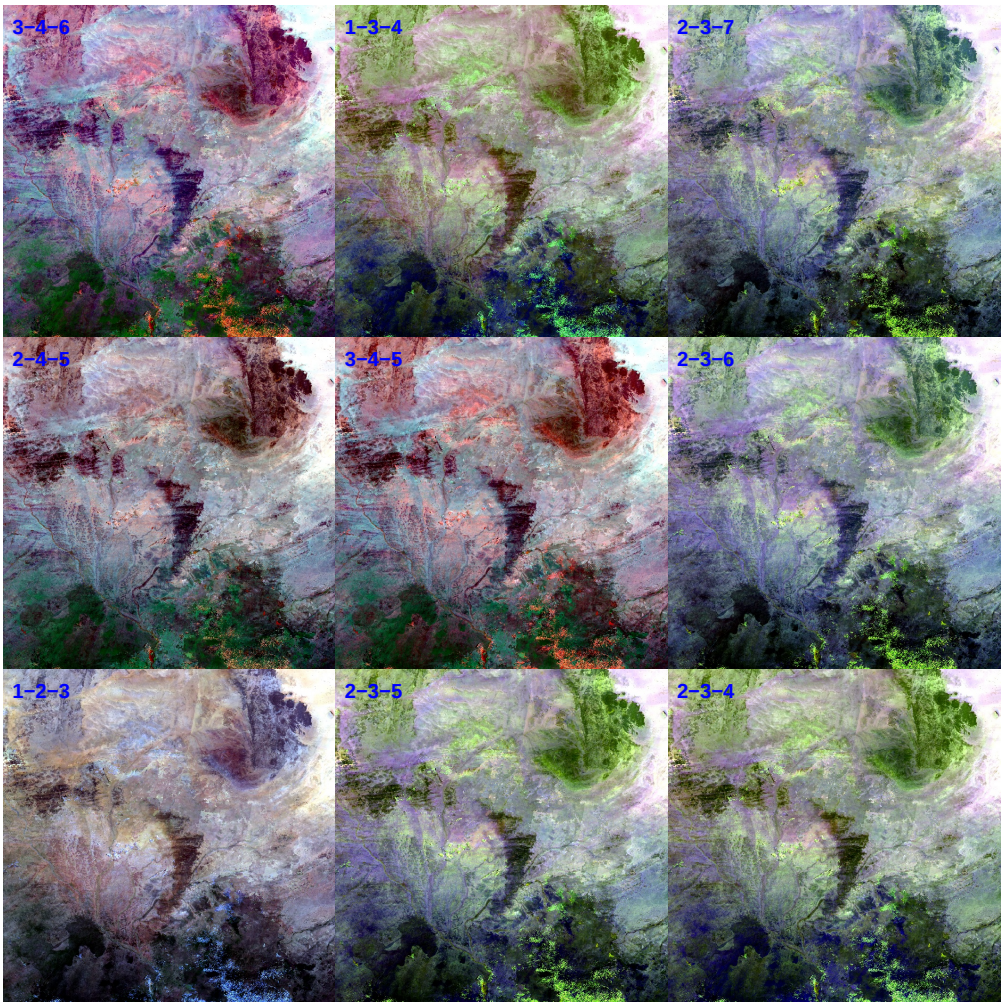


Figura 9.25: Combinación de bandas espectrales con los nueve menores valores OIF

9.6 Razón de Bandas

La **razón de bandas** es una técnica que se utiliza para mostrar variaciones espectrales de forma eficaz. Se basa en resaltar las diferencias espectrales que son exclusivas de los materiales que se busca visualizar. Los materiales superficiales idénticos pueden dar diferentes valores de reflectancia debido a la pendiente y aspecto topográfico, sombras o cambios estacionales en el ángulo e intensidad de iluminación del sol. Estas variaciones afectan la interpretación del usuario y puede dar a lugar resultados erróneos. La operación de razón de bandas puede transformar los datos, pero sin reducir los efectos de dichas condiciones ambientales.

Las imágenes de razón de bandas son conocidas por mejorar el contraste espectral entre las bandas consideradas en la operación y se han utilizado con éxito en la identificación y mapeo de tipos de suelo.

A partir del conocimiento teórico de las propiedades espectrales de los minerales, es bien reconocido que las proporciones de las bandas R/B, SWIR2/SWIR3, SWIR2/NIR se analizan para detectar óxidos de hierro, minerales que contienen hidroxilo y óxidos ferrosos, respectivamente.

En Mia and Fujimitsu [2012] encontramos algunas de las razones de banda más comunes en el ámbito de la identificación de minerales, por ejemplo, la razón de bandas:

$$Abram = \begin{cases} R & SWIR2/SWIR3 \\ G & R/G \\ B & NIR/SWIR2 \end{cases} \tag{9.3}$$

Ha probado ser tan útil que se ha denominado con nombre propio, **Razón de Abram** y muestra el óxido de hierro alterado hidrotermalmente en colores verde, los minerales arcillosos en color rojo (Figura 9.26).

El código R para construir la razón es utilizar la operación directo sobre los MDR y componiendo en un stack cada uno de los resultados:

```
#cálculo de cada componente
b1 <- imagen3$swir2 / imagen3$swir3
b2 <- imagen3$r / imagen3$g
b3 <- imagen3$nir / imagen3$swir2
#creación del objeto 'brick'
ratio <- brick(b1,b2,b3)
#mapa
plotRGB(ratio, r=1, g=2, b=3, stretch="lin")
```

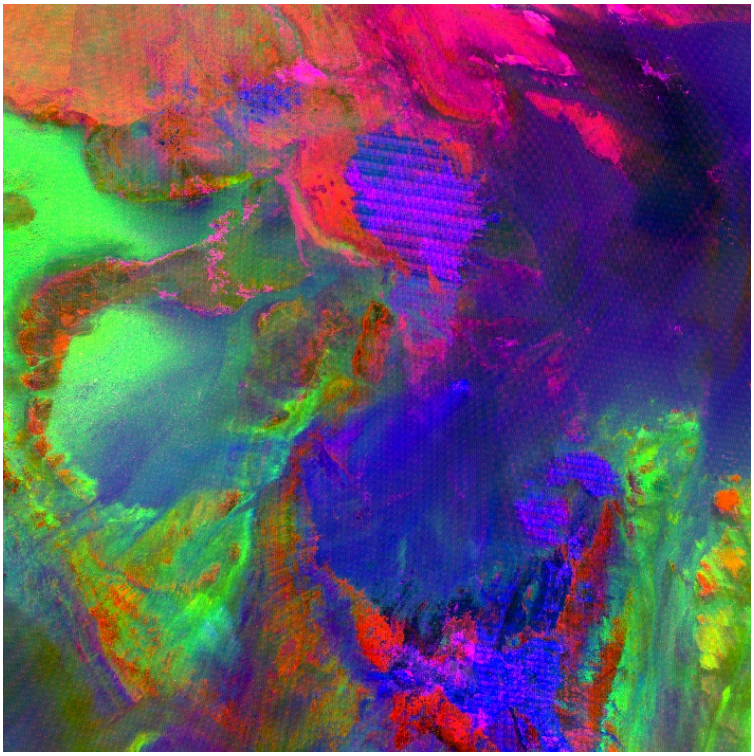


Figura 9.26: Razón de bandas Abram.

La siguiente razón muy utilizada se denomina **razón de Kaufmann** (Figura 9.27), en la cual los minerales que contienen iones de hierro, las zonas con vegetación y los minerales hidroxilos se muestran respectivamente en color rojo, verde y azul.

$$Kaufmann = \begin{cases} R & SWIR3/NIR \\ G & NIR/R \\ B & SWIR2/SWIR3 \end{cases} \tag{9.4}$$

```
#componentes
b1 <- imagen3$swir3 / imagen3$nir
b2 <- imagen3$nir / imagen3$r
b3 <- imagen3$swir2 / imagen3$swir3
#composición de bandas
ratio <- brick(b1,b2,b3)
#impresión de la imagen
plotRGB(ratio, r=1, g=2, b=3, stretch="lin")
```

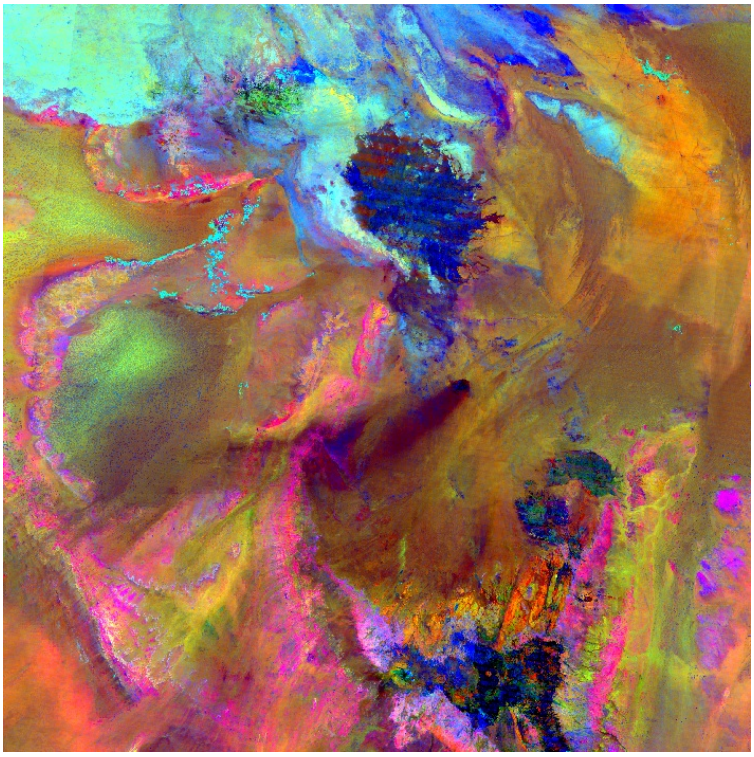


Figura 9.27: Razón de bandas Kaufmann.

La **razón Chica-Olma**:

$$Chica - Olma = \begin{cases} R & SWIR2/SWIR3 \\ G & SWIR2/NIR \\ B & R/B \end{cases} \tag{9.5}$$

Se obtienen los minerales arcillosos alterados como rojo, iones férricos como verde y óxidos ferrosos en color azul (Figura 9.28).

```
b1 <- imagen3$swir2 / imagen3$swir3
b2 <- imagen3$swir2 / imagen3$nir
b3 <- imagen3$r / imagen3$b
ratio <- stack(b1,b2,b3)
plotRGB(ratio, r=1, g=2, b=3, stretch="lin")
```

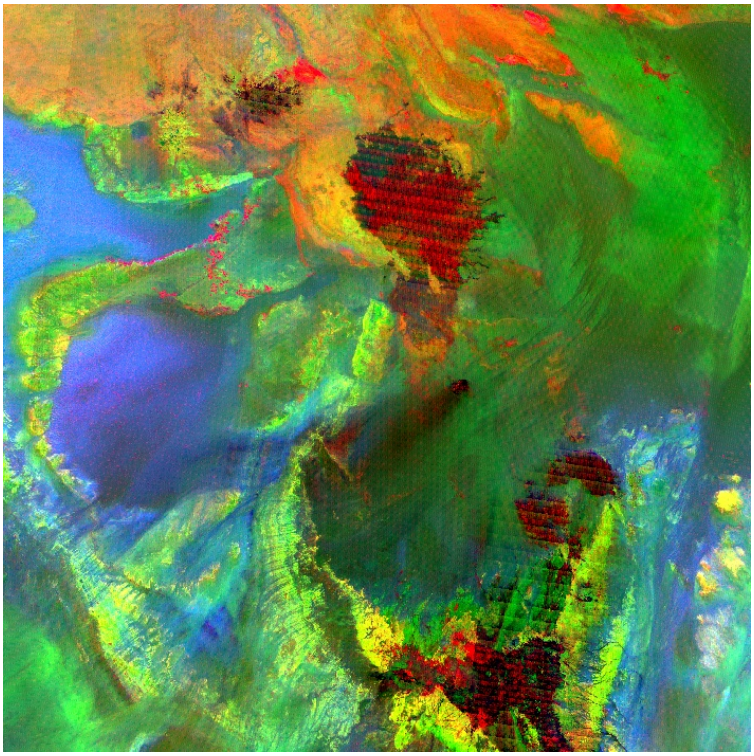


Figura 9.28: Razón de bandas Chica-Olma.

Finalmente debemos mencionar la **razón de Sultán** (Sultan et al. [1987]) que es el pilar de la investigación minera utilizando datos EO. Se define como:

$$Sultán = \begin{cases} R & SWIR2/B \\ G & SWIR2/SWIR3 \\ B & (SWIR2/NIR) \times (R/NIR) \end{cases} \tag{9.6}$$

Donde granitos aparecen en verde, cuarzos en amarillo, anfibolitas en azul, biotitas en café o serpentinas en rojo, (Figura 9.29).

```
b1 <- imagen2$swir2 / imagen2$b
b2 <- imagen2$swir2 / imagen2$swir3
b3 <- (imagen2$swir2 / imagen2$nir) * (imagen2$r / imagen2$nir)
ratio <- stack(b1, b2, b3)
plotRGB(ratio, r=1, g=2, b=3, stretch="lin")
```

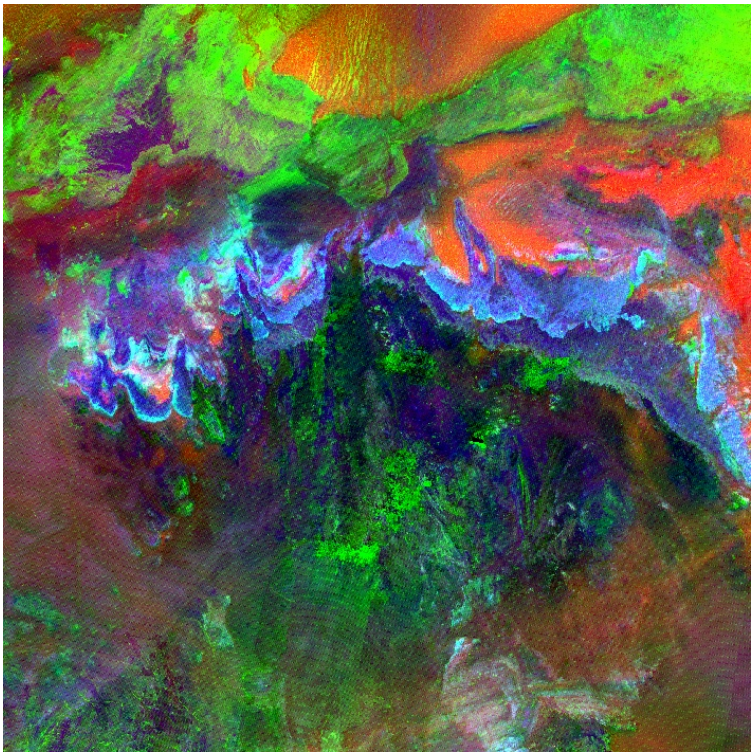


Figura 9.29: Razón de bandas Sultán.

9.7 Gráficos de Dispersión

En el punto 8.12.4 se ha utilizado el MDR *recorte_1.grd* que leemos desde disco y utilizaremos en los procesos siguientes.

```
recorte <- brick(here("raw","recorte_1.grd"))
```

Los *gráficos de dispersión* de una imagen se usan para examinar la asociación entre bandas y su relación con entidades y materiales sensados de interés. Los valores de píxel de una banda (*variable 1*) se visualizan a lo largo del eje x y los de otra banda (*variable 2*) se visualizan a lo largo del eje y. Así, los píxeles de la superficie terrestre se trazan en *un espacio R-NIR*, tienden a formar un patrón triangular (Figura 9.30, superior), que pueden definirse físicamente por:

- *Una línea base* más baja de sustrato espectros de fondo (sin vegetación verde) que se extienden linealmente hacia afuera desde el origen, comúnmente conocido como ***línea del suelo***.
- *Un ápice verde* de máxima reflectancia NIR y mínima reflectancia R, indicativo de una vegetación verde muy densa.

```
sr <- sampleRandom(recorte[[4:5]], 10000)
plot(sr[,c(1,2)], main = "Espacio Red-NIR", asp= 1, pch=16,
     col=scales::alpha("steelblue4",0.1))
```

La línea del suelo abarca píxeles de suelo brillantes (más alejados del origen), suelos oscuros (más cercanos al origen) y agua (más cercanos al origen). Los diferentes tipos de cobertura terrestre y biomasa seguirán el patrón triangular de píxeles en el espacio R-NIR, pero con líneas de suelo diferentes o ápices de verdor con pequeña variación en su posición.

Se debe tener en cuenta que la mayoría de los píxeles se trazarán dentro de la nube triangular de respuestas espectrales, que indican coberturas boscosas abiertas que se componen tanto de suelo, agua y vegetación.

En los casos extremos de regiones desérticas o con muy baja vegetación vamos a encontrar una fuerte línea de suelo, pero muy poca dispersión en el triángulo vegetal tal como vemos en el caso de la imagen MODIS de Algeria (Figura 9.30, inferior).

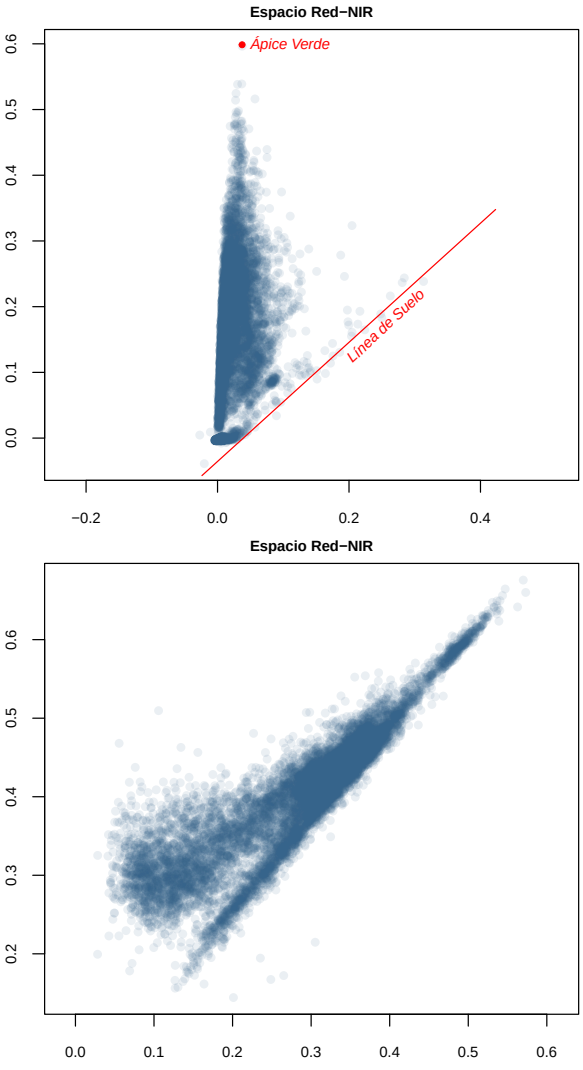


Figura 9.30: Gráfico de Dispersión R/NIR en Imagen Landsat Desembocadura Río Mataquito(superior) e Imagen MODIS de la región Tamonrasset, Algeria (inferior).

9.7.1 Correlación entre Bandas

Además de las técnicas presentadas en 9.5 y 9.6 descritas para las bandas MODIS también se pueden aplicar a las imágenes de las bandas Landsat 8 OLI y TIRS.

La forma de la nube de marcas en el gráfico de dispersión entrega valiosa información de la relación entre dos respuestas espectrales, una técnica para estudiar de manera general esta relación para toda una imagen `x`, es utilizar la función `splom(x, plot.loess=T)` del paquete *rasterVis* (Perpiñán and Hijmans [2021]) (Figura 9.31).

El panel gráfico resultante se compone de varios sub-paneles que muestran la relación entre bandas (allí donde se interceptan filas y columnas), una serie de gráficos de densidad en la diagonal y en la mitad inferior agrega una línea de regresión LOESS.

- **Gráfico de Densidad**: Un gráfico de densidad visualiza la *distribución de datos* en un intervalo (o período de tiempo continuo). Este gráfico es una variación del *histograma* que utiliza el suavizado vía kernel para la traza de valores, lo que permite ir suavizando el ruido. Los picos de un gráfico de densidad ayudan a mostrar dónde se concentran los valores durante el intervalo. Una ventaja que tienen los gráficos de densidad sobre los histogramas es que son mejores para **determinar la forma de distribución** porque no se ven afectados por la cantidad de *'bins'* utilizados (las típicas barras utilizadas en un histograma clásico). Por ejemplo, un histograma que consta de solo 4 bins no produciría una forma de distribución lo suficientemente distinguible como lo haría un histograma por ejemplo de 20 bins. Sin embargo, con los gráficos de densidad, esto no es problema.

- **Regresión Loess**: Es una técnica *no paramétrica* que utiliza la regresión ponderada *local* para ajustar una curva suave a través de puntos en un diagrama de dispersión. Las curvas de Loess pueden revelar tendencias y ciclos en datos que pueden ser difíciles de modelar con una curva paramétrica.

```
library(rasterVis)
splom(x = recorte,
      varname.cex=1,
      plot.loess=TRUE)
```

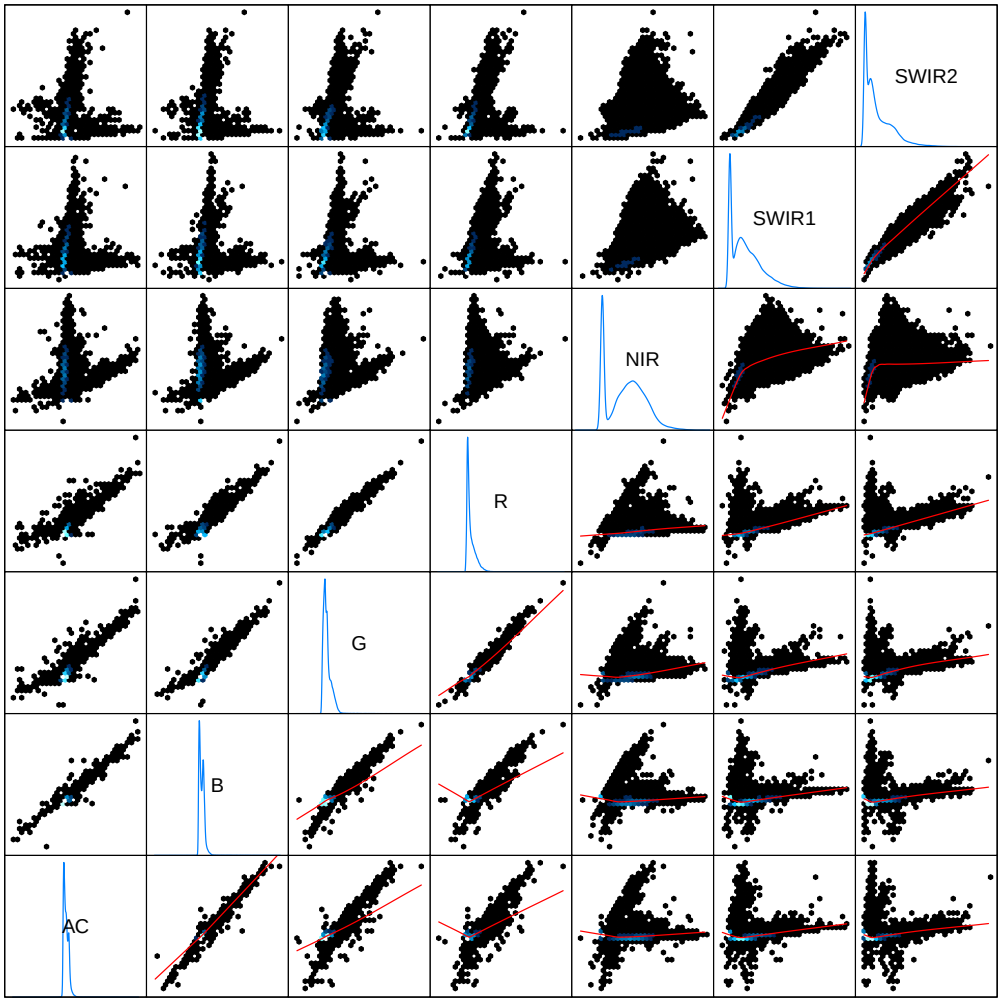


Figura 9.31: Gráficos de Dispersión para la imagen Landsat 8 OLI de la desembocadura del Río Mataquito Chile.

Umbrales (Thresholding)

Los métodos del valor umbral son un grupo de algoritmos cuya finalidad es segmentar gráficos rasterizados, es decir, separar los objetos de una imagen que nos interesen del resto. Con la ayuda de los métodos de valor umbral en las situaciones más sencillas se puede decidir qué píxeles conforman los objetos que buscamos y qué píxeles son sólo el entorno de estos objetos. Este método es especialmente útil para separar el texto de un documento del fondo de la imagen (papel amarillento, con manchas y arruguitas, por ejemplo) y así poder llevar a cabo el reconocimiento óptico de texto (OCR) con más garantías de obtener el texto correcto. Esto es especialmente útil si queremos digitalizar libros antiguos, en los que el contraste entre el texto (que ya ha perdido parte de sus pigmentos) y el papel (oscurecido y manoseado) no es demasiado elevado.

Como con todos los métodos de segmentación se trata de asignar cada píxel a un cierto grupo, llamado comúnmente "segmento". La imagen que se debe *segmentar*, como cualquier otro elemento gráfico rasterizado, está compuesta por valores numéricos (uno o más valores de color para cada píxel), la pertenencia de un píxel a un cierto segmento se decide mediante la comparación de su nivel de gris (u otro valor unidimensional) con un cierto valor umbral.

Umbralizar una imagen es un tipo especial de cuantificación que separa los valores de los píxeles en dos clases, dependiendo de un valor umbral dado q que suele ser constante. La operación de umbral mapea todos los píxeles a uno de dos valores de intensidad fijos a_0 o a_1, es decir,

$$f_{threshold}(a) = \begin{cases} a_0, & \text{si } a < q, \\ a_1, & \text{si } a \geq q \end{cases} \tag{9.7}$$

Con $0 < q \leq a_{max}$. Una aplicación común es **binarizar** la imagen con valores $a_0 = 0$ y $a_1 = 1$.

9.8 Método de Otsu

El método de Otsu, llamado así en honor a Nobuyuki Otsu (Figura 9.32) que lo inventó en 1979 (Otsu [1979]), utiliza técnicas estadísticas, para resolver el problema. En concreto, se utiliza la varianza, que es una medida de la dispersión de valores –en este caso se trata de la dispersión de los niveles de gris-.

Figura 9.32: Profesor Nobuyuki Otsu.

El método de Otsu calcula el valor umbral de forma que la dispersión dentro de cada segmento sea lo más pequeña posible, pero al mismo tiempo la dispersión sea lo más alta posible entre segmentos diferentes. Para ello se calcula el cociente entre ambas variancias y se busca un valor umbral para el que este cociente sea máximo.

9.8.1 Paquetes del Proyecto BioConductor

La imagen obtenida con el índice MNDWI (ver 9.20.3, figura 9.33) con valores -1 a 1, donde los valores negativos corresponden a terreno natural y positivos a cuerpo de agua.

```
paleta <- diverging_hcl(n= 11,
                        palette= "Broc",
                        rev=T)
plot(mndwi,
     col=colorRampPalette(paleta)(16),
     axes=F,
     box=F,
     legend=F,
     zlim=c(-1,1))
```

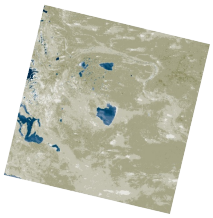


Figura 9.33: Índice MNDWI, Sector Lago Cardiel, Provincia de Santa Cruz, Argentina.

Y buscamos umbralizar la imagen para destacar y extraer los cuerpos de agua mayores utilizando el método de Otsu en la

función del paquete *EBImage*.

EBImage proporciona una funcionalidad de propósito general para el procesamiento y análisis de imágenes (Pau et al. [2010]) y es un paquete creado y mantenido por la iniciativa **Bioconductor**.

El proyecto Bioconductor se inició en 2001 y está supervisado por un equipo central, con sede principalmente en Roswell Park Comprehensive Cancer Center, y por otros miembros procedentes de instituciones estadounidenses e internacionales.

Su misión es promover el análisis estadístico y la comprensión de los ensayos biológicos de alto rendimiento actuales y emergentes. Bioconductor se basa en paquetes escritos principalmente en el lenguaje de programación R. Bioconductor está comprometido con el desarrollo de software distribuido, colaborativo y de código abierto y con la investigación alfabetizada y reproducible. Habilitar comunidades de usuarios y desarrolladores es una parte esencial de nuestra misión. Las juntas consultivas científicas, técnicas y comunitarias supervisan el proyecto, www.bioconductor.org.

Para la mantención e instalación de sus propios paquetes instalamos en primer lugar el paquete para gestión, **BiocManager** (Morgan [2021]).

```
install.packages("BiocManager")
```

Y para activar en la sesión actual:

```
library(BiocManager)
```

A continuación, se puede instalar el paquete deseado (por ejemplo EBImage) utilizando la función `install()` del paquete *BiocManager*.

```
install("EBImage")
```

Y luego cargar en la sesión actual:

```
library(EBImage)
```

9.9 Caso Práctico

Las funciones del paquete requieren la información de nuestro modelo ráster en formato de array o matriz, para ello utilizamos la función correspondiente `as.array(x)` que toma el modelo `x` y devuelve un vector con los datos:

```
arr <- as.array(mndwi)
str(arr)
```

```
 num [1:7831, 1:7831, 1] NA NA NA NA NA NA NA NA NA NA ...
```

La función `otsu()` retorna o devuelve un valor umbral o de corte para binarizar la imagen (Figura 9.34).

```
(corte <- otsu(x = arr,
               range = c(-1,1),
               levels=256))
```

```
[1] 0.2226562
```

```
#Aplicamos el valor de corte
cuerpo_de_agua <- mndwi #respaldo
cuerpo_de_agua[cuerpo_de_agua >= 0.2226562] <- 1
cuerpo_de_agua[cuerpo_de_agua < 0.2226562] <- NA

plot(cuerpo_de_agua, col= "dodgerblue3", axes=F,
     box=T, legend=F)
```

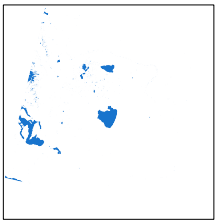


Figura 9.34: Cuerpos de agua. Imagen binarizada del índice MNDWI. Sector Lago Cardiel, Provincia de Santa Cruz, Argentina.

9.10 Procesamiento de Ráster Binario

El resultado del proceso anterior se compone de una imagen binaria, valor 1 para los píxeles asignados como cuerpo de agua y el resto NA.

A continuación vamos a revisar algunos *post procesos*, *operaciones* o *filtros* sobre la imagen de este tipo.

9.10.1 Parches

Cada grupo de píxeles asignados con valor uno adyacentes o en contacto forman un *parche*, el proceso de *contar* o *identificar* cada uno de ellos lo realizaremos con la función *clump* del paquete

raster.

La función `clump()` detecta grupos (parches) de píxeles conectados. Cada grupo obtiene una identificación única. *NA* y *cero* se utilizan como valores de fondo (es decir, estos valores se utilizan para separar grupos). Para la definición de las celdas adjacentes se puede usar el kernel de vecinos tipo **reina** o **torre** (Figuras 9.35 y 9.36), usando el argumento de `directions`. Cuando todos los vecinos correspondientes al tipo pertenecen al parche, el píxel central se asigna a ese id.

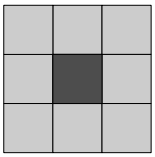


Figura 9.35: Caso de Vecindad tipo 'Reina' donde se utilizan los 8 vecinos a la celda central.

Para archivos más grandes que se procesan en trozos, el número de grupos más alto no es necesariamente igual al número de grupos (a menos que use argumento `gaps= FALSE`).

```
cuerpos_id <- clump(x = cuerpo_de_agua,
            directions=8,
            gaps=F)
```

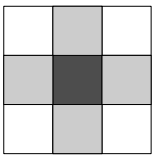


Figura 9.36: Caso de Vecindad tipo 'Torre' donde se utilizan los cuatro vecinos, superior, derecho, inferior e izquierdo.

Usamos la función `range()` para identificar el máximo **ID** retornado (que coincide con el número de *'parches'* encontrados) (Figura 9.37).

```
cellStats(x= cuerpos_id, stat= range)
```

```
[1]    1 7374
```

Se detectan 7374 parches de píxeles; recordar que la numeración solo corresponde a un orden aleatorio de definición y no tiene relación con sistema coordenado, ni ubicación relativa, ni tamaño.

```
paleta <- qualitative_hcl(n= 11,
                          palette= "Dark 3",
                          rev=F)
plot (cuerpos_id,
      col=colorRampPalette(paleta)(64),
      axes=F, box=T, legend=F)
```

Figura 9.37: Cuerpos de agua identificados por función clump().

9.10.2 Cálculo de Áreas

El paquete *bfastSpatial* (Dutrieux and DeVries [2014]) pre procesa datos de series de tiempo en formato ráster para que pue-

dan ser analizados con algoritmos de detección de cambios como **bfast**[1]. Utiliza clases del paquete ráster e incluye utilidades para ejecutar dichos algoritmos y post procesar los resultados.

El paquete se aloja en el sitio github.com y existe un instalador de paquetes especialmente diseñado para esta modalidad. Wickham et al. [2021] desarrollan el paquete **devtools** con una colección de herramientas para el desarrollo de paquetes.

```
install.packages("devtools")
library(devtools)
```

A continuación, instalamos el paquete *bfastSpatial* usando la función `install_github()`:

```
install_github('dutri001/bfastSpatial')
```

Y agregamos a nuestra sesión actual:

```
library(bfastSpatial)
```

Para calcular el tamaño de los grupos de píxeles en un RasterLayer. El área del grupo se asigna a cada grupo y se genera un nuevo ráster con las dimensiones idénticas al original, con los valores que representan el tamaño por grupo.

La definición de un grupo de píxeles sigue la lógica utilizada en la función `clump()`, donde las diagonales se incluyen o se ignoran al definir los grupos (diferencia en vecindad reina o torre). Se puede proporcionar un factor `f` de conversión opcional basado en el tamaño de píxel.

```
paleta <- sequential_hcl(n= 11,
                         palette= "Blues 2", rev=T)
plot (areas_cuerpo,
      col=colorRampPalette(paleta)(64),
      axes=F, box=T, legend=F)
```

Para nuestro caso, las imágenes Landsat que hemos utilizado, poseen un pixel de 30 m de lado, o también lo podemos definir como un píxel de 900 m^2 (30 $m \times 30$ m) y si consideramos que una hectárea son 10.000 m^2 el factor lo podemos calcular como: `f = 900/10000` (Figura 9.38).

[1] **BFAST**, *(Breaks for Additive Season and Trend)*, es un algoritmo que integra la descomposición de series de tiempo en componentes de *tendencia*, *temporada* y *resto* con métodos para **detectar** y **caracterizar** cambios dentro de series de tiempo.

Figura 9.38: Área de los Cuerpos de Agua (ha). Sector Lago Cardiel, Provincia de Santa Cruz, Argentina.

```
areas_cuerpo <- clumpSize(x = cuerpo_de_agua,
                          f = 900/10000)
```

Otra forma de explorar el resultado es mediante el uso de la función `hist()` (Figura 9.39) para visualizar la distribución de áreas de los polígonos.

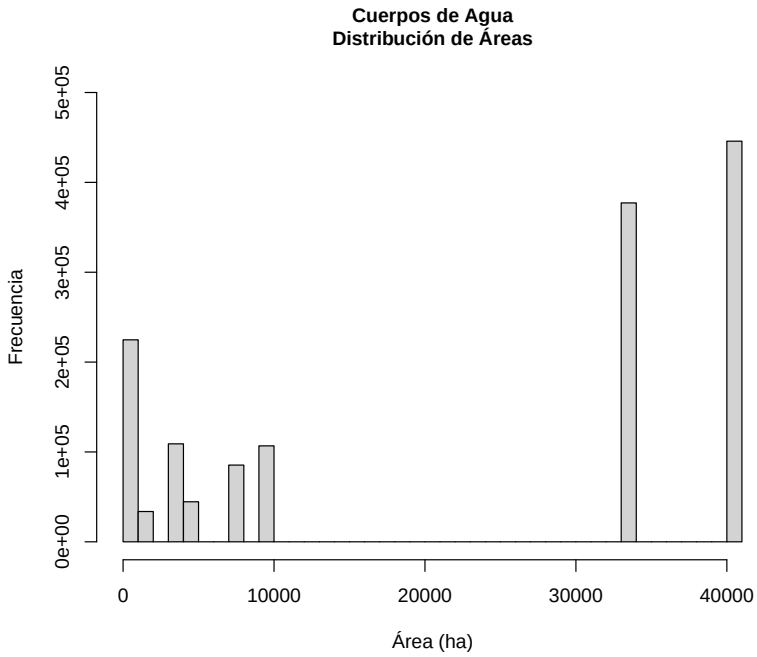


Figura 9.39: Histograma de áreas estimado mediante la función 'clumpSize()'

De la inspección podemos buscar un valor de área para separar los polígonos que deseamos mantener, en nuestro ejemplo seleccionamos el valor de 20.000 ha o de 2×10^8 m.

La función `areaSieve(x, thresh)` del paquete bfastSpatial. Aplica un umbral de área `thresh` a una capa ráster `x`.

Se proporciona un umbral en metros cuadrados (se generalizará en el futuro) y se eliminan todos los 'grupos' de píxeles más pequeños que este umbral (Figura 9.40).

```
cuerpos_mayores <- areaSieve(x = areas_cuerpo,
                             thresh = 2e8)
id_cuerpos_mayores <- clump(cuerpos_mayores)
```

```
plot(id_cuerpos_mayores,
     col=c("blue","green"),
     axes=F, box=T, legend=F)
```

Figura 9.40: Cuerpos de agua superiores a 20.000 ha. Sector Lago Cardiel, Provincia de Santa Cruz, Argentina.

9.10.3 Conversión a una capa Vectorial

La función `boundaries()` del paquete *raster* detecta *bordes* (bordes son pixeles que tienen más de una clase en la vecindad denominadas 'reina' o 'torre'), asignando un valor cero a aquellos pixeles dentro del área y un valor uno a los pixeles límite o borde.

```
cuerpo_borde <- boundaries(x = id_cuerpos_mayores,
                           type="inner",
                           classes=F,
                           directions=8)
```

En el siguiente paso vamos a signar los píxeles de borde (1) a NA para dejar solo una clase (cero):

```
cuerpo_borde[cuerpo_borde == 1] <- NA
```

Y finalmente la función `rasterToPolygons()` convierte de modelo ráster a modelo vectorial.

```
vector_cuerpo <- rasterToPolygons(cuerpo_borde,
                                  dissolve = TRUE)
```

```
spplot(vector_cuerpo,
       xlim=c( 242000,243000 ),
       ylim=c( -5432000,-5433000 ),
       colorkey=FALSE, col.regions="gray95")
```

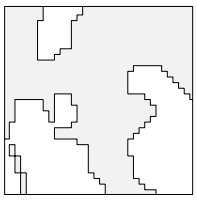


Figura 9.41: Detalle de la conversión de ráster a vectorial.

La conversión directa genera polígonos con un exceso de vértices (Figura 9.41) y una posible solución es la contenida en el paquete *rmapshaper* (Teucher and Russell [2021]) cuya principal ventaja es la disponibilidad del algoritmo de simplificación con reconocimiento topológico en la función `ms_simplify()`.

```
install.packages("rmapshaper")
library(rmapshaper)
```

Esto significa que los límites compartidos entre *polígonos adyacentes* siempre se mantienen intactos, sin espacios ni superpo-

siciones, incluso con altos niveles de simplificación (Figura 9.42).

```
vector_cuerpo_agua <- ms_simplify(vector_cuerpo,
                                  explode = T,
                                  keep = 0.1)
```

```
spplot(vector_cuerpo_agua,
       xlim=c( 242000,243000 ),
       ylim=c( -5432000,-5433000 ),
       colorkey=FALSE,
       col.regions="gray95")
```

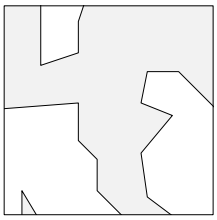


Figura 9.42: Detalle de la conversión de ráster a vectorial.

Volvemos a proyección geográfica WGS84 (se mantiene la proyección de la imagen Landsat original):

```
crs <- CRS("+proj=longlat +datum=WGS84")
vector_geo <- spTransform(x = vector_cuerpo_agua,
                          CRSobj = crs)
```

Finalmente, o exportamos los vectores a un archivo formato *shapefile* (propietario de ESRI) para usar en otros programas de análisis geográficos:

```
shapefile(vector_geo,
          here("raw","salida.shp"), overwrite=T)
```

O como en el punto 3.12 se define el uso del paquete *mapview* para crear sesiones interactivas de cartografía (Figura 9.43) a la cual agregamos los polígonos recién creados.

```
library(mapview)
mapview(vector_geo)
```

```
mapview(vector_geo, map.types = c( "OpenTopoMap"), legend=F,
        alpha.regions=0.35, col.regions="deeppink", color="deeppink2",
        lwd=3)
```

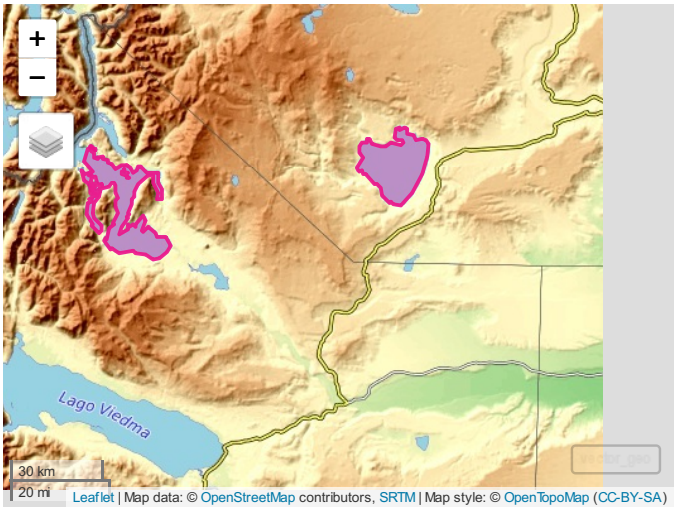


Figura 9.43: Polígonos resultantes proyectados y usados como capa adicional en una sesión 'mapview'.

Revisión Histórica

9.11 Historia con Landsat

Usando el proceso descrito en 8.10 descargamos una serie de imágenes del área geográfica de Las Vegas, Nevada y el Lago Mead EEUU, que abarcan el período 1985 al 2020 cada cinco años (Cuadro 9.3, Figura 9.44).

Cuadro 9.3: Set de Imágenes ciudad Las Vegas, Nevada, EEUU. Período 1985 al 2020. Misiones Landsat 5, 7 y 8.

Año	ID
1985	LT05_L2SP_039035_19850619_20200918_02_T1
1990	LT05_L2SP_039035_19900617_20200916_02_T1
1996	LT05_L2SP_039035_19960601_20200911_02_T1
2000	LT05_L2SP_039035_20000612_20200906_02_T1
2005	LE07_L2SP_039035_20050618_20200915_02_T1
2010	LE07_L2SP_039035_20100616_20200911_02_T1
2014	LC08_L2SP_039035_20140619_20200911_02_T1
2020	LC08_L2SP_039035_20200619_20200823_02_T1

Para buscar una respuesta espectral específica del agua utilizamos el índice $MNDWI$ (9.20.3)

```
MNDWI <- spectralIndices(imagen, swir2="SWIR2",
                         green="G", indices = "MNDWI")
```

Y para destacar el espejo de agua del Lago enmascaramos los valores del índice inferiores a 0.243 (Xu [2006]):

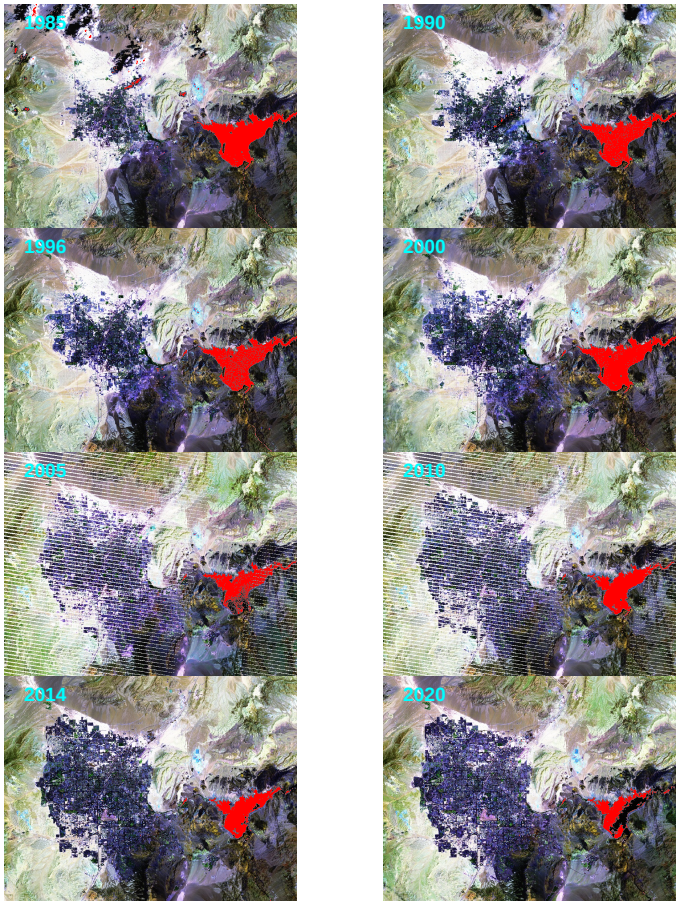
```
MNDWI[MNDWI <= 0.243] <- NA
```

Y agregamos como una capa adicional sobre una composición

de bandas *urbana* (r=SWIR2, g= SWIR1, b= R) para destacar el uso de suelo urbano (Figura 9.44).

En las cuales podemos observar el constante aumento del área poblada (colores morados) y la disminución del tamaño del espejo de agua del lago Mead.

También en las imágenes correspondientes a los años 2005 y 2010 podemos observar la presencia de los artefactos[2] productos de la falla del SLC (ver 8.8.1).



[2] Un **artefacto** se refiere a cualquier defecto visible en una imagen digital.

Figura 9.44: Crecimiento Urbano 1985-2020. Las Vegas, Nevada, EEUU. Composición Urbana (SWIR2/SWIR1/R) e índice MNDWI (rojo).

Índices Espectrales

9.12 Fundamentos

Un índice espectral es la combinación aritmética de las bandas de una imagen, con el objeto de centrarse en la detección de señales de interés y evitar el efecto de otras fuentes. Para ello se intenta **maximizar la sensibilidad** a ciertas propiedades de la superficie y **normalizar** los resultados para reducir los efectos debidos a las condiciones de la toma de la imagen (ángulo solar, ángulo de vista, estado de la atmósfera, topografía, ruido, etc.)

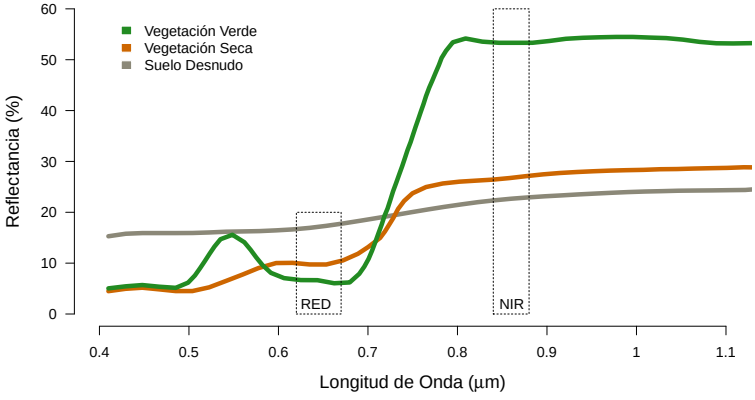


Figura 9.45: Contraste Rojo - Infrarrojo cercano en la vegetación.

9.13 Consideraciones Vegetación

Los Índices de Vegetación (VI) son técnicas simples y robustas para extraer información cuantitativa sobre la cantidad de vegetación, o verdor, por cada píxel de una imagen.

Ellos típicamente implican transformaciones espectrales de dos o más bandas, una en la región espectral R ($0.6 - 0.7\ \mu m$) y otra en el NIR ($0.7 - 1.1\ \mu m$), donde la dispersión de las hojas es significativa. Característica bastante única para la *vegetación fotosintéticamente activa*, ya que la vegetación en la senescencia tiende a reducir en NIR mientras aumenta en R (Figura 9.45). Los suelos suelen tener poco o ningún contraste entre las bandas R y NIR.

La reflectancia R y NIR se combinan en VI espectrales para mejorar los valores provenientes de la vegetación mientras se minimizan las influencias de la no vegetación.

Los VI no solo se utilizan como una medida óptica directa del verdor, sino que también son útiles como una medida indirecta de *variables biofísicas*, (por ejemplo, LAI, fAPAR, fracción de verdor de vegetación (Fv), biomasa y fotosíntesis).

El término *verdor* en sí puede definirse como la composición de señales de: contenido de clorofila foliar, área foliar, cobertura del dosel y estructura del dosel.

En efecto, un VI es un 'integrador' de muchas variables que juntas determinan la *señal fotosintética* de la vegetación activa. El término verdor reconoce tanto las fortalezas como las limitaciones de VI para recuperar directamente variables biofísicas específicas.

Por un lado, los VI exhiben una *alta sensibilidad a las variaciones espaciales y temporales* del verdor. Por otro lado, generalmente no pueden diferenciar entre *múltiples causas* responsables del cambio en el verdor, por ejemplo: si las variaciones en VI se deben a un cambio en el LAI o en el contenido de clorofila de las hojas.

Los VIs están restringidos en su capacidad de recuperar directamente variables biofísicas específicas, ya que representan un compuesto de muchas propiedades de vegetación y dosel de hojas. Sin embargo, para áreas locales y tipos específicos de cobertura del suelo, muchos han encontrado que los VIs son útiles para construir modelos semiempíricos para la extracción de información biofísica de la vegetación.

9.13.1 Interacción EEM - Vegetación

La vegetación tiene una reflectividad baja en el espectro visible, con un pico en el color verde debido a la clorofila, y muy alta en el infrarrojo reflejado o próximo (NIR) debido a la escasa absorción de energía por parte de las plantas en esta banda. En el infrarrojo medio hay una disminución especialmente importante en aquellas longitudes de onda en las que el agua de la planta absorbe la energía (Figura 9.46).

Esta curva tan contrastada se debilita en el caso de la vegetación enferma en la que disminuye el infrarrojo y aumenta la reflectividad en el rojo y azul. Se observa también que la reflectividad de una planta depende de su contenido en agua. Cuando el contenido de agua aumenta disminuye la reflectividad ya que aumenta la absorción de radiación por parte del agua contenida en la planta (Figura 9.48).

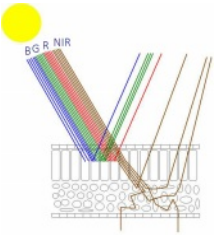


Figura 9.46: Interacción de la Energía Electromagnética con la estructura interna de la hoja vegetal.

9.13.2 Parámetro Foliar

Los parámetros relativos a la hoja afectan la respuesta espectral son:

- **Pigmentos de la Hoja**, el contenido de clorofila (a o b) y carotenoides tienen distintas magnitudes de absorción a diversas longitudes de onda.
- **Estructura Interna**, la estructura física de la hoja afecta la reflectividad, un mesofilo más ancho con mayor número de cavidades de aire, es mayor, por ejemplo, árboles caducos mayor reflectancia en el NIR que árboles coníferos (ej. pino), ellos poseen un mayor contenido de celulosa que absorbe NIR.
- **Contenido del Agua**, afecta la reflectividad ya que absorbe energía en la región Visible y NIR.

9.13.3 Parámetros Dosel

Los árboles son estructuras complejas con múltiples capas de hojas y ramas. La radiación solar interactúa con las hojas a nivel celular y la radiación que pasa a través de una hoja interactúa con la siguiente hoja que se encuentra.

Figura 9.47: Comparación de cobertura foliar entre dos plantas teóricas. A la izquierda el LAI estimado 3. A la derecha 0.7.

- **Índice de Área Foliar (o LAI)**. El área de hojas verdes por unidad de área. En la figura 9.47 se muestran dos hipotéticas coberturas vegetales. LAI se calcula de varios métodos tanto directos como indirectos, pero el principio es el mismo, establecer la relación entre el área cubierta por *la totalidad* de las hojas (área foliar) versus el área de la unidad de medida espacial (generalmente un metro cuadrado).

- **Distribución del ángulo de las hojas (o LAD)** del dosel de una planta se refiere a la descripción matemática de la orientación angular de las hojas en la vegetación. Específicamente, si cada hoja está representada conceptualmente por una placa plana pequeña, su orientación se puede describir con el cenit y los ángulos azimutales de la superficie normal a esa placa. El LAD de un dosel vegetal tiene un impacto significativo en la reflectancia, transmitancia y absorción de la luz solar en la capa de vegetación y, por lo tanto, también en su crecimiento y desarrollo.

- **Proporción de hojas/madera** afecta por diferencia en los mecanismos interactivos con la EEM.

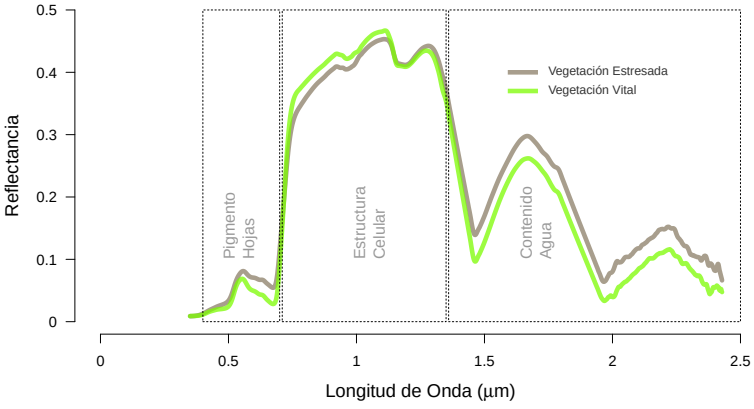


Figura 9.48: Factores que dominan la respuesta espectral para dos tipos de vegetación: vital y con estrés hídrico.

9.13.4 Preparación del área de estudio

Utilizando el rasterLayer *recorte* generado en el punto 8.12.3 vamos a realizar algunos ajustes para preparar nuestra imagen

de pruebas.

Chequear los nombres de las capas disponibles para facilitar su identificación:

```
names(recorte)
```

```
[1] "AC"    "B"     "G"     "R"     "NIR"   "SWIR1" "SWIR2"
```

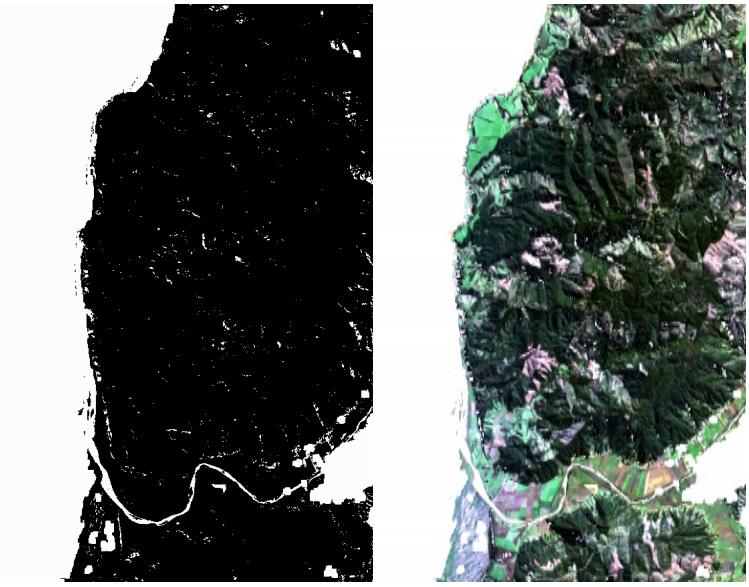


Figura 9.49: Máscara por valor 21824, pixel Claro (izquierda) y enmascarado de la zona de estudio (derecha).

Ajustamos a la zona de estudio una máscara con los valores de banda *QA_PIXEL* con el valor de pixel *21824* que corresponde a pixel *claro* (Figura 9.49, izquierda).

```
mascara_recorte <- crop(x = mascara,
                        y = extent(753411.8, 770155.5,
                                   6114039, 6140000))
mascara_recorte[mascara_recorte == 21824] <- 1
```

Y enmascaramos nuestra imagen original para obtener un rasterLayer con valores ajustados a la respuesta física esperada (Figura 9.49, derecha).

```
area_estudio <- mask(recorte, mascara_recorte)
```

9.14 Diseño de Índices Espectrales

Los índices espectrales son imágenes que se calculan a partir de imágenes multi banda. Las imágenes destacan un fenómeno concreto que está presente y atenúan otros factores que degradan los efectos de la imagen.

Por ejemplo, un *índice de vegetación* mostrará la vegetación que está en buen estado con un color brillante en la imagen del índice, mientras que la vegetación que no lo está tendrá valores más bajos y el terreno desnudo será oscuro.

Dado que el sombreado debido a las variaciones del terreno (colinas y valles) afecta a la intensidad de las imágenes, los índices se crean de manera que el *color de un objeto se resalte* en lugar de que lo haga el brillo o la intensidad del objeto.

El valor de un índice de vegetación para un pino en buen estado que está sombreado en un valle *será similar* al de un pino que está a pleno sol.

A menudo, estos índices se crean añadiendo y sustrayendo bandas, con lo que se generan varias relaciones de bandas.

Están vinculados a bandas concretas que están en parte definidas del espectro electromagnético. Como resultado, es posible que solo sean válidos para ciertos sensores o clases de sensores, y es fundamental que se usen las bandas adecuadas en el cálculo.

9.15 Índices Generales de Vegetación

9.15.1 Índice Razón Vegetal (RVI)

La razón de bandas NIR y Red puede indicar la presencia de vegetación. Para definir un índice que sea independiente del sensor, la razón debe ser definida en términos de un parámetro geofísico tal como es la reflectancia (ρ). El *Índice Razón Vegetal (RVI)* (Birth and McVey [1968]) fue uno de los primeros de los índices usados en teledetección,

$$RVI = \frac{\rho_{NIR}}{\rho_{red}} \tag{9.8}$$

También se describe en la literatura como *Simple Ratio (SR)*.

El resultado no está normalizado y se ve muy afectado por valores extremos presentes en alguna de las capas.

El paquete RStoolbox (Leutner et al. [2019]) presenta una función optimizada y práctica para el cálculo de una multitud de índices.

```
rvi <- spectralIndices(area_estudio,
                       red = "R",
                       nir = "NIR",
                       indices = "SR")
```

Al no estar normalizado, los valores se deben ajustar, y para facilitar el proceso de buscar los límites apropiados podemos realizar un análisis básico mediante el estudio de la distribución de los valores, por ejemplo, revisar la frecuencia de los valores:

```
head(freq(rvi), n=5)
```

```
     value count
[1,] -1821     1
[2,] -1249     1
[3,] -1107     1
[4,]  -973     1
[5,]  -899     1
```

O visualizar de manera gráfica mediante un histograma (Figura 9.50).

Y así, removemos todos los valores extremos y muy escasos que probablemente son producto de considerar valores anómalos de reflectancia. Para ello usamos las técnicas de subconjunto y condiciones lógicas:

El código R toma los valores **mayores a 50 y valores menores a 0** les asigna un valor *NaN* para removerlos, (también pueden ser asignados valores de *NULL* o *NA*).

```
rvi[rvi > 50 | rvi < 0] <- NaN
```

Finalmente, obtenemos el mapa Índice RVI (Figura 9.51).

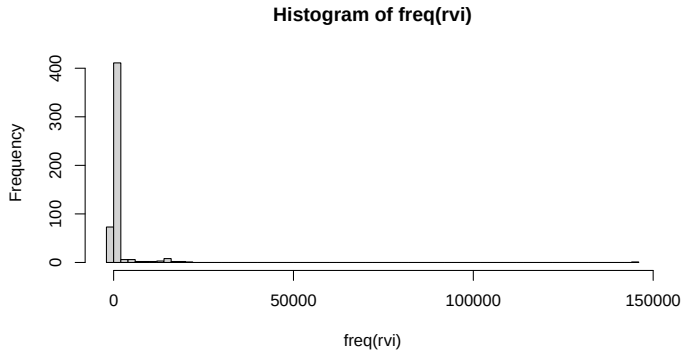


Figura 9.50: Histograma de valores en rasterLayer RVI 'hist(freq(rvi))'.

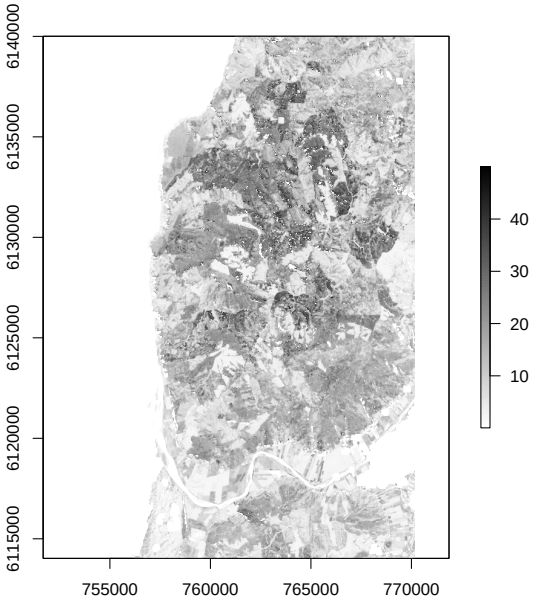


Figura 9.51: Índice RVI. Sector Desembocadura Río Mataquito.

Los valores cercanos a uno indican respuesta similar en las bandas R y NIR (por ejemplo, el suelo desnudo) y a medida que aumenta el valor más intenso el indicador de verdor vegetal.

La proporción o razones son útiles por su capacidad para reducir muchas formas de ruido multiplicativo, tales como diferencias de iluminación, atenuación atmosférica, sombras de nubes y variaciones topográficas (pendiente, aspecto), todas las cuales influyen en múltiples bandas de manera similar y, por tanto, puede minimizarse mediante la relación de bandas.

Las proporciones, sin embargo, son sensibles al fondo del suelo debajo de las copas con vegetación con valores que serán positivamente sesgados cuando los suelos sean oscuros o húmedos.

9.15.2 Índice Vegetal de Diferencia Normalizada (NDVI)

El NDVI es una medida del estado fitosanitario basado en la forma en que una planta refleja la luz en las frecuencias R y NIR (Rouse et al. [1973]).

La *clorofila* (un indicador de salud) *absorbe una gran cantidad de luz visible* y la estructura celular de las hojas *refleja intensamente la luz infrarroja cercana (NIR)*. Cuando la planta se deshidrata, enferma, sufre enfermedades, etc. el mesófilo esponjoso [3] se deteriora y la planta absorbe más luz infrarroja cercana, en lugar de reflejarla. Así pues, la observación de cómo cambia la NIR en comparación con la luz roja proporciona una indicación precisa de la presencia de clorofila, que está vinculada con la salud de las plantas.

[3] **Mesófilo**, es un término botánico que designa el tejido que se encuentra entre las epidermis del anverso y reverso de las hojas (wikipedia).

El NDVI se calcula con la siguiente fórmula:

$$NDVI = \frac{\rho_{NIR} - \rho_{red}}{\rho_{NIR} + \rho_{red}} \tag{9.9}$$

Este índice define valores desde -1.0 *a* 1.0, donde:

- *valores negativos*: están formados principalmente por nubes, agua y nieve.
- *valores negativos cercanos a cero*: están formados principalmente por rocas y suelo descubierto.
- *valores pequeños* (< 0.1): corresponden a áreas sin rocas, arena o nieve.

- *valores moderados (*0.2 *a* 0.3*)*: representan arbustos y praderas.
- *valores grandes (*0.6 *a* 0.8*)*: indican bosques templados y tropicales.

El NDVI tiene éxito como medida de vegetación porque es lo suficientemente estable para permitir comparaciones significativas de cambios estacionales e interanuales en el crecimiento y actividad de la vegetación.

Se utiliza ampliamente para monitorear la vegetación a escala continental y global, pero parece ser un pobre indicador de la biomasa vegetal si la cobertura del suelo es baja, como en las regiones áridas y semiáridas. También tiende a saturarse y perder sensibilidad por una densa cubierta vegetal, como las copas de los árboles de la selva tropical.

Como es un índice basado en proporciones, el NDVI mantiene los beneficios de *razón de bandas* y *normaliza fuentes multiplicativas* de ruido extraño para producir valores más estables.

El resultado para el área de trabajo lo podemos examinar en el área de estudio (Figura 9.52).

```
ndvi <- spectralIndices(area_estudio, red = "R", nir = "NIR",
                        indices = "NDVI")
```

9.16 Índices con Reducción de Efecto Suelo

Los índices de esta categoría intentan compensar las variaciones en el cantidad y condición del fondo del suelo (brillo y contenido de humedad).

9.16.1 Índice Vegetación Ajustado por Suelo (SAVI)

El NDVI, aunque es una medida de vegetación, también responde a variaciones en el suelo debajo de las copas de vegetación, lo que dificulta distinguir de manera confiable los cambios en la vegetación de los causados por el suelo.

Los suelos son muy heterogéneos espacialmente, y varían en menor medida temporalmente (generalmente a través de procesos de secado y eventos de humectación). En estudios de campo,

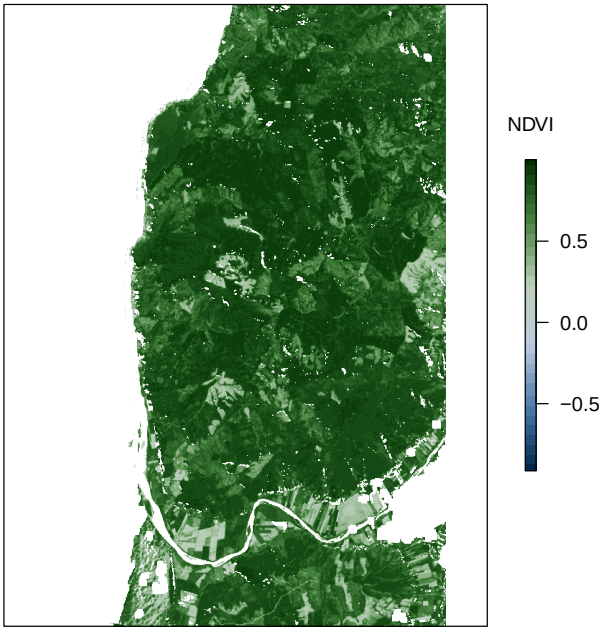


Figura 9.52: Índice NDVI. Sector Desembocadura Río Mataquito.

se han demostrado relaciones biofísicas para variar el NDVI significativamente con el brillo del fondo del suelo subyacente, a más oscuridad o humedad de los suelos dan resultado valores de NDVI mucho más altos (hasta el doble) que los suelos más secos o más brillantes, para condiciones equivalentes de vegetación (Figura 9.53).

El índice ajustado por suelo o SAVI es un índice muy superior para ambientes de baja cobertura vegetal (Huete [1988]), se expresa como:

$$SAVI = \left(\frac{\rho_{NIR} - \rho_{red}}{\rho_{NIR} + \rho_{red} + L} \right) (1 + L) \qquad (9.10)$$

Donde L es una constante calculada empíricamente [4] para minimizar la sensibilidad del índice a las variaciones de reflectancia del suelo. Si L es cero, SAVI es igual al NDVI.

Si la cobertura de vegetación es intermedia L estará alrededor de 0.5. El factor $(1 + L)$ asegura el rango de SAVI sea igual al NDVI $[-1, +1]$.

[4] La palabra empírico, viene del griego *experimentado*, es un adjetivo que indica que algo está basado en la propia *experiencia* y *observación* sin haber implicado una suposición o deducción lógica.

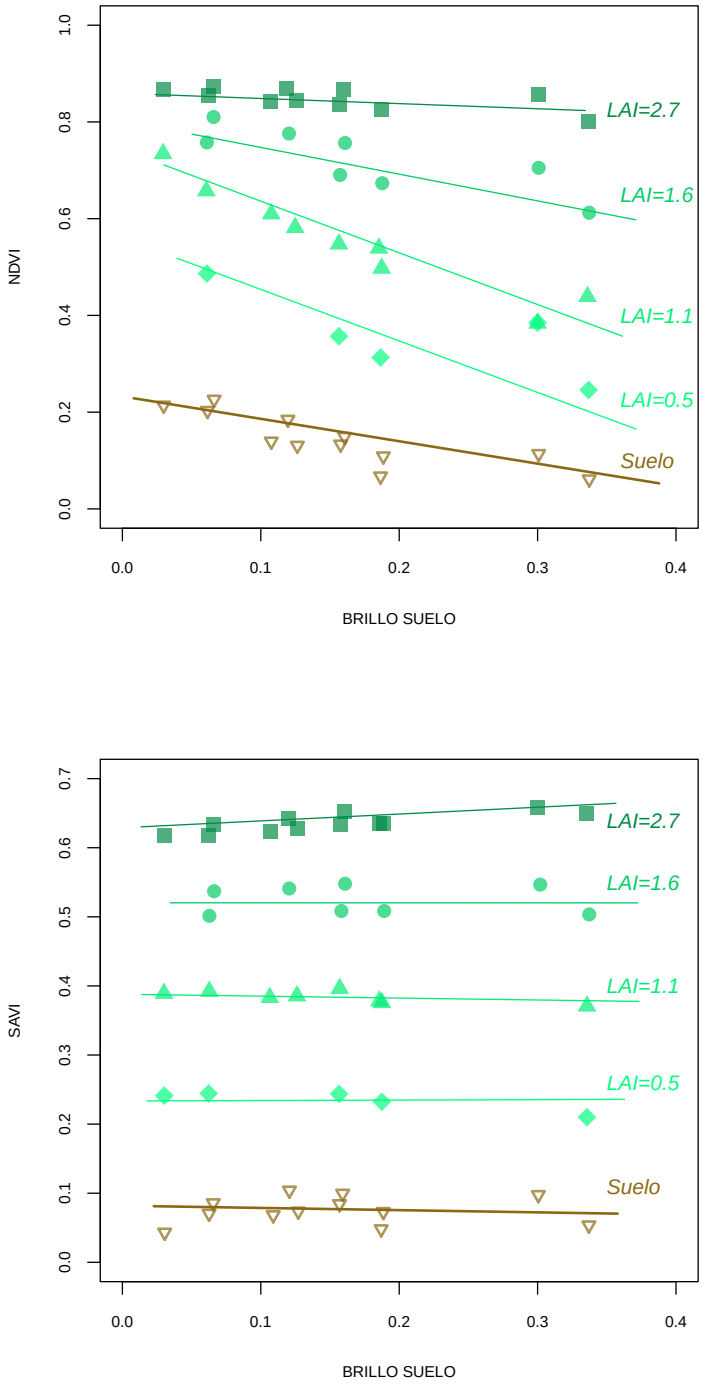


Figura 9.53: Influencia del brillo del suelo en el NDVI (superior) y SAVI (inferior). Adaptado de Huete et al., 1988.

```
savi <- spectralIndices(area_estudio, red = "R", nir = "NIR",
                        coefs = list(L = 0.5), indices = "SAVI")
```

9.16.2 SAVI Modificado o MSAVI

Variantes del índice SAVI son MSAVI y MSAVI2 (Qi et al. [1994]), se presenta en la figura 9.54.

MSAVI solo modifica la ecuación que calcula el parámetro L:

$$L = 1 - \frac{2s(\rho_{NIR} - \rho_{red})(\rho_{NIR} - s\rho_{red})}{(\rho_{NIR} + \rho_{red})} \tag{9.11}$$

Donde s es la pendiente de la recta *línea de suelo* en el gráfico R vs NIR (Figura 9.30).

```
msavi <- spectralIndices(area_estudio, red = "R", nir = "NIR",
                         indices = "MSAVI")
```

La expresión de la segunda modificación es:

$$MSAVI2 = \frac{2\rho_{NIR} + 1 - \sqrt{(2\rho_{NIR} + 1)^2 - 8(\rho_{NIR} - \rho_{red})}}{2} \tag{9.12}$$

```
msavi2 <- spectralIndices(area_estudio, red = "R", nir = "NIR",
                          indices = "MSAVI2")
```

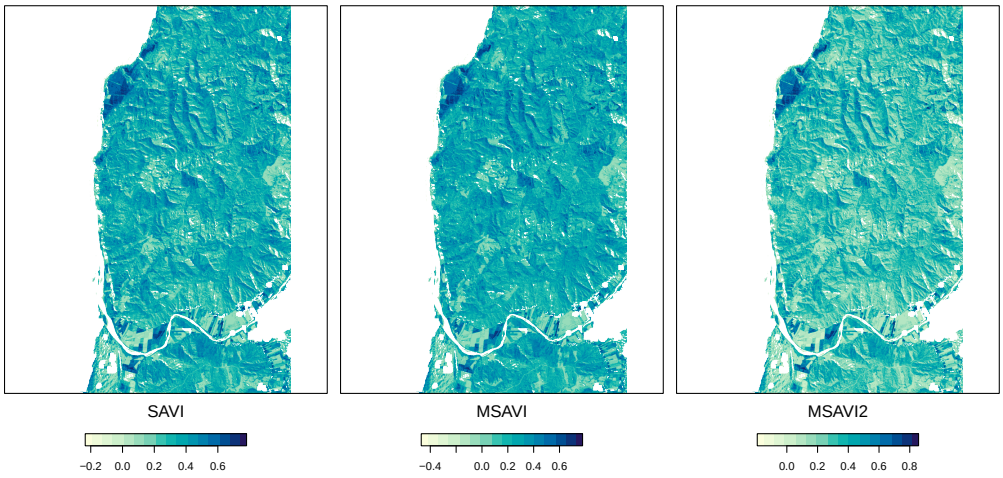


Figura 9.54: Índices SAVI y sus modificaciones. Sector Desembocadura Río Mataquito.

9.17 Reducción Efectos Atmosféricos

Los índices en esta categoría intentan minimizar los efectos atmosféricos que pueden variar dentro y entre imágenes.

9.17.1 Índices EVI

Los VI anteriores realizan ajustes para minimizar el efecto del suelo en el cálculo. Otro de los factores externos no relativos a la vegetación que afecta el cálculo es la *atmósfera*. La inclusión de un término empírico para la corrección atmosférica define el *Índice de Vegetación Mejorado (o EVI)* (Huete et al. [1999]), figura 9.55, izquierda. La banda azul en el EVI se utiliza principalmente para estabilizar las influencias de aerosoles en la banda roja resultante de la corrección residual y errónea de aerosoles.

$$EVI = G\left(\frac{\rho_{NIR} - \rho_{red}}{L + \rho_{NIR} + C_1\rho_{red} - C_2\rho_{blue}}\right) \tag{9.13}$$

Donde los parámetros empíricos son $G = 2.5$, $L = 1$, $C_1 = 6$, y $C_2 = 7.5$. L es el factor de ajuste para el fondo de dosel, C_1 y C_2 son los pesos de resistencia de aerosoles.

EVI se vuelve cada vez más sensible a la banda NIR en los doseles de moderada a altas cantidades de vegetación con una mayor profundidad óptica de penetración. Por lo tanto, EVI describe mejor las variaciones estructurales biofísicas del dosel y esta menos propenso a la saturación en áreas de alta biomasa.

A partir de 2000, y después del lanzamiento de los dos sensores MODIS en Terra y Aqua por la NASA (8.1.1), EVI fue adoptado como un producto estándar por la NASA y se hizo extremadamente popular entre los usuarios debido a su capacidad para eliminar los ruidos de fondo y de la atmósfera. , así como su falta de saturación, un problema típico de NDVI. EVI se distribuye actualmente de forma gratuita por el USGS LP DAAC [5].

[5] lpdaac.usgs.gov

```
evi <- spectralIndices(area_estudio, red= "R", nir= "NIR", blue= "B",
       indices= "EVI", coefs= list(G= 2.5, C1= 6, C2= 7.5, L_evi= 1))
```

9.17.2 EVI2 o EVI a dos bandas

Dos razones impulsan la búsqueda de un EVI de dos bandas (Jiang et al. [2008]):

- Extendiendo el EVI hacia atrás en el tiempo, usando el registro AVHRR. Los sensores AVHRR carecen de una banda azul, por lo que no es posible usar una versión EVI de tres bandas. Esto podría conducir potencialmente a un registro de EVI de 30 años que complementa el registro de NDVI.

- La banda azul siempre ha sido problemática y su relación señal/ruido es bastante pobre. Esto se debe principalmente a la naturaleza de la energía reflejada en esta parte del espectro terrestre, que es extremadamente baja.

$$EVI2 = G\left(\frac{\rho_{NIR} - \rho_{red}}{1 + \rho_{NIR} + 2.4\rho_{red}}\right) \tag{9.14}$$

Donde el parámetro empírico es $G = 2.5$. EVI2 tiene la mejor similitud con el EVI de 3 bandas, particularmente cuando los efectos atmosféricos son insignificantes y la calidad de los datos es buena. También se puede utilizar para sensores sin una banda azul, figura 9.55, derecha.

```
evi2 <- spectralIndices(area_estudio,
                        red = "R", nir = "NIR", indices = "EVI2",
                        coefs = list(G = 2.5, C1=2.4, L_evi=1))
```

Una de las aplicaciones más exitosas de EVI fue reportada por Alfredo Huete y sus colegas (Huete et al. [2006]) a principios de 2006. Por lo general, se considera que la selva amazónica tiene una temporada de crecimiento monótona, donde el crecimiento de la vegetación no tiene un patrón particular. Usando el producto MODIS EVI, Huete y sus colegas pudieron demostrar, *por primera vez*, que contrariamente a esa noción, la selva amazónica exhibe un crecimiento distinto durante la estación seca con serias implicaciones para nuestra comprensión actual del ciclo del carbono y los sumideros y, posteriormente, las cuestiones relacionadas con los gases de efecto invernadero y

el calentamiento global. Queda por ver si eso es el resultado del cambio climático o es un comportamiento normal (Wikipedia).

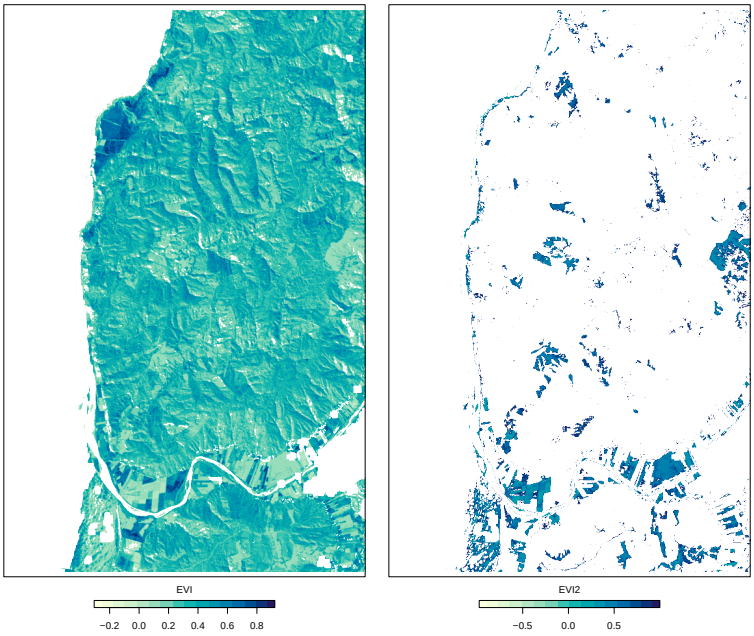


Figura 9.55: Índices EVI y EVI2. Sector Desembocadura Río Mataquito.

9.17.3 Índice de Vegetación Resistente a la Atmósfera, ARVI.

Índice de vegetación que no es relativamente propenso a los factores atmosféricos (como el aerosol). La fórmula del índice ARVI descrita por Kaufman y Tanré (Kaufman [1992]) es básicamente NDVI corregida para el efecto de dispersión atmosférica en el espectro de la reflectancia roja, usando medidas en longitudes de onda azules, figura 9.56, izquierda.

Fórmula del índice de vegetación ARVI:

$$ARVI = \frac{\rho_{NIR} - \rho_{red} - y(\rho_{red} - \rho_{blue})}{\rho_{NIR} + \rho_{red} - y(\rho_{red} - \rho_{blue})} \qquad (9.15)$$

Donde y es el cociente derivado de los componentes de la reflectancia atmosférica en los canales azul y rojo. Si no se conoce

el modelo de aerosoles para el momento de la toma, usar el valor $y = 1$ (Bannari et al. [1995]).

En comparación con otros índices, el índice de agricultura ARVI también es más robusto a los efectos topográficos, lo que lo convierte en una herramienta de monitoreo altamente efectiva para las regiones montañosas tropicales a menudo contaminadas por hollín proveniente de la agricultura de roza y quema. Cuando usarlo, en regiones con alto contenido de aerosoles atmosféricos (por ejemplo, lluvia, niebla, polvo, humo, contaminación del aire).

```
arvi_numerador <- (area_estudio$NIR - area_estudio$R) -
  1 * (area_estudio$R - area_estudio$B)

arvi_denominador <- (area_estudio$NIR + area_estudio$R) -
  1 * (area_estudio$R - area_estudio$B)

arvi <- arvi_numerador/arvi_denominador

arvi[arvi > 1] <- NA
arvi[arvi < -1] <- NA
```

9.17.4 ARVI2 Modificado.

El índice ARVI2 fue diseñado (Kaufman [1992]) para ser más resistente a los efectos atmosféricos y más sensible a una amplia gama de concentraciones de clorofila, figura 9.56, derecha.

$$ARVI2 = -0.18 + 1.17\left(\frac{\rho_{NIR} - \rho_{red}}{\rho_{NIR} + \rho_{red}}\right) \tag{9.16}$$

```
arvi2 <- -0.18 + 1.17 *
  ((area_estudio$NIR - area_estudio$R)/
     (area_estudio$NIR + area_estudio$R))

arvi2[arvi2 > 1] <- NA
arvi2[arvi2 < -1] <- NA
```

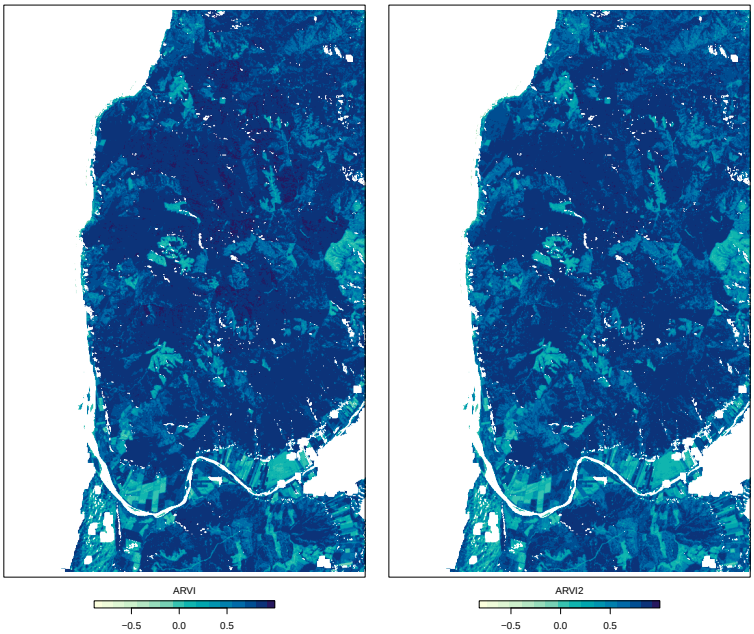


Figura 9.56: Índices ARVI y ARVI2. Sector Desembocadura Río Mataquito.

9.17.5 Índice de Vegetación Libre de Aerosoles AFRI

El Índice de Vegetación Libre de Aerosoles (Karnieli et al. [2001]) se basa en la capacidad de las bandas SWIR, es penetrar en la columna atmosférica incluso cuando existen aerosoles como humo o sulfatos (Figura 9.57). En consecuencia, estos índices tienen una aplicación importante en la evaluación de la vegetación en presencia de humo, contaminación antropogénica o columnas volcánicas. Bajo condiciones de cielo claro se comporta de manera muy similar a NDVI.

```
NIR <- area_estudio$NIR
SWIR1 <- area_estudio$SWIR1
afri <- NIR - 0.66 *  (SWIR1 / (NIR + 0.66 * SWIR1))
afri[afri < -1] <- NA
```

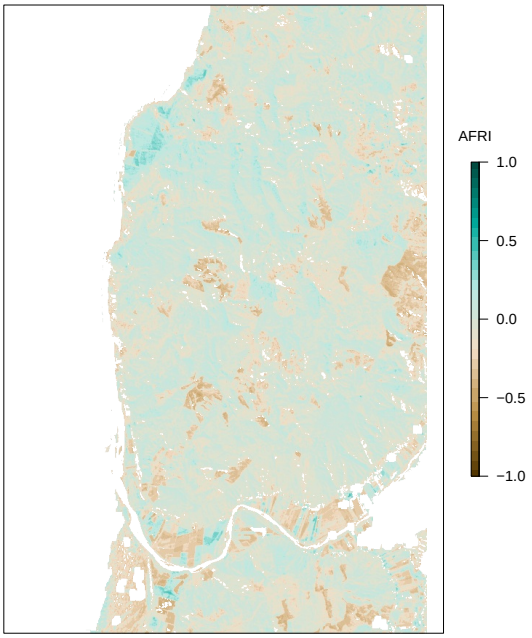


Figura 9.57: Índice AFRI. Sector Desembocadura Río Mataquito.

9.18 Reducción Efectos del Suelo y Atmosféricos

9.18.1 Índice Global de Monitoreo Ambiental GEMI

GEMI (Pinty and Verstraete [1992]) es un índice no lineal que aplica una corrección atmosférica estandarizada a los valores de infrarrojo cercano y rojo que tienen que ser convertidos a reflectancia (Figura 9.58). Los estudios comparativos han demostrado que es muy sensible a las variaciones del suelo de fondo en áreas de baja cobertura vegetal pero menos sensible a los efectos atmosféricos.

$$GEMI = eta(1 - 0.25\ eta) - \frac{\rho_{red} - 0.125}{1 - \rho_{red}} \tag{9.17}$$

$$eta = \frac{2(\rho_{NIR}^2 - \rho_{red}^2) + 1.5\ \rho_{NIR} + 0.5\ \rho_{red}}{\rho_{NIR} + \rho_{red} + 0.5} \tag{9.18}$$

```
gemi <- spectralIndices(area_estudio, red = "R", nir = "NIR",
                        indices = "GEMI")
```

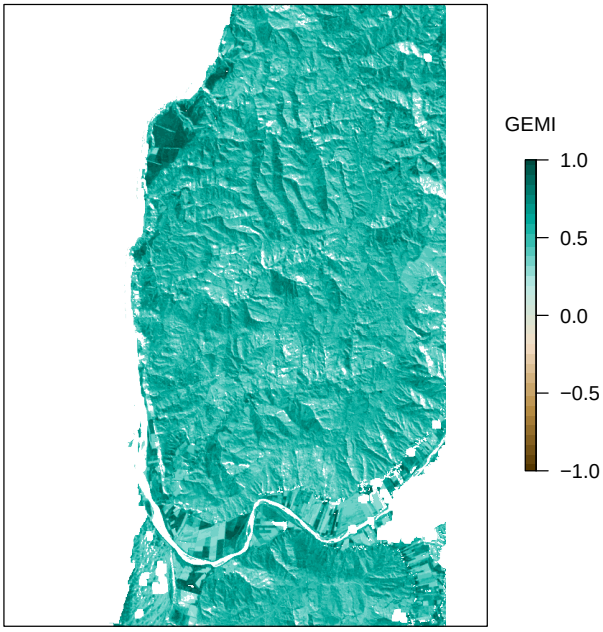


Figura 9.58: Índice GEMI. Sector Desembocadura Río Mataquito.

9.18.2 Comparativa VI

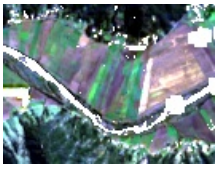


Figura 9.59: Río Mataquito, RGB.

Usando la función interactiva *drawExtent* seleccionamos una nueva área de interés centrada en las zonas de cultivo entorno al río Mataquito (Figura 9.59) y exploramos en detalle el funcionamiento de los distintos VI desarrollados (Figuras 9.60).

```
detalle <- drawExtent()
```

9.19 Índice Relativos a Incendios Forestales

Los graves incendios de inicios de agosto del 2021, en la Provincia de Muğla, cerca de la localidad de Bayir en Grecia, son el lamentable marco para realizar el estudio de los índices relativos al fuego y zonas quemadas. Se ha descargado una imagen *Landsat 8*, del 8 de agosto del 2021 con incendio activo y una imagen del 20 de julio de 2021 pre-incendio para efectos de comparación.

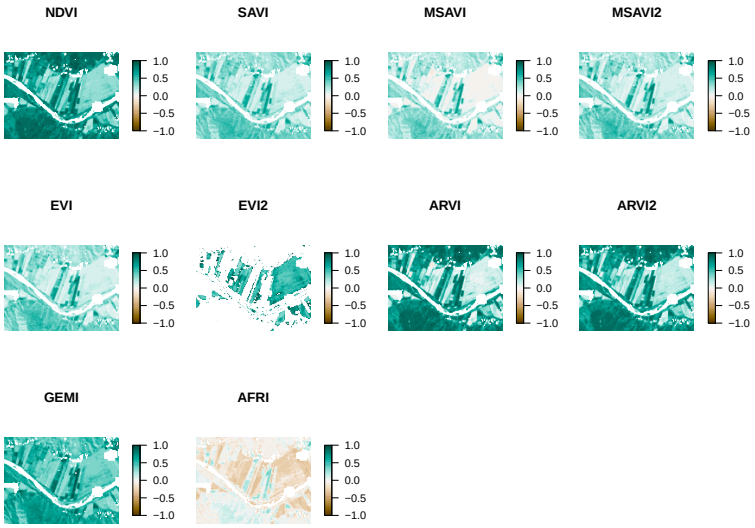


Figura 9.60: Índice Vegetales. Sector Desembocadura Río Mataquito.

9.19.1 Extracción de los Datos

Una vez descargadas las imágenes de la zona geográfica y fechas correspondientes (utilizando las instrucciones descritas en 8.10), procesamos en R para extraer las imágenes tif para componer nuestro rasterBrick (adaptados de 8.11).

1. Comprobamos nuestra carpeta de data cruda */raw/sats/* nuestras escenas descargadas en formato *tar* para comprobar la posición en el vector de aquellas que vamos a extraer en esta sesión.

```
tar_ <- dir(path = here("raw","sats"), pattern ="tar")
```

2. Construimos una serie de objetos para almacenar datos intermedios que vamos a utilizar. Vector *carpeta* con el nombre de la carpeta donde descomprimir el contenido de los archivos tar. Vector *origen* con la ruta y nombre completo de los archivos tar. Vector *destino* ruta de la carpeta donde extraeremos los archivos.

```
carpeta_ <- sub(".tar","", tar_)
origen_ <- paste(here("raw","sats"), tar_, sep = "/")
destino_ <- paste(here("raw","sats"), carpeta_, sep = "/")
```

3. Del punto *1* identificamos los índices de las imágenes a procesar (en mi caso 2 y 3) y usamos una estructura *for* para repetir el proceso de *untar* los archivos desde *origen* a *destino*.

```
for (i in 2:3){
  untar(tarfile = origen_[i], exdir = destino_[i])
}
```

4. Revisamos el contenido final de la carpeta destino_[3] usando el patrón "TIF".

```
tifs_pre <- dir(path = destino_[2], pattern ="TIF")
tifs_fire <- dir(path = destino_[3], pattern ="TIF")
```

Finalmente revisamos las imágenes que corresponden a las bandas que vamos utilizar en los ejercicios siguiente, un set pre-incendio forestal *tifs_pre* y un set durante el incendio *tifs_fire* :

```
tifs_pre[c(3:9,12)]
tifs_fire[c(3:9,12)]
```

En los siguientes pasos construimos un rasterBrick con las bandas y nombres organizados para facilitar los pasos siguientes.

```
nombres <- c("AC","B","G","R","NIR","SWIR1","SWIR2","TERMAL")
```

Extraer bandas, construir un rasterStack y asignar nombres para la escena pre incendio:

```
bandas_pre <- file.path(destino_[2], tifs_pre[c(3:9,12)])
imagen_pre <- stack(bandas_pre)
names(imagen_pre) <- nombres
```

Y del incendio:

```
bandas_fire <- file.path(destino_[3], tifs_fire[c(3:9,12)])
imagen_fire <- stack(bandas_fire)
names(imagen_fire) <- nombres
```

Para cada una, convertimos de DN o valores digitales a reflectancia (ρ) utilizando las constantes definidas en el cuadro 8.2 para las imágenes OLI y TIRS. Ver figura 9.61 para una composición RGB de ambas imágenes procesadas.

Figura 9.61: Imagen Landsat 8 OLI/TIRS, Región de Provincia de Muğla, Turquía. Escena pre incendio del 20 de junio del 2021 (superior) y en pleno incendio en 5 agosto 2021 (inferior). Composición de Bandas RGB.

```
imagen_pre_fire <- (imagen_pre[[1:7]] * 0.0000275) + -0.2
names(imagen_pre_fire) <- nombres[1:7]
imagen_pre_fire$TERMAL  <- (imagen_pre[[8]] * 0.00341802 + 149)

imagen_in_fire <- (imagen_fire[[1:7]] * 0.0000275) + -0.2
names(imagen_in_fire) <- nombres[1:7]
imagen_in_fire$TERMAL  <- (imagen_fire[[8]] * 0.00341802 + 149)
```

Y utilizando una sesión interactiva se define un área más ajustada entorno al incendio forestal:

```
ext <- drawExtent()
```

Y recortamos nuestra imagen para trabajar en la zona de interés.

```
area_incendio_pre <- crop(x = imagen_pre_fire, y = ext)
area_incendio <- crop(x = imagen_in_fire, y = ext)
```

Un interesante ejercicio es comparar combinaciones de banda para conocer el comportamiento de un fuego activo y la pluma de humo (Figura 9.62).

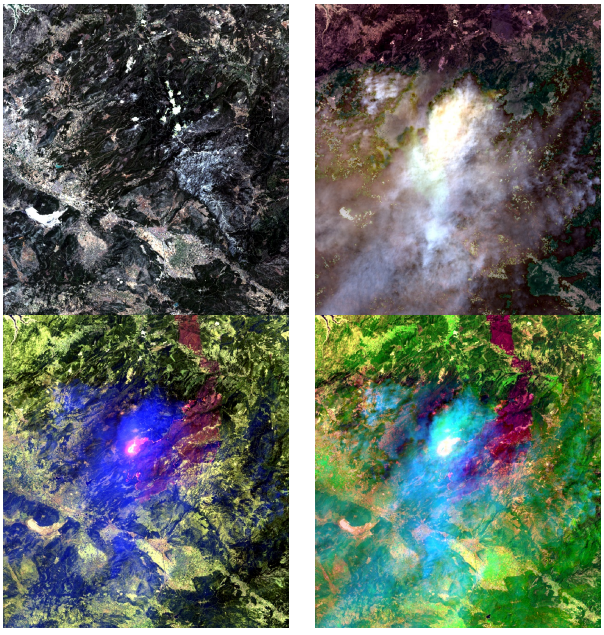


Figura 9.62: Composición de Bandas, Color verdadero RGB Pre incendio (izquierda, arriba), Color verdadero RGB en incendio (derecha, arriba), SWIR2/SWIR1/R (izquierda, abajo) y SWIR2/NIR/B (derecha, abajo).

9.19.2 Índice Área Quemada BAI

Este índice resalta la tierra quemada en el espectro del rojo al infrarrojo cercano, al enfatizar la señal de carbón en las imágenes posteriores al incendio. El índice se calcula a partir de la distancia espectral de cada píxel a un punto espectral de referencia, donde convergen las áreas quemadas recientemente (Chuvievo [1998]). Figura 9.63.

$$BAI = \frac{1}{(0.1 - \rho_{red})^2 + (0.06 - \rho_{NIR})^2} \tag{9.19}$$

```
bai <- 1 / ((0.1 - area_incendio$R)^2 + (0.06- area_incendio$NIR)^2)
bai[bai > 1000] <- NA
```

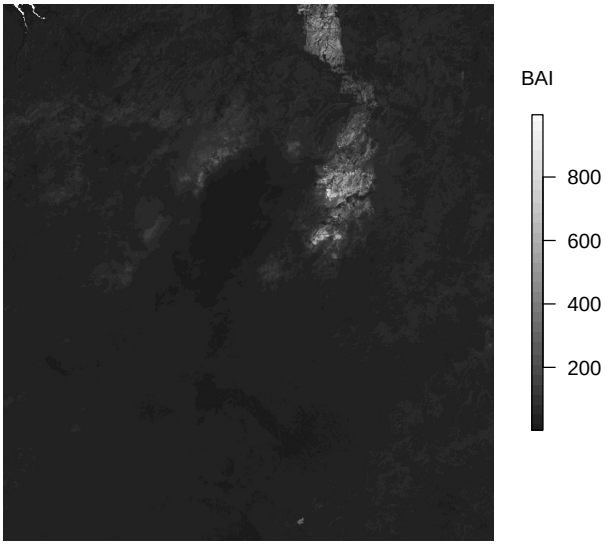


Figura 9.63: Índice BAI. Sector Desembocadura Río Mataquito.

9.19.3 Índice Razón Normalizado Quemado NBRI

Este índice destaca las áreas quemadas en zonas de incendios. La fórmula es similar a NDVI, excepto que utiliza longitudes de onda de infrarrojo cercano (NIR) e infrarrojo de onda corta (SWIR) (Key et al. [2002]). Figura 9.64.

$$NBRI = \frac{\rho_{NIR} - \rho_{SWIR}}{\rho_{NIR} + \rho_{SWIR}} \tag{9.20}$$

```
nbri <- spectralIndices(area_incendio,
                        nir ="NIR",
                        swir3="SWIR2",
                        indices = "NBRI")
```

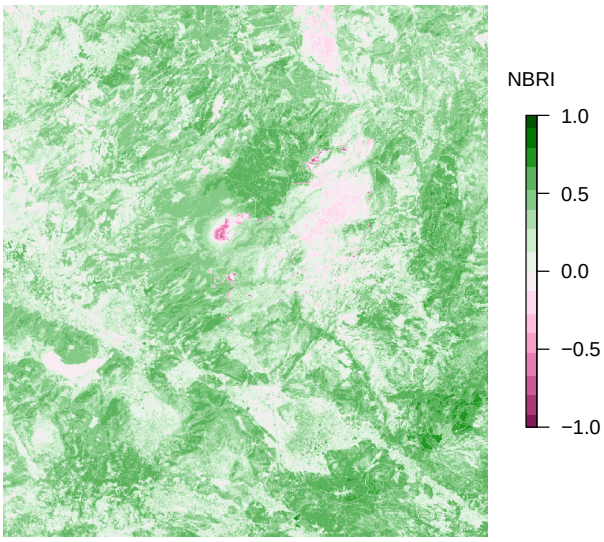


Figura 9.64: Índice NBRI. Sector Desembocadura Río Mataquito.

9.19.4 NBRI Diferenciado, DNBRI

El programa del servicio geológico de EE.UU. (USGS) FIREMON (Key and Benson [2005]) publica una categorización de la severidad de quemaduras (Cuadro 9.4) de un incendio forestal basado en el NBRI *Diferenciado*:

$$\Delta NBRI = NBRI_{pre} - NBRI_{post} \tag{9.21}$$

Donde $NBRI_{pre}$ es el índice calculado en la misma zona geográfica **previa ocurrencia** del incendio y $NBRI_{post}$ el índice calculado **después** del incendio.

Calculamos el índice NBRI para la imagen pre-incendio:

```
nbri_pre <- spectralIndices(area_incendio_pre, nir ="NIR",
                            swir3="SWIR2", indices = "NBRI")
```

Y calculamos la diferencia:

Cuadro 9.4: Categorías de gravedad de quemaduras según proyecto FIREMON (USGS).

Límite Inferior	Límite Superior	Gravedad Quemadura
-1	-0.25	Alto rebrote post fuego
-0.25	-0.1	Bajo rebrote post fuego
-0.1	1	Sin quemar
0.1	0.27	Baja
0.27	0.44	Moderada a baja
0.44	0.66	Moderada-alta
0.66	1	Alta

```
dnbri <- nbri_pre - nbri
```

Y generamos el mapa de Severidad mediante la diferencia de NBRI (Figura 9.65) utilizando la técnica de reclasificar y luego convertir a factores para un posterior tratamiento estadístico.

1. Creamos una matriz con rangos y nuevos valores:

2. Aplicamos la reclasificación mediante el uso de la función `reclassify(x, rcl)` donde el ráster `x` (re)clasifica grupos de valores por otros valores definidos en una matriz `rcl`.

3. Se convierte en *factores*:

```
categoria <- ratify(firemon, count= T, ordered=T)
```

4. Vector con un texto descripción por cada rango.

```
clases <- c("Alto rebrote post fuego","Bajo rebrote post fuego",
            "Sin quemar","Quemadura baja gravedad",
            "Quemadura de gravedad moderada a baja",
            "Quemadura de gravedad moderada a alta",
            "Quemadura de alta gravedad")
```

5. Asociamos el vector como atributo a la tabla del ráster:

```
levels(categoria)[[1]] <- cbind(levels(categoria)[[1]],
                                clase=clases)
```

Así, podemos revisar la frecuencia por categoría:

```
levels(categoria)
```

```
[[1]]
  ID   COUNT                                clase
1  1 1042319               Alto rebrote post fuego
2  2  384064               Bajo rebrote post fuego
3  3  251916                           Sin quemar
4  4   21972               Quemadura baja gravedad
5  5    1202 Quemadura de gravedad moderada a baja
6  6    1471 Quemadura de gravedad moderada a alta
7  7     447            Quemadura de alta gravedad
```

Y consultar por coordenadas (previa conversión a número de celda):

```
id <- cellFromXY(categoria,c(617397, 4132006))
```

Como *ids* de celdas *(300, 500, 4)* usando la función `factorValues(x, v)`, donde `x` corresponde a un rasterLayer y `v` un vector de identificación de celdas:

```
factorValues(x = categoria,
             v = categoria[c(id, 300, 500, 4)])
```

```
    COUNT                                clase
1    1471 Quemadura de gravedad moderada a alta
2  384064               Bajo rebrote post fuego
3 1042319               Alto rebrote post fuego
4  251916                           Sin quemar
```

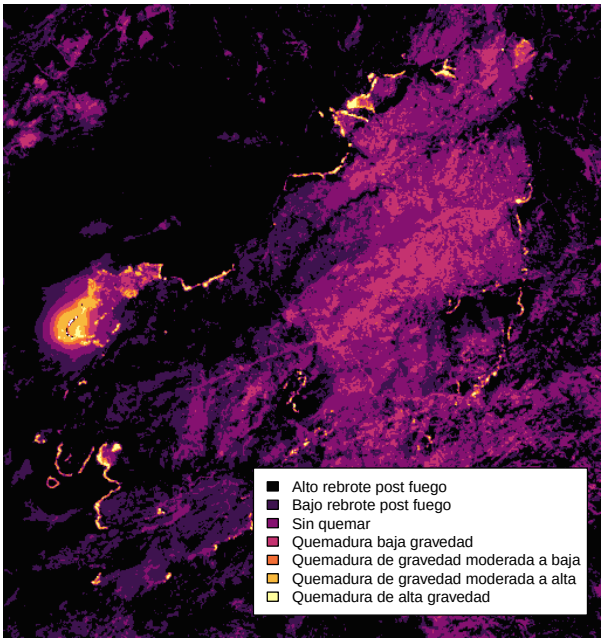


Figura 9.65: SEVERIDAD, medido vía Diferencia de NBRI pre y post incendio de Muğla, Turquía.

9.19.5 Índice Razón Normalizado Quemado Termal 1 NBRT1

Este índice utiliza una banda térmica para mejorar el NBR (Holden et al. [2005]). Da como resultado una mejor separabilidad entre tierra quemada y no quemada (Figura 9.66).

$$NBRT1 = \frac{\rho_{NIR} - \rho_{SWIR}\left(\frac{termal}{1000}\right)}{\rho_{NIR} + \rho_{SWIR}\left(\frac{termal}{1000}\right)} \tag{9.22}$$

La banda térmica en esta ecuación debe calibrarse a las temperaturas de brillo (en Kelvins).

```
nir <- area_incendio$NIR
swir <- area_incendio$SWIR2
termal <- area_incendio$TERMAL
nbrt1 <- (nir-swir*(termal/1000))/(nir+swir*(termal/1000))
```

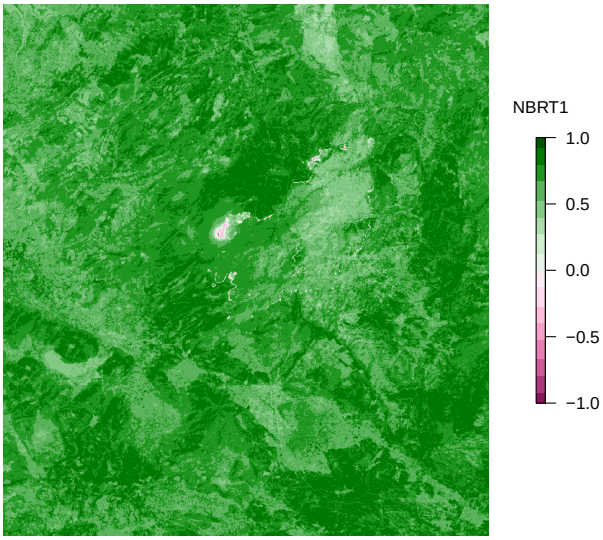


Figura 9.66: Índice NBRT1. Sector Desembocadura Río Mataquito.

9.20 Índices Diseñados para Agua

9.20.1 Preparación Imagen de prueba

Descargamos una imagen Landsat 8, centrada en el lago Cardiel, Provincia de Santa Cruz, Argentina. La actividad más importante que se desarrolla en sus aguas, es la pesca comercial. En la década del 40 un avión que transportaba alevinos con destino final en el lago Fagnano (Tierra del Fuego), se vio sorprendido por una tormenta; la demora en el vuelo llevó al piloto a tomar una decisión: liberarlos en el primer espejo de agua que apareciera antes que las crías murieran, el lago Cardiel se convirtió en su nuevo hogar.

Al contener una alta composición iónica, no son aptas para el consumo humano (wikipedia).

```
tifs_agua <- dir(path = destino_[4], pattern ="TIF")
```

Extraer bandas, construir un rasterStack y asignar nombres para la escena pre-incendio:

```
bandas_agua <- file.path(destino_[4], tifs_agua[c(3:9,12)])
imagen_test <- stack(bandas_agua)
nombres <- c("AC","B","G","R","NIR","SWIR1","SWIR2","TERMAL")
```

```
names(imagen_test) <- nombres
```

Para cada una, convertimos de DN o valores digitales a reflectancia (ρ) utilizando las constantes definidas en el cuadro 8.2 para las imágenes OLI y TIRS. Ver figura 9.67 para una composición *batimétrica* (R/G/AC).

```
imagen_agua <- (imagen_test[[1:7]] * 0.0000275) + -0.2
names(imagen_agua) <- nombres[1:7]
imagen_agua$TERMAL  <- (imagen_test[[8]] * 0.00341802 + 149)
```

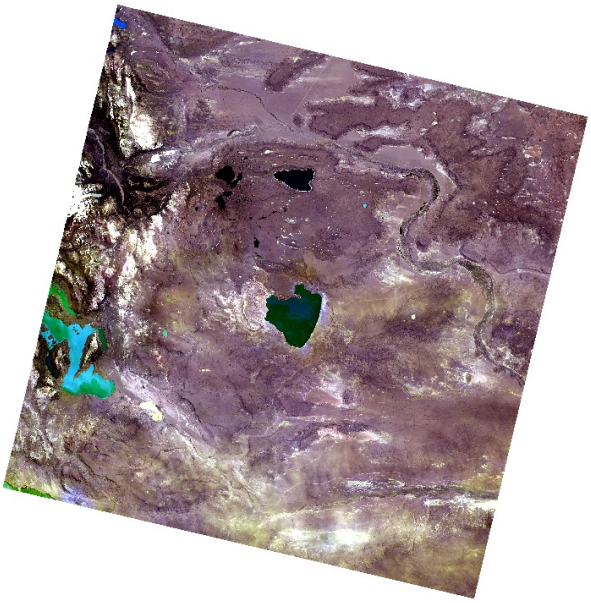


Figura 9.67: Imagen Landsat 8 OLI/TIRS, Sector Lago Cardiel, Provincia de Santa Cruz, Argentina. Composición 'BATIMÉTRICA' (R-G-AC).

9.20.2 NDWI

Como el índice de vegetación de diferencia normalizada ($NDVI$) (Figura 9.68) que se ha utilizado ampliamente para identificar vegetación, $NDWI$ es un índice para detectar cuerpos de agua empleando bandas *green* (verde) y banda NIR (McFeeters [1996]). La expresión del índice es:

$$NDWI = \frac{\rho_{green} - \rho_{NIR}}{\rho_{green} + \rho_{NIR}} \tag{9.23}$$

El rango del valor $NDWI$ es de -1 a 1. Características del agua es probable que tengan valores positivos, mientras que la vegetación terrestre y el suelo desnudo generalmente tienen valores negativos. Por lo tanto, 0 se considera a menudo un umbral predeterminado para el $NDWI$.

```
ndwi <- spectralIndices(imagen_agua, nir = "NIR",
                        green = "G", indices = "NDWI")
```

Y la modificación Gao [1996] (Figura 9.69):

$$NDWI2 = \frac{\rho_{NIR} - \rho_{SWIR2}}{\rho_{NIR} + \rho_{SWIR2}} \tag{9.24}$$

```
ndwi2 <- spectralIndices(imagen_agua, nir = "NIR",
                         swir2 = "SWIR2", indices = "NDWI2")
```

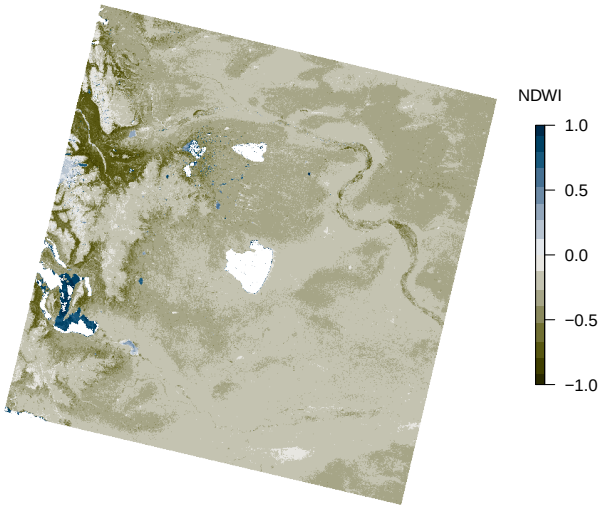


Figura 9.68: Índice NDWI, Sector Lago Cardiel, Provincia de Santa Cruz, Argentina.

9.20.3 MNDWI

El Índice de agua de diferencia normalizada modificado o MNDWI (Xu [2006]) (Figura 9.70), utiliza las bandas verde y SWIR para realzar las entidades que se encuentran en mar abierto. También disminuye las entidades de área construidas que a menudo se correlacionan con el mar abierto en otros índices.

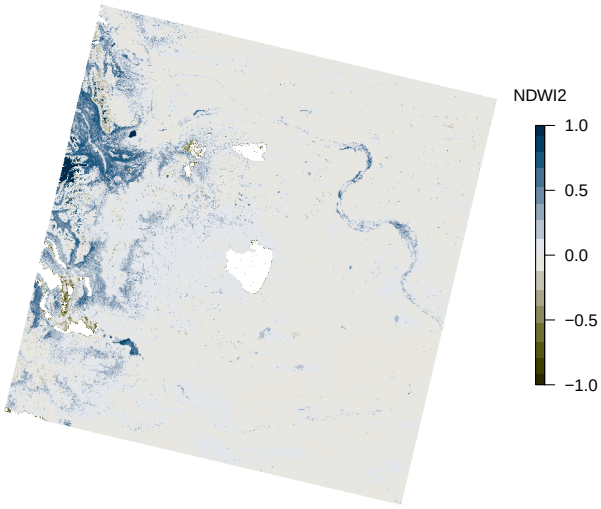


Figura 9.69: Índice NDWI2, Sector Lago Cardiel, Provincia de Santa Cruz, Argentina.

$$MNDWI = \frac{\rho_{green} - \rho_{SWIR}}{\rho_{green} + \rho_{SWIR}} \tag{9.25}$$

```
mndwi <- spectralIndices(imagen_agua, swir2 = "SWIR2",
                         green = "G", indices = "MNDWI")
```

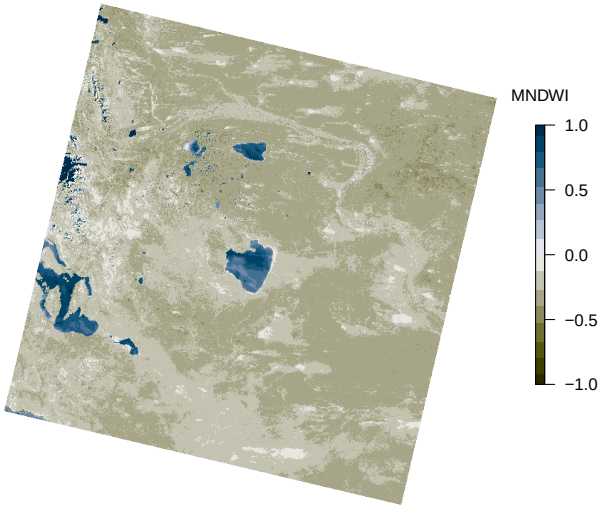


Figura 9.70: Índice MNDWI, Sector Lago Cardiel, Provincia de Santa Cruz, Argentina.

9.20.4 EWI

Una modificación de MNDWI (Xu [2006]) (ver 9.20.3) introduciendo $(NDVI + n)$ en el denominador y m en el nominador para *resaltar* información de agua, *comprimir* la información de fondo y *reducir* la diferencia de información de fondo entre *suelo* y *vegetación* es presentado en Pei et al. [2007].

Considerando que el valor NDVI de un píxel de agua pura será cero o negativo, se toma el parámetro $n = 0.5$ como valor empírico para prevenir cualquier resultado anormal y reducir la diferencia de EWI entre suelo y vegetación, El término del numerador $m = 0.1$ (ecuación 9.26) se obtuvo a través de un gran número de observaciones experimentales.

$$EWI = \frac{\rho_{green} - \rho_{SWIR} + m}{(\rho_{green} + \rho_{SWIR})(NDVI + n)} \tag{9.26}$$

Necesitamos calcular NDVI:

```
ndvi <- spectralIndices(imagen_agua, red="R", nir="NIR", indices = "NDVI")
```

Definir los parámetros m y n:

```
m <- 0.1; n <- 0.5
```

Y aplicamos la formula para el cálculo del índice EWI (Figura 9.71):

```
ewi_1 <- (imagen_agua$G - imagen_agua$SWIR1 + m)
ewi_2 <- (imagen_agua$G + imagen_agua$SWIR1)*(ndvi + n)
ewi <- ewi_1 / ewi_2
```

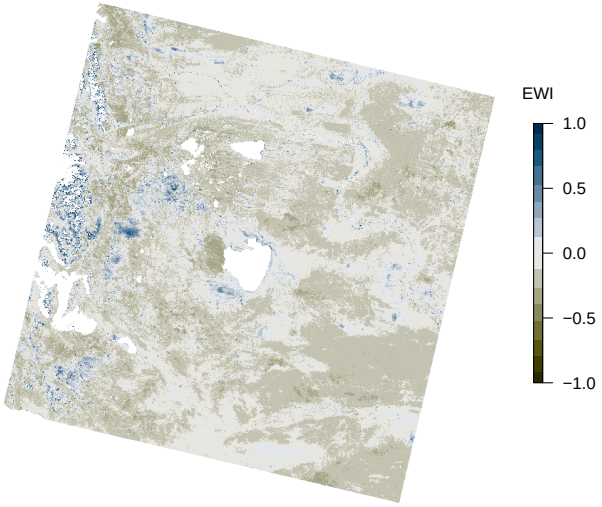


Figura 9.71: Índice EWI. Sector Lago Cardiel, Provincia de Santa Cruz, Argentina.

9.20.5 AWEI

$AWEI_{ns}$ (Feyisa et al. [2014]) *no sombra*, (Figura 9.72) es un índice formulado para eliminar de forma eficaz los píxeles *no agua*, incluidas las superficies oscuras construidas en áreas con fondo urbano y $AWEI_s$ *sombre* (Figura 9.73), está formulado principalmente para mejorar aún más la precisión eliminando los píxeles de sombra que $AWEI_{ns}$ no puede eliminar de forma eficaz.

El subíndice ns en la ecuación 9.27 se incluye para especificar que el índice es adecuado para situaciones en las que las sombras *no son un problema importante*. El subíndice s en la ecuación 9.28 indica que la ecuación está destinada a eliminar los píxeles de sombra y mejorar la precisión de extracción de agua en áreas con sombra y / u otras superficies oscuras. Pero en áreas con superficies muy reflectantes como hielo, nieve y techos reflectantes en áreas urbanas, la ecuación 9.28 *puede clasificar erróneamente superficies como el agua.*

$$AWEI_{ns} = 4(\rho_{green} - \rho_{SWIR}) - 0.25\rho_{NIR} + 2.75\rho_{SWIR} \quad (9.27)$$

$$AWEI_s = \rho_{blue} + 2.5\rho_{green} - 1.5(\rho_{NIR} + \rho_{SWIR}) - 0.25\rho_{SWIR} \tag{9.28}$$

```
awei_ns_1 <- 4 * (imagen_agua$G - imagen_agua$SWIR1)
awei_ns_2 <- 0.25*(imagen_agua$NIR + 2.75 * imagen_agua$SWIR1)
awei_ns <- awei_ns_1 - awei_ns_2

awei_s_1 <- imagen_agua$B +  2.5 * imagen_agua$G
awei_s_2 <- 1.5 * (imagen_agua$NIR + imagen_agua$SWIR1) -
  0.25 * imagen_agua$SWIR1
awei_s <- awei_ns_1 - awei_ns_2
```

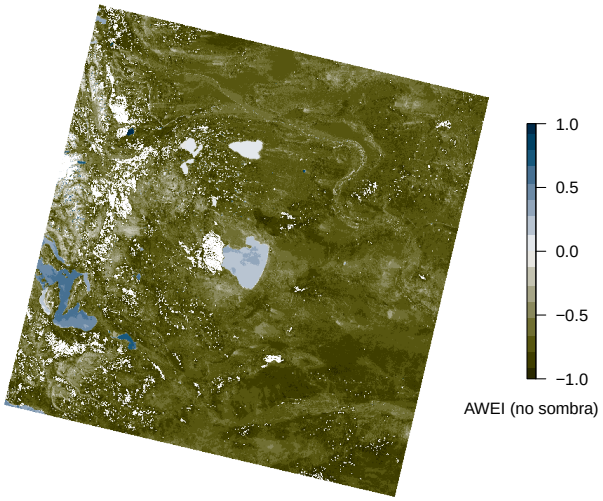


Figura 9.72: Índice AWEI, no sombra. Sector Lago Cardiel, Provincia de Santa Cruz, Argentina,

9.20.6 NDWI ponderado, WNDWI

Tanto $NDWI$ como $MNDWI$ pueden separar bien la vegetación del agua porque tanto los valores espectrales de vegetación en las bandas NIR y $SWIR$ son mayores que los de la banda verde.

Sin embargo, para un píxel de agua turbia, el valor de la banda NIR suele ser mayor que la banda *green* (verde), mientras que la banda $SWIR$ es más pequeña que la banda verde; para el píxel de vegetación en un área de sombra, la banda $SWIR$ es en su

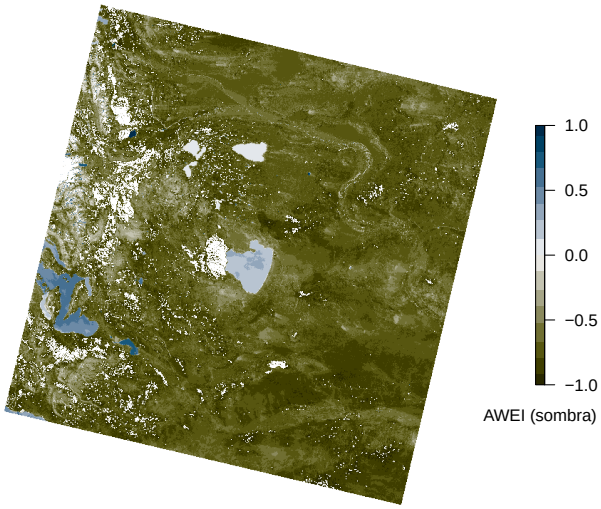


Figura 9.73: Índice AWEI, sombra. Sector Lago Cardiel, Provincia de Santa Cruz, Argentina,

mayoría más pequeña que la banda verde, mientras que la banda NIR es generalmente más grande que la banda verde.

Si calculamos los índices de agua $NDWI$ y $MNDWI$ y recortamos en el valor 0, NDWI no puede clasificar correctamente el *agua turbia* en la clase de agua y MNDWI también clasificará erróneamente *la vegetación en el área de sombra* como agua.

Con el fin de reducir los errores, se propone un nuevo índice de agua llamado $WNDWI$ que usa un valor medio de NIR y SWIR para sustituir la banda NIR o SWIR. La expresión de WNDWI es el siguiente (Guo et al. [2017]), figura 9.74:

$$WNDWI = \frac{\rho_{green} - \alpha\rho_{NIR} - (1-\alpha)\rho_{SWIR}}{\rho_{green} + \alpha\rho_{NIR} + (1-\alpha)\rho_{SWIR}} \tag{9.29}$$

Donde $\alpha \in [0, 1]$ es un coeficiente de ponderación. Específicamente, si $alpha = 1$ el índice $WNDWI$ se convierte en $NDWI$ y cuando $\alpha = 0$ el $WNDWI$ es idéntico con $MNDWI$.

```
a <- 0.1
wndwi_1 <- imagen_agua$G -
  a * imagen_agua$NIR -
  (1 - a) * imagen_agua$SWIR2
```

```
wndwi_2 <- imagen_agua$G +
  a * imagen_agua$NIR +
  (1 - a) * imagen_agua$SWIR2

wndwi <- wndwi_1 / wndwi_2
```

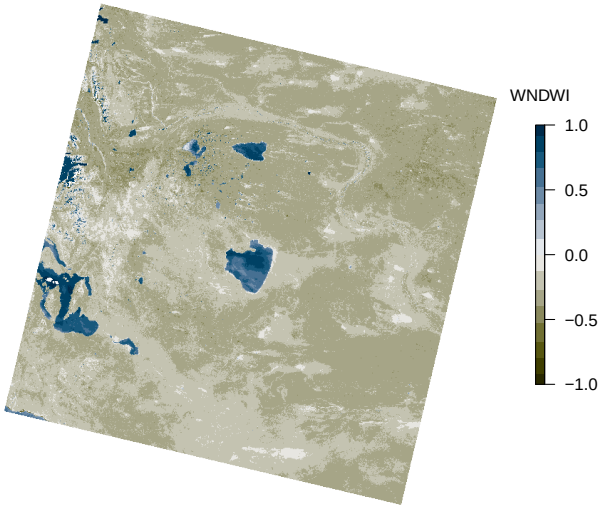


Figura 9.74: Índice WNDWI parámetro alpha de 0.1, Sector Lago Cardiel, Provincia de Santa Cruz, Argentina.

Algal Bloom

Figura 9.75: Tweet del 2 de febrero de 2019, donde se alerta de un brote de cianobacterias en las costas de Uruguay.

9.21 Algal Bloom 2019 Montevideo

En un tweet publicado el 2 de febrero de 2019 (Figura 9.75) se informa de la presencia anómala de **cianobacterias** en la costa frente a Montevideo (fenómeno denominado *algal bloom*).

Usando los pasos descritos en 8.15 hemos descargado dos imágenes Sentinel-2 que cubren la zona costera frente a la ciudad de Montevideo (Uruguay) en las fechas 31 de enero 2019 y 16 mayo de 2019 (fechas en el brote y postbrote).

Las cianobacterias (Cyanobacteria del griego cyano, "azul"), es un filo del dominio Bacteria que comprende las bacterias capaces de realizar fotosíntesis oxigénica. Son los únicos procariontes que llevan a cabo ese tipo de fotosíntesis, por ello también se les llama oxifotobacterias. Algunas cianobacterias producen toxinas y pueden envenenar a los animales que habitan el mismo ambiente o beben el agua. Se trata de una gran variedad de géneros y especies; algunas producen toxinas muy específicas y otras producen un espectro más o menos amplio de tóxicos. El fenómeno se hace importante sólo cuando hay una floración (una explosión demográfica), lo que ocurre a veces en aguas dulces o salobres, si las condiciones de temperatura son favorables y abundan los nutrientes, sobre todo el fósforo (Wikipedia).

Primero creamos los objetos R, con nombres de archivos *zip* y construimos los nombres de carpeta abreviados *nombre*.

```
zipes <- dir(path = here("raw","sen"),
             pattern =".zip$")
carpetas <- sub(".zip","", zipes)
```

```
(nombre <- substr(x = carpetas, 1, 26))
```

```
[1] "S2A_MSIL2A_20190131T134211" "S2B_MSIL2A_20190516T134219"
[3] "S2B_MSIL2A_20210905T152639"
```

Se descomprimen las imágenes correspondientes al estudio (atento, sus índices no necesariamente sean las mismas o en el mismo orden). En mi máquina corresponden a los índices 1 y 2 del arreglo *nombre*.

La escena para el momento de la explosión de algas es el índice **1** y repetimos para la fecha post fenómeno **2**:

```
zip::unzip(zipfile = here("raw","sen", zipes[1]),
           exdir = here("raw","sen",nombre[1]),
           junkpaths=T)
```

De cada una se extraen las imágenes jp2:

```
jps1 <- list.files(here("raw","sen",nombre[1]),
                   pattern = ".jp2$",
                   recursive = T,
                   full.names = T)

jps_sin <- list.files(here("raw","sen",nombre[2]),
                      pattern = ".jp2$",
                      recursive = T,
                      full.names = T)
```

9.21.1 Crear rasterStack 31 de enero

Pasos para crear una escena Sentinel-2, BOA:

1. Seleccionar resolución 10 m:

```
id1 <- str_detect(jps1, "_10m")
sen2_10m_img <- jps1[id1]
basename(sen2_10m_img[-6]) #sin TCI
```

```
[1] "T21HWB_20190131T134211_AOT_10m.jp2"
[2] "T21HWB_20190131T134211_B02_10m.jp2"
[3] "T21HWB_20190131T134211_B03_10m.jp2"
[4] "T21HWB_20190131T134211_B04_10m.jp2"
```

```
[5] "T21HWB_20190131T134211_B08_10m.jp2"
[6] "T21HWB_20190131T134211_WVP_10m.jp2"
```

```
nombres_10m <- c("AOT","B2","B3","B4","B8","WVP")
```

2. Crear stack de layer o capas y se asignan los nombres de capas para fecha uno:

```
sen2_10m <- stack(sen2_10m_img[-6])
names(sen2_10m) <- nombres_10m
```

9.21.2 Crear rasterStack 16 de mayo

Procesar la fecha post fenómeno:

1. Selección de resolución de 10 m:

```
id_sin <- str_detect(jps_sin, "_10m")
sen2_10m_img_sin <- jps_sin[id_sin]
basename(sen2_10m_img_sin[-6]) #sin TCI
```

```
[1] "T21HWB_20190516T134219_AOT_10m.jp2"
[2] "T21HWB_20190516T134219_B02_10m.jp2"
[3] "T21HWB_20190516T134219_B03_10m.jp2"
[4] "T21HWB_20190516T134219_B04_10m.jp2"
[5] "T21HWB_20190516T134219_B08_10m.jp2"
[6] "T21HWB_20190516T134219_WVP_10m.jp2"
```

2. Crear stack y renombre de capas:

```
sen2_10m_sin <- stack(sen2_10m_img_sin[-6])
names(sen2_10m_sin) <- nombres_10m
```

9.21.3 Procesamiento de las escenas

Se define un objeto *extent*[6] para realizar una vista de detalle en la costa frente a Montevideo:

```
ext_mon <- c(558000, 594000, 6132000, 6142000)
```

Y con dicha extensión se realiza el recorte usando la función *crop* sobre ambas escenas (Figura 9.76):

[6] Recordatorio: Posterior a realizar un *plot()* o *plotRGB()* utilizar *drawExtent()* interáctivo para definir áreas geográficas sobre la imagen misma.

```
montevideo <- crop(sen2_10m, ext_mon)
montevideo_sin <- crop(sen2_10m_sin, ext_mon)

plotRGB(montevideo_sin,
        r="B4",
        g="B3",
        b="B2",
        stretch="lin",
        ext=ext_mon)
```

Figura 9.76: Detalle Costas de Montevideo, 16 de mayo 2019, composición RGB. Resolución 10m.

9.21.4 Tratamiento B11

El filtro espectral utilizado para visualizar la actividad fotogénica de las bacterias utiliza la respuesta espectral *SWIR1* o *B11* en Sentinel-2 se distribuye en resolución de 20 m y las debemos extraer para ambas fechas:

```
swir_20m <- raster(jps1[26])
swir_20m_sin <- raster(jps_sin[26])
```

Cambiar la resolución usando la función `disaggregate(x, fact, method)`, permite crear un nuevo rasterLayer con mayor resolución espacial, que a partir de `x`, basado en el número de celdas especificados en `fact` (por ejemplo, fact= 1 queda igual, fact= 2 se reduce a la mitad), debe ser definido por un entero que afecta filas y columnas o un vector de dos elementos para definir por separado en ambas dimensiones.

Los nuevos valores si se desean interpolar se debe incluir el parámetro `method` y actualmente solo existe la opción `bilinear`.

Para cambiar la resolución en sentido opuesto la función `aggregate(x, fact, fun)`, permite crear un nuevo rasterLayer con menor resolución espacial, a partir de `x`, basado en el factor de reducción especificados en `fact` (por ejemplo, fact= 2 resulta en un ráster con $2 \times 2 = 4$ veces menos número de celdas), debe ser definido por un entero que afecta filas y columnas o un vector de dos elementos para definir por separado en ambas dimensiones.

```
swir_10m <- disaggregate(x= crop(swir_20m,ext_mon),
                         fact= 2, method= "bilinear")
swir_10m_sin <- disaggregate(x= crop(swir_20m_sin, ext_mon),
                             fact= 2, method= "bilinear")
```

Así tenemos dos rasterLayer de B11 con una resolución de 10 m con la misma extensión de nuestras escenas. A continuación, se agregan al stack, asigna nombre y reordenar las capas.

```
montevideo <- addLayer(montevideo, swir_10m)
names(montevideo)[7] <- "B11"
montevideo <- subset(montevideo, c(2:5,7))

montevideo_sin <- addLayer(montevideo_sin, swir_10m_sin)
names(montevideo_sin)[7] <- "B11"
montevideo_sin <- subset(montevideo_sin, c(2:5,7))
```

Y llevamos de ND a BOA o reflectancia superficial utilizando el factor 1/10000:

```
montevideo_sr <- montevideo * 1e-4
montevideo_sin_sr <- montevideo_sin * 1e-4

montevideo_sr <- clamp(montevideo_sr,0,1)
montevideo_sin_sr <- clamp(montevideo_sin_sr,0,1)
```

9.21.5 FAI

A diferencia de las superficies terrestres, el agua absorbe fuertemente la luz en las longitudes de onda RED-NIR-SWIR. De hecho, debido a esta alta absorción, el agua es opaca o "negra" en las longitudes de onda SWIR incluso en los ambientes más turbios; esto proporciona la base para la utilización de estas lon-

gitudes de onda para corregir los efectos atmosféricos. Por otro lado, las algas flotando en la superficie del agua tienen mayor reflectancia en el NIR que, en otras longitudes de onda, y se puede distinguir fácilmente de las aguas circundantes (Hu [2009]).

Así se presenta el índice FAI o Índice de Algas Flotantes (Hu [2009]):

$$FAI = \rho_{NIR} - \left(\rho_{RED} + (\rho_{NIR} - \rho_{SWIR}) * \left(\frac{\lambda_{NIR} - \lambda_{RED}}{\lambda_{SWIR} - \lambda_{RED}}\right)\right) \tag{9.30}$$

Y su implementación en R y nuestra escena. En pleno fenómeno:

```
fai <- montevideo_sr$B8 - (montevideo_sr$B4 +
                             (montevideo_sr$B8
                              - montevideo_sr$B11) *
                             (835-665)/(1613-665)
                           )
```

Y post fenómeno:

```
fai_sin <- montevideo_sin_sr$B8 - (montevideo_sin_sr$B4 +
                                     (montevideo_sin_sr$B8 -
                                        montevideo_sin_sr$B11) *
                                     (835-665)/(1613-665)
                                   )
```

Para revisar y chequear los resultados podemos realizar distintas visualizaciones, por ejemplo:

1. Color Verdadero o R,G,B (B4,B3,B2):

Combinación que más se aproxima a los colores reales. Es ideal para realzar información del agua: turbidez, corrientes y sedimentos en suspensión.

Las bandas visibles dan respuesta a la luz que ha penetrado más profundamente, y por tanto sirven para discriminar el agua poco profunda y sirven para distinguir aguas turbias, corrientes, batimetría y zonas con sedimentos.

El azul oscuro indica aguas profundas, el azul claro indica aguas de media profundidad. La vegetación se muestra en tonalidades verdes. El suelo aparece en tonos marrones y tostados. El suelo desnudo y la roca aparecen en tonos amarillentos y plateados (Figura 9.77).

Figura 9.77: Detección de algas, composición Color Verdadero R,G,B.

2. Banda NIR (B8):

La respuesta vegetal es más fuerte en este rango espectral (Figura 9.78).

Figura 9.78: Detección de algas, respuesta banda NIR.

3. Falso Color NIR,R,G (B8,B4,B3):

La banda del infrarrojo cercano es útil para identificar los límites entre el suelo y el agua, también es sensible a la clorofila, permitiendo que se observen variaciones de la vegetación, que aparecen en tonos rojo.

Los cuerpos de agua con sedimentos en suspensión aparecen en tonos azul claro y los que poseen pocos sedimentos en suspensión en azul oscuro y tonos de marrón para mayor presencia de sedimentos; Las áreas urbanas y el suelo expuesto aparecen en tonos azules (Figura 9.79).

4. Falso Color SWIR,NIR,R (B11,B8,B4):

Esta combinación con dos bandas en la región del infrarrojo (SWIR y NIR) muestra una mayor diferenciación entre el suelo y el agua.

Figura 9.79: Detección de algas, composición Falso Color NIR,R,G.

La vegetación se muestra en diversas tonalidades de verde y rosa, que varían en función del tipo y de las condiciones de ubicación; las áreas urbanas y el suelo expuesto se presentan en tonos rosados; el agua, independiente de la cantidad de sedimentos en suspensión, aparece en negro o azul muy obscuro (Figura 9.80).

Figura 9.80: Detección de algas, composición Falso Color SWIR,NIR,R.

5. Índice FAI:

Producto obtenido con la fórmula 9.30 en la escena post explosión de algas (16 mayo 2019) y el día 31 enero 2019 en pleno proceso de brote de algas (Figura 9.81, Superior e inferior respectivamente).

Donde se aprecia la aparición de trazos claros sobre el fondo negro que corresponde al agua. Dichas trazas no necesariamente son iguales en todas las composiciones debido a que cada una responde a las distintas sensibilidades espectrales y no necesariamente corresponden al fenómeno estudiado.

En la figura 9.82 se destaca en detalle una zona con presencia de cianobacterias en varias composiciones espectrales. Solo la imagen producto de FAI en pleno proceso corresponde a la extensión determinada por el índice, que ha sido estimada con la

Figura 9.81: Detección de algas, índice FAI post fenómeno (superior), algal bloom (inferior).

configuración de bandas y ponderadores óptimo. Las trazas y señas en las otras combinaciones pueden ser productos de las algas u otros fenómenos asociados (Figura 9.82).

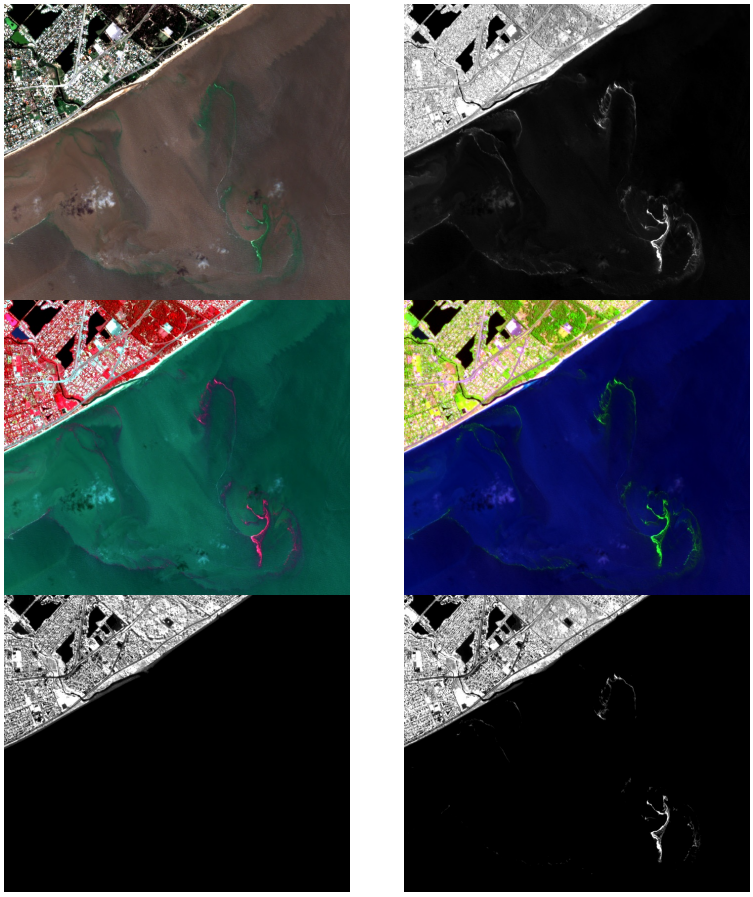


Figura 9.82: Vista de Detalle en la detección de algas. Composición Color Verdadero, NIR, Falso Color 1 y 2 e Índice FAI (post y algal bloom).

Clasificación

9.22 Conceptos y Fundamentos

La **clasificación** de imágenes digitales es el proceso de *asignar píxeles a clases*. Por lo general, cada píxel se trata como una unidad individual compuesta por valores en varias bandas espectrales. Por la comparación de píxeles entre sí contra píxeles de identidad conocida, es posible integrar grupos de píxeles similares en clases asociadas a las categorías de interés del usuario de los datos de teledetección.

Estas clases forman regiones en un mapa o una imagen, por lo que después de la clasificación, la imagen digital se presenta como un mosaico de parches uniformes, cada uno identificado por un color o símbolo. Estas clases son, en teoría, homogéneas: Los píxeles dentro de las clases son espectralmente más similares entre sí que a los píxeles en otras clases. En la práctica, por supuesto, cada clase mostrará cierta diversidad, ya que cada escena exhibirá cierta variabilidad dentro de las clases.

La clasificación de imágenes es uno de los campos importante en la EO y teledetección. En algunos casos, la clasificación en sí misma puede ser el objeto del análisis. Por ejemplo, la clasificación del uso de la tierra a partir de datos de teledetección produce una imagen similar a un mapa como producto final del análisis. En otros casos la clasificación puede ser sólo un paso intermedio en un análisis más elaborado, en el que los datos clasificados forman una de varias capas de datos en un SIG.

9.23 Tipos de Clasificación

Una distinción básica en la clasificación de imágenes separa en *clasificación supervisada* de *clasificación no supervisada.* Los procedimientos de clasificación supervisados requieren una considerable interacción con el analista, quien debe orientar la clasificación **identificando áreas** en la imagen que se sabe que pertenecen a cada categoría.

Clasificación no supervisada, por otro lado, requiere una interacción mínima con el analista en la búsqueda de **grupos naturales de píxeles** presentes en la imagen. La distinción entre supervisados y la clasificación sin supervisión es útil, especialmente para los estudiantes que están aprendiendo por primera vez sobre clasificación de imágenes, aunque las dos estrategias no son tan claramente distinguibles como estas definiciones sugieren, ya que algunos métodos no encajan perfectamente en ninguna de las categorías. Siendo denominados **clasificadores híbridos** aquellos que comparten características de ambos métodos.

9.23.1 Clasificación Supervisada

Una clasificación supervisada supone un *conocimiento previo* del área de estudio, ya sea por fuentes externas o por trabajo de campo. Esta familiaridad con la zona de estudio permite que el intérprete identifique en la imagen áreas suficientemente representativas de cada una de las categorías buscadas. Estas áreas se conocen como "campos de entrenamiento".

El término indica que tales áreas se utilizan para entrenar a la computadora en el reconocimiento de las diferentes categorías. En otras palabras, a partir de los píxeles de los campos de entrenamiento, la computadora calcula los ND que definen cada una de las clases, y luego asigna el resto de los píxeles de la imagen a una de esas categorías por sus ND.

La selección de los campos de entrenamiento puede basarse en propiedades ópticas conocidas de cada categoría, (observaciones de campo en lugares conocidos, fotografías aéreas y terrestres, entre otras fuentes de información). Se recomienda obtener la información de referencia lo más cerca posible (tiempo y distancia)

como sea posible al momento de la adquisición de la imagen, para evitar una mala asignación de píxeles basada sobre datos de referencia obsoletos. Siempre que sea posible, también se recomienda hacer mediciones radiométricas de campo, que le dan al intérprete información relevante sobre posibles factores de ruido o para ayudar a seleccionar correctamente las bandas adecuadas para la clasificación.

Una vez completa la selección de campos de entrenamiento, las estadísticas básicas de cada categoría se calculan, media, rango, desviación estándar, matriz de varianza-covarianza, etc. para todos los píxeles incluidos en el campo de entrenamiento asignado a esa clase.

Estas estadísticas se utilizarán más adelante para clasificar el resto de los píxeles de la imagen. Por tanto, las medidas extraídas de los campos de entrenamiento se asumen ser buenos representantes de cada clase. De lo contrario, el intérprete estaría forzando la computadora para clasificar áreas heterogéneas, y la clasificación sería incorrecta.

Dado que la fase de entrenamiento es la más relevante en la clasificación de imágenes, la selección de píxeles de entrenamiento debe hacerse con gran detalle y precisión (Figura 9.83). Cuando las categorías son homogéneas, un solo campo de entrenamiento por categoría puede ser suficiente para evitar confusión con otras clases (Tipo A).

Figura 9.83: Selección de Campos de Entrenamiento para Clase 'MAR'. Tipos de Selección, A Homogéneas, B Transición y C Variación Espectral

Se debe prestar especial atención para evitar seleccionar campos en áreas de transición (Tipo B), ya que las estadísticas de clase resultantes pueden no ser buenos representantes de la cobertura objetivo. El error opuesto sería seleccionar sitios de entrenamiento solo en aquellos lugares donde la categoría es muy homogénea, falta la variación espectral real de esa cobertura (Tipo C).

Por último, se debe considerar en la selección de los sitios de entrenamiento requisitos estadísticos que aseguren que son buenos representantes de las diferentes categorías. Este proceso implica una forma peculiar de muestreo espacial. Por esta razón, es mejor utilizar algunos de los criterios habituales en este tipo de técnica para elegir el tamaño y distribución de la muestra más

adecuada, así como para realizar estimaciones de eso.

En cuanto al tamaño, se requiere seleccionar un **mínimo** de $m + 1$ píxeles por categoría, siendo m el número de bandas que integran el análisis. Sin embargo, es muy conveniente *exceder este límite mínimo*. Varios autores (Davis et al. [1978]) recomiendan seleccionar un número de píxeles entre $10m$ y $100m$ por categoría. Este número mínimo está estrechamente relacionado con el grado de asociación espacial entre los ND de la imagen.

El muestreo convencional considera que las muestras seleccionadas son aleatorias e independiente y esta suposición no es válida (los campos de entrenamiento tienen píxeles vecinos afectados por la autocorrelación espacial). Por lo tanto, es más recomendable elegir *varios campos de tamaño pequeño que uno más grande*, ya que este último tenderá a subestimar la variabilidad de esa categoría. Algunos autores recomiendan *seleccionar píxeles aleatorios*, dentro de campos de mayor tamaño, en lugar de incluirlos a todos.

En cuanto a la distribución, lo mejor es considerar las características de la imagen en sí misma y tratar de incluir las variaciones espaciales de cada categoría: pendiente, aspecto, cobertura de la gradiente, densidad, contenido de humedad, tipo de suelo, etc.

Se debe considerar que la variabilidad de una cubierta es directamente proporcional a la resolución del sensor. En otras palabras, cuanto mayor sea el detalle espacial registrado en la imagen, mayor es su sensibilidad para detectar las variaciones internas de una categoría, que estaría oculta debajo de un píxel más grande. Dado que mayor heterogeneidad significa una mayor mezcla con clases similares y un mayor riesgo de confusiones en las asignaciones posteriores, un aumento en la resolución espacial puede complicar la clasificación digital. Este hecho, que parece contradecir el sentido común (mayor resolución espacial, mayor precisión), ha sido probado por varios autores y se conoce como **ruido de escena**.

9.23.2 Clasificación No Supervisada

Este método intenta definir las clases espectrales presentes en la imagen mediante la búsqueda de grupos de píxeles con ND similares. No se requieren conocimientos previos del área de estudio, y el trabajo del intérprete se centra más en el etiquetado de los grupos resultantes que en proporcionar información de entrada al algoritmo de agrupamiento. El supuesto básico detrás de los métodos no supervisados está en la *existencia de grupos únicos de píxeles con características espectrales similares* y, por lo tanto, una cobertura similar.

Sin embargo, como se dijo anteriormente, la similitud espectral no implica necesariamente una cobertura homogénea porque la misma cobertura puede tener diferentes respuestas espectrales, mientras que la misma señal espectral puede referirse a más de una cobertura. En consecuencia, el intérprete debe tener sentido de las clases resultantes, principalmente etiquetándolas, pero también combinándolas o dividiéndolas según las necesidades del usuario final.

El entrenamiento no supervisado se basa en seleccionar el **espacio de discriminación** (conjunto de bandas que se utilizarán para la agrupación) y el **algoritmo de agrupación**. Las variables utilizados para la agrupación pueden ser las bandas originales o el resultado de alguna transformación, como *Análisis de Componentes Principales (PCA)* o *Índices Vegetales (VI)*. Eventualmente, variables auxiliares como MDR o precipitaciones también se pueden utilizar en la agrupación, siempre que estas variables se midan en una escala cuantitativa.

Existe un buen número de algoritmos para agrupar los datos similares pero todos ellos comparten el punto crítico que es la decisión sobre sus parámetros de control, los cuales deben ser apropiados para cada problema de clasificación, ya que no existen reglas universales seguir.

Por lo general, el intérprete no conoce el número de grupos espectrales presente en la imagen, o los umbrales convenientes de varianza máxima, o la distancia adecuada entre grupos. De esta forma, el análisis no supervisado puede ser considerado un proce-

so exploratorio en el que el usuario selecciona valores arbitrarios que son modificados después de ver los resultados. El análisis sin supervisión puede indicar, por ejemplo, si las clases de usuario final son espectralmente separables o no, o su variabilidad espectral, o si tendrán una distribución bimodal o multimodal.

9.24 Tipos de Clases

Clases informacionales son categorías de interés para los usuarios de los datos, por ejemplo, diferentes tipos de unidades geológicas, diferentes tipos de bosques, o los diferentes tipos de uso de la tierra que transmiten información a los planificadores, gerentes, administradores y científicos que utilizan información derivada de datos de teledetección. Finalmente, estas clases son la información que deseamos derivar de los datos; son el objeto de nuestro análisis.

Desafortunadamente, estas clases no se registran directamente en los sensores; podemos derivarlos solo indirectamente, utilizando la evidencia contenida en los brillos o ND grabados en cada imagen. Por ejemplo, la imagen no puede mostrar directamente unidades geológicas, sino solo las diferencias en la topografía, la vegetación, el color del suelo, la sombra y otros factores que llevan al analista a concluir que ciertas condiciones geológicas existen en áreas específicas.

Clases espectrales son grupos de píxeles que son uniformes con respecto a los ND en varias bandas espectrales.

Si es posible definir una relación entre una clase espectral en la imagen y una clase informacional, la imagen forma una valiosa fuente de información. Por lo tanto, *clasificación en teledetección* procede relacionar categorías espectrales a categorías informativas.

Si la unión se puede hacer con confianza, entonces es probable que la información sea confiable. Si las categorías espectrales e informativas no corresponden, entonces es poco probable que la imagen sea una fuente útil para esa fuente de información.

Rara vez podemos esperar encontrar coincidencias exactas uno a uno entre información y clases espectrales. Cualquier clase in-

formativa incluye variaciones espectrales que surgen de variaciones naturales dentro de la clase. Por ejemplo, una región de la clase informativa "Bosque" sigue siendo "bosque", a pesar de que puede mostrar variaciones en la edad, la composición de especies, densidad y vigor, que conducen a diferencias en la apariencia espectral de una sola clase informativa. Además, otros factores, como variaciones en la iluminación y sombreado pueden producir variaciones adicionales en clases espectralmente uniformes.

Por lo tanto, las clases informativas se componen generalmente de numerosas *subclases espectrales* (grupos espectralmente distintos de píxeles que juntos pueden ensamblarse para formar una clase informativa).

Con la clasificación digital, a menudo debemos tratar subclases espectrales como unidades distintas durante la clasificación, para luego varias de esas clases espectrales asociarlas bajo un símbolo único para la imagen o mapa final que utilizarán los usuarios.

9.25 Ventajas y Desventajas del Método No Supervisado

9.25.1 Ventajas

Las ventajas de la clasificación no supervisada (en relación con la clasificación supervisada) pueden ser enumerados de la siguiente manera:

No se requieren amplios conocimientos previos de la región. O, más exactamente, la naturaleza del conocimiento requerido para la clasificación no supervisada difiere del requerido para clasificación supervisada. Para realizar una clasificación supervisada, conocimiento detallado de Se requiere que el área a ser examinada seleccione ejemplos representativos de cada clase a ser mapeado. Para realizar una clasificación sin supervisión, no se requieren conocimientos previos detallados, pero se requiere conocimiento de la región para interpretar el significado de los resultados producidos por el proceso de clasificación.

Se minimiza la posibilidad de errores humanos. Para realizar una clasificación sin supervisión, el operador tal vez pueda

especificar sólo el número de categorías deseadas (o, posiblemente, límites mínimos y máximos en el número de categorías) y, a veces, restricciones que gobierna la distinción y uniformidad de los grupos. Muchas de las decisiones detalladas necesarios para la clasificación supervisada no son necesarios para la clasificación no supervisada, por lo que al analista se le presentan menos oportunidades de error. Si el analista tiene inexactitud preconceptos con respecto a la región, esos preconceptos tendrán poca oportunidad de influir en la clasificación. Las clases definidas por la clasificación no supervisada a menudo son mucho más uniformes con respecto a la composición espectral que los generados por clasificación supervisada.

Las clases únicas se reconocen como unidades distintas. Tales clases, tal vez de muy pequeñas extensión de área, pueden permanecer no reconocidos en el proceso de clasificación supervisada y podría incorporarse inadvertidamente a otras clases, generando error e imprecisión a lo largo de toda la clasificación.

9.25.2 Desventajas y limitaciones

Las desventajas y limitaciones de la clasificación sin supervisión surgen principalmente de dependencia de agrupaciones "naturales" y dificultades para hacer coincidir estos grupos con la información categorías que son de interés para el analista.

La clasificación no supervisada identifica clases espectralmente homogéneas dentro del datos que no necesariamente corresponden a las categorías informativas que son de interés al analista. Como resultado, el analista se enfrenta al problema de emparejar espectros clases generadas por la clasificación a las clases informativas que son requeridas por el usuario final de la información. Rara vez existe una correspondencia simple uno a uno entre los dos conjuntos de clases.

El analista tiene un control limitado sobre el menú de clases y sus identidades específicas. Si es necesario generar un menú específico de clases informativas (por ejemplo, para hacer coincidir a otras clasificaciones para otras fechas o regiones adyacentes), el uso de clasificación no supervisada puede ser insatisfactorio.

Las propiedades espectrales de clases informativas específicas cambiarán con el tiempo (de forma estacional, así como a lo largo de los años). Como resultado, las relaciones entre información las clases y las clases espectrales no son constantes, y las relaciones definidas para una imagen no pueden extenderse a los demás.

9.26 Ventajas y Desventajas del Método Supervisado

9.26.1 Ventajas

Las ventajas de la clasificación supervisada, en relación con la clasificación no supervisada, pueden enumerarse de la siguiente manera.

El analista tiene el control de un conjunto seleccionado de categorías adaptadas a un propósito específico y una región geográfica. Esta cualidad puede ser de vital importancia si se hace necesario generar una clasificación para el propósito de la comparación con otra clasificación de la misma área en una fecha diferente o si la clasificación debe ser compatible con las de las regiones vecinas.

La clasificación supervisada está vinculada a áreas específicas de identidad conocida. Control otorgado a través del proceso de selección de áreas de entrenamiento.

El analista que utiliza la clasificación supervisada no se enfrenta al problema de emparejar categorías espectrales en el mapa final con las categorías informativas de interés. Esta tarea, ha sido abordado durante el proceso de selección de datos de entrenamiento.

El operador puede detectar errores graves en la clasificación examinando los datos de entrenamiento.

9.26.2 Desventajas y limitaciones

Las desventajas de la clasificación supervisada son numerosas.

El analista impone una estructura de clasificación sobre los datos. Hay que recordar que la clasificación no supervisada busca clases “naturales”. Estas clases definidas por el operador pueden no coincidir con las clases que existen dentro de los datos y, por lo tanto, pueden no ser distintas o bien definidas en el espacio de

datos multidimensional.

Los datos de entrenamiento a menudo se definen principalmente con referencia a categorías informativas y sólo secundariamente con referencia a propiedades espectrales. Un área de entrenamiento que sea "100% bosque" puede ser precisa con respecto a la definición de "bosque", pero aún puede ser muy diversa con respecto a la densidad, la edad, el sombreado y por lo tanto forman un área de entrenamiento pobre.

Los datos de entrenamiento seleccionados por el analista pueden no ser representativo de las condiciones encontradas en toda la imagen. Esto podría ser cierto a pesar de los mejores esfuerzos del analista, especialmente si el área a clasificar es grande, compleja o inaccesible.

La selección detallada de los datos de entrenamiento puede ser una tarea costosa y que requiere mucho tiempo y ser una empresa tediosa. Incluso si se dispone de amplios recursos el analista puede experimentar problemas para hacer coincidir las áreas de entrenamiento según se definan en mapas y fotografías aéreas y en la imagen a clasificar.

Es posible que la clasificación supervisada no pueda reconocer y representar categorías especiales o únicas no representadas en los datos de entrenamiento. Posiblemente porque no son conocidos por el analista o porque ocupan muy pequeñas áreas de la imagen.

9.27 Análisis de las Estadísticas de Entrenamiento

Independientemente del método utilizado para definir la fase de entrenamiento, antes de realizar la clasificación de la imagen completa, se recomienda analizar las estadísticas de entrenamiento para determinar si las categorías seleccionadas son realmente discriminables. Si no, el intérprete debe averiguar si el proceso de entrenamiento necesita una revisión, o las variables, o categorías de entrada son relevantes para el objetivo final.

En el caso de la imagen Sentinel-2, de 20m (8.15.2) que usaremos como ejemplo de clasificación, se han seleccionado y creado cinco clases (Figura 9.84):

- *Agua*: Polígono en agua de mar.

Figura 9.84: Cinco áreas de entrenamiento, cinco clases. Imagen Sentinel-2, 20m, RGB.

- *Suelo*: Polígono en suelo descubierto [7].
- *Urbano*: Polígono sobre área urbana.
- *Matorral*: Vegetación baja
- *Bosque*: Vegetación Alta y Densa.

[7] Para efecto de demostración el polígono de suelo natural se extiende sobre una extensa área, cubriendo un rango muy amplio de respuestas espectrales.

Los cuadros 9.5 y 9.6 muestran la media y desviación estándar de los píxeles incluidos en las clases de entrenamiento de las categorías definidas.

Cuadro 9.5: Promedio de Clases.

Banda[nm]	Agua	Suelo	Urbano	Matorral	Bosque
B1[444]	511.29	1036.33	1567.38	628.00	338.20
B2[496.6]	451.71	1979.78	1939.46	777.95	376.72
B3[560]	295.92	2744.06	2366.46	1093.63	660.12
B4[664.5]	232.07	3304.28	2668.69	1743.84	440.84
B5[703.9]	242.96	3716.78	2994.23	2073.42	1035.20
B6[740.2]	243.71	3787.61	2960.62	2253.58	2621.84
B7[782.5]	260.34	3961.17	3063.23	2474.16	3147.08
B8A[864.8]	250.58	4024.56	3049.62	2714.74	3413.80
B11[1613.7]	211.47	4136.72	3323.54	4161.79	1345.96
B12[2202.4]	164.82	3028.39	3163.23	3538.68	664.88

Nota:
Valores son expresados en reflectancia x 10.000

La **media** proporciona información sobre el *comportamiento espectral medio de cada categoría*, mientras que la **desviación estándar** contiene su homogeneidad interna. En este caso, los valores más altos de desviación estándar son para la capa *suelo*, mientras que los valores más bajos los valores son para el *agua*. A pesar de la gran complejidad espectral de las zonas urbanas, no muestran desviaciones estándar altas, probablemente debido al efecto de homogeneización producido por tener un tamaño de píxel de resolución media.

También se han desarrollado métodos gráficos y numéricos para evaluar las estadísticas de entrenamiento. Entre los primeros, el más básico es un gráfico de *firma espectral* (ver 7.11.1), que muestra las bandas en el eje x y los ND de cada categoría en el eje y, (Figura 9.85).

Cuadro 9.6: Desviación Estándar de Clases.

Banda[nm]	Agua	Suelo	Urbano	Matorral	Bosque
B1[444]	8.60	165.55	146.15	38.92	60.80
B2[496.6]	10.00	299.72	145.13	53.12	13.00
B3[560]	10.54	410.38	112.50	65.82	22.86
B4[664.5]	9.18	423.45	94.21	68.73	20.12
B5[703.9]	10.55	459.88	116.39	80.68	36.26
B6[740.2]	9.92	407.65	108.25	85.67	78.97
B7[782.5]	10.76	388.62	105.40	91.70	124.31
B8A[864.8]	10.45	361.39	105.71	94.74	134.54
B11[1613.7]	12.30	551.83	86.17	106.10	25.38
B12[2202.4]	10.25	517.08	96.06	74.72	29.89

Nota:
Valores son expresados en reflectancia x 10.000

Al ajustar y convertir los valores de ND a reflectancias (usando factor 1/10000), estos gráficos serán similares a las curvas espectrales que se muestran en 9.45.

Siguiendo nuestro ejemplo, nos basamos en las primeras diez bandas reflectantes del sensor MSI para construir la firma espectral que proporciona una primera evaluación de las tendencias espectrales de cada categoría: *trazos paralelos* y *líneas cercanas* indican superposiciones probables entre categorías, mientras que las *intersecciones entre líneas* muestra las bandas que tienen más probabilidades de separarse entre categorías.

El gráfico de firma espectral muestra solo la *tendencia promedio* de cada categoría. Para saber si pueden ser discriminados de los demás, también es necesario considerar los valores de dispersión, ya que proporcionarán información sobre una superposición potencial con categorías similares.

Una herramienta gráfica para evaluar esas posibles superposiciones es el *gráfico de dispersión espectral*. Este es un gráfico de barras que incluye el rango digital cubierto por cada categoría, comúnmente expresado como el promedio ± 1 o 2 desviaciones estándar de la media. En la Figura 8.42, se muestran diez bandas de estos gráficos, mostrando las distribuciones de las cinco catego-

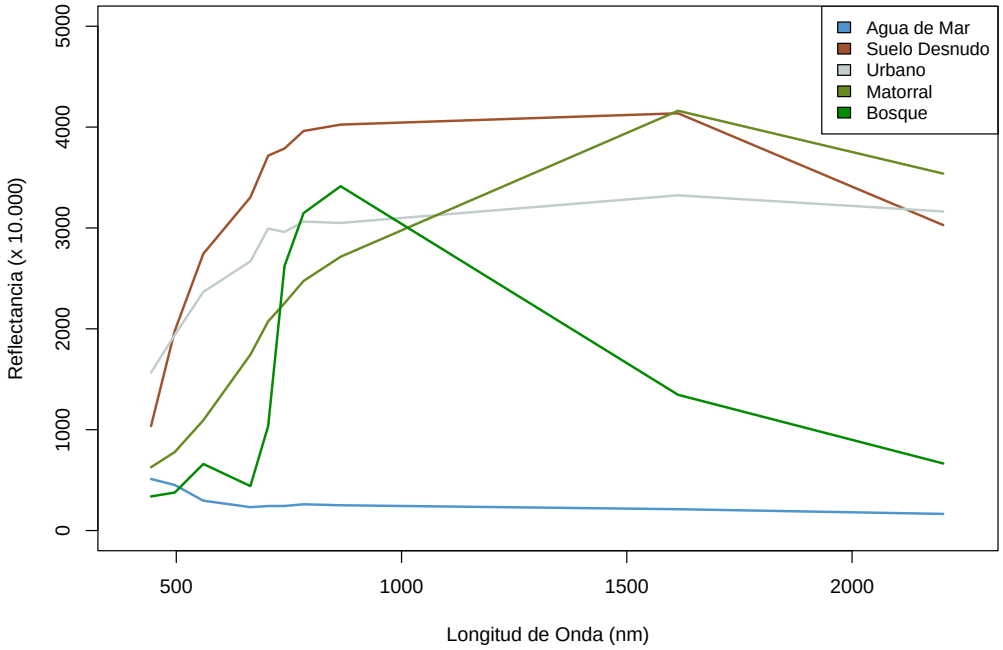


Figura 9.85: Firmas Espectrales de las clases de entrenamiento.

rías o clases definidas para los datos extraídos desde una imagen Sentinel-2. Observamos cómo ciertas coberturas muestran claras superposiciones en algunas bandas (por ejemplo, suelo y urbano en la banda 2), pero están claramente separadas en otras (en la banda 1, especialmente), mientras que otras muestran una mayor similitud en varias de las bandas (urbano y matorral), y el agua se discrimina claramente del resto en las cuatro bandas.

Finalmente, además de los procedimientos gráficos, existen algunos criterios cuantitativos que proporcionan más información para discriminar entre las categorías de entrada. El índice más simple es la **distancia normalizada (DNo)**, que se calcula como la *diferencia absoluta entre los promedios de dos categorías, como una fracción de su desviación estándar* (Ecuación 9.31), (Davis et al. [1978]):

$$DNo_{A,B} = \frac{|\,\overline{ND_A} - \overline{ND_B}\,|}{S_A + S_B} \tag{9.31}$$

Este cálculo se aplica a cada par de bandas que participan en la clasificación, promediando su valor para obtener una **Matriz**

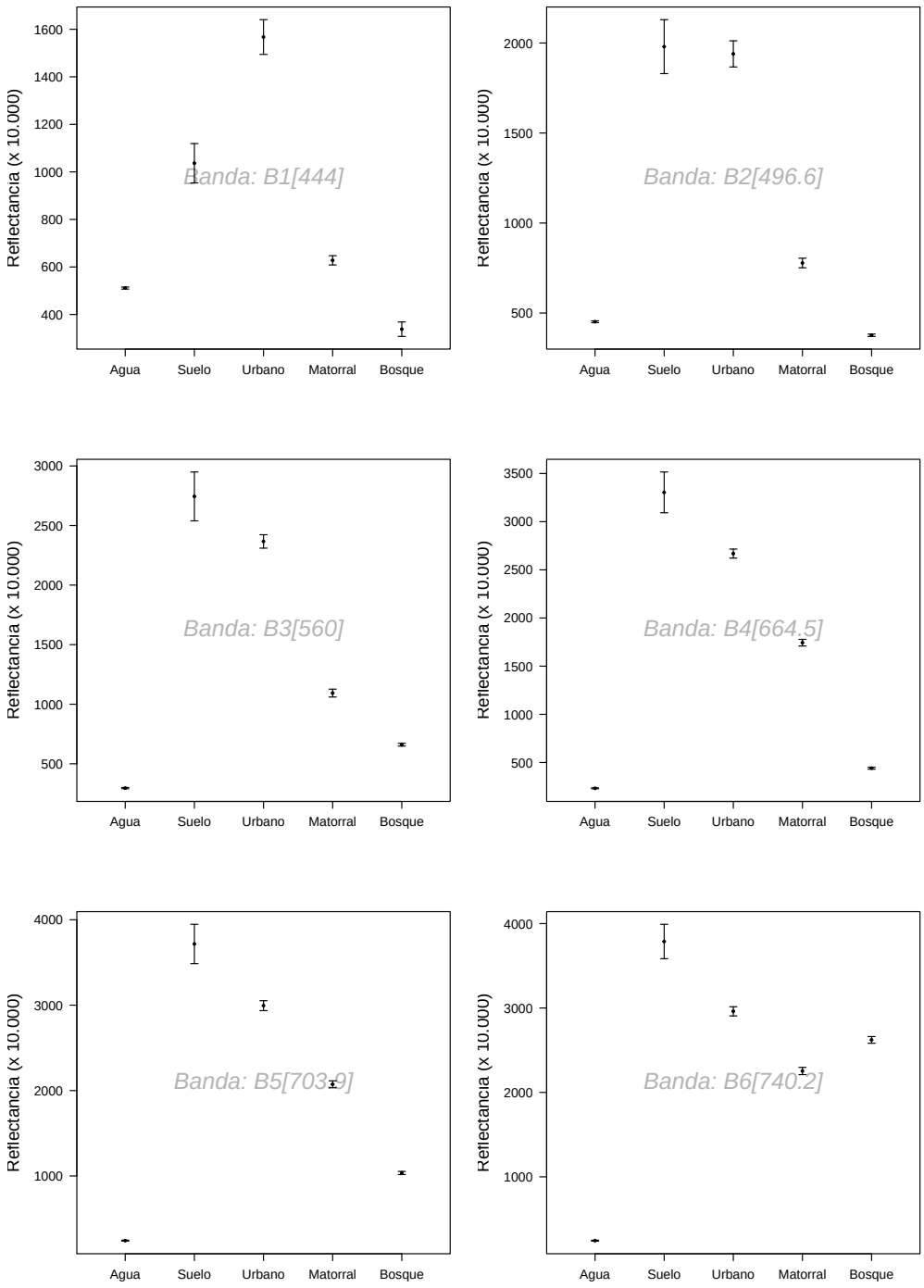


Figura 9.86: Gráficos de Dispersión Espectral.

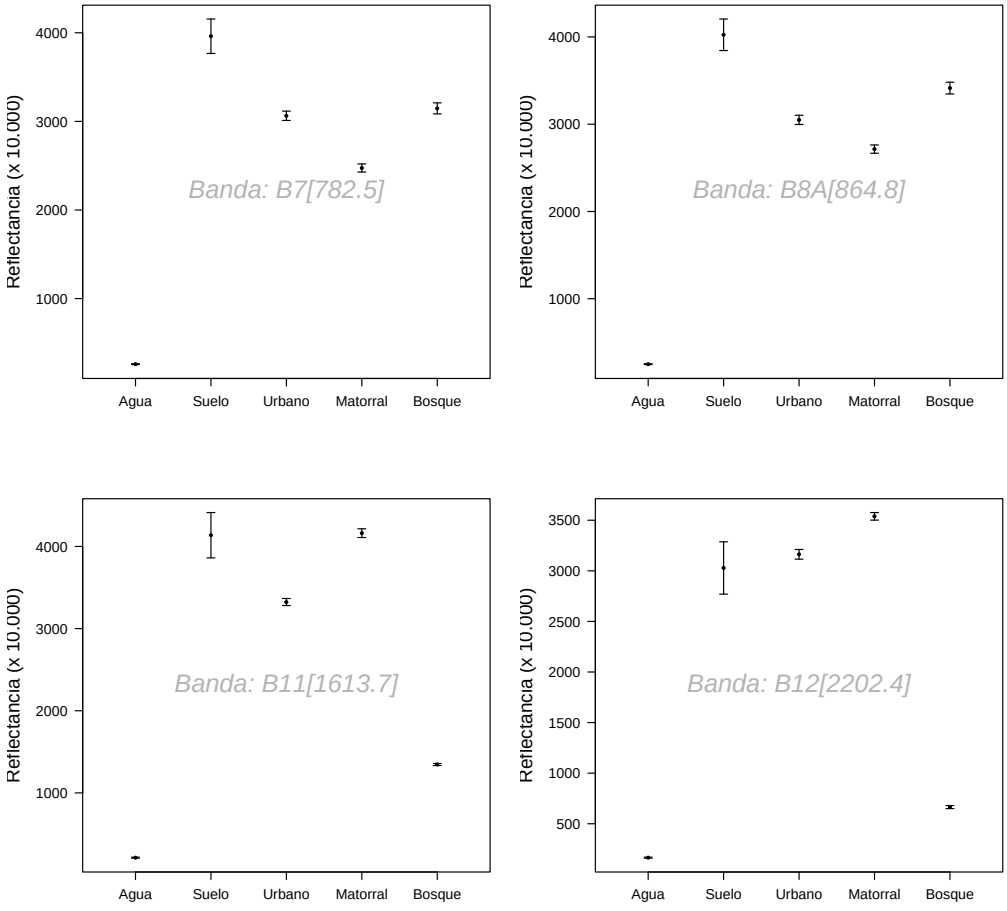


Figura 9.87: Gráficos de Dispersión Espectral.

de Separabilidad (Cuadro 9.7, 9.8 y 9.9).

Para conocer que tan *cerca* o *lejos* se encuentra una clase de otra se debe buscar el par de ellas en las filas y el valor por banda espectral en la columna correspondiente, o en valor promedio en la última columna.

Los valores se comportan de manera relativa, por ejemplo, ¿a partir de qué magnitud se consideran *cercanas*? o ¿qué es una clase *lejana* de otra? Una forma de analizar los resultados es buscar los valores máximos o mínimos y a partir de ellos ubicar los valores particulares que me interesan, así puedo tener una vista con un marco de referencia.

Cuadro 9.7: Tabla de Separabilidad para las cinco clases de entrenamiento. Para las bandas espectrales y su promedio. Datos obtenidos a partir de imagen Sentinel-2.

ClaseA	ClaseB	B1	B2	B3	B4
Agua	Suelo	3.01	4.93	5.82	7.10
Agua	Urbano	6.82	9.59	16.83	23.57
Agua	Matorral	2.46	5.17	10.45	19.40
Agua	Bosque	2.49	3.26	10.90	7.12
Suelo	Urbano	1.704	0.091	0.722	1.228
Suelo	Matorral	1.997	3.406	3.466	3.170
Suelo	Bosque	3.08	5.13	4.81	6.46
Urbano	Matorral	5.08	5.86	7.14	5.68
Urbano	Bosque	5.94	9.88	12.61	19.49
Matorral	Bosque	2.91	6.07	4.89	14.66

Cuadro 9.8: Continuación.

ClaseA	ClaseB	B5]	B6	B7	B8A
Agua	Suelo	7.38	8.49	9.27	10.15
Agua	Urbano	21.67	22.99	24.13	24.10
Agua	Matorral	20.06	21.03	21.61	23.43
Agua	Bosque	16.92	26.75	21.37	21.82
Suelo	Urbano	1.254	1.603	1.818	2.087
Suelo	Matorral	3.040	3.110	3.096	2.872
Suelo	Bosque	5.40	2.40	1.59	1.23
Urbano	Matorral	4.67	3.65	2.99	1.67
Urbano	Bosque	12.83	1.81	0.37	1.52
Matorral	Bosque	8.88	2.24	3.12	3.05

Cuadro 9.9: Continuación.

ClaseA	ClaseB	B11	B12	Promedio
Agua	Suelo	6.96	5.43	**6.85**
Agua	Urbano	31.61	28.20	**20.95**
Agua	Matorral	33.37	39.70	**19.67**
Agua	Bosque	30.11	12.46	**15.32**
Suelo	Urbano	1.275	0.220	**1.20**
Suelo	Matorral	0.038	0.862	**2.51**
Suelo	Bosque	4.83	4.32	**3.93**
Urbano	Matorral	4.36	2.20	**4.33**
Urbano	Bosque	17.73	19.84	**10.20**
Matorral	Bosque	21.42	27.47	**9.47**

9.28 Problema de Asociar la Firma Espectral con una Clase

Los resultados del proceso de construcción de las áreas de entrenamiento hasta ahora proporcionan agrupaciones uniformes de píxeles: clases que son uniformes con respecto a los valores espectrales que componen cada píxel. Las clases *son significativas* solo en la medida en que puedan *coincidir* con una o más clases informativas que sean de interés para el usuario del producto final.

Lamentablemente, a menudo, las clases espectrales no coinciden a la perfección con las clases informativas. En algunos casos, en las clases informativas pueden ser productos de mezclas o arreglos complejos de varios otros componentes. Por ejemplo, un parche boscoso se encuentra disperso en una pequeña área en un entorno más extenso de pastizales. Si estas áreas boscosas son pequeñas en relación con la resolución espacial del sensor, entonces la respuesta espectral general para tal área diferirá de *Bosque* o *Pastizal*, debido al efecto de la mezcla de píxeles en la respuesta espectral. Esta región entonces puede ser asignada a una clase distinta de las clases de bosques o pastizales.

Incluso las clases informativas nominalmente uniformes pueden ser la manifestación de un conjunto de clases espectrales. Por ejemplo, un área boscosa puede registrarse como varios grupos espectrales, debido quizás a variaciones en densidad, edad, aspecto, sombras, y otros factores que alteran las propiedades espectrales de una región boscosa, pero no lo suficiente para alterar el hecho de que la región pertenece a la clase informativa *Bosque*.

Así, un problema práctico serio con la **clasificación no supervisada** es que las coincidencias entre clases espectrales e informativas no siempre son posibles; las categorías informativas pueden no tener contrapartes espectrales directas o viceversa. Es más, no se tiene control sobre la naturaleza de las categorías generadas por clasificación no supervisada.

Si se realiza un estudio para comparar los resultados con los de una región adyacente o de una fecha diferente, por ejemplo, puede ser necesario tener el mismo conjunto de categorías de

información en ambos mapas o no se puede hacer la comparación. Si se va a utilizar una clasificación no supervisada, podría ser difícil generar los mismos conjuntos de categorías informativas en ambas imágenes.

9.29 Concepto de Endmember

Los píxeles puros (compuestos de la misma cobertura o superficie) se conocen como **endmember** o *miembros finales*. Los *endmembers* son los objetos macroscópicos disponibles en una escena. Puede ser vegetación, agua, mineral, suelo, hoja de un manzano, animal, metal o cualquier material natural o artificial.

La extracción de endmember es el proceso de seleccionar una colección de espectros de firmas puras de objetos o coberturas del suelo presentes en una imagen de teledetección.

La extracción se realiza generalmente de dos formas:

1. Derivándolas directamente de las imágenes de teledetección, sobre píxeles claramente identificados como puros.

2. De una biblioteca espectral que contiene los espectros de coberturas conocidas medidas en el campo o laboratorio.

Estos espectros de firmas puras son muy importantes para las tareas de clasificación supervisadas ya que servirán de patrón de comparación.

Métodos de Clasificación en R

9.30 Preparación de la Imagen

Previo a la aplicación de distintos métodos de clasificación, debemos preparar nuestra imagen para reducir posibles problemas y facilitar la lectura e interpretación de los resultados.

En el punto 8.15.3 generamos la imagen Sentinel-2 de resolución de 60 m con las siguientes bandas o capas:

```
names(sen2_60m)
```

```
 [1] "MSK_CLD" "MSK_SNW" "AOT"     "B1"      "B2"      "B3"
 [7] "B4"      "B5"      "B6"      "B7"      "B9"      "B11"
[13] "B12"     "B8A"     "SCL"     "WVP"
```

En los procesos de clasificación se utilizan los valores de reflectancia así que usando la función *subset* extraemos y ordenamos por su longitud de onda solo las bandas de respuestas espectrales.

```
proc_60m <- subset(sen2_60m,c(4:10,14,11:13))
names(proc_60m)
```

```
 [1] "B1"  "B2"  "B3"  "B4"  "B5"  "B6"  "B7"  "B8A" "B9"  "B11" "B12"
```

Y utilizando el factor 1/10000 se convierte a reflectancia BOA y ajustamos al rango válido $[0, 1]$.

```
sen2_60m_sr <- proc_60m/10000

sen2_60m_sr <- clamp(x = sen2_60m_sr, lower= 0, upper= 1)
```

9.31 Técnicas para Extracción de Muestras

9.31.1 DrawPoly()

El método más simple consiste de interactuar con la imagen en el mismo editor de RStudio.

1. Generamos la imagen con la función *plotRGB*.

```
plotRGB(sen2_60m_sr, r="B4", g="B3",
        b="B2", stretch="lin")
```

2. Generamos un objeto *extent* de la zona de interés.

```
ext <- drawExtent()
```

3. Refrescamos nuevamente la imagen ajustada a la ventana definida por *ext*.

```
plotRGB(sen2_60m_sr,
        r="B4", g="B3", b="B2",
        stretch="lin", ext= ext)
```

4. Utilizando las opciones descritas en 4.5.2 seleccionamos las muestras de entrenamiento para cada una de las clases necesarias (Figura 9.84). Por ejemplo, dibujamos el polígono sobre el agua con valores homogéneos y representativos:

```
poli_mar <- drawPoly()
```

Y vamos repitiendo por cada clase necesaria.

9.31.2 Dibujo RGB

Una alternativa más compleja es utilizar una sesión interactiva de cartografía web con un set de herramientas para el trazado y edición de los polígonos que definen áreas de interés.

1. Primero cargamos los paquetes en la sesión actual (debiesen estar instalados de los capítulos anteriores).

```
library(mapview)
library(mapedit)
```

2. Generamos una sesión de mapa web con nuestra imagen como una capa adicional y la paleta de dibujo activa. No es necesario que el CRS de la imagen coincida con el mapa web ya que la función *viewRGB* se encarga de ello de manera transparente.

```
poli_mar <- viewRGB(sen2_60m_sr) %>% editMap()
```

3. El objeto retornado es un objeto del tipo *list* que se compone de varios elementos dependiendo de lo realizado en la sesión de dibujo (puede ser *dibujado*, *editado* o *borrado*) y tenemos acceso a la geometría utilizando el formato:

```
geom <- poli_mar$drawn$geometry
```

4. Y lo transformamos al formato de objeto espacial usando la función *as_Spatial()* del paquete **sf**.

```
poli_mar <- sf::as_Spatial(geom)
```

5. Retornamos el objeto al CRS de la imagen Sentinel-2 del ejemplo (Figura 9.88).

```
crs_60m <- crs(sen2_60m)
poli_mar_proj <- spTransform(poli_mar,crs_60m)
```

Y repetimos por cada clase o categoría.

9.32 Extracción de Datos Espectrales

Generados para el ejercicio se seleccionan las siguientes clases (Figura 9.84):

```
poli_suelo
poli_mar
poli_urba
poli_cultivo
poli_bosque
```

Figura 9.88: Dos polígonos de muestreo para clases agua en la esquina Superior derecha.

Y por cada área de entrenamiento se extraen los valores medios utilizando la función *extract* del paquete **raster**, aplicando la función *mean* y removiendo los valores NA (*na.rm=T*).

```
perfil_mar <- extract(x = proc_60m, y = poli_mar,
                      fun=mean, na.rm=T)
```

Y repetimos para cada una de las áreas de entrenamiento.

```
perfil_suelo <- extract(x = proc_60m, y = poli_suelo,
                        fun=mean, na.rm=T)
perfil_urba <- extract(x = proc_60m, y = poli_urba,
                       fun=mean, na.rm=T)
perfil_cultivo <- extract(x = proc_60m, y = poli_cultivo,
                          fun=mean, na.rm=T)
perfil_bosque <- extract(x = proc_60m, y = poli_bosque,
                         fun=mean, na.rm=T)
```

Y creamos un objeto del tipo *matrix* combinando los valores por fila y mantenemos columnas por cada banda espectral.

```
endmem <- rbind(perfil_mar,perfil_suelo,perfil_urba,
                perfil_cultivo, perfil_bosque)
```

```
rownames(endmem) <- c("Agua","Suelo","Urbano","Matorral","Bosque")
```

Una muestra para estudiar el formato es:

```
endmem[,1:3]
```

```
                B1          B2          B3
Agua     0.05112943 0.04517105 0.02959187
Suelo    0.10363333 0.19797778 0.27440556
Urbano   0.15673846 0.19394615 0.23664615
Matorral 0.06280000 0.07779474 0.10936316
Bosque   0.03382000 0.03767200 0.06601200
```

Clasificación Supervisada en R

En las páginas anteriores hemos realizado la extracción de endmember (9.29) vía búsqueda en la imagen Sentinel-2 que servirán de patrón de búsqueda en toda la imagen para clasificarla en esas cinco clases.

Los métodos han sido implementados en el paquete RStoolbox (Leutner et al. [2019]).

9.33 Mapeador de Ángulos Espectrales (SAM)

El mapeador de ángulos espectrales (SAM) es una herramienta que permite realizar una clasificación rápida de similitud espectral entre espectros en una imagen *test*, con endmembers de *referencia*. Los espectros de referencia pueden ser espectros de laboratorio o de campo o extraídos de la imagen (Kruse et al. [1993]).

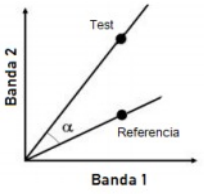


Figura 9.89: Relación entre espectros, uno de test, otro de referencia. La medida de separación angular (a) corresponde a la medida obtenida por el método SAM.

El algoritmo determina la similitud espectral entre dos espectros calculando el *ángulo a*, entre los dos espectros, tratándolos como vectores en un espacio con dimensionalidad igual al número de bandas *nb* (Figura 9.89).

Para un tipo de cobertura dado (Figura 9.90), el vector correspondiente a su espectro estará a lo largo de una línea que pasa por el origen, siendo la magnitud del vector más pequeña A o más grande B dependiendo de su estado bajo una iluminación más baja o alta, respectivamente.

Al comparar el vector de un tipo de cobertura desconocida C con un material conocido con un vector espectral endmember D; ambas coinciden si el ángulo a es menor que un valor de tolerancia

especificado (Figura 9.91).

El algoritmo SAM generaliza esta interpretación geométrica al espacio nb dimensional. El cálculo consta tomando el arco coseno del producto escalar de los espectros. SAM determina la similitud de un espectro de referencia t con un espectro de referencia r aplicando la ecuación 9.32.

$$a = \cos^{-1}\left(\frac{\sum_{i=1}^{nb} t_i r_i}{\left(\sum_{i=1}^{nb} t_i^2\right)^{1/2}\left(\sum_{i=1}^{nb} r_i^2\right)^{1/2}}\right) \tag{9.32}$$

Esta medida de similitud es insensible a factores externos porque el ángulo entre dos vectores es invariante con respecto a las longitudes de los vectores. Como resultado, espectros endmember capturados en laboratorio pueden ser comparados con espectros de reflectancia aparente detectados a distancia, que inherentemente tienen un factor de adicional relacionado con los efectos de la iluminación topográfica.

Para cada píxel, SAM calcula el ángulo entre el vector definido por los valores de píxel y cada vector del endmember. El resultado de esto es una capa ráster para cada miembro del extremo que contiene el ángulo espectral. Cuanto más pequeño es el ángulo espectral, más similar es un píxel a una clase de referencia.

La función `sam(img, em, angles)` ejecuta la clasificación de la imagen *img* usando los endmember *em* y devuelve un modelo ráster clasificado en el número de categorías definidas (Figura 9.92) si el parámetro *angles* es falso.

```
clases <- sam(img = proc_20m_crop, em = endmem,
              angles = F)
```

También, si desea más control en la clasificación y aplicar umbrales (9.7.1) de los valores de ángulos máximos o clasificar cada píxel a un miembro final más similar, se puede generar un objeto RasterBrick con tantas bandas como clases contiene el objeto *em* definiendo el parámetro *angles* a verdadero (Figura 9.93). Cada capa tiene el valor angular y mientras menor el valor, más similar a la clase es.

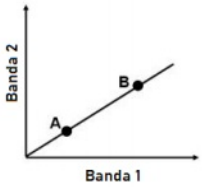


Figura 9.90: Efecto de la iluminación en el vector espectral de una cobertura.

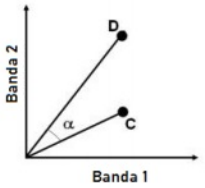


Figura 9.91: Medida de separación (a) entre dos coberturas (C, D).

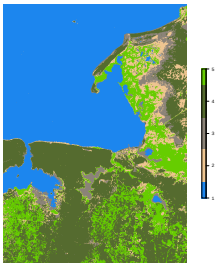


Figura 9.92: Clasificación de imagen, método SAM (parámetro 'angles' =F)

```
clases_sam <- sam(img = proc_20m_crop, em = endmem,
                  angles = T)
```

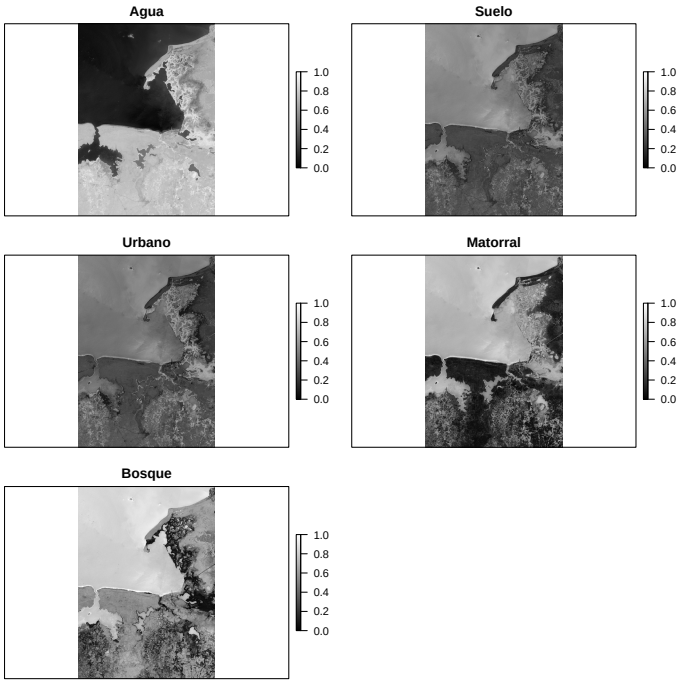


Figura 9.93: Valor Angular asignado por el cálculo SAM para cada categoría. Mientras más bajo el valor, más parecido al espectro de referencia.

9.34 Análisis de Mezcla Espectral de Múltiples Endmember (MESMA)

El análisis de mezcla espectral de miembros múltiples (MESMA) es una técnica para estimar la proporción de cada píxel que está cubierto por una serie de tipos de cobertura conocidos; en otras palabras, busca determinar la composición probable de cada píxel de la imagen.

Los píxeles que contienen más de un tipo de cobertura se denominan **píxeles mixtos**. Los píxeles "**puros**" contienen solo una cobertura o clase. Por ejemplo, un píxel mixto puede contener vegetación, suelo desnudo y parches de suelo desnudo. Un píxel puro contendría solo una cobertura, ejemplo, vegetación.

Los píxeles mixtos pueden causar problemas en las clasificaciones de imágenes (tanta clasificación supervisada o no supervisada) porque el píxel pertenece a más de una clase, pero solo se puede asignar a una única clase.

Una forma de abordar el problema de los píxeles mixtos es utilizar idealmente el análisis de subpíxeles con imágenes **hiperespectrales**.

MESMA fue formulado por Roberts et al. [1998] utilizando imágenes del Espectrómetro de Imágenes Infrarrojas Visibles en el Aire (AVIRIS) en las montañas de Santa Mónica de California, EEUU.

Básicamente, la técnica modela datos espectrales medidos de forma remota como combinaciones lineales de espectros puros, (endmember), que acepta variación en el número y tipos por píxel. Así, la vegetación se compone por un conjunto único de endmember, como por fracciones.

Se expresa como:

$$P_{i\lambda} = \sum_{k=1}^{N} f_{ki} \times P_{k\lambda} + \varepsilon_{i\lambda} \tag{9.33}$$

y

$$\sum_{k=1}^{N} f_{ki} = 1 \tag{9.34}$$

Donde un píxel compuesto $P_{i\lambda}$ en una ubicación i se modela como la suma de N endmember, $P_{k\lambda}$, cada uno de los cuales cubre una fracción f_{ki} del píxel.

El término residual $\varepsilon_{i\lambda}$ describe la parte no modelada de la radiancia, y el modelo elegido para cada píxel es el que minimiza la raíz del error cuadrático medio (RMSE) sobre el número incluido de bandas utilizadas en la desmezcla, B (Formula 9.35).

$$RMSE = \left[\sum_{k=1}^{B} (\varepsilon_{i\lambda})^2 / B \right]^{1/2} \tag{9.35}$$

El paquete RStoolbox implementa la función `mesma(img, em, method, iterate, tolerance)` que ejecuta la clasificación de

la imagen *img* usando los endmember *em*, mediant el método *method* que en la actualidad solo se dispone de la solución estadística denominada 'NNLS' (Franc et al. [2005]). También dispone de los parámetros *iterate* y *tolerance* que permite realizar ajustes finos (los valores por defecto son 400 y 1e-8 respectivamente).

Dependiendo de la configuración de iteración y tolerancia, la suma de las probabilidades de presencia estimadas por píxel podría variar alrededor de 1 (Figura 9.94).

```
clases_mesma <- mesma(img = proc_20m_crop, em = endmem,
                      method = "NNLS")
```

Los valores representan la probabilidad de la presencia estimada de cada endmember, los valores deben tender al rango $[0, 1]$.

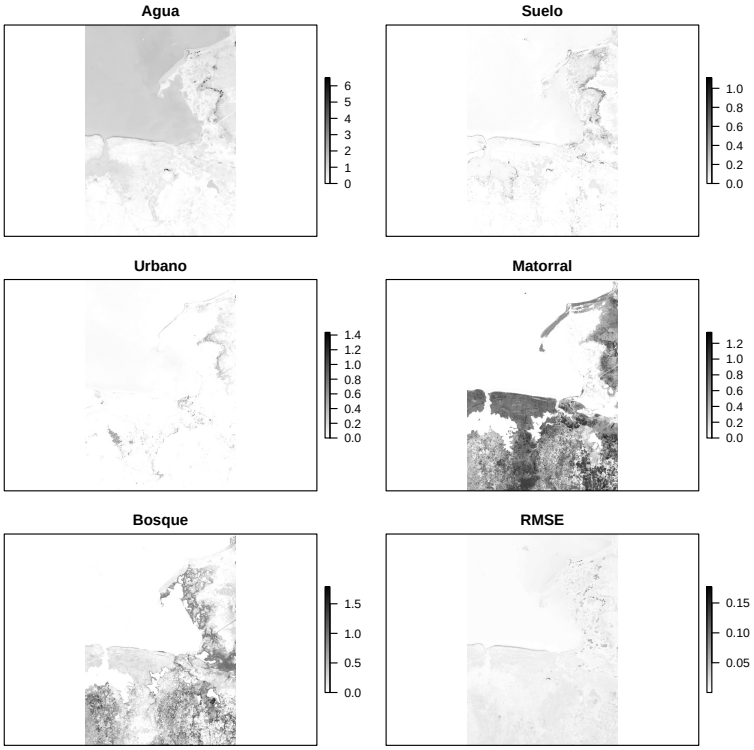


Figura 9.94: Coberturas estimadas con MESMA, y error RMSE.

Clasificación No Supervisada en R

La clasificación no supervisada se puede definir como la identificación de grupos o patrones dentro de datos multiespectrales. La clasificación no supervisada es la definición, identificación, etiquetado y mapeo de estas clases naturales.

9.35 Método Agrupamiento kmeans

El **algoritmo Kmeans** es un algoritmo iterativo que intenta dividir el conjunto de datos en K subgrupos distintos no superpuestos (clústeres) donde cada punto de datos pertenece a un solo grupo, he intenta hacer que los puntos de datos *intra-clúster* sean lo más similares posibles y al mismo tiempo mantiene los clústeres lo más diferentes (lejos) posible. Asigna puntos de datos a un grupo de modo que la suma de la distancia al cuadrado entre los puntos de datos y el centroide del grupo (media aritmética de todos los puntos de datos que pertenecen a ese grupo) sea mínima. Mientras menos variación tengamos dentro de los conglomerados, más homogéneos (similares) serán los puntos de datos dentro del mismo conglomerado.

La forma en que funciona el algoritmo kmeans es la siguiente:

- Especificar número de agrupaciones K.
- Inicializa los centroides barajando primero el conjunto de datos y luego seleccionando aleatoriamente K puntos de datos para los centroides sin reemplazo.
- Sigue iterando hasta que no haya cambios en los centroides. Es decir, la asignación de puntos de datos a grupos no está cambiando.

- Calcula la suma de la distancia al cuadrado entre los puntos de datos y todos los centroides.
- Asigna cada punto de datos al grupo más cercano (centroide).
- Calcula los centroides de los grupos tomando el promedio de todos los puntos de datos que pertenecen a cada grupo.

9.35.1 Algunas Consideraciones

Dado que los algoritmos de agrupación en clústeres que incluyen kmeans utilizan mediciones basadas en la distancia para determinar la similitud entre los puntos de datos, se recomienda *estandarizar* los datos para que tengan una *media de cero* y *una desviación estándar de uno*, ya que casi siempre las características de cualquier conjunto de datos tendrían diferentes unidades de medida. como la edad frente a los ingresos.

Dada la naturaleza *iterativa* de kmeans y la inicialización aleatoria de los centroides al comienzo del algoritmo, diferentes inicializaciones pueden conducir a diferentes clusters, ya que el algoritmo de kmeans puede quedarse atascado en un óptimo local y no converger al óptimo global. Por lo tanto, se recomienda ejecutar el algoritmo utilizando *diferentes inicializaciones de centroides* y seleccionar los resultados de la ejecución que arrojaron la suma más baja de la distancia al cuadrado.

9.36 Cálculo del Número de Grupos

El primer paso para aplicar el método `kmeans()` del paquete *stats* que se encuentra en el grupo base de R(R Core Team [2021]). Requiere construir un objeto *matrix* y para efectos de velocidad de procesamiento extraer un subconjunto de los datos espectrales.

Primero convertimos el objeto ráster a matrix con la función `values()`:

```
data <- values(proc_20m_crop)
```

Y usamos `sample(x, size)` que toma una muestra del tamaño especificado `size` de los elementos de `x` usando con o sin reemplazo, para construir un índice para rescatar un subconjunto de los datos:

```
muestras <- data[sample(x= nrow(data), size= 2000), ]
```

Para normalizar o estandarizar los datos utilizamos la función `scale(x, center, scale)` también presente en los paquetes *base*, centramos y escalamos los datos para que cumplan las condiciones de tener un promedio (μ) cero y una desviación estándar (σ) uno:

```
normalizado <- scale(x = muestras,
                     center = T,
                     scale = T)
```

La función `kmeans(x, centers, nstart, iter.max)` requiere de algunos parámetros para su uso, `x` debe ser una matriz numérica, cada columna corresponderá a una banda. `centers` corresponde al número k, `nstart` define el número de reinicios aleatorios para cada centro y finalmente `iter.max` el número de iteraciones permitidas.

```
kmeans <- kmeans(x = normalizado,
                 centers = 3,
                 nstart=5,
                 iter.max = 15)
```

El resultado es un objeto *kmeans*, que se compone de varios elementos, y detallaremos las más importantes.

El número de muestras en cada kluster (*total muestra* = 2000):

```
kmeans$size
```

```
[1] 592 535 873
```

El objetivo es conseguir una gran similitud dentro de cada grupo y una baja similitud entre cada grupo.

Estadísticamente:

- Alta similitud dentro de un grupo = baja varianza dentro del grupo, o *withinss*.
- Baja similitud entre los grupos = alta varianza entre los conglomerados, o *betweenss*.

Luego, si se calcula toda la varianza en los datos (*totss*). En la *agrupación óptima*, dado que las agrupaciones son muy dife-

rentes entre sí, la mayor parte de la varianza total se explica por la varianza entre los grupos. Y, por supuesto, dado que la varianza dentro de cada grupo es muy pequeña, explicaría solo una pequeña fracción de la varianza total en los datos.

En resumen, el objetivo es maximizar *betweenss/totss*. Sin embargo, si elige un número de cluster igual que el número de observaciones, entonces *totalss* es exactamente igual a *betweenss* y la proporción deseada será 1 (o 100%).

Con nuestros datos:

```
round(kmeans$betweenss/kmeans$totss, digits = 2) * 100
```

```
[1] 75
```

Así, el 77% es la medida de la *varianza total en el conjunto de datos* que se *explica por el cluster*.

Entonces, para obtener un alto porcentaje, puede simplemente aumentar el número de clústeres, pero se pierde el punto de la agrupación. Una forma de lidiar con esto es usar el *método del codo* (9.36.1) para elegir el número razonable de clústeres.

Entonces, la idea es encontrar un valor de k para el cual el modelo no esté sobreajustado y, al mismo tiempo, agrupe los datos según la distribución real. A continuación, resolveremos este problema de encontrar el mejor número de grupos.

9.36.1 Método Elbow

El **método elbow** (codo) analiza el porcentaje de varianza explicado como una función del número de clusters: se debe elegir un número de grupos tal que agregar otro no proporcione un modelamiento mucho mejor de los datos.

Más precisamente, si se grafica el porcentaje de varianza explicado por los cluster contra el número de cluster, los primeros agregarán mucha información (explicando mucha varianza), pero en algún momento la ganancia será marginal, dando un cambio en la pendinete del perfil del gráfico.

El número de grupos se elige en este punto, de ahí el *criterio del codo*. Este codo no siempre puede identificarse claramente.

```
ver_resid <- sapply(1:10,
                    function(k){
                      kmeans(x = normalizado,
                             centers = k,
                             nstart=15,
                             iter.max = 15 )$tot.withinss
                      })
```

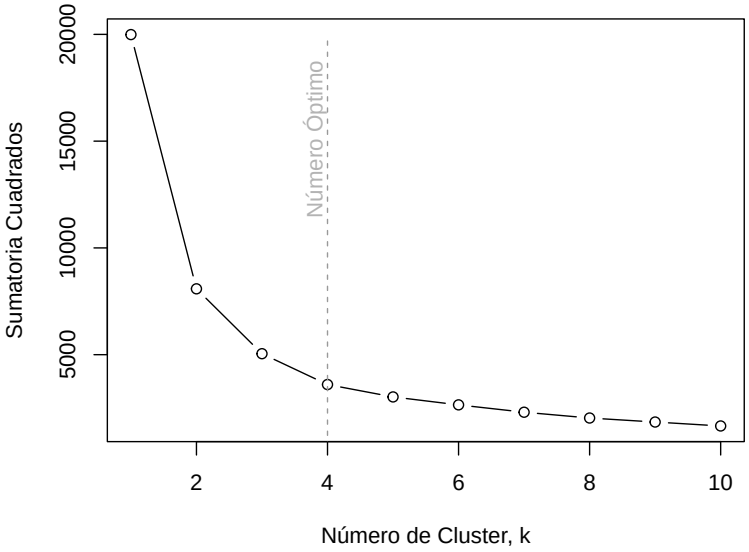


Figura 9.95: Número Óptimo de Cluster, método de Elbow.

Por lo tanto, para $k = 4$, la relación *betweenss/totss* tiende a cambiar lentamente y permanece menos cambiante en comparación con otros k's. Por lo tanto, para estos datos, $k = 5$ debería ser una buena opción para el número de clusters (Figura 9.95).

```
kmeans <- kmeans(x = normalizado,
                 centers = 4,
                 nstart=5,
                 iter.max = 15)
```

```
round(kmeans$betweenss/kmeans$totss * 100,2)
```

```
[1] 81.99
```

9.37 Clasificación No Supervisada unsuperClass()

En el paquete *RStoolbox* vamos a utilizar la función `unsuperClass(img, nSamples, nClasses, nStarts,nIter, norm)`, que automáticamente realiza las tareas de extracción de una muestra, normalizar y aplicar *kmeans*.

El parámetro `img` corresponde a un modelo ráster, `nSamples` el tamaño de la muestra, `nStarts` corresponde al número de veces que se reinician los centroides por cluster, `nIter` máximo número de iteraciones permitido y `norm` para normalizar la data.

```
ucl <- unsuperClass(img= proc_20m_crop,
                    nSamples = 10000,
                    nClasses = 4,
                    nStarts = 5,
                    nIter = 15,
                    norm = T)
```

Para estudiar el resultado del modelo kmeans, se accesa al modelo con:

```
ucl$model
```

y la imagen clasificada (Figura 9.96):

```
ucl$map
```

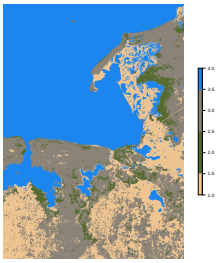


Figura 9.96: Clasificación no supervisada (kmeans, cuatro categorías).

9.38 Análisis de Componentes Principales (PCA)

La aplicación de PCA es la **reducción de dimensionalidad** (variables), perdiendo la menor cantidad de información (varianza) posible: cuando contamos con un gran número de variables cuantitativas posiblemente correlacionadas (indicativo de existencia de información redundante), PCA permite reducirlas a un número menor de variables transformadas (componentes principales) que expliquen gran parte de la variabilidad en los datos.

La función `rasterPCA(img)` del paqute *RStoolbox* aplica PCA a un objeto ráster `img`:

```
rpc <- rasterPCA(img = proc_20m_crop)
```

```
names(rpc)
```

```
[1] "call"  "model" "map"
```

Los componentes principales se extraen *secuencialmente*, el primer componente principal representa *la mayor variación* en los datos. Intuitivamente, el primer componente principal es un vector que apunta en la dirección en la que los datos están más "dispersos". El segundo componente principal se configura de manera similar, pero con la restricción adicional de que no debe estar correlacionado con el primero. Cada componente principal subsiguiente captura un porcentaje cada vez más bajo de variación en los datos y no está correlacionado con los componentes principales extraídos previamente.

Puede haber tantos componentes principales como campos numéricos en los datos. Sin embargo, normalmente es posible capturar la variación en los datos utilizando los primeros componentes principales en lugar del conjunto completo de campos numéricos originales.

Usando nuestros datos:

```
summary(rpc$model)
```

```
Importance of components:
                          Comp.1     Comp.2     Comp.3      Comp.4
Standard deviation     0.2676557 0.08418481 0.04035661 0.012318347
Proportion of Variance 0.8875991 0.08780742 0.02017870 0.001880046
Cumulative Proportion  0.8875991 0.97540649 0.99558519 0.997465237
                           Comp.5       Comp.6      Comp.7
Standard deviation     0.009474782 0.0071721600 0.005235233
Proportion of Variance 0.001112249 0.0006373289 0.000339575
Cumulative Proportion  0.998577486 0.9992148150 0.999554390
                            Comp.8       Comp.9      Comp.10
Standard deviation     0.0039348223 0.0035497892 2.807506e-03
Proportion of Variance 0.0001918289 0.0001561237 9.765738e-05
Cumulative Proportion  0.9997462189 0.9999023426 1.000000e+00
```

Encontramos que el primer componente explica un 88.75 de la varianza [8] en los datos, el segundo explica una varianza acumulada de 97.54, el tercero un 99.56.

Así, cada componente principal generado por PCA será una combinación lineal de las variables originales, y serán además independientes o no correlacionadas entre sí.

Si revisamos los tres primeros componentes vemos el peso de

[8] La **varianza** es la desviación estándar elevada al cuadrado. O al revés, la desviación estándar es la raíz cuadrada de la varianza. La desviación estándar se hace para poder trabajar en las *unidades de medida iniciales*.

cada variable (nuestras bandas espectrales) en la composición de cada componente principal.

```
loadings(rpc$model)[,1:3]
```

```
        Comp.1     Comp.2      Comp.3
B1   0.01610711  0.1247363  0.25276544
B2   0.04765859  0.1597188  0.36147302
B3   0.11275525  0.1307055  0.44535795
B4   0.16145098  0.2563452  0.43551327
B5   0.23093928  0.1305818  0.41477686
B6   0.33959696 -0.3043268  0.16047681
B7   0.39383405 -0.4321115  0.08621945
B8A  0.44436667 -0.4643966 -0.05857362
B11  0.52998220  0.3668333 -0.39617846
B12  0.39570428  0.4790786 -0.22763280
```

Quizás, si expresamos en términos porcentuales los aportes de cada variable comprenderemos el objetivo del método PCA, una reducción d evariables expresando en un nuevo set de variables que resumen mejor los datos.

Extraemos las desviaciones estándar por variable en cada componente, la elevamos al cuadrado y realizamos la suma por cada uno,

```
componentes <- rpc$model$loadings
varianza <- componentes[1:10,1:3]^2
suma_columna <- colSums(varianza)
```

y calculamos el porcentaje (Cuadro 9.10):

```
porcentual <- (varianza * 100) /suma_columna
```

9.38.1 Consideraciones de Uso

Los componentes principales se pueden usar en lugar del grupo de variables originales en los métodos estadísticos descritos, evitando los problemas que pueden ocurrir cuando se usan variables altamente correlacionadas, pero se debe considerar que la interpretación se dificulta.

El método se puede utilizar para determinar qué grupos de

Cuadro 9.10: Importancia de cada banda en la construcción de los primeros tres componentes principales.

	Comp.1	Comp.2	Comp.3
B1	0.03	1.56	6.39
B2	0.23	2.55	13.07
B3	1.27	1.71	19.83
B4	2.61	6.57	18.97
B5	5.33	1.71	17.20
B6	11.53	9.26	2.58
B7	15.51	18.67	0.74
B8A	19.75	21.57	0.34
B11	28.09	13.46	15.70
B12	15.66	22.95	5.18

Nota:
Valores expresados en %.

variables probablemente estén altamente relacionados entre sí y ayudar a guiar las decisiones sobre cuáles omitir de un proceso.

La capacidad de reducir una gran cantidad de variables en una pequeña cantidad de componentes principales suele ser una ventaja al visualizar las relaciones en los datos.

Finalmente, cada componente se puede visualizar como una capa (Figura 9.97) o usarlos en una composición RGB (Figura 9.98).

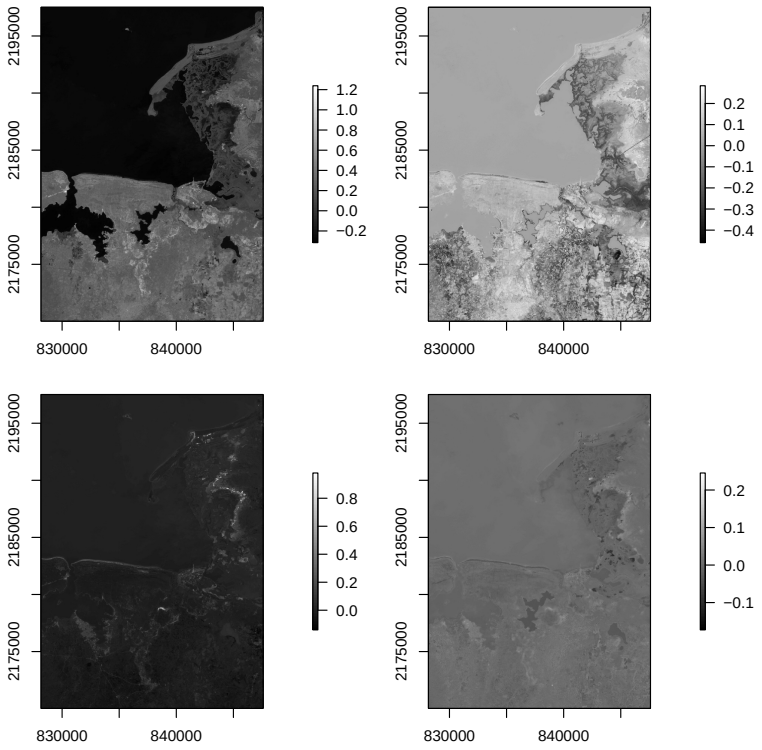


Figura 9.97: Representación gráfica de los cuatro primeros componentes.

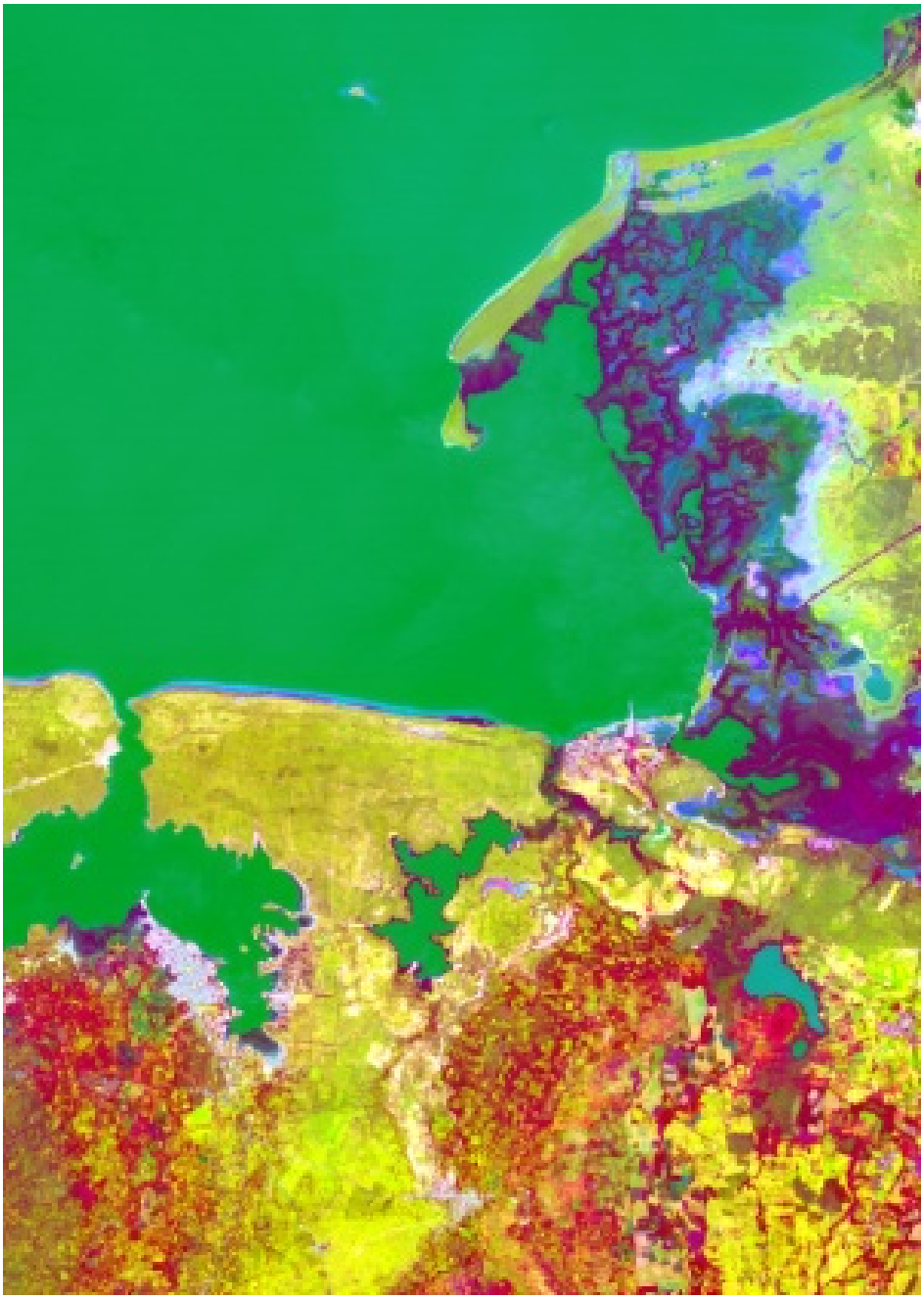


Figura 9.98: Composición RGB, r=comp.1, g=comp.2, b=comp.3.

Índice alfabético

Bibliografía

Chapter 1 - a systematic view of remote sensing. In Shunlin Liang and Jindi Wang, editors, *Advanced Remote Sensing (Second Edition)*, pages 1–57. Academic Press, second edition edition, 2020. ISBN 978-0-12-815826-5. DOI: https://doi.org/10.1016/B978-0-12-815826-5.00001-5. URL `https://www.sciencedirect.com/science/article/pii/B9780128158265000015`.

JJ Allaire, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. *rmarkdown: Dynamic Documents for R*, 2021. URL `https://github.com/rstudio/rmarkdown`. R package version 2.11.

Christopher Amante and Barry W Eakins. Etopo1 arc-minute global relief model: procedures, data sources and analysis. 2009.

Tim Appelhans, Florian Detsch, Christoph Reudenbach, and Stefan Woellauer. *mapview: Interactive Viewing of Spatial Data in R*, 2020a. URL `https://CRAN.R-project.org/package=mapview`. R package version 2.9.0.

Tim Appelhans, Kenton Russell, and Lorenzo Busetto. *mapedit: Interactive Editing of Spatial Data in R*, 2020b. URL `https://CRAN.R-project.org/package=mapedit`. R package version 0.6.0.

Alice M Baldridge, SJ Hook, CI Grove, and GJRSoE Rivera. The aster spectral library version 2.0. *Remote Sensing of Environment*, 113(4):711–715, 2009.

A. Bannari, D. Morin, F. Bonn, and A. R. Huete. A review of vegetation indices. *Remote Sensing Reviews*, 13(1-2):95–120, 1995.

Richard A. Becker, Allan R. Wilks. R version by Ray Brownrigg. Enhancements by Thomas P Minka, and Alex Deckmyn. *maps: Draw Geographical Maps*, 2018. URL `https://CRAN.R-project.org/package=maps`. R package version 3.3.0.

Henrik Bengtsson. *R.utils: Various Programming Utilities*, 2020. URL `https://CRAN.R-project.org/package=R.utils`. R package version 2.10.1.

Joseph K. Berry. *Beyond Mapping: Concepts, Algorithms, and Issues in GIS*. October 1996. ISBN 978-0-470-23676-5.

G. S. Birth and G. McVey. Measuring the color of growing turf with a reflectance spectrophotometer. *Agronomy Journal*, 60: 640–643, 1968.

Roger Bivand, Tim Keitt, and Barry Rowlingson. *rgdal: Bindings for the 'Geospatial' Data Abstraction Library*, 2021. URL `https://CRAN.R-project.org/package=rgdal`. R package version 1.5-23.

Roger S. Bivand, Edzer Pebesma, and Virgilio Gomez-Rubio. *Applied spatial data analysis with R, Second edition*. Springer, NY, 2013. URL `https://asdar-book.org/`.

Newton Henry Black. *Practical Physics*. Macmillan, 1913. URL `https://books.google.cl/books?id=BT8AAAAAYAAJ`.

Scott Chasalow. *combinat: combinatorics utilities*, 2012. URL `https://CRAN.R-project.org/package=combinat`. R package version 0.0-8.

María Martín; Emilio Chuvievo. Cartografia de grandes incendios forestales en la península ibérica a partir de imágenes noaa-avhrr. *Serie Geográfica*, pages 109–128, 1998.

William S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the*

American Statistical Association, 74(368):829–836, 1979. DOI: 10.1080/01621459.1979.10481038. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1979.10481038.

William S. Cleveland. Lowess: A program for smoothing scatterplots by robust locally weighted regression. *The American Statistician*, 35(1):54–54, 1981. ISSN 00031305. URL http://www.jstor.org/stable/2683591.

Helen Couclelis. People manipulate objects (but cultivate fields): Beyond the raster-vector debate in gis. In A. U. Frank, I. Campari, and U. Formentini, editors, *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, pages 65–77, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg. ISBN 978-3-540-47333-6.

Paul SP Cowpertwait and Andrew V Metcalfe. *Introductory time series with R*. Springer Science & Business Media, 2009.

Gábor Csárdi, Kuba Podgórski, and Rich Geldreich. *zip: Cross-Platform 'zip' Compression*, 2021. URL https://CRAN.R-project.org/package=zip. R package version 2.2.0.

Daniel Baston. *exactextractr: Fast Extraction from Raster Datasets using Polygons*, 2021. URL https://CRAN.R-project.org/package=exactextractr. R package version 0.6.1.

Shirley M Davis, David A Landgrebe, Terry L Phillips, Philip H Swain, Roger M Hoffer, John C Lindenlaub, and Leroy F Silva. Remote sensing: the quantitative approach. *New York*, 1978.

Jean Pierre Decorps. *GADMTools: Easy Use of 'GADM' Maps*, 2021. URL https://CRAN.R-project.org/package=GADMTools. R package version 3.8-2.

David J Diner, Gregory P Asner, Roger Davies, Yuri Knyazikhin, Jan-Peter Muller, Anne W Nolin, Bernard Pinty, Crystal B Schaaf, and Julienne Stroeve. New directions in earth observing: Scientific applications of multiangle remote sensing.

Bulletin of the American Meteorological Society, 80(11):2209–2228, 1999.

Loïc Dutrieux and Ben DeVries. bfastSpatial: Set of utilities and wrappers to perform change detection on satellite image time-series, December 2014. URL `https://github.com/loicdtx/bfastSpatial`.

Ian S Evans. General geomorphometry, derivatives of altitude, and descriptive statistics. In *Spatial analysis in geomorphology*, pages 17–90. Routledge, 2019.

Jeffrey S. Evans. *spatialEco*, 2021. URL `https://github.com/jeffreyevans/spatialEco`. R package version 1.3-6.

Gudina L Feyisa, Henrik Meilby, Rasmus Fensholt, and Simon R Proud. Automated water extraction index: A new technique for surface water mapping using landsat imagery. *Remote Sensing of Environment*, 140:23–35, 2014.

James John Flannery. The relative effectiveness of some common graduated point symbols in the presentation of quantitative data. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 8(2):96–109, 1971.

Vojtěch Franc, Václav Hlaváč, and Mirko Navara. Sequential coordinate-wise algorithm for the non-negative least squares problem. In *International Conference on Computer Analysis of Images and Patterns*, pages 407–414. Springer, 2005.

Antony Galton. A formal theory of objects and fields. In Daniel R. Montello, editor, *Spatial Information Theory*, pages 458–473, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-45424-3.

Bo-cai Gao. Ndwia normalized difference water index for remote sensing of vegetation liquid water from space. *Remote Sensing of Environment*, 58(3):257–266, 1996.

Jose A. Guijarro. *climatol: Climate Tools (Series Homogenization and Derived Products)*, 2019. URL `https://CRAN.R-project.org/package=climatol`. R package version 3.1.2.

Qiandong Guo, Ruiliang Pu, Jialin Li, and Jun Cheng. A weighted normalized difference water index for water extraction using landsat imagery. *International journal of remote sensing*, 38(19):5430–5445, 2017.

Bryan A. Hanson. *SpecHelpers: Spectroscopy Related Utilities*, 2017. URL `github.com/bryanhanson/SpecHelpers`. R package version 0.2.7.

Ian Harris, Timothy J. Osborn, Phil Jones, and David Lister. Version 4 of the cru ts monthly high resolution gridded multivariate climate dataset. *Sci Data*, 7(1):109, 2020. ISSN 2052-4463. DOI: 10.1038/s41597-020-0453-3. URL `https://doi.org/10.1038/s41597-020-0453-3`.

T Hengl and HI Reuter. *Geomorphometry: Concepts, software, applications*. Number 33. Elsevier, 2009. ISBN 9780080921884.

Robert J. Hijmans. *geosphere: Spherical Trigonometry*, 2019. URL `https://CRAN.R-project.org/package=geosphere`. R package version 1.5-10.

Robert J. Hijmans. *raster: Geographic Data Analysis and Modeling*, 2021. URL `https://CRAN.R-project.org/package=raster`. R package version 3.4-10.

Z. A. Holden, A. M. S. Smith, P. Morgan, M. G. Rollins, and P. E. Gessler. Evaluation of novel thermally enhanced spectral indices for mapping fire perimeters and comparisons with fire atlas data. *International Journal of Remote Sensing*, 26(21): 4801–4808, 2005. DOI: 10.1080/01431160500239008. URL `https://doi.org/10.1080/01431160500239008`.

Jeffrey Hollister, Tarak Shah, Alec L. Robitaille, Marcus W. Beck, and Mike Johnson. *elevatr: Access Elevation Data from Various APIs*, 2020. URL `https://github.com/usepa/elevatr/`. R package version 0.3.1, doi:10.5281/zenodo.4282962.

Berthold KP Horn. Hill shading and the reflectance map. *Proceedings of the IEEE*, 69(1):14–47, 1981.

Chuanmin Hu. A novel ocean color index to detect floating algae in the global oceans. *Remote Sensing of Environment*, 113(10): 2118–2129, 2009.

A. R. Huete. A soil-adjusted vegetation index (savi), 1988.

Alfredo R. Huete, Kamel Didan, Yosio E. Shimabukuro, Piyachat Ratana, Scott R. Saleska, Lucy R. Hutyra, Wenze Yang, Ramakrishna R. Nemani, and Ranga Myneni. Amazon rainforests green-up with sunlight in dry season. *Geophysical Research Letters*, 33(6), 2006. DOI: https://doi.org/10.1029/2005GL025583. URL `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2005GL025583`.

A.R. Huete, C. Justice, and W. Van Leeuwen. Modis vegetation index (mod 13). algorithm theoretical basis document version 3, 1999.

Ross Ihaka and Robert Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996. DOI: 10.1080/10618600.1996.10474713. URL `https://www.tandfonline.com/doi/abs/10.1080/10618600.1996.10474713`.

JabRef Development Team. Jabref — an open-source, cross-platform citation and reference management software, 2021. URL `https://www.jabref.org/`.

Mark E Jakubauskas, David R Legates, Jude H Kastens, et al. Harmonic analysis of time-series avhrr ndvi data. *Photogrammetric engineering and remote sensing*, 67(4):461–470, 2001.

Susan K Jenson and Julia O Domingue. Extracting topographic structure from digital elevation data for geographic information system analysis. *Photogrammetric engineering and remote sensing*, 54(11):1593–1600, 1988.

Z. Jiang, A. R. Huete, K. Didan, and T. Miura. Development of a two-band enhanced vegetation index without a blue band, 2008.

Arnon Karnieli, Yoram J. Kaufman, Lorraine Remer, and Andrew Wald. Afri — aerosol free vegetation index. *Remote Sensing of Environment*, 77(1):10–21, 2001. ISSN 0034-4257. DOI: https://doi.org/10.1016/S0034-4257(01)00190-0. URL `https://www.sciencedirect.com/science/article/pii/S0034425701001900`.

D. Kaufman, Y. J.;Tanre. Atmospherically resistant vegetation index (arvi) for eos-modis, 1992.

Dan Kelley and Clark Richards. *oce: Analysis of Oceanographic Data*, 2021. URL `https://CRAN.R-project.org/package=oce`. R package version 1.4-0.

CH Key and NC Benson. Landscape assessment: remote sensing of severity, the normalized burn ratio and ground measure of severity, the composite burn index. *FIREMON: Fire effects monitoring and inventory system Ogden, Utah: USDA Forest Service, Rocky Mountain Res. Station*, 2005.

C.H. Key, N. Benson, D. Ohlen, S. Howard, R. McKinley, and Zhu Z. The normalized burn ratio and relationships to burn severity: ecology, remote sensing and implementation. San Diego, CA, 2002. J.D. Greer, ed. Rapid Delivery of Remote Sensing Products.

Rami Krispin. *TSstudio: Functions for Time Series Analysis and Forecasting*, 2020. URL `https://CRAN.R-project.org/package=TSstudio`. R package version 0.1.6.

Fred A Kruse, AB Lefkoff, JW Boardman, KB Heidebrecht, AT Shapiro, PJ Barloon, and AFH Goetz. The spectral image processing system (sips)—interactive visualization and analysis of imaging spectrometer data. *Remote sensing of environment*, 44(2-3):145–163, 1993.

Benjamin Leutner, Ned Horning, and Jakob Schwalb-Willmann. *RStoolbox: Tools for Remote Sensing Data Analysis*, 2019. URL `https://CRAN.R-project.org/package=RStoolbox`. R package version 0.2.6.

Kuo-Nan Liou. *An introduction to atmospheric radiation*. Elsevier, 2002.

Christopher T. Lloyd, Heather Chamberlain, David Kerr, Greg Yetman, Linda Pistolesi, Forrest R. Stevens, Andrea E. Gaughan, Jeremiah J. Nieves, Graeme Hornby, Kytt MacManus, Parmanand Sinha, Maksym Bondarenko, Alessandro Sorichetta, and Andrew J. Tatem. Global spatio-temporally harmonised datasets for producing high-resolution gridded population distribution datasets. *Big Earth Data*, 3(2):108–139, 2019. DOI: 10.1080/20964471.2019.1625151. URL `https://doi.org/10.1080/20964471.2019.1625151`. PMID: 31565697.

Lawrence W Martz and Eeltje De Jong. Catch: A fortran program for measuring catchment area from digital elevation models. *Computers & Geosciences*, 14(5):627–640, 1988.

Matteo Mattiuzzi and Florian Detsch. *MODIS: Acquisition and Processing of MODIS Products*, 2021. URL `https://github.com/MatMatt/MODIS`. R package version 1.2.3.9000.

Stuart K McFeeters. The use of the normalized difference water index (ndwi) in the delineation of open water features. *International journal of remote sensing*, 17(7):1425–1432, 1996.

Susan K. Meerdink, Simon J. Hook, Dar A. Roberts, and Elsa A. Abbott. The ecostress spectral library version 1.0. *Remote Sensing of Environment*, 230:111196, 2019. ISSN 0034-4257. DOI: https://doi.org/10.1016/j.rse.2019.05.015. URL `https://www.sciencedirect.com/science/article/pii/S0034425719302081`.

Md Mia and Yasuhiro Fujimitsu. Mapping hydrothermal altered mineral deposits using landsat 7 etm+ image in and around kuju volcano, kyushu, japan. *Journal of Earth System Science*, 121, 08 2012. DOI: 10.1007/s12040-012-0211-9.

Kirill Müller. *here: A Simpler Way to Find Your Files*, 2020. URL `https://CRAN.R-project.org/package=here`. R package version 1.0.1.

Martin Morgan. *BiocManager: Access the Bioconductor Project Package Repository*, 2021. URL `https://CRAN.R-project.org/package=BiocManager`. R package version 1.30.16.

David G Morris and Richard G Heerdegen. Automatically derived catchment boundaries and channel networks and their hydrological applications. *Geomorphology*, 1(2):131–141, 1988.

Babak Naimi. *rts: Raster Time Series Analysis*, 2019. URL `https://CRAN.R-project.org/package=rts`. R package version 1.0-49.

Minato Nakazawa. *pyramid: Draw Population Pyramid*, 2019. URL `https://CRAN.R-project.org/package=pyramid`. R package version 1.5.

Erich Neuwirth. *RColorBrewer: ColorBrewer Palettes*, 2014. URL `https://CRAN.R-project.org/package=RColorBrewer`. R package version 1.1-2.

Victor Olaya. *Sistemas de Información Geográfica*. lulu.com, June 2020. ISBN 9781716777660.

Jeroen Ooms. *magick: Advanced Graphics and Image-Processing in R*, 2021. URL `https://CRAN.R-project.org/package=magick`. R package version 2.7.2.

Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

Eric Pante and Benoit Simon Bouhet. Analysing bathymetric data in r with marmap. 2015.

Eric Pante and Benoit Simon-Bouhet. marmap: A package for importing, plotting and analyzing bathymetric and topographic data in r. *PLoS ONE*, 8(9):e73051, 2013. doi:10.1371/journal.pone.0073051.

Gregoire Pau, Florian Fuchs, Oleg Sklyar, Michael Boutros, and Wolfgang Huber. Ebimage an r package for image processing with applications to cellular phenotypes. *Bioinformatics*, 26 (7):979–981, 2010. DOI: 10.1093/bioinformatics/btq046.

Edzer Pebesma. Simple Features for R: Standardized Support for Spatial Vector Data. *The R Journal*, 10(1):439–446, 2018. DOI: 10.32614/RJ-2018-009. URL `https://doi.org/10.32614/RJ-2018-009`.

Edzer J. Pebesma and Roger S. Bivand. Classes and methods for spatial data in R. *R News*, 5(2):9–13, November 2005. URL `https://CRAN.R-project.org/doc/Rnews/`.

YAN Pei, You-jing ZHANG, and ZHANG Yuan. A study on information extraction of water system in semi-arid regions with the enhanced water index (ewi) and gis based noise remove techniques [j]. *Remote Sensing Information*, 6, 2007.

D Jo Pennock, BJ Zebarth, and E De Jong. Landform classification and soil distribution in hummocky terrain, saskatchewan, canada. *Geoderma*, 40(3-4):297–315, 1987.

Oscar Perpiñán and Robert Hijmans. *rasterVis: Visualization Methods for Raster Data*, 2021. URL `https://oscarperpinan.github.io/rastervis/`. R package version 0.50.3.

Carla Pezzulo, Graeme M. Hornby, Alessandro Sorichetta, Andrea E. Gaughan, Catherine Linard, Tomas J. Bird, David Kerr, Christopher T. Lloyd, and Andrew J. Tatem. Sub-national mapping of population pyramids and dependency ratios in africa and asia. *Scientific Data*, 4(1):170089, 2017. ISSN 2052-4463. DOI: 10.1038/sdata.2017.89. URL `https://doi.org/10.1038/sdata.2017.89`.

Marius Philipp. *S5Processor: Sentinel-5P L2 Processing Tool*, 2021a. R package version 0.1.0.

Marius Philipp. *rHarmonics: R package for harmonic modelling of time-series data*, 2021b. R package version 0.1.0.

David Pierce. *ncdf4: Interface to Unidata netCDF (Version 4 or Earlier) Format Data Files*, 2019. URL `https://CRAN.R-project.org/package=ncdf4`. R package version 1.17.

B. Pinty and M. M. Verstraete. Gemi: a non-linear index to monitor global vegetation from satellites. *Plant Ecology*, 101 (1):15–20, 1992.

PROJ contributors. *PROJ coordinate transformation software library*. Open Source Geospatial Foundation, 2021. URL `https://proj.org/`.

J. Qi, A. Chehbouni, A. R. Huete, Y. H. Kerr, and S. Sorooshian. A modified soil adjusted vegetation index. *Remote Sensing of Environment*, 48(2):119–126, 1994.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL `https://www.R-project.org/`.

John Alan Richards. *Remote sensing digital image analysis*. Springer, fifth edition, 2013. ISBN 978-3-642-30061-5.

Dar A Roberts, M Gardner, Rick Church, S Ustin, G Scheer, and RO Green. Mapping chaparral in the santa monica mountains using multiple endmember spectral mixture models. *Remote sensing of environment*, 65(3):267–279, 1998.

Jr. Rouse, J.W., R.H. Haas, J.A. Schell, and D.W. Deering. Monitoring the vernal advancement and retrogradation (green wave effect) of natural vegetation, 1973.

Barry Rowlingson. *geonames: Interface to the "Geonames" Spatial Query Web Service*, 2019. URL `https://CRAN.R-project.org/package=geonames`. R package version 0.999.

RStudio Team. *RStudio: Integrated Development Environment for R*. RStudio, PBC., Boston, MA, 2020. URL `http://www.rstudio.com/`.

Peter Shary, LARISA Sharaya, and Andrey Mitusov. The problem of scale-specific and scale-free approaches in geomorphometry. *Geografia Fisica e Dinamicca Quaternaria*, 28:81, 01 2005.

Peter A Shary. Land surface in gravity points classification by a complete system of curvatures. *Mathematical Geology*, 27(3): 373–390, 1995.

Peter A Shary, Larisa S Sharaya, and Andrew V Mitusov. Fundamental quantitative methods of land surface analysis. *Geoderma*, 107(1-2):1–32, 2002.

Robert H Shumway and David S Stoffer. Time series analysis and its applications: with r examples, 2017.

David Sjoberg. *hablar: Non-Astonishing Results in R*, 2020. URL `https://CRAN.R-project.org/package=hablar`. R package version 0.3.0.

Philip N. Slater. Remote sensing. optics and optical systems. *Remote sensing. Optics and optical systems*, 1980.

M. Smith, M. Goodchild, and P. Longley. *Geospatial analysis - a comprehensive guide to principles, techniques and software tools (6. ed.)*. 2018.

Andy South. *rworldxtra: Country boundaries at high resolution.*, 2012. URL `https://CRAN.R-project.org/package=rworldxtra`. R package version 1.01.

Andy South. *rnaturalearth: World Map Data from Natural Earth*, 2017. URL `https://CRAN.R-project.org/package=rnaturalearth`. R package version 0.1.0.

Reto Stauffer, Georg J. Mayr, Markus Dabernig, and Achim Zeileis. Somewhere over the rainbow: How to make effective use of colors in meteorological visualizations. *Bulletin of the American Meteorological Society*, 96(2):203–216, 2009. DOI: 10.1175/BAMS-D-13-00155.1.

Mohamed Sultan, Raymond E Arvidson, Neil C Sturchio, and Edward A Guinness. Lithologic mapping in arid regions with landsat thematic mapper data: Meatiq dome, egypt. *Geological Society of America Bulletin*, 99(6):748–762, 1987.

Michael D. Sumner. *graticule: Meridional and Parallel Lines for Maps*, 2021. URL `https://CRAN.R-project.org/package=graticule`. R package version 0.1.6.

Susumu Tanimura, Chusi Kuroiwa, and Tsutomu Mizota. Proportional symbol mapping in r. *Journal of Statistical Software*, 15:1–7, 2006.

Andrew J. Tatem, Andres J. Garcia, Robert W. Snow, Abdisalan M. Noor, Andrea E. Gaughan, Marius Gilbert, and Catherine Linard. Millennium development health metrics: where do africa's children and women of childbearing age live? *Population Health Metrics*, 11(1):11, 2013. ISSN 1478-7954. DOI: 10.1186/1478-7954-11-11. URL `https://doi.org/10.1186/1478-7954-11-11`.

Martijn Tennekes. tmap: Thematic maps in R. *Journal of Statistical Software*, 84(6):1–39, 2018. DOI: 10.18637/jss.v084.i06.

Andy Teucher and Kenton Russell. *rmapshaper: Client for 'mapshaper' for 'Geospatial' Operations*, 2021. URL `https://CRAN.R-project.org/package=rmapshaper`. R package version 0.4.5.

W. R. Tobler. Analytical cartography. *The American Cartographer*, 3(1):21–31, 1976. DOI: 10.1559/152304076784080230. URL `https://doi.org/10.1559/152304076784080230`.

John W Tukey. *Exploratory data analysis*, volume 2. Reading, Mass., 1977. ISBN 9780201076165.

M.P.J. van der Loo and E. De Jonge. *Learning RStudio for R Statistical Computing*. Community experience distilled. Lightning Source, 2012. ISBN 9781782160601. URL `https://books.google.cl/books?id=vOTHmAEACAAJ`.

E Vermote. Mod09a1 modis / terra surface reflectance 8-day l3 global 500m sin grid v006, 2015.

Heinrich Walter, E Harnickell, and Dieter Mueller-Dombois. Climate diagram maps. *Ind. Countries and the ecological climatic regions of the earth. Suppl. to the veg. monographs*, 8(11), 1975.

Hadley Wickham, Jim Hester, and Winston Chang. *devtools: Tools to Make Developing R Packages Easier*, 2021. URL `https://CRAN.R-project.org/package=devtools`. R package version 2.4.2.

Margaret F. J. Wilson, Brian O'Connell, Colin Brown, Janine C. Guinan, and Anthony J. Grehan. Multiscale terrain analysis of multibeam bathymetry data for habitat mapping on the continental slope. *Marine Geodesy*, 30(1-2):3–35, 2007. DOI: 10.1080/01490410701295962. URL `https://doi.org/10.1080/01490410701295962`.

Yihui Xie. animation: An R package for creating animations and demonstrating statistical methods. *Journal of Statistical Software*, 53(1):1–27, 2013. URL `http://www.jstatsoft.org/v53/i01/`.

Yihui Xie and JJ Allaire. *tufte: Tufte's Styles for R Markdown Documents*, 2021. URL `https://CRAN.R-project.org/package=tufte`. R package version 0.10.

Yihui Xie, J.J. Allaire, and Garrett Grolemund. *R Markdown: The Definitive Guide*. Chapman and Hall/CRC, Boca Raton, Florida, 2018a. URL `https://bookdown.org/yihui/rmarkdown`. ISBN 9781138359338.

Yihui Xie, Christian Mueller, Lijia Yu, and Weicheng Zhu. *animation: A Gallery of Animations in Statistics and Utilities to Create Animations*, 2018b. URL `https://yihui.name/animation`. R package version 2.6.

Yihui Xie, Christophe Dervieux, and Emily Riederer. *R Markdown Cookbook*. Chapman and Hall/CRC, Boca Ra-

ton, Florida, 2020. URL https://bookdown.org/yihui/rmarkdown-cookbook. ISBN 9780367563837.

Hanqiu Xu. Modification of normalised difference water index (ndwi) to enhance open water features in remotely sensed imagery. *International journal of remote sensing*, 27(14):3025–3033, 2006.

Margaret Young and Ian S Evans. Statistical characterization of altitude matrices by computer. report 5. terrain analysis: Program documentation. Technical report, 1978.

Achim Zeileis, Kurt Hornik, and Paul Murrell. Escaping RGBland: Selecting colors for statistical graphics. *Computational Statistics & Data Analysis*, 53(9):3259–3270, 2009. DOI: 10.1016/j.csda.2008.11.033.

Achim Zeileis, Jason C. Fisher, Kurt Hornik, Ross Ihaka, Claire D. McWhite, Paul Murrell, Reto Stauffer, and Claus O. Wilke. colorspace: A toolbox for manipulating and assessing colors and palettes. *Journal of Statistical Software*, 96(1):1–49, 2020. DOI: 10.18637/jss.v096.i01.

Lyle W Zevenbergen and Colin R Thorne. Quantitative analysis of land surface topography. *Earth surface processes and landforms*, 12(1):47–56, 1987.

Wei Zheng, Zhang Xia, Zhang Bing, and Li Xing. General and specific methods studying on bands selection of hyperspectral remote sensing data. In *Proceedings. 2005 IEEE International Geoscience and Remote Sensing Symposium, 2005. IGARSS '05.*, volume 5, pages 3223–3226, 2005. DOI: 10.1109/IGARSS.2005.1526527.