

Introducción

Capítulo 1

Un viaje del sol a los píxeles.

En esta primera práctica nos familiarizaremos con las interfaces gráficas del QGIS y de R-studio analizando la imagen Landsat 8 de noviembre de 2016, desde el punto de vista espectral. Son nuestros objetivos:

- Abrir una imagen en QGIS.
- Crear archivos vectoriales y digitalizar coberturas en QGIS.
- Abrir un archivo raster y vectorial en R.
- Realizar un análisis estadístico de la imagen y de las distintas coberturas digitalizadas en R.

1.0.1. Exploración de imágenes con el QGIS

Comenzamos abriendo la imagen LC82240782016304LGN00.vrt que se encuentra en la carpeta `raster_data/LC82240782016304`, la cual corresponde al departamento de Iguazú en la provincia de Misiones. Fue obtenida por el satélite Landsat 8 durante el mes de noviembre de 2016.

En el menú *Capa → Añadir capa → Añadir capa ráster*. Navegamos hasta la carpeta `raster_data/LC8224078201630` y abrimos la imagen LC82240782016304LGN00.vrt. La encontraremos en el *Panel de capas* de QGIS. Podemos usar las herramientas para movernos en la imagen (Figura 1.1).



Figura 1.1 – Herramientas para moverse dentro de la imagen. De izquierda a derecha: 1. Desplazar mapa, 2. Desplazar mapa a la selección, 3. Acerca zoom, 4. Alejar zoom, 5. zoom a la resolucion nativa, 6. zoom general, 7. zoom a la selección, 8. zoom a la capa, 9. zoom anterior, 10. zoom siguiente, 11. Actualizar.

Para realizar cambios en la visualización y explorar las propiedades de una capa, hacemos click derecho sobre ella y luego seleccionamos la opción *Propiedades*. Allí podemos ir a la pestaña *General* para ver datos como el nombre de la capa¹, la cantidad de filas y columnas del archivo, el valor digital no válido, el sistema de coordenadas entre otros (Figura 1.2).

En la pestaña *Estilo* podemos cambiar la visualización de la capa. Allí elegimos de qué color mostrar cada una de las bandas. Para cambiar el realce hacemos click en el botón *Cargar* para seleccionar los valores máximos y mínimos (Figura 1.3).

La herramienta *Identificar un objeto espacial* nos permite extraer valores de la imagen. Al habilitarla veremos datos como el valor de reflectancia del píxel seleccionado que pueden mostrarse como Árbol, Tabla o Grafo según uno desee (Figura 1.4).

¹Es un buen momento para ponerle uno más sencillo

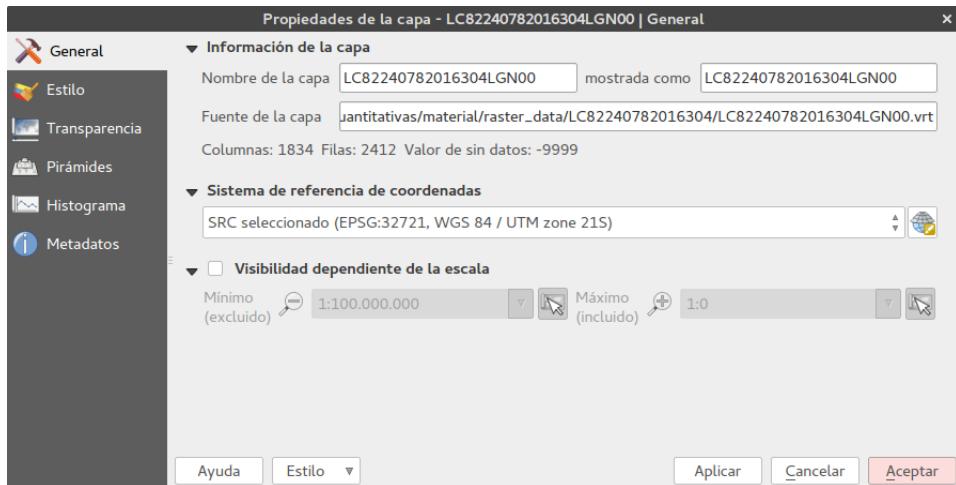


Figura 1.2 – Pestaña general de propiedades de una capa. Podemos ver los datos más importantes como la cantidad de filas y columnas, el nombre y el sistema de referencia.

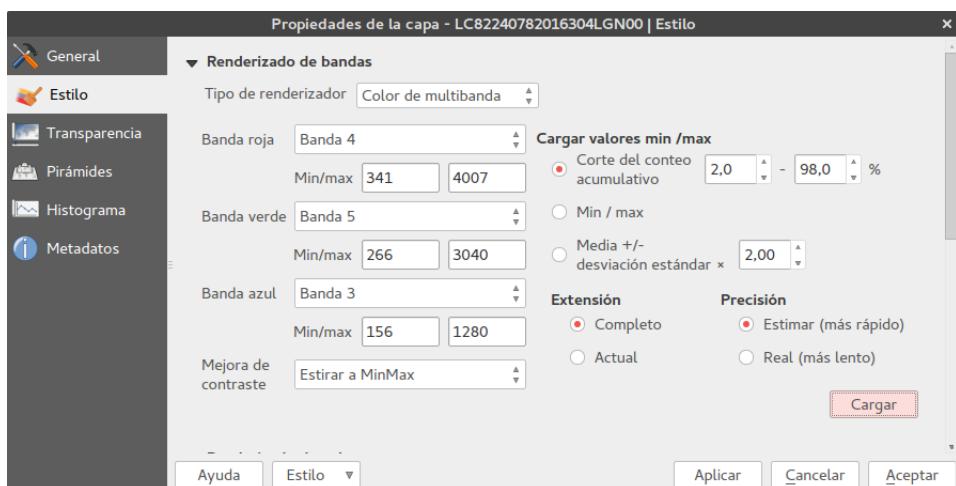


Figura 1.3 – Estilos de visualización de una capa raster.

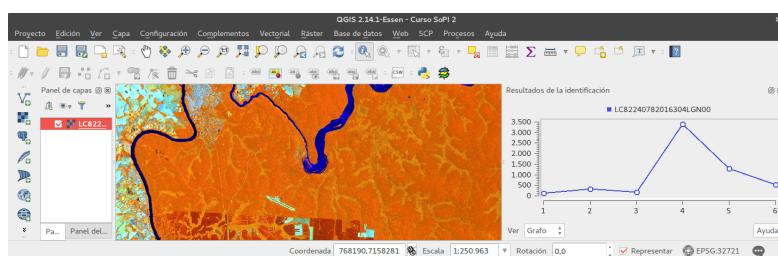


Figura 1.4 – Identificación de un píxel correspondiente a la selva paranaense mostrada como grafo.

Actividad 1.0.1. Cambie la combinación de bandas de la imagen Landsat 8 a color real y explórela. Identifique zonas de coberturas uniformes. Pruebe cambiar de combinación de bandas y decida si las zonas siguen siendo uniformes después de cada cambio.

Actividad 1.0.2. Encuentre el sistema de coordenadas en el cual se obtenga la imagen. ¿Cuántas filas y columnas tiene?

Actividad 1.0.3. Utilizando la herramienta identificar objetos espaciales encuentre los valores de reflectancia de distintas coberturas. Grafique estos valores en función de la longitud de onda y en el espacio espectral.

1.0.2. Creación de capas vectoriales

La capas vectoriales nos será de utilidad en este curso, para extraer datos cuantitativos de las capas raster.

Con la herramienta *nueva capa de archivo shape* es posible crear una nueva capa vectorial. Para esto hacemos click en el botón que se encuentra en el panel lateral. Podemos agregar los campos que sean necesarios para nuestra capa vectorial. En este caso, serán los campos: MC_ID, como entero de longitud 1 y Comment, como texto de 80 caracteres. Elegimos el sistema de coordenadas correspondiente a la imagen anterior, guardamos en la carpeta `vector_data/`, con el nombre `firmas.shp` (Figura 1.5).

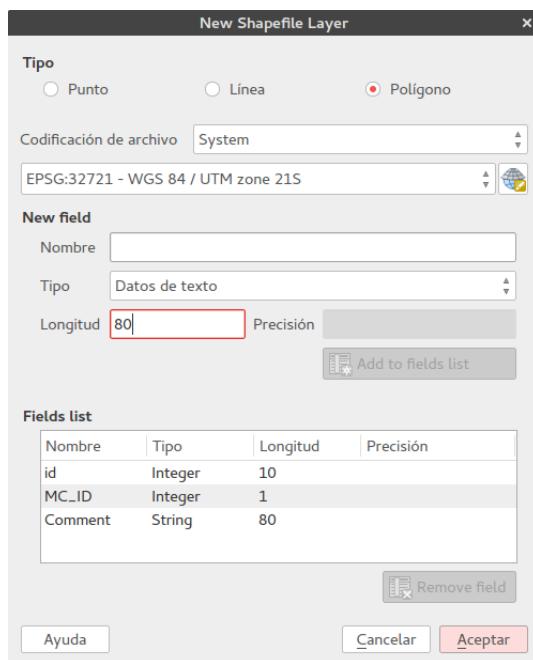


Figura 1.5 – Creación de una nueva capa vectorial.

Una vez creada, utilizamos la barra de herramientas de QGIS, (Figura 1.6), para agregarle geometrías. Para esto hacemos click en el botón de agregar geometría y digitalizamos una zona uniforme dentro de la imagen.



Figura 1.6 – Herramientas de edición vectorial. De izquierda a derecha: 1. Comutar edición, 2. Guardar cambios a la capa, 3. Añadir objeto espacial, 4. Añadir cadena circular, 5. Mover objeto espacial, 6. Herramienta de nodos, 7. Borrar lo seleccionado, 8. Cortar objetos espaciales, 9. Copiar objetos espaciales, 10. Pegar objetos espaciales.

Al terminar, QGIS pedirá un número de ID para la capa que debe ser correlativo. Además podremos ingresar en este momento los valores del resto de los campos de nuestro objeto espacial (Figura 1.7).

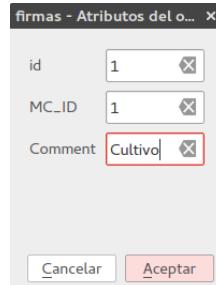


Figura 1.7 – Valores de los campos del nuevo polígono creado.

Es importante recordar que debemos estar en el modo de edición para poder hacerlo. Al terminar debemos desactivar esa opción.

Actividad 1.0.4. Digitalice coberturas uniformes dentro de la imagen. Recuerde obtener al menos un polígono por cada categoría de uso y cobertura presente.

En caso de necesitar cambiar la visualización de la capa vectorial, podemos entrar a sus propiedades². Podemos acceder a la tabla de datos de la capa vectorial haciendo click derecho sobre ella y eligiendo la opción *Abrir tabla de atributos*.

1.0.3. Exploración raster en R

Veamos como abrir y trabajar con las imágenes satelitales en R. La forma de realizar operaciones es escribir comandos en la consola de R-studio y ejecutarlos presionando enter. Para trabajar con imágenes satelitales debemos utilizar algunas librerías adicionales. Para cargarlas usamos el comando `library(raster)`. De esta forma agregamos funciones que nos facilitaran el trabajo raster.

Además, deberemos situar nuestra carpeta de trabajo donde se encuentran las carpetas que descargamos. Para esto nos movemos en el explorador de archivos hasta ella y hacemos click en usar la carpeta como carpeta de trabajo (Figura 1.8).

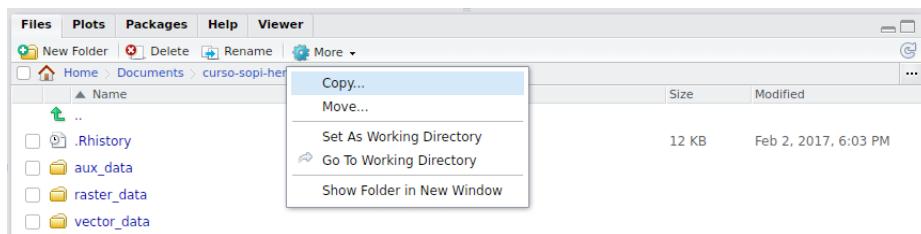


Figura 1.8 – Configuración del directorio de trabajo desde la interfaz gráfica.

También podemos utilizar el comando `setwd(.)` para configurar el directorio de trabajo. En este caso hay que especificar la ruta completa hasta él.

Una vez en la carpeta, existen varias maneras de abrir una imagen. Con el comando `raster`, abrimos una única banda; `brick`, abrimos un archivo multibanda, ;y `stack`, abrimos distintas bandas por separado. Veamos algunos ejemplo:

²Puedes utilizar el estilo precargado ubicado en la carpeta `aux_data`

Ejemplo 1.0.1. Abrimos la imagen completa del archivo de Landsat 8 y consultamos sus propiedades.

```
1 ref.2016 <- brick("raster_data/LC82240782016304/LC82240782016304LGN00.vrt")
2 ref.2016
```

obtenemos de resultado el siguiente texto

```
class      : RasterBrick
dimensions : 2412, 1834, 4423608, 6  (nrow, ncol, ncell, nlayers)
resolution : 30.00402, 30.00265 (x, y)
extent     : 731118.6, 786146, 7101531, 7173897 (xmin, xmax, ymin, ymax)
coord. ref. : +proj=utm +zone=21 +south +datum=WGS84 +units=m +no_defs
              +ellps=WGS84 +towgs84=0,0,0
data source : ./material/raster_data/LC82240782016304/LC82240782016304LGN00.vrt
names       : LC82240782016304LGN00.1, LC82240782016304LGN00.2, ...
min values  : -33, 192, ...
max values  : 2774, 3265, ...
```

En él, podemos ver la clase a la que corresponde el archivo, en este caso un *RasterBrick*, las dimensiones, el tamaño de píxel, extensión de la capa, proyección, cual es la ruta al archivo, las bandas y sus valores máximos y mínimos.

Cambiemos el nombre a las bandas y la convertimos a reflectancia entre 0 y 1.

```
1 names(ref.2016) <- c("blue", "green", "red", "nir", "swirl1", "swirl2")
2 ref.2016 <- ref.2016/1e4
3 rasterOptions(addheader = "ENVI")
4 writeRaster(ref.2016, "raster\_data/processed/ref2016")
```

Analicemos el código línea por línea.

- La primera abre la imagen como un raster de múltiples bandas.
 - La segunda, cambia los nombres de cada banda a los que figuran en la lista entre paréntesis. Es importante resaltar que el número de nombres debe ser el mismo que el de bandas.
 - En tercer lugar, convertimos el archivo de números enteros entre 0 y 10000 a valores entre 0 y 1.
 - La cuarta linea es necesaria correrla una sola vez por sesión. La misma agrega el header de ENVI a nuestro output para poder abrir el archivo desde QGIS
 - La quinta línea guarda el archivo raster con el nombre **ref2016** . En este caso estamos usando el formato nativo de R.

Podemos graficar una combinación de bandas con el comando `plotRGB(ref.2016,r=4,g=5,b=3,stretch='lin')` (Figura 1.9)

Para graficar las bandas por separador hacemos `plotRGB(ref.2016)` (Figura 1.10).

Actividad 1.0.5. Abra el archivo guardado en QGIS y vuelva a mirar la firma espectral para distintas coberturas. ¿Entre qué valores se encuentra ahora?

Veamos como trabajar más en detalle con los valores de nuestra imagen.

Ejemplo 1.0.2. Hagamos un análisis estadístico de la imagen ejecutando el comando `summary` (ref. 2016)

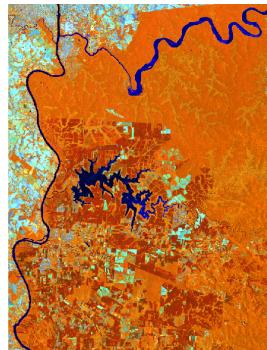


Figura 1.9 – Combinación de bandas nir-swir1-red en R.

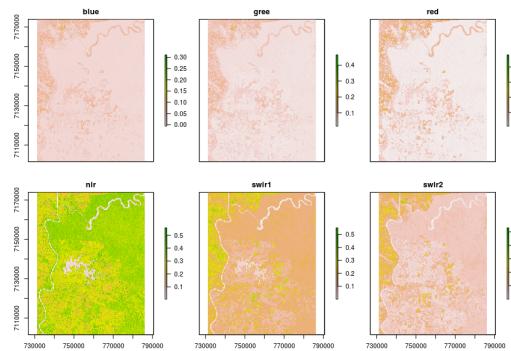


Figura 1.10 – Gráfico de bandas con realce automático para cada una.

Calculamos los histogramas de todas las bandas con el comando `hist(ref.2016)` y el scatter plot entre dos bandas como `plot(ref.2016$red, ref.2016$nir)`.

En caso de querer todos los scatterplots e histogramas en un solo gráfico usamos el comando `pairs(ref.2016)`.

1.0.4. Manejo vectorial en R

Hasta ahora estamos analizando la imagen completa, pero podemos analizar determinadas zonas utilizando un archivo vectorial.. También será posible muestrear la imagen usando otro raster, pero lo veremos más adelante.

Para trabajar con vectores en R utilizaremos la librería `library(rgdal)`.

Ejemplo 1.0.3. Veamos como realizar el análisis básico de un vector en R. Comenzamos leyéndolo

```
1   firmas <- readOGR(dsn="vector\_data/", layer="firmas")
```

Notamos en este caso que debemos indicar por separado la carpeta que contiene al shapefile en `dsn` y el nombre de la capa que queremos abrir como `layer`.

Podemos mostrar las propiedades del vector llamando a la variable `firmas` obteniendo

```
class      : SpatialPolygonsDataFrame
features   : 8
extent     : 738692.8, 767774.6, 7133396, 7165265 (xmin, xmax, ymin, ymax)
coord. ref. : +proj=utm +zone=21 +south +datum=WGS84 +units=m +no_defs
              +ellps=WGS84 +towgs84=0,0,0
```

```

variables : 3
names      : id, MC_ID,      Comment
min values : 0,      1,      Alto
max values : 9,      8, Suelo desnudo

```

Para graficar los vectores y la imagen juntos hacemos

```

1 plotRGB( ref.2016 , stretch="lin" )
2 plot(firmas , add=TRUE, col='red')

```

donde la primera línea grafica la imagen de fondo y la segunda agrega el shapefile sobre ella.

Actividad 1.0.6. Muestre las propiedades de la capa raster y vectorial y verifique que se encuentren en el mismo sistema de coordenadas.

Veamos como extraer datos de un archivo raster con un vector con la función `extract`. Esta toma dos argumentos, el vector que queremos utilizar y la capa raster sobre la cual hacer la consulta.

Ejemplo 1.0.4. Graficar en un scatterplot de dos bandas mostrando la zona del espacio ocupada por una cobertura.

```

1 datos <- extract( ref.2016 , firmas )

```

de esta forma realizamos la extracción de todos los datos de la imagen a una lista

```

1 plot( ref.2016$red , ref.2016$nir )
2 points( as.data.frame(datos[1])$red , as.data.frame(datos[1])$nir , col="green" ,
3         pch = ".")

```

Agregamos el scatterplot al muestreo obteniendo

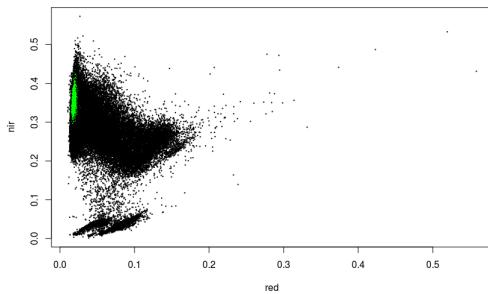


Figura 1.11 – Resultado del scatterplot para las bandas roja y nir. Se muestra en verde datos correspondientes a la selva paranaense.

La función `extract` nos permite también aplicar una función a los datos extraídos antes de entregarlos al usuario. Veamos como usarla para calcular datos de interés sobre las coberturas y guardarlos en un archivo vectorial.

Ejemplo 1.0.5. Extraer los promedios y desvío standar de un raster y agregarlos a un vector. Primero extraemos los valores de promedio y desvío

```

1 promedio <- extract(ref.2016, firmas, fun=mean)
2 desvio <- extract(ref, firmas, fun=sd)

```

renombramos luego las columnas como promedio y desvío seguido de la banda a la que pertenecen,

```

1 colnames(promedio) <- paster("mean", colnames("promedio"), sep="_")
2 colnames(desvio) <- paster("sd", colnames("desvio"), sep="_")

```

finalmente agregamos los archivos a un nuevo shapefile

```

1 firmas@data <- cbind(firmas@data, promedio, desvio)
2 writeOGR(firmas, sdn="vector_data/processed/", "firmas_datos",
           driver="ESRI Shapefile")

```

Finalmente, veamos como usar una capa vectorial para graficar las firmas espectrales y graficarlas para distintas coberturas

Ejemplo 1.0.6. Graficar las firmas espectrales en función de la longitud de onda para cada polígono. Utilizaremos dos nuevas librerías, `reshape2` y `lattice`

Comenzamos convirtiendo en dataframe a nuestros promedios, donde cada columna corresponde a una firma espectral

```

1 df <- t(promedio)
2 colnames(df) <- vector@data$Comment

```

Agregamos luego una columna con las longitudes de onda en nanometros. Luego reformamos el dataframe para que podamos subsetearlo, poniendo finalmente los nombres a cada columna

```

1 df$wl <- as.matrix(c(485, 560, 660, 830, 1650, 2215))
2 df <- melt(df, id.vars="wl", variable.name="cobertura")
3 names(df) <- c("wl", "Cobertura", "Reflectancia")

```

El dataframe resultante debería ser:

wl	Cobertura	Reflectancia
1 485	Alto	0.012926561
2 560	Alto	0.034730646
3 660	Alto	0.018491884
4 830	Alto	0.354564681
5 1650	Alto	0.133750642
...		

Repetimos el proceso para el desvío standar

```

1 ddf <- t(desvio)
2 colnames(ddf) <- vector@data$Comment
3 ddf$wl <- as.matrix(c(485, 560, 660, 830, 1650, 2215))
4 ddf <- melt("wl", "Cobertura", "Desvio")
5 ddf$desvio <- ddf$desvio
6 ddf$MC_ID <- as.character(vector@data$MC_ID[match(df$Cobertura,
                                                       vector@data$Comment)])

```

El resultado será ahora

```
wl      Cobertura Reflectancia      Desvio
1  485      Alto  0.012926561 0.0007772473
2  560      Alto  0.034730646 0.0018113004
3  660      Alto  0.018491884 0.0011561294
4  830      Alto  0.354564681 0.0166801398
5 1650      Alto  0.133750642 0.0075157929
...
...
```

En primer lugar pondremos todas las firmas espectrales juntas, separadas por color, con la librería **lattice**

```
1  xyplot(Reflectancia ~ wl, data=df, groups = Cobertura,
2         auto.key=list(space="top", columns=4),
3         ty=c("l", "p"))
```

Aquí la primer línea dice que grafiquemos la reflectancia como función de la longitud de onda, obteniendo los datos del dataframe DF y agrupandolos según la columna cobertura. La siguiente línea agrega la leyenda en la parte superior de la figura con 4 columnas. Por último en la tercera línea pedimos que el gráfico tenga líneas y puntos (Figura 1.12).

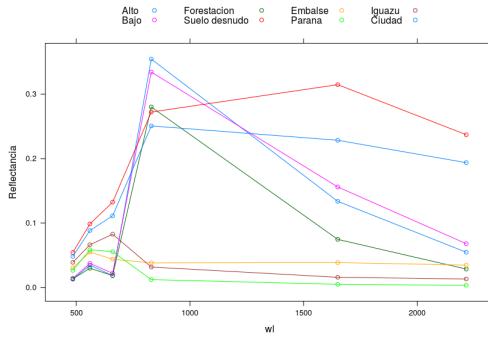


Figura 1.12 – Firmas espectrales

Si queremos agruparlo por categoría de uso y cobertura cambiamos la formula **Reflectancia** **wl** por **Reflectancia** **wl** | **MC_ID**

```
1  xyplot(Reflectancia ~ wl | MC_ID, data=df, groups = Cobertura,
2         auto.key=list(space="top", columns=4),
3         ty=c("l", "p"))
```

Si queremos graficar solo un subset de datos (Figura 1.13).

```
1  xyplot(Reflectancia ~ wl | MC_ID, data=df, groups = Cobertura,
2         auto.key=list(space="top", columns=4), ty=c("l", "p"),
3         subset = Cobertura %in% c("Alto", "Bajo"))
```

Actividad 1.0.7. Grafique la media y el desvío standar para las distintas coberturas que pudo identificar en el punto uno.

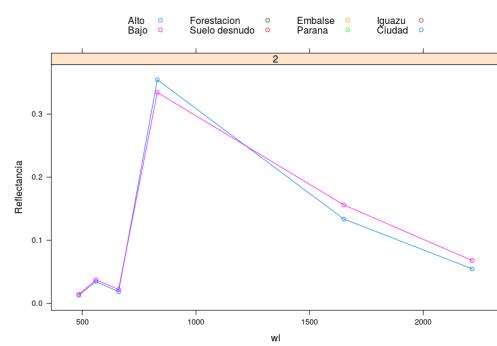


Figura 1.13 – Firmas espectrales

Capítulo 2

Rebotando por la atmósfera

En esta segunda actividad práctica nos centraremos en la corrección radiométrica de imágenes satelitales. Son nuestros objetivos:

- Abrir una imagen satelital desde el metadato.
- Convertir los valores de la imagen a reflectancia a tope de la atmósfera.
- Corregir la imagen satelital utilizando los métodos de *dos* y *cost*
- Corregir la imagen satelital utilizando el *6S* en su versión web.

2.0.1. Cálculo de reflectancia a tope de la atmósfera

Para poder convertir una imagen a reflectancia a tope de la atmósfera vamos a necesitar la información adicional que hallaremos en su metadato. Para abrirla desde el metadato utilizaremos las funciones disponibles en la librería **RStoolbox**.

Ejemplo 2.0.1. Abrimos la imagen Landsat 7 del año 2000 desde el metadato y la mostraremos en combinación color real y analicemos sus propiedades.

```
1 meta.2000 <- readMeta("raster_data/LE72240782000188EDC00/LE72240782000188EDC00_-  
MTL.txt")
```

Podemos mostrar las distintas variables incluidas en el objeto usando el signo \$ y su nombre. Por ejemplo, `meta.2000$SOLAR_PARAMETERS` da como resultado

```
azimuth elevation distance  
37.38251 31.14409 1.01670
```

Usando el metadato podemos cargar la imagen completa con el comando `stackMeta`. Eliminamos, en este caso, las bandas 6 y 7 por ser térmicas.

```
1 dn.2000 <- stackMeta(meta.2000)  
2 dn.2000 <- dn.2000[[-6:-7,]]  
3 dn.2000
```

El resultado es un objeto *raster stack* como el que sigue

```
class      : RasterStack  
dimensions : 2412, 1834, 4423608, 6 (nrow, ncol, ncell, nlayers)  
resolution : 30.00402, 30.00265 (x, y)  
extent     : 731118.6, 786146, 7101531, 7173897 (xmin, xmax, ymin, ymax)  
coord. ref. : +proj=utm +zone=21 +south +datum=WGS84 +units=m +no_defs
```

```
+ellps=WGS84 +towgs84=0,0,0
names      : B1_dn, B2_dn, B3_dn, B4_dn, B5_dn, B7_dn
min values :      0,      0,      0,      0,      0,      0
max values :  255,  255,  255,  255,  255,  255
```

Mostramos la imagen en combinación color real con `plotRGB(dn.2000, r=3, g=2, b=1, stretch="lin")` (Figura 2.1).



Figura 2.1 – Imagen en combinación de bandas color real de la zona de interés.

De esta forma tenemos el archivo en número digital con todos sus metadatos para convertirlo a reflectancia y realizar las distintas correcciones.

Existen dos formas de convertirla a reflectancia a tope de la atmósfera: a mano, utilizando las herramientas algebraicas de R o con la función específica de `RStoolbox`.

Ejemplo 2.0.2. Cálculo de reflectancia a tope de la atmósfera utilizando el metadato paso por paso

```
1 dn2ref.2000 <- meta.2000$CALREF[1:6,]
2 elev.2000 <- pi*meta.2000$SOLAR_PARAMETERS['elevation']/180
```

extraemos primero del metadato los parámetros de calibración en reflectancia y el ángulo de elevación solar. Convertimos luego la imagen a reflectancia y la dividimos por el seno del ángulo solar. Luego cambiamos los nombres de las bandas

```
1 toam.2000 <- (dn.2000*dn2ref.2000$gain+dn2ref.2000$offset)/sin(elev.2000)
2 names(toam.2000) <- c("blue","green","red","nir","swirl","swir2")
```

Otra forma de realizarlo es utilizando la función `radCor`. En este caso, debemos tener la imagen en DN, el metadato y cual es la cantidad que queremos calcular.

```
1 toa.2000 <- radCor(dn.2000, metaData = meta.2000, method = "apref")
```

Podemos comparar los resultados de ambos métodos inspeccionando los objetos `toam.2000` y `toa.2000`.

```
class      : RasterBrick
dimensions : 2412, 1834, 4423608, 6  (nrow, ncol, ncell, nlayers)
resolution : 30.00402, 30.00265  (x, y)
extent     : 731118.6, 786146, 7101531, 7173897  (xmin, xmax, ymin, ymax)
coord. ref. : +proj=utm +zone=21 +south +datum=WGS84 +units=m +no_defs
              +ellps=WGS84 +towgs84=0,0,0
data source : in memory
names       : blue,          green,          red,          nir,          swir1,        ...
min values  : -0.01976113, -0.02181530, -0.02029439,  0.01934678, -0.02781926, ...
max values  :  0.6106812,   0.5609009,   0.6079443,   0.8696885,  0.8640919, ...
y

class      : RasterStack
dimensions : 2412, 1834, 4423608, 6  (nrow, ncol, ncell, nlayers)
resolution : 30.00402, 30.00265  (x, y)
extent     : 731118.6, 786146, 7101531, 7173897  (xmin, xmax, ymin, ymax)
coord. ref. : +proj=utm +zone=21 +south +datum=WGS84 +units=m +no_defs
              +ellps=WGS84 +towgs84=0,0,0
names       : B1_tre,        B2_tre,        B3_tre,        B4_tre,        B5_tre, ...
min values  : 0.00000000, 0.00000000, 0.00000000, 0.01934678, 0.00000000, ...
max values  : 0.6106812,  0.5609009,  0.6079443,  0.8696885,  0.8640919, ...
```

Actividad 2.0.1. Inspeccione la reflectancia a tope de la atmósfera para todas las bandas. Para esto realice los histogramas, graficos de dispersión, calcule la media, el desvío standar y cualquier otra medida estadística que desee.

2.0.2. Cálculo de reflectancia corregida atmosféricamente por métodos estadísticos

La función `radCor` dispone de un parámetro para hacer distintos tipos de correcciones atmosféricas. Ya utilizamos `apref`, que nos permitió calcular la reflectancia a tope de la atmósfera, veamos ahora como aplicar el método de substracción de cuerpo oscuro.

Ejemplo 2.0.3. Aplicaremos el metodo de *simple dos* para corregir la imagen. En este caso solamente restaremos el mínimo en cada banda a la imagen para las bandas donde existe haze, es decir en la zona del visible y del infrarrojo cercano.

Estimamos el haze primero y luego corregimos la imagen haciendo

```

1 haze.2000 <- estimateHaze(dn.2000, darkProp = 0.01, hazeBands = 1:4, plot=TRUE)
2 sdos.2000 <- radCor(dn.2000, metaData = meta.2000,
3                         hazeValues = haze.2000,
4                         hazeBands = c("B1_dn", "B2_dn", "B3_dn", "B4_dn"),
5                         method="sdos")

```

en este caso los valores de haze estimados son

B1_dn B2_dn B3_dn B4_dn
41 27 20 15

Para hacer un análisis de lo que pasa, vamos a graficar los histogramas de cada banda para la imagen en reflectancia TOA y corregida por el método *simple dos*. Usaremos el paquete **rasterVis**.

```

5      lines=TRUE, points=FALSE))
6  B2 <- densityplot(~B2_tre+B2_sre, data=toa.boa, xlab="Reflectancia",
7                      ylab="", main="Banda verde", plot.points=FALSE, xlim=c(0,0.3)
8                      ,
9                      key=simpleKey(text=c("Tope de la atmósfera",
10                               "Corrección Simple DOS"),
11                               lines=TRUE, points=FALSE))
12 B3 <- densityplot(~B3_tre+B3_sre, data=toa.boa, xlab="Reflectancia",
13                      ylab="", main="Banda roja", plot.points=FALSE, xlim=c(0,0.3),
14                      key=simpleKey(text=c("Tope de la atmósfera",
15                               "Corrección Simple DOS"),
16                               lines=TRUE, points=FALSE))
17 B4 <- densityplot(~B4_tre+B4_sre, data=toa.boa, xlab="Reflectancia",
18                      ylab="", main="Banda nir", plot.points=FALSE, xlim=c(0,0.3),
19                      key=simpleKey(text=c("Tope de la atmósfera",
20                               "Corrección Simple DOS"),
21                               lines=TRUE, points=FALSE))
22 print(B1, split = c(1, 1, 2, 2), more=TRUE)
23 print(B2, split = c(2, 1, 2, 2), more=TRUE)
24 print(B3, split = c(1, 2, 2, 2), more=TRUE)
25 print(B4, split = c(2, 2, 2, 2), more=FALSE)

```

las primeras 4 funciones crean los histogramas para cada banda corregida, mientras que las últimas 4 líneas los imprimen en una grilla (Figura 2.2). Notamos en este caso que la corrección se vuelve

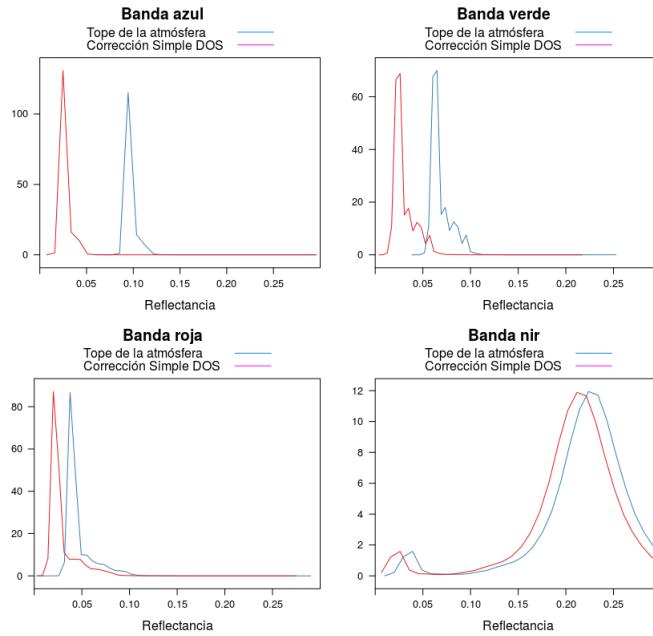


Figura 2.2 – Graficos de densidad para las distintas bandas donde se muestra el nivel de corrección en cada una.

menos importante a medida que crece la longitud de onda. Además, la corrección solo cambia la posición de la distribución y no su forma.

Actividad 2.0.2. Analice los valores de haze obtenidos por la función `stimate haze` y grafíquelos como función de la longitud de onda en escala logarítmica. ¿Qué observa?

Actividad 2.0.3. Utilice el método `costz` para corregir la imagen a reflectancia a tope de la superficie. Puede ayudarse con el comando `?radCor`

Actividad 2.0.4. Guarde el archivo raster generado por cada uno de los métodos de corrección. Abralos en QGIS y compárelos visualmente. Obtenga firmas espectrales con los distintos métodos de corrección.

2.0.3. 6S

Veamos ahora como operar con el 6S para obtener una estimación de los parámetros atmosféricos. Para ello utilizaremos la versión web del 6S que se encuentra disponible en <http://6s.ltdri.org/pages/run6SV.html>.

Ingresaremos a la página y haremos click en el botón *Submit query*. Iremos luego configurando paso a paso nuestro modelo de la atmósfera, haciendo siempre click en el botón *submit query* para ir al paso siguiente.

Los parametros para nuestro modelo son

1. Geometrical conditions
 - TM (Landsat)
 - Month: 4, Day:13, GTM decimal hour: 13.60, Longitude: -63.8606, Latitude: -24.9937.
2. Atmospheric Model
 - Select Atmospheric Profile: Mid latitude summer
 - Select aerosol model: Continental Model
 - Visibility: 60
3. Target & sensor altitude
 - Select targe altitude: sea level
 - Select sensor altitude: satellite level
4. Spectral conditions
 - Select spectral conditions: choose band
 - Select band: 1st band of tm (landat 5)
5. Ground reflectance
 - Ground reflectance type: homogeneous surface
 - Directional effect: no directional effect
 - Specify surface reflectance: input constant value of ro
 - input constant value for ro: 0
6. Signal
 - Atmospheric correction mode: no atmospheric correction

Todos estos valores los encontramos en el metadato de la imagen.

En *7.Results* podemos ver el resultado haciendo click en *Output file*. Extraemos los valores de

- global gas. trans. - total
- total sca. trans. - total
- spherical albedo - total
- reflectance I - total

Una vez ejecutado el proceso puede usarse el siguiente código para corregir todas las bandas utilizando R.

```

1  a <- c(0.98,0.90,...) # Global gas transmitance
2  b <- c(0.81,0.90,...) # Total scattering transmitance
3  g <- c(0.15,0.10,...) # Spherical albedo
4  r <- c(0.08,0.05,...) # Reflectance I
5  sss.2000 <- (toa.2000/(a*b)-r/b)/(1+g*(toa.2000/(a*b)-r/b))

```

Actividad 2.0.5. Realice una extracción de firmas espectrales para distintas coberturas de cada uno de los archivos raster obtenidos y grafíquelos juntos. Compare los resultados con la firma espectral realizada a partir de la imagen corregida por el USGS.

Actividad 2.0.6. Haga un gráfico de densidades que muestre los distintos métodos de corrección atmosféricos para cada banda.

Actividad 2.0.7. Para cada banda calcule la diferencia promedio entre las imágenes en reflectancia a tope de la atmósfera, las distintas correcciones y la imagen en reflectancia entregada por el USGS.

Capítulo 3

Un ábaco espectral

En esta tercer práctica comenzaremos a trabajar con operaciones matemáticas entre las bandas y relacionaremos los resultados con variables biofísicas medibles en el terreno. Son nuestros objetivos:

- Calcular los índices de vegetación a partir de las imágenes en reflectancia.
- Calcular la línea de suelo como parámetro para obtener índices de vegetación.
- Realizar modelos empíricos que relacionen variables biofísicas medidas a campo con los índices espectrales.
- Construir mapas a partir de los modelos empíricos antes mencionados.

3.0.1. Cálculo de índices entre bandas

Utilizaremos las librerías `raster` y `RStoolbox`. Para poder usar mejores paletas de colores utilizaremos la librería `RColorBrewer`. Por último, puede ayudar cargar la librería `rasterVis` para realizar algunos de los gráficos.

Comenzamos primero cargando la imagen desde el metadato y convirtiéndola a reflectancia entre cero y uno.

```
1  xml.2016 <- readMeta("raster_data/LC82240782016304/LC82240782016304LGN00.xml")
2  ref.2016 <- stackMeta(xml.2016, quantity = "sre")
3  scaleF <- getMeta(ref.2016,xml.2016, what = "SCALE_FACTOR")
4  ref.2016 <- ref.2016 * scaleF
5  ref.2016 <- ref.2016[[-1,]]
6  names(ref.2016) <- c("blue","green","red","nir","swirl1","swirl2")
```

Luego podemos realizar operaciones entre las bandas llamando a cada una por separado. Veamos como ejemplo el calculo de NDVI.

Ejemplo 3.0.1. Cálculo de NDVI a mano.

```
1  ndvi.2016 <- (ref.2016$nir-ref.2016$red)/(ref.2016$nir+ref.2016$red)
2  cols = colorRampPalette(brewer.pal(9,"YlGn"))(256)
3  plot(ndvi.2016, col=cols, zlim = c(0,1))
```

En este caso estamos

- Calculando el NDVI a mano usando la formula $(\rho_n - \rho_r)/(\rho_n + \rho_r)$.
- Obteniendo una rampa de color entre amarillo y verde.

- Graficando el NDVI, usamos como colores la rampa anterior y ajustandolo entre 0 y 1, es decir, los valores menores a 0 se mostrarán todos del mismo color.

Obtenemos entonces el resultado de la figura 3.1

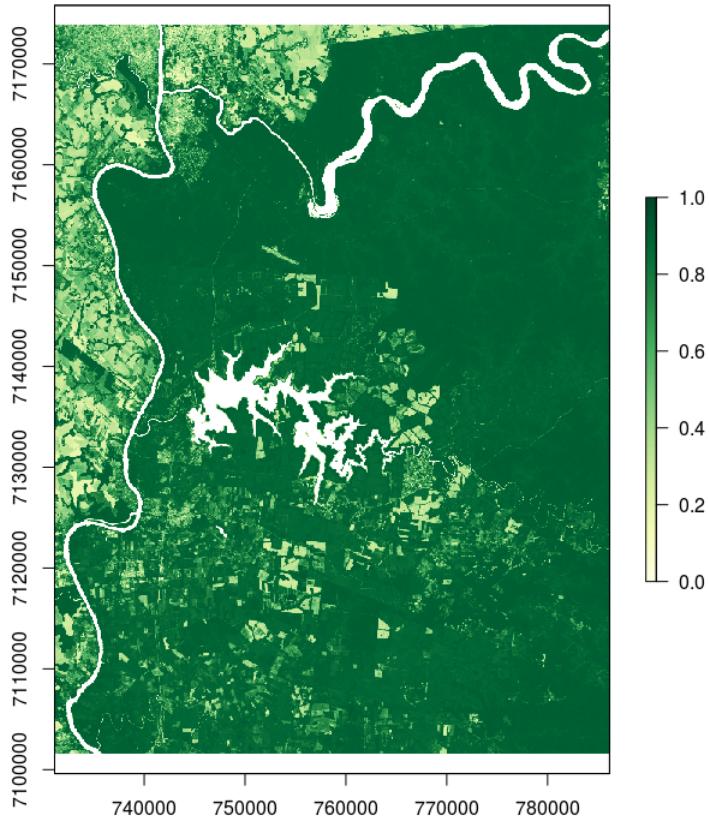


Figura 3.1 – Mapa del NDVI en escala de verdes.

El paquete `RStoolbox` tiene varias herramientas que nos ayudan a calcular los índices espectrales. Veamos por ejemplo como calcular el NDVI y el EVI.

Ejemplo 3.0.2. Para calcular los índices mediante la función `spectralIndices` debemos especificar con que raster trabajamos y que bandas corresponden a cada longitud de onda

```

1  indices.2016 <- spectralIndices(ref.2016,
2                                     blue="blue", red="red", nir="nir",
3                                     indices=c("NDVI", "EVI"))
4  plot(indices.2016, col=cols, zlim=c(0,1))

```

En este caso, vemos que la función `spectralIndices` necesita al menos 3 parámetros

- Una imagen.
- A que zona del espectro corresponde cada banda.

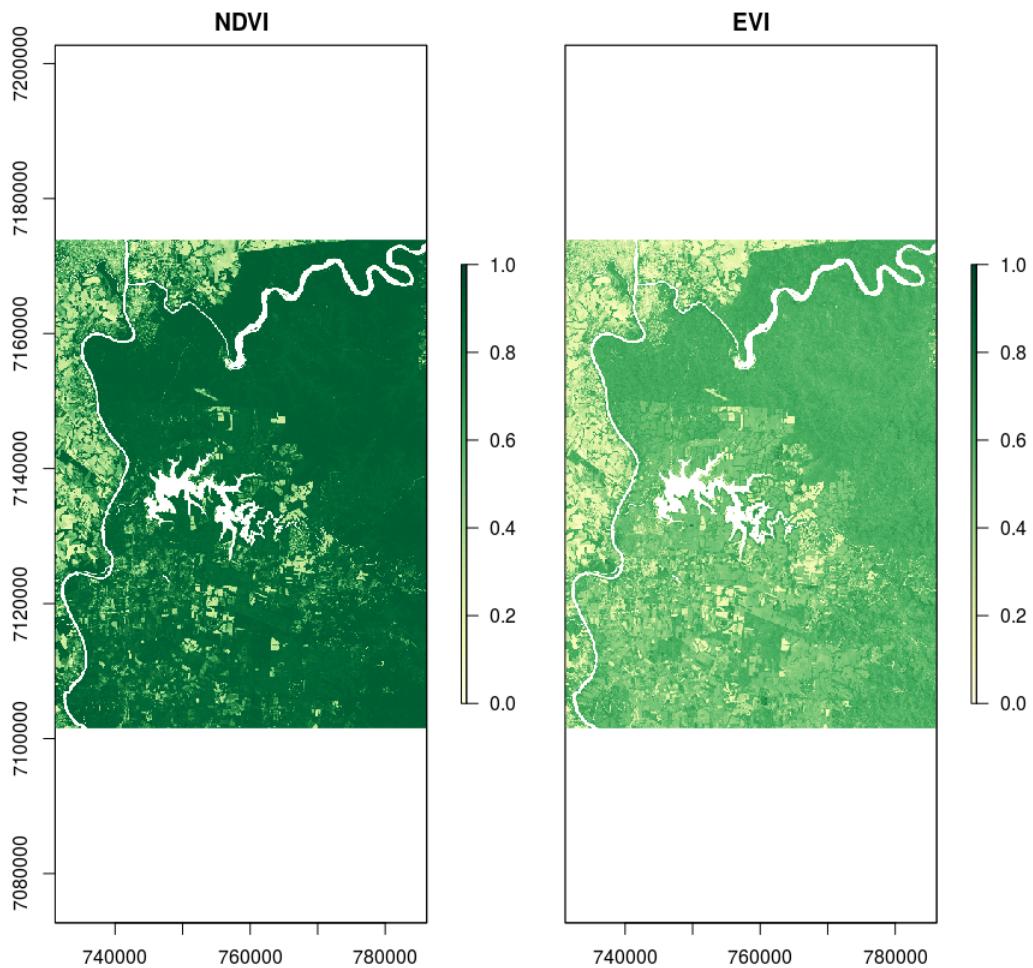


Figura 3.2 – Graficos del EVI y el NDVI para la imagen seleccionada.

- Los índices que queremos calcular

Actividad 3.0.1. Calcule el NDVI y el EVI para el año 2000 utilizando la imagen landsat 7.

Actividad 3.0.2. Calcule y grafique todos los índices posibles que involucren a las bandas roja e infrarrojo cercano de Landsat 8. Puede ayudarse con el comando `?spectralIndices`.

3.0.2. Cálculo de la línea de suelo

La línea de suelo es una cantidad que definimos en teledetección que aporta información sobre las imágenes que estamos analizando y sirve para incorporar los efectos de la reflectancia del suelo en el cálculo de índices. Veamos como hacerlo con la librería `landsat`

Ejemplo 3.0.3. Calculemos la línea de suelo a partir de la imagen Landsat 8. Para esto necesitaremos enmascarar las zonas con cobertura de agua y nubes.

```

1 mask.2016 <- raster("raster_data/LC82240782016304/LC82240782016304LGN00_cfmask.tif")
2 masked.2016 <- mask(ref.2016, mask=mask.2016, inverse=TRUE,
3                         maskvalue=0, updatevalue=255)
4 masked.2016[masked.2016<=0] <- 255

```

de esta forma enmascaramos todos los valores con nubes, agua y donde la reflectancia obtenida es cero con el valor 255. Calculamos ahora la línea de suelo y la mostramos en un scatterplot

```

1 bsl.2016 <- BSL(as.matrix(masked.2016$red), as.matrix(masked.2016$nir),
2                         method="quantile")
3 plot(ref.2016$red, ref.2016$nir)
4 abline(bsl.2016$BSL, col="red")

```

Obtenemos como resultado el gráfico de (Figura 3.3). Podemos consultar los demás parámetros imprimiendo la variable `bsl.2016`.

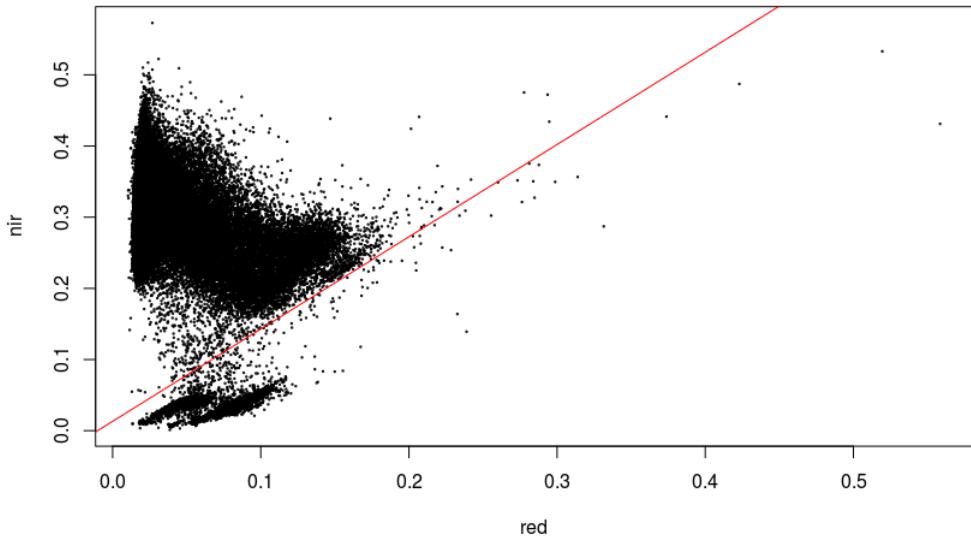


Figura 3.3 – Línea de suelo sobre el scatterplot nir-red

Actividad 3.0.3. Calcule el tSAVI utilizando la línea de suelo.

Actividad 3.0.4. Calcule la línea de suelo sin enmascarar la imagen y dibuje el scatterplot conto con ella.

Actividad 3.0.5. Obtenga la línea de suelo y calcule el índice tSAVI para la imagen del año 2000.

3.0.3. Estimación de parámetros biofísicos

Finalmente, veamos como se puede obtener datos biofísicos a partir de los índices de vegetación para realizar mapas de porcentaje de cobertura, productividad, etc.

Ejemplo 3.0.4. Comenzamos calculando el NDVI para el año 2016, y utilizando la capa muestreo, hacemos una extracción estadística sobre la misma.

```

1 vector <- readOGR(dsn="vector_data/", layer="muestreo")
2 datos <- extract(ndvi.2016, vector)
3 DF <- data.frame(vector@data, datos)
4 pairs(DF)

```

Obtendremos un gráfico que presenta los scatterplots entre las bandas, su correlación e histogramas. Vemos que la superficie cubierta por vegetación varía linealmente con el NDVI. Por lo tanto, los utilizaremos para hacer un ajuste de nuestro modelo

```

1 lm.2016 <- lm(fcover ~ ndvi, data=muestreo)
2 plot(muestreo$ndvi, mustreo$fcover)
3 abline(lm.2016, col="red")
4 summary(lm.2016)

```

de esta forma obtenemos los parámetros de nuestro ajuste.

Para aplicar el modelo a nuestro raster hacemos

```

1 fcover.2016 <- predict(ndvi.2016, lm.2016)
2 plot(fcover.2016)

```

Actividad 3.0.6. Genere los modelos de lai, fapar y fcover para el año 2016 y realice mapas de dichas variables.

Actividad 3.0.7. Utilizando los modelos obtenidos para 2016 realice los mapas de lai, fapar y fcover del año 2000. ¿Que suposición está haciendo? Compare distintas zonas de los modelos en el qgis.

Capítulo 4

Geometría espectral

En nuestra cuarta práctica trabajemos con rotaciones en el espacio espectral. Apuntamos a poder incorporar conceptos como dimensionalidad y correlación entre bandas, que nos ayuden a utilizar mejor la información satelital.

Son nuestros objetivos:

- Aplicar la transformada tasseled cap a una imagen multiespectral e interpretar el significado de cada banda.
- Aplicar la transformada por componentes principales a una imagen multiespectral e interpretar el resultado.
- Aplicar la transformada por componentes principales a un stack de bandas multiespectrales de distintas fechas e interpretar los resultados.
- Extraer información de series temporales de índices espectrales.

4.0.1. Cálculo de la transformada tasseled cap para una imagen landsat

Comencemos calculando la transformada tasseled cap para una imagen Landsat 8. Ésta será la primer rotación que utilizaremos en el curso cuya la interpretación es bastante sencilla. Usaremos el paquete **RStoolbox**.

Ejemplo 4.0.1. Para cacular la transformada por componentes principales comenzamos abriendo la imagen Landsat 8 desde el metadado y convirtiéndola a reflectancia a tope de la cobertura, como vimos en las clases anteriores.

```
1  xml.2016 <- readMeta("raster_data/LC82240782016304/LC82240782016304LGN00.xml")
2  ref.2016 <- stackMeta(xml.2016, quantity = "sre")
3  scaleF <- getMeta(ref.2016,xml.2016, what = "SCALE_FACTOR")
4  ref.2016 <- ref.2016 * scaleF
5  ref.2016 <- ref.2016[[-1,]]
6  names(ref.2016) <- c("blue","green","red","nir","swirl","swir2")
```

Analizamos ahora el espacio rojo-infrarrojo cercano y rojo-verde para la imagen

```
1  B1 <- xyplot(nir~red,data=ref.2016)
2  B2 <- xyplot(red~green,data=ref.2016)
3  print(B1,split=c(1,1,2,1),more=TRUE)
4  print(B2,split=c(2,1,2,1),more=FALSE)
```

obteniendo como scatterplots (Figura 4.1).

Calculemos ahora la transformada tasseled cap. Para eso usamos la función **tasseledCap** del paquete **RStoolbox**.

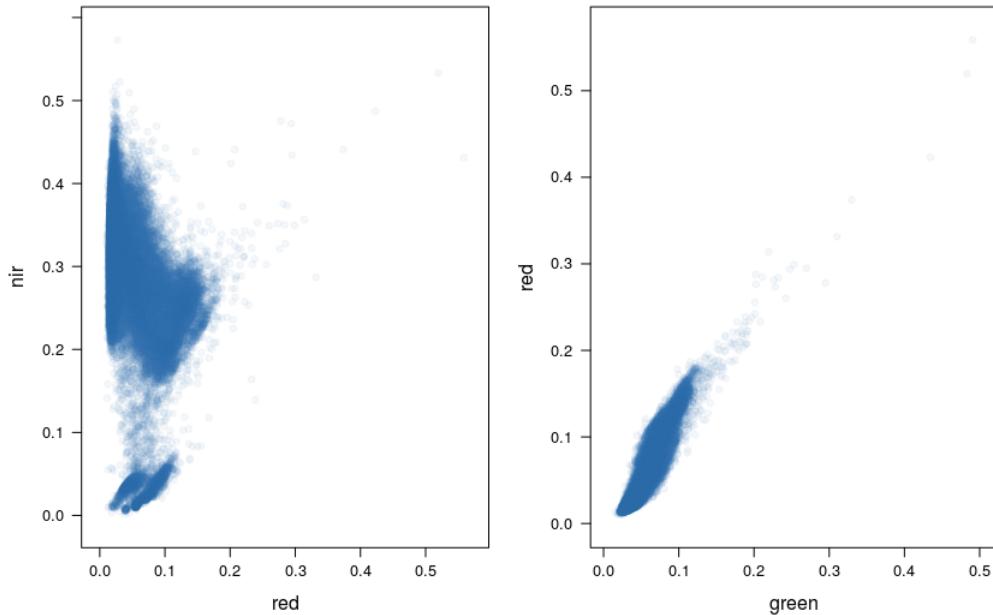


Figura 4.1 – Scatterplot verde-rojo y nir-red.

```
1 tsc.2016 <- tasseledCap(ref.2016, sat="Landsat8OLI")
```

Tendremos una imagen de tres bandas, *brillo, verdor* y *humedad*. Podemos graficar cada una de las bandas por separado con el comando `plot(tsc.2016)` o todas juntas con `plotRGB(tsc.2016,r=1,g=2,b=3,stretch="lin")`

Actividad 4.0.1. Calcule la transformada tasseled cap para la imagen Landsat 7 del año 2000.

4.0.2. Cálculo de la transformada por componentes principales

Veamos ahora como calcular la transformada por componentes principales. Utilizaremos la herramienta `rasterPCA` del paquete `RStoolbox`.

Ejemplo 4.0.2. Comencemos analizando la transformada por componentes principales de la imagen de 2016. Miremos primero los scatterplots con el comando `pairs(ref.2016)` obteniéndolos (Figura 4.2)

Mirando el resumen de la imagen vemos que hay varias bandas muy correlacionadas entre sí, como las del visible, mientras que otras lo están poco, como el infrarrojo cercano y el infrarrojo de onda corta. Por lo tanto, esperamos que no todas las bandas sean necesarias para explicar el comportamiento de la imagen, al menos en el nivel de detalle más bajo.

Apliquemos entonces la transformada por componentes principales y veamos que sucede

```
1 pca.2016 <- rasterPCA(ref.2016)
2 summary(pca.2016$model)
```

El sumario del modelo obtenidos,

Importance of components:				
	Comp.1	Comp.2	Comp.3	Comp.4 ...
Standard deviation	0.08079854	0.07808556	0.01242745	0.006488765 ...

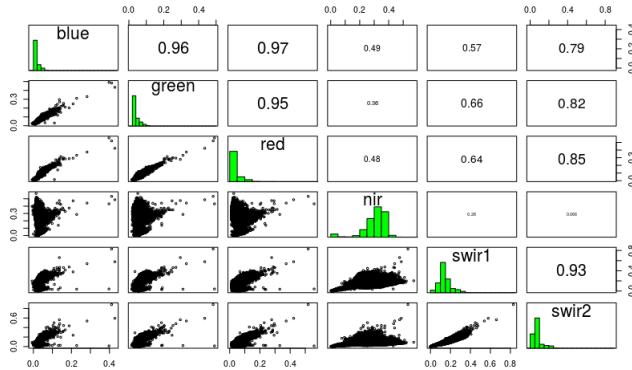


Figura 4.2 – Scaterplots y coeficientes de correlación para la imagen Landsat 8.

```
Proportion of Variance 0.50850204 0.47492732 0.01202957 0.003279516 ...
Cumulative Proportion 0.50850204 0.98342936 0.99545892 0.998738441 ...
```

Al observar las varianzas, vemos que las 3 primeras explican más que el 99.5 % de la variabilidad de la imagen. Es decir que, de las 6 bandas de Landsat 8 en esta imagen, 3 nos alcanza para explicar casi todo el comportamiento. Analisemos la primera, usando el comando `loadings(pca.2016$model)` para ver como son las componentes.

```
loadings:
  Comp.1 ...
blue
green
red -0.128 ...
nir -0.575 ...
swir1 -0.663 ...
swir2 -0.451 ...
```

La primer componente pesa siempre con el mismo signo, a todas las bandas. Podemos interpretarla como un brillo negativo y esperar que sea más alta cuando miramos zonas de la imagen que tengan menos reflectancia en promedio. La segunda componente, presenta diferencia entre los valores del infrarrojo cercano y el resto de la bandas. Esta será alta en presencia de vegetación y baja en su ausencia. La componente tres se comporta de forma similar pero con las bandas del infrarrojo medio, por lo tanto podemos interpretarla como una componente que varía según el contenido de humedad.

Actividad 4.0.2. Calcule y analice la transformada por PCA de la imagen Landsat 7 del año 2000.

4.0.3. Algunas ideas sobre series temporales

Empecemos esta sección con una actividad

Actividad 4.0.3. Aplique la transformada por componentes principales al stack de bandas del año 2000 y 2016.

Veamos que pasa al trabajar con series temporales de índices.

Ejemplo 4.0.3. Otra aplicación de la transformada por componentes principales es el análisis de series temporales.

```

1  ndvi.list <- list.files("raster_data/MOD13Q1/NDVI/", pattern = "*.tif$",
2                           full.names = TRUE)
3  ndvi.stack <- stack(ndvi.list)

```

una vez abierta la imagen la convertimos a valores entre -1 y 1 e interpolamos los valores faltantes.

```

1  ndvi.stack <- ndvi.stack/1e4
2  ndvi.stack <- approxNA(ndvi.stack)
3  writeRaster(ndvi.stack, "ndvi-series.tif")

```

Una vez interpoladas las fechas donde no había datos de NDVI, podemos abrirla en el QGIS y analizar distintas zonas de la imagen.

Utilizando la herramienta de identificar objetos espaciales podemos consultar como es el comportamiento de la serie temporal para cada píxel.

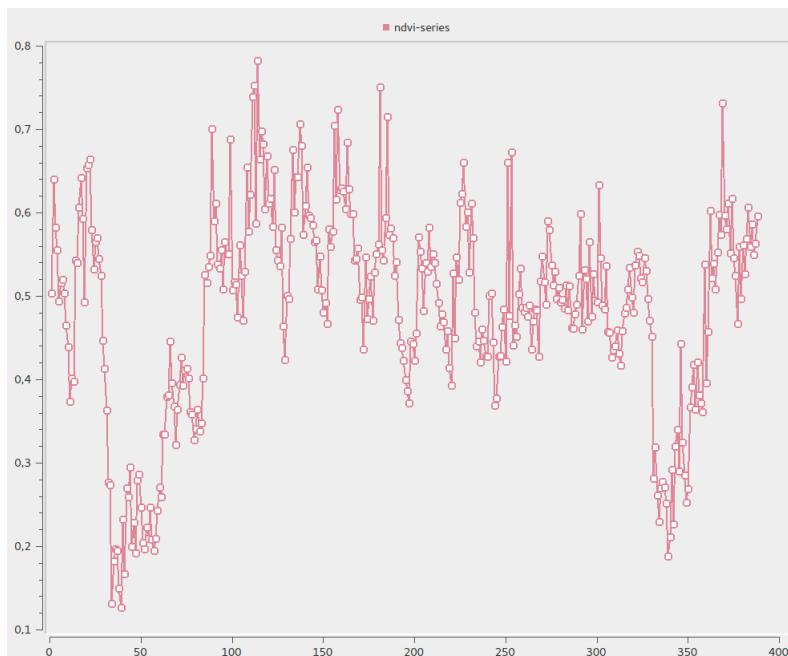


Figura 4.3 – Serie temporal de valores de NDVI.

Vemos que distintas zonas tienen distintos comportamientos intra e interanual. Podemos analizar el promedio y el desvío standar para cada píxel de la imagen

```

1  ndvi.mean <- mean(ndvi.stack)
2  plot(ndvi.mean)
3  ndvi.sd <- calc(ndvi.stack, fun=sd)
4  plot(ndvi.sd)

```

Actividad 4.0.4. Grafique las primeras 4 componentes de la transformada por componentes principales de la imagen del stack de NDVI. ¿Qué zonas puede identificar en la primera? ¿Qué zonas se distinguen en la segunda? ¿Qué comportamiento encuentra en la tercera y cuarta?

Capítulo 5

Clasificación no supervisada de imágenes

En la quinta clase del curso, trabajaremos con clasificación no supervisada de imágenes satelitales. Son nuestros objetivos:

- Usar la herramienta de clasificación no supervisada.
- Aprender a indentificar clases espectrales en categoías de uso y cobertura.
- Incorporar información no espectral a las clasificaciones como pueden ser datos temporales o información espacial.
- Aplicar la transformada por componentes principales para reducir la dimensionalidad y seleccionar los datos mas relevantes previos a las clasificaciones.

5.0.1. Clasificación mediante el método k-means

Cargaremos primero la imagen landsat 8 y habilitaremos la opcion para escribir el header de ENVI.

```
1 rasterOptions(addheader = "ENVI")
2 xml.2016 <- readMeta("raster_data/LC82240782016304/LC82240782016304LGN00.xml")
3 ref.2016 <- stackMeta(xml.2016, quantity = "sre")
4 scaleF <- getMeta(ref.2016,xml.2016, what = "SCALE_FACTOR")
5 ref.2016 <- ref.2016 * scaleF
6 ref.2016 <- ref.2016[[-1,]]
7 names(ref.2016) <- c("blue","green","red","nir","swirl1","swirl2")
```

Veamos como clasificar una imagen usando el método k-means en R. Vamos a usar los paquetes **raster** y **RStoolbox**

Ejemplo 5.0.1. Comenzamos seteando la semilla para el geneador de números aleatorios con el comando **set.seed(42)**. De esta forma la serie de números aleatorios es la misma para todos.

Luego clasificamos la imagen y la guardamos como vimos en las clases anteriores.

```
1 kmeans.2016 <- unsuperClass(ref.2016, nClasses = 5, nStarts = 100,
2                               nSamples = 100)
3 writeRaster(kmeans.2016$map, "raster_data/processed/kmeans2016",
4             datatype="INT1U")
```

En este caso estamos solamente usando 5 clases espectrales. Podemos ahora graficar por separado cada una de las clases

```

1   clases.2016 <- layerize(kmeans.2016$map)
2   plot(clases.2016)

```

Abriremos la imagen ahora en el qgis e identificaremos cada una de las clases realiendo interpretacion visual de la imagen.

Para realizar la identificacion primero vamos al menu *propiedades de la imagen* → *Estilo* → *Tipo de renderizacion* → *Unibanda pseudocolor*. Elegimos de modo Intervalo Igual y en numero de clases ponemos con el minimo en 1 y el maximo en 5. En estilo de color elegimos colores aleatorios. Iremos luego cambiando los colores uno a uno por un color brillante e identificado a que cobertura pertenece dicha clase espectral.

Construiremos con ella una tabla como la siguiente

id	class
1	2
2	7
3	1
4	1
5	1

que guardaremos en un archivo de texto con el nombre **class**. El mismo lo utilizaremos para realizar la fusion de clases.

Una vez conocidas las categorias de uso y cobertura correspondientes a cada clase espectral podemos combinarlas

```

1   clases.2016 <- read.delim("class")
2   reclas.2016 <- subs(kmeans.2016$map, clases.2016)

```

Ejemplo 5.0.2. Hagamos un analisis espectral de las imagenes clasificadas antes y despues de la fusion. Para esto utilizaremos a la imagen clasificada como mascara para asignar los colores al scatterplot. Para esto usaremos la libreria **rasterVis**.

Agregamos la imagen clasificada a las bandas de la imagen en reflectancia y luego hacemos los scatterplot segun la clase espectral o de informacion como

```

1   stack.2016 <- stack(ref.2016, kmeans.2016$map, reclas.2016)
2   xyplot(nir+swirl~red, groups=layer, data=stack.2016)
3   xyplot(nir+swirl~red, groups=class, data=stack.2016)

```

Vemos que las clases espectrales pueden unirse en clases de informacion y que algunas clases de informacion no se separan en distintas clases espectrales (figuras a y b).

Ejemplo 5.0.3. Repitamos este analisis pero con 100 clases espectrales. El algoritmo es el mismo pero ahora vamos a usar el parametro nClasses igual a 100.

```

1   kmeans.2016b <- unsuperClass(ref.2016, nClasses = 5, nStarts = 100,
2                                   nSamples = 100)
3   writeRaster(kmeans.2016b$map, "raster_data/processed/kmeans2016b",
4               datatype="INT1U")
5   clasesb.2016 <- read.delim("class100")
6   reclasb.2016 <- subs(kmeans.2016b$map, clasesb.2016)
7   plot(reclasb.2016)

```

Comparamos nuevamente los espacios de fase entre la clasificacion obtenida a partir de 5 y 100 clases espectrales

```

1 stackb.2016 <- stack(ref.2016, kmeansb.2016$map, reclasb.2016)
2 xyplot(nir+swirl~red, groups=class, data=stack.2016)
3 xyplot(nir+swirl~red, groups=class, data=stackb.2016)

```

Vemos en este caso que somos capaces de separar todas las clases de informacion en el espacio de fases.

Actividad 5.0.1. Vuelva a repetir la clasificacion utilizando la imagen obtenida de la transformada por componentes principales descartando las bandas que aporten menos informacion.

Actividad 5.0.2. Repita la clasificacion para la imagen landsat 7 del año 2000.

5.0.2. Informacion espacio-temporal

Con el fin de mejorar la clasificacion podemos incorporar informacion espacial y temporal a la informacion radiometrica. De esta forma estaremos, al momento de clasificar los pixeles, ya no trabajando en un espacio espectral si no en uno mas amplio. Veamos como hacer esto.

Ejemplo 5.0.4. Para incorporar informacion espacial sobre el contexto del píxel, podemos hacerlo tanto antes como después de la clasificación. Nos centraremos en este caso en como hacerlo antes.

Calculamos primero la variabilidad local de brillo para la banda pancromatica de la imagen Landsat 8 como

```

1 pan.2016 <- raster("raster_data/LC82240782016304LGN00/LC82240782016304LGN00_B8.
TIF")
2 window <- matrix(1,nrow=5, ncol=5)
3 sd.2016<-focal(pan.2016,w=window, fun=sd)

```

Una vez obtenida la banda de desvio standar, podemos agregarla a las demás, luego de remuestrearla haciendo

```

1 sd.2016 <- aggregate(sd.2016, fact=2, fun=mean)
2 stack(ref.2016, sd.2016)

```

y finalmente calculamos la transformada por componentes principales como

```

1 pca.2016 <- rasterPCA(stack(ref.2016, sd.2016), spca = TRUE)

```

Vemos en este caso, analizando el sumerio, que la banda de textura agrega informacion con respecto a la solo disponible al utilizar las bandas en radiancia. Clasificamos ahora la imagen como hicimos antes utilizando el algoritmo k-means y la guardamos para su posterior comparacion.

Actividad 5.0.3. Repita el ejemplo para la imagen landsat 7 del año 2000.

Ejemplo 5.0.5. Para incorporar informacion del contexto temporal, agregaremos a la informacion espectral informacion sobre la variacion del indice de vegetacion durante el año en que se tomo la imagen. Primero cargamos las imagenes modis del 2016

```

1 list.2016 <- list.files("raster_data/MOD13Q1/NDVI/", pattern = "MOD13Q1.A2016+.
*tif", full.names = TRUE)
2 ndvi.2016 <- stack(list.2016)/1e4
3 sd.2016 <- calc(ndvi.2016, fun = sd)
4 me.2016 <- calc(ndvi.2016, fun = mean)

```

Una vez hecho esto, resampleamos el promedio y el desvio de la serie temporal y lo unimos a las imagenes en reflectancia, sobre las cuales calcularemos la transformada por componentes principales.

```
1 sd.2016 <- resample(sd.2016, ref.2016, method="ngb")
2 me.2016 <- resample(me.2016, ref.2016, method="ngb")
3 pca.2016 <- rasterPCA(stack(ref.2016, me.2016, sd.2016), spca = TRUE)
```

podemos ahora seleccionar las 4 componentes que explican el 99 % de comportamiento de la imagen y clasificarla.

Actividad 5.0.4. Repita el ejemplo para la imagen Landsat 7 del año 2000.

Capítulo 6

Clasificacion supervisada de imágenes

En la sexta clase del curso, continuamos trabajando con algoritmos de clasificación de imágenes, centrandonos en este caso en los no supervisados. Son nuestros objetivos:

- Poder realizar clasificaciones no supervisadas utilizando los distintos algoritmos que se encuentran en R.
- Calcular la distancia espectral entre como forma de determinar la separabilidad de dos clases espectrales.
- Comparar utilizando la entropia de un pixel que coberturas presentan mayor confusión al momento de la clasificación.

Cargaremos primero la imagen landsat 8 y habilitaremos la opción para escribir el header de ENVI. Usaremos en primer lugar los paquetes **raster**, **rgdal** y **RStoolbox**.

```
1 rasterOptions(addheader = "ENVI")
2 xml.2016 <- readMeta("raster_data/LC82240782016304/LC82240782016304LGN00.xml")
3 ref.2016 <- stackMeta(xml.2016, quantity = "sre")
4 scaleF <- getMeta(ref.2016,xml.2016, what = "SCALE_FACTOR")
5 ref.2016 <- ref.2016 * scaleF
6 ref.2016 <- ref.2016[[-1,]]
7 names(ref.2016) <- c("blue","green","red","nir","swirl1","swirl2")
8 vector <- readOGR(dsn="vector_data/", layer="entrenamiento")
```

6.0.1. Clasificador por máxima verosimilitud

Empecemos con la clasificación por el método de máxima verosimilitud, para esto necesitamos del paquete

```
1 sup.2016 <- superClass(ref.2016, vector, responseCol = "MC_ID",
2                           model = "mlc")
3 plot(sup.2016, col=rainbow(8))
```

y realizar el scatterplot de dichas variables como.

```
1 ref.mlc <- stack(ref.2016,sup.2016$map)
2 xyplot(nir~red, groups=MC_ID, data=ref.mlc)
```

Cambiando el algoritmo de clasificacion en el parametro `model` podemos calcular distintas clasificaciones supervisadas. Algunas de las vistas en clase son `rf`, `svmRadial`, `kNN`. Cada una de ellas usa alguna libreria adicional de las cargadas antes.

Ejemplo 6.0.1. Una forma de mejorar las clasificaciones supervisadas basadas en el espacio espectral es clasificar por separado distintas clases espectrales y luego unirlas en la misma clase de informacion. Veams como hacerlo.

```
1 sup.2016b <- superClass(ref.2016, vector, responseCol = "C_ID",
2                           model = "mlc")
```

Una vez realizada la clasificacion, debemos substituir los valores de cada pixel por el de la clase de informacion correspondiente. Para ello hacemos

```
1 subs.2016 = vector@data[c(3,1)]
2 sub.2016 <- reclassify(sup.2016b$map, subs.2016)
3 writeRaster(sub.2016, "raster_data/processed/mlc2016",
4             datatype="INT1U")
5 plot(sub.2016, col=rainbow(8))
```

Podemos finalmente comparar las dos imagenes clasificadas lado a lado ejecuntando el comando `plot(stack(sup.2016$map, sub.2016), col=rainbow(8))`

Actividad 6.0.1. Realice clasificaciones por los distintos metodos y comparelas visualmente.

Actividad 6.0.2. Agregue las bandas de textura y evolucion temporal del NDVI y vuelva a clasificar las imagenes.

6.0.2. Entropia de la clasificacion

Para poder comparar en que zonas los clasificadores presentan mas o menos dispersion podemos calcular la entropia de las distintas clasificaciones en cada pixel. Para esto utilizaremos la funcion `rasterEntropy`.

Ejemplo 6.0.2. Para esto comenzamos corriendo la clasificacion para distintos modelos, los apilados y despues calculamos la entropia de los mismos

```
1 set.seed(42)
2 sup.2016 <- superClass(ref.2016, vector, responseCol = "C_ID",
3                           model = "mlc")
4 mlc.2016 <- reclassify(sup.2016$map, subs.2016)
5
6 library(randomForest)
7 sup.2016 <- superClass(ref.2016, vector, responseCol = "C_ID",
8                           model = "rf")
9 rf.2016 <- reclassify(sup.2016$map, subs.2016)
10
11 library(kernlab)
12 sup.2016 <- superClass(ref.2016, vector, responseCol = "C_ID",
13                           model = "svmLinear")
14 svm.2016 <- reclassify(sup.2016$map, subs.2016)
15
16 prediction_stack <- stack(mlc.2016, rf.2016, svm.2016)
17 names(ensemble) <- c("mlc", "rf", "svm")
18
19 model_entropy <- rasterEntropy(prediction_stack)
```

Podemos graficar la entropia de las clasificaciones como `model_entropy` y ver que zonas presentan mas diferencias a la hora de la clasificacion y cuales no.

```
1 plot(stack(prediction_stack , model_entropy) , col=rainbow(8))
```

Actividad 6.0.3. Repita las clasificaciones por los metodos de arriba agregando la banda textura y de variaciion temporal del ndvi. Apilela junto con las clasificaciones por k-means de la clase anterior. ¿A que cobertura pertenecen las zonas con mayor variabilidad?

Capítulo 7

Técnicas pos-clasificación

Veamos finalmente, en esta práctica, algunas técnicas de posprocesamiento para imágenes satelitales que nos ayudaran a mejorar los valores extraidos y conocer la incertezza en su estimación. Son nuestros objetivos:

- Aplicar filtros modales a las clasificaciones para eliminar píxeles aislados.
- Poder calcular la presición total para la clasificación y sus precisiones del usuario y productor.
- Estimar el error de para las areas calculadas a partir de la imagen clasificada.

Vamos a usar los paquetes `raster`, `rgdal`, `RStoolbox` y `rasterVis`.

7.0.1. Filtrado de clasificaciones

Una de las primeras tareas a realizar luego de clasificar una imagen es aplicar un filtro a la misma que permite eliminar píxeles aislados. Al hacerlo podremos descartar, por ejemplo, pixeles de ciudad en el medio de la selva o de selva en medio de la ciudad. Esta es otra forma de incorporar el contexto espacial a nuestras clasificaciones

Ejemplo 7.0.1. Veamos como aplicar un filtro por moda a una imagen clasificada.

Comencemos cargando una imagen clasificada en R

```
1 mlc.2016 = raster(raster_data/processed/mlc2016)
```

Para aplicar el filtro de 3x3 creamos una matriz con unos de 3x3 y usamos el comando `focal` con la función `modal`

```
1 window <- matrix(1,nrow=3, ncol=3)
2 mlc.3x3<-focal(mlc.2016,w=window,fun=modal)
3 writeRaster(mlc.3x3,"raster_data/processed/mlc3x3", datatype = "INT1U",
   overwrite=TRUE)
```

En el caso de un filtro por moda, estaremos dejando el mayor que mas veces aparezca entre los que rodean al pixel.

Podemos en este caso mostrar la imagen con y sin el filtro junto con la entropía de la clasificación para ver en que zonas se produjeron mas cambios.

```
1 plot(stack(mlc.2016, mlc.3x3, rasterEntropy(stack(mlc.2016, mlc.3x3,))))
```

Analizando la entropía vemos en que zonas se presentan mayor diferencia entre los valores antes y despues de aplicar el filtro.

Actividad 7.0.1. Aplique filtros de 5x5 y 7x7 para filtrar la imagen. Que problemas desaparecen? que dificultad introducen.

Actividad 7.0.2. Aplique el filtro de 3x3 a la imagen correspondiente al año 2000.

7.0.2. Validacion de clasificaciones

El segundo procesamiento pos clasificacion, y tal vez el mas importante, es el calculo de la matriz de confusión para nuestra clasificación. Haremos esto en dos partes: crearemos primero el set de datos para la validación y luego lo usaremos para calcular dicha matriz.

Ejemplo 7.0.2. Veamos como crear un set de puntos aleatorios de muestreo con QGIS. En este caso utilizaremos como unidad de muestreo al pixel, pero de forma similar pueden usarse polígonos o grupos de polígonos.

Comenzamos abriendo la imagen filtrada en QGIS. Una vez hecho esto, vamos a vectorizar la clasificación utilizando la herramienta *Raster → Conversion → Poligonizar*. Guardamos el archivo como **3x3.shp** en la carpeta de **vector_data**.

Una vez hecho esto utilizamos la herramienta *Dividir capa vectoria* y lo guardamos en la carpeta **vector_data/split**. Cargamos luego las capas en QGIS, elegimos la herramienta *Puntos aleatorios en los límites de la capa* y hacemos click en el botón *Ejecutar proceso por lotes*. Seleccionamos cada capa y en numero de puntos ponemos la cantidad correspondiente a cada categoría. Para calcular dicha cantidad usamos el comando **freq.2016 <- freq(mlc.3x3)** para conocer las frecuencias de aparición de cada categoría y luego con el comando **freq.2016[,2] <- round(freq.2016[,2]/sum(freq.2016[,2])*250+50)** distribuimos los píxeles en cada categoría con 50 fijos y 50 según la frecuencia de aparición. Con una distancia mínima de 100m.

Elegimos luego en que carpeta guardar los puntos aleatorios y con que nombre y hacemos click en aceptar. Finalmente, una vez creadas las capas de puntos aleatorios las unimos utilizando la herramienta *Combinar capas vectoriales*.

Una vez unidas las capas editamos la tabla de atributos para agregar el campo **MC_ID** y, realizando un análisis visual en distintas combinaciones de bandas, asignamos a cada punto el ID de la clase de información a la que pertenece.

Para hacerlo podemos utilizar la misma imagen que usamos para clasificar o una de mayor resolución espacial. Una vez terminado y guardada la capa tendremos nuestra capa de validación para continuar.

Actividad 7.0.3. Construya de esta forma una serie de puntos de validación para la imagen Landsat 7 del año 2000.

Ejemplo 7.0.3. Una vez obtenidos los puntos de validación podemos calcular la matriz de confusión. Para esto debemos cargar el polígono de validación y la calculamos con la función **validateMap**.

```
1  valid.2016 <- readOGR(dsn="vector_data/", layer="validacion")
2  val.2016 <- validateMap(mlc.3x3, valData = valid,
3                           responseCol = "MC_ID")
```

al inspeccionar el elemento **val.2016\$performance** obtenemos la matriz de confusión, la precisión global como *Acuracy*, la precisión del productor como *Sensitivity* y la del usuario como *Pos Pred Value*. begin

Confusion Matrix and Statistics

		Reference				
		1	2	5	7	8
Prediction	1	150	27	14	6	2
	2	5	164	0	0	0

5	6	0	21	0	0
7	1	0	0	35	5
8	3	0	0	6	54

Overall Statistics

Accuracy : 0.8497
 95% CI : (0.8153, 0.8799)
 No Information Rate : 0.3828
 P-Value [Acc > NIR] : < 2.2e-16

 Kappa : 0.7888
 Mcnemar's Test P-Value : NA

Statistics by Class:

	Class: 1	Class: 2	Class: 5	Class: 7	Class: 8
Sensitivity	0.9091	0.8586	0.60000	0.74468	0.8852
Specificity	0.8533	0.9838	0.98707	0.98673	0.9795
Pos Pred Value	0.7538	0.9704	0.77778	0.85366	0.8571
Neg Pred Value	0.9500	0.9182	0.97034	0.97380	0.9839
Prevalence	0.3307	0.3828	0.07014	0.09419	0.1222
Detection Rate	0.3006	0.3287	0.04208	0.07014	0.1082
Detection Prevalence	0.3988	0.3387	0.05411	0.08216	0.1263
Balanced Accuracy	0.8812	0.9212	0.79353	0.86570	0.9323

Una vez obtenidas la matriz de confusion y las areas mediante el comando `freq()`, podemos usar el script de R `areas.R`. Este nos devolvera un archivo csv con la el area del mapa, *adj_area*, y su error *CI_adj_area* como se ve al ejecutar el comando `ov[c(1:5,11,12)]`.

1	2	5	7	8	adj_area	CI_adj_area	
1	0.316	0.057	0.030	0.013	0.004	134705.31	11396.012
2	0.015	0.487	0.000	0.000	0.000	216164.27	9459.761
5	0.006	0.000	0.021	0.000	0.000	19868.04	6181.366
7	0.000	0.000	0.000	0.016	0.002	12715.55	4163.619
8	0.002	0.000	0.000	0.003	0.028	13510.27	2682.288

Actividad 7.0.4. Construya la matriz de confusion y obtenga la presicion global para todas las clasificaciones del año 2000 y 2016. Que algoritmo funciona mejor con la imagen? A partir de la misma obtenga estimaciones de area para el año 2000 y 2016 y utilicela para estimar la perdida de selva paranaense durante este periodo.