Análisis geo-espacial y Minería

Francisco Barreras

May 7, 2015

El Proyecto

- La minería y en particular la minería ilegal tienen externalidades locales sobre la población y el medio ambiente.
- La medición de dichas externalidades está limitada a la disponibilidad de datos.
- Existe un censo minero para 2011, pero está bastante incompleto y la actividad minera es muy dinámica.
- ¿Cómo identificar la minería ilegal?

Remote Sensing

- Con imágenes satelitales de resolución relativamente alta, podrían identificarse sitios de minería basándose en el color de los pixeles.
- En los mejores satélites, un pixel corresponde a un área cuadrada de $30x30m^2$, así que el reconocimiento basado en formas no es tan viable.

Software de Análisis Espacial

En el dominio del análisis geoespacial existen varios programas para procesamiento de imágenes, por ejemplo:

- ArcGis
- QGIS
- GRASS

La mayoría de estos sirven para visualizar y editar objetos espaciales, pero en **R** todas estas herramientas existen y están integradas al entorno de objetos vectoriales usual, lo que facilita el análisis estadístico.

El paquete SP

El paquete 'SP' trabaja con objetos de la clase *SpatialPolygons*. Básicamente son objetos georreferenciados que pueden o no incluir más datos. Los principales objetos son:

- **SpatialPoints**
- **SpatialLines**
- **SpatialPolygons**
- **SpatialGrid**

Estos objetos tienen una estructura de **DataFrame**, por lo que se pueden filtrar de acuerdo a sus **ID's** o algunos valores de sus variables.

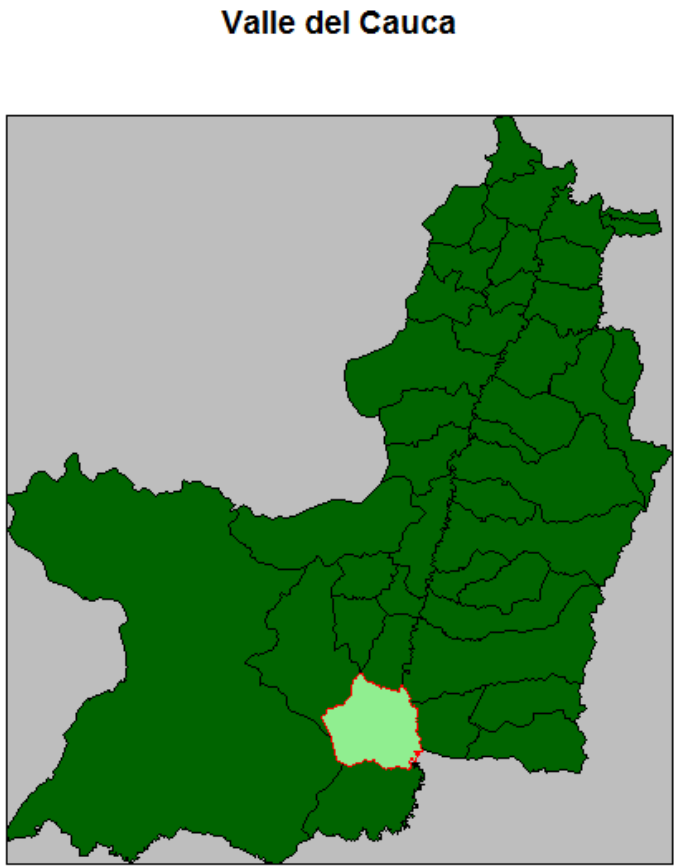
Filtrar municipios por un departamento

Si se filtra el ShapeFile de los municipios (**SpatialPolygonsDataFrame**) por el departamento '**VALLE**' obtenemos solamente los polígonos de los municipios del Valle:

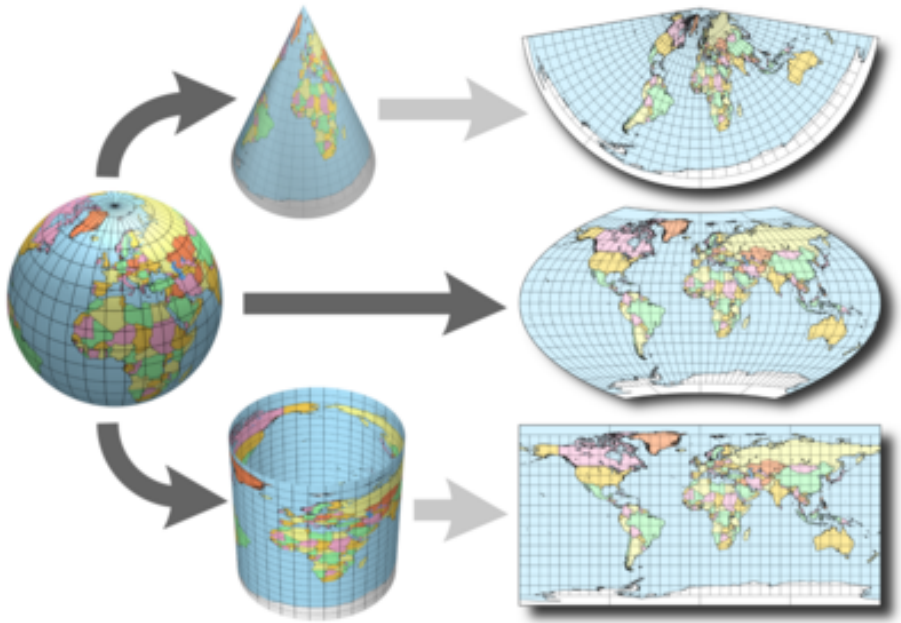
```
Console C:/Users/USER/Copy/Mineriallegal/
> municipios.sf
class       : SpatialPolygonsDataFrame
features    : 1100
extent      : -2197404, -790005.4, 3036709, 4861945  (xmin, xmax, ymin, ymax)
coord. ref. : +proj=eqdc +lat_0=-32 +lon_0=-60 +lat_1=-5 +lat_2=-42 +x_0=0 +y_0=0 +ellps=aust_SA +units=m +no_d
variables   : 7
names       : CODANE2, PERIMETER, MUNICIPIO, DEPARTAMEN,        AREA,   id,     area_m2
min values  :   05001,       0.0, Abejorral,   AMAZONAS,           0,    0,    16657200
max values  :   99773, 2251110.0, Zipaquira,    VICHADA, 65664299008, 1099, 69207400000
> valle.sf<-subset(municipios.sf,municipios.sf$DEPARTAMEN=="VALLE")
```

Plot de los municipios

Podemos graficar los municipios y escoger a Cali, y pintarlo en otro color.



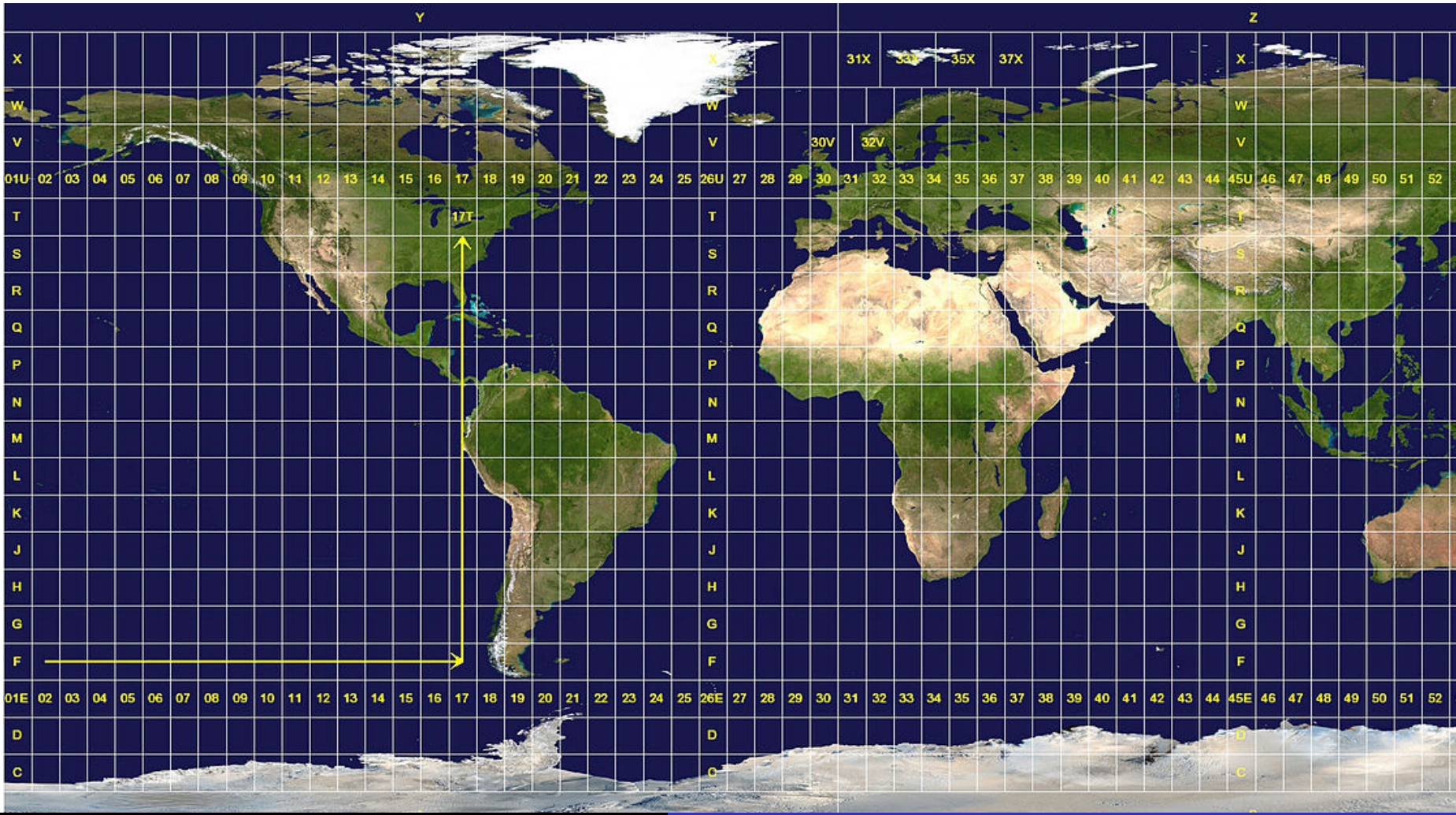
Proyección



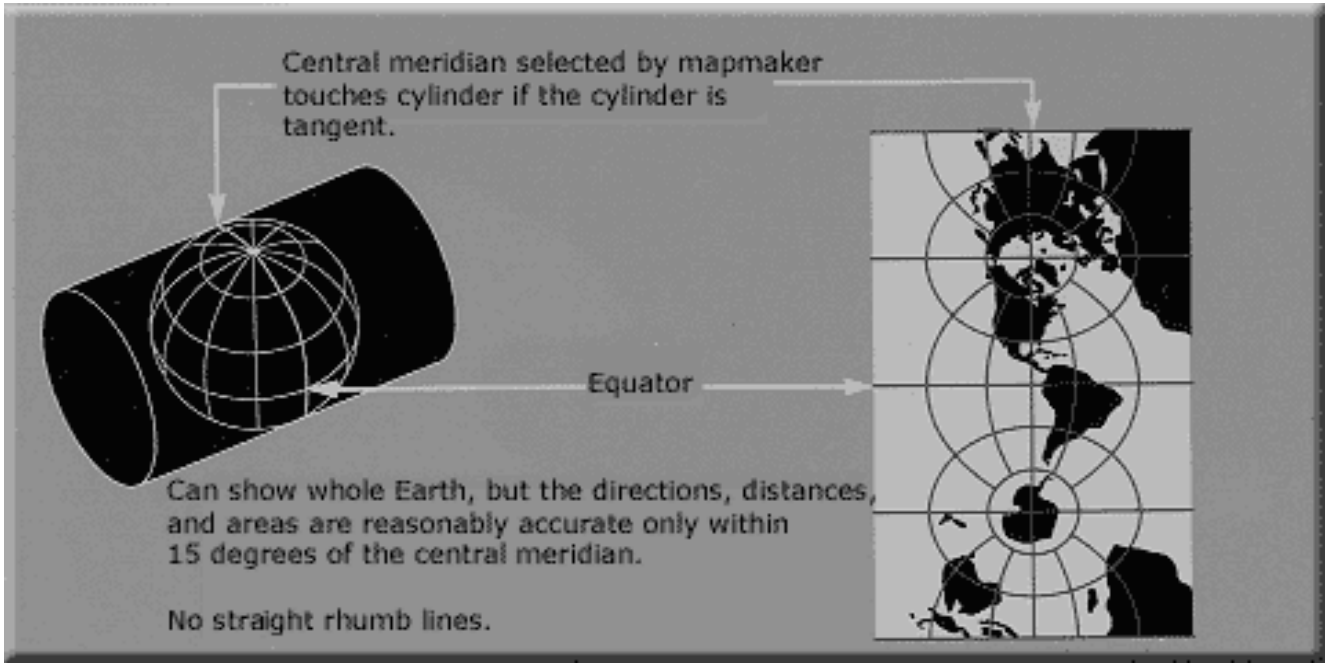
Proyección

- Todos los objetos espaciales deben tener un sistema de coordenadas incluído.
- Distintas proyecciones planas conservan distintas características como **Área**, **Distancias**, **Formas**, etc.
- Si el análisis que se busca es muy local, la proyección **UTM** conserva relativamente bien la topología de los polígonos.
- **UTM** = Universal Transverse Mercator. No es una sola proyección, sino que divide el mundo en zonas y tiene una proyección para cada zona.

GLOBAL UTM GRID



Proyección para una zona UTM



UTM

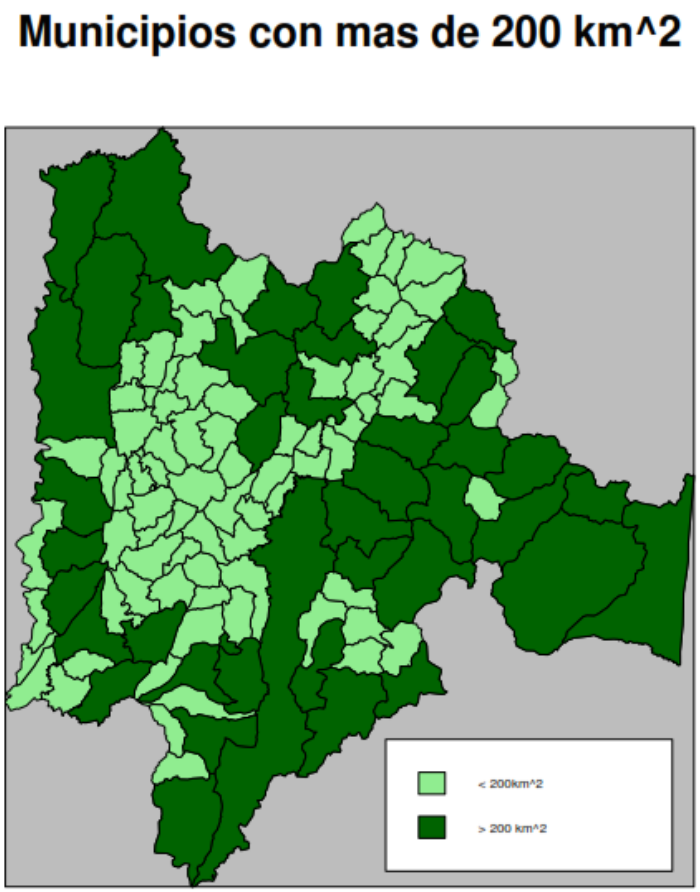
- Cada zona UTM abarca 6° de longitud y su origen está en la intersección entre su meridiano central y la linea del Ecuador.
- La proyección UTM está medida en metros. Las dos dimensiones se llaman **Eastings** y **Northings**.
- Los **Northings** de cierto punto se definen como el número de metros desde la linea ecuatorial.
- Para evitar lidiar con números negativos, los **Eastings** están definidos de forma que el meridiano central está a **500.000** Eastings.

Obviamente para poder hacer análisis con varios objetos, todos deben estar en la misma proyección. Por conveniencia se usa la proyección **UTM 18**, el comando **spTransform()** proyecta a las coordenadas deseadas:

```
Console C:/Users/USER/Copy/Mineriallegal/
> proj4string(valle.sf)
[1] "+proj=eqdc +lat_0=-32 +lon_0=-60 +lat_1=-5 +lat_2=-42 +x_0=0 +y_0=0 +ellps
> valle.sf<-spTransform(valle.sf,CRS("+proj=longlat +datum=WGS84"))
> proj4string(valle.sf)
[1] "+proj=longlat +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0"
> #Ahora el archivo del valle está en coordenadas LatLon.
>
```

Ejemplo

A manera de ejemplo, pintemos en verde los municipios de Cundinamarca con área superior a $200 km^2$

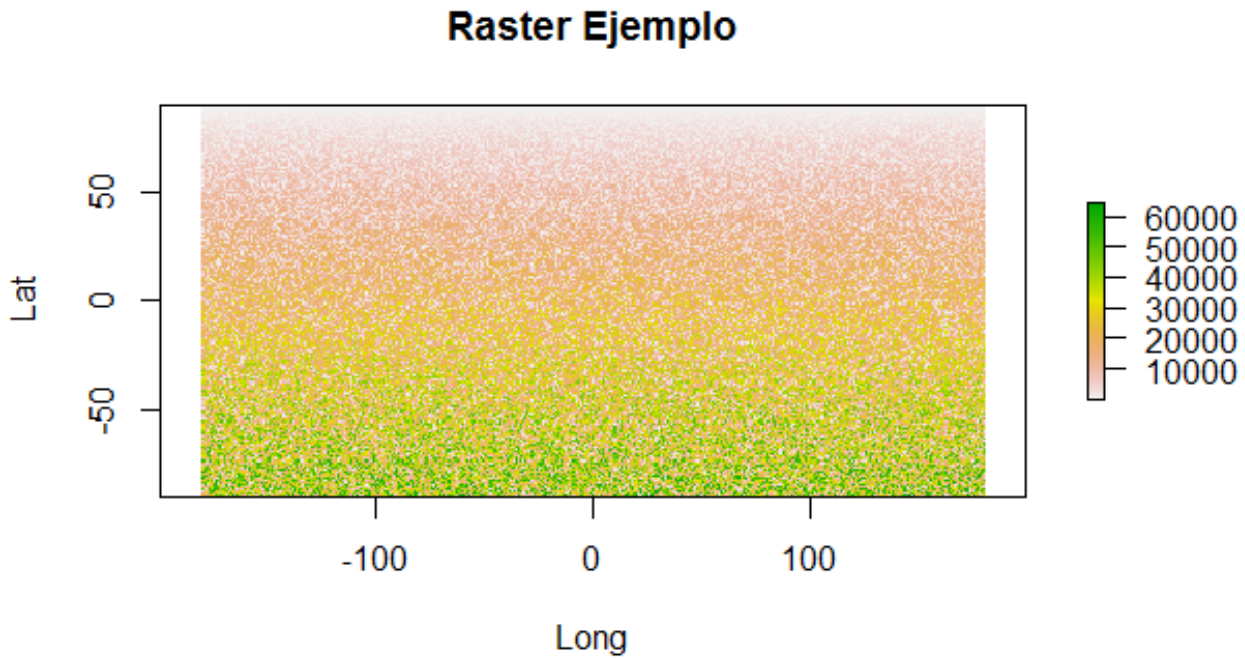


El paquete Raster

- Los objetos de la clase 'SpatialPolygons' están "vectorizados". Son polígonos construidos con ecuaciones, **independientemente** de la escala.
- Otro objeto conceptualmente distinto es el de mapa de bits. Simplemente un grid de pixeles, con cierta **resolución**, **origen** y coordenadas para cada esquina.
- Estos objetos son de la clase '**Raster**', y son precisamente el tipo de objeto que necesitamos para leer las imágenes satelitales.

Un Raster que cubre el mundo

```
> ras<-raster()
> vector<-1:ncell(ras)
> ras[]<-runif(vector,max=vector)
> plot(ras,main="Raster Ejemplo")
```



Los objetos del tipo Raster suelen ser pesados, y es normal que vengan empaquetados con distintas capas. Los objetos más relevantes son:

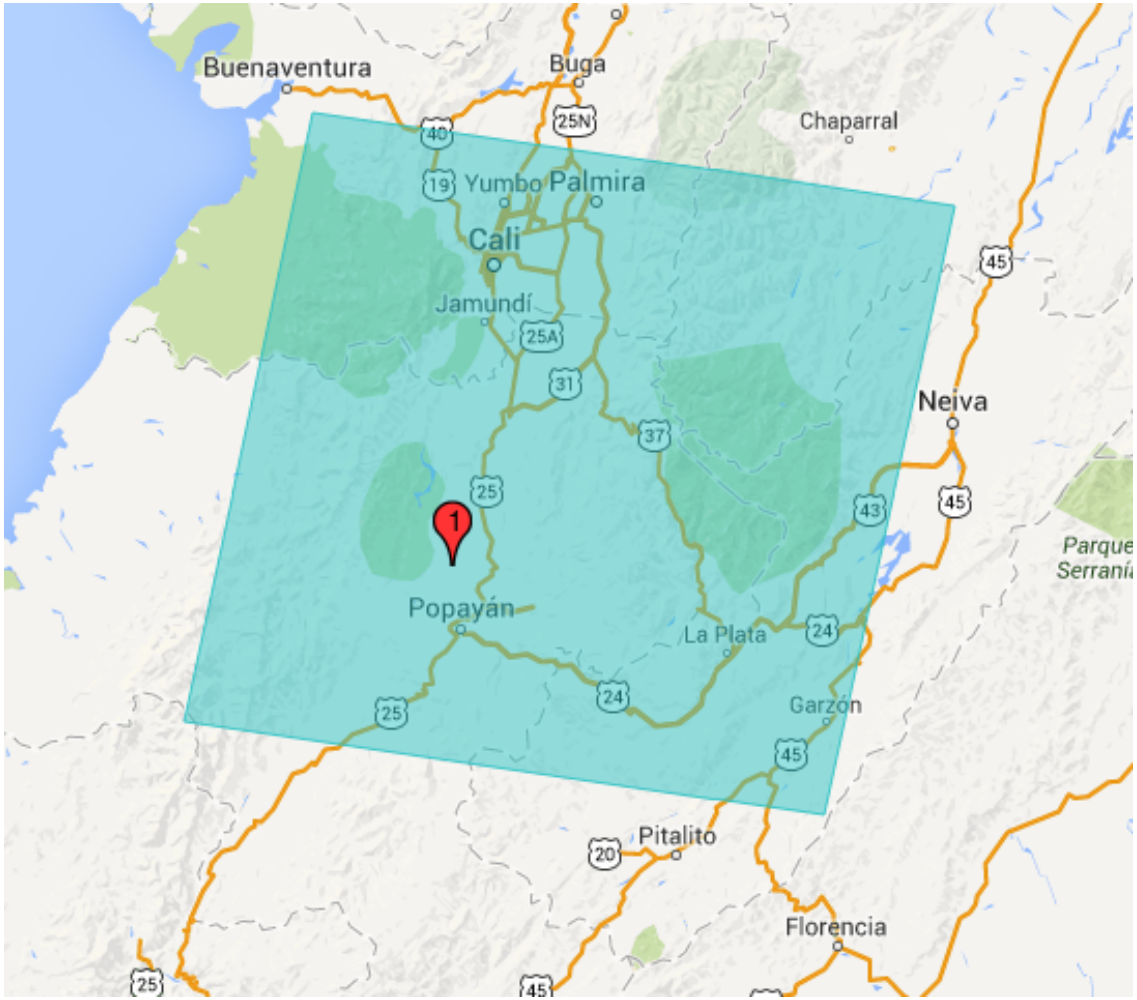
- **RasterLayer**
- **RasterBrick**
- **RasterStack**

Parámetros de una escena Landsat 7

```
Console C:/Users/USER/Copy/Mineriallegal/
> scene_LE70070572010343EDC00
class       : RasterStack
dimensions  : 6921, 7971, 55167291, 8  (nrow, ncol, ncell, nlayers)
resolution  : 30, 30  (x, y)
extent      : 609585, 848715, 376485, 584115  (xmin, xmax, ymin, ymax)
coord. ref. : +proj=utm +zone=18 +datum=WGS84 +units=m +no_defs +ellps=WGS84 +towgs84=0,0,0
names       : b1.r, b2.r, b3.r, b4.r, b5.r, b6_1.r, b6_2.r, b7.r
min values  :    0,    0,    0,    0,    0,      0,      0,    0
max values  :  255,  255,  255,  255,  255,    255,    255,  255
```

- 638 MegaBytes
- Area $240x210Km^2$ aproximadamente.
- **55 167 291** Pixeles

Las escenas de Landsat7 son muy grandes

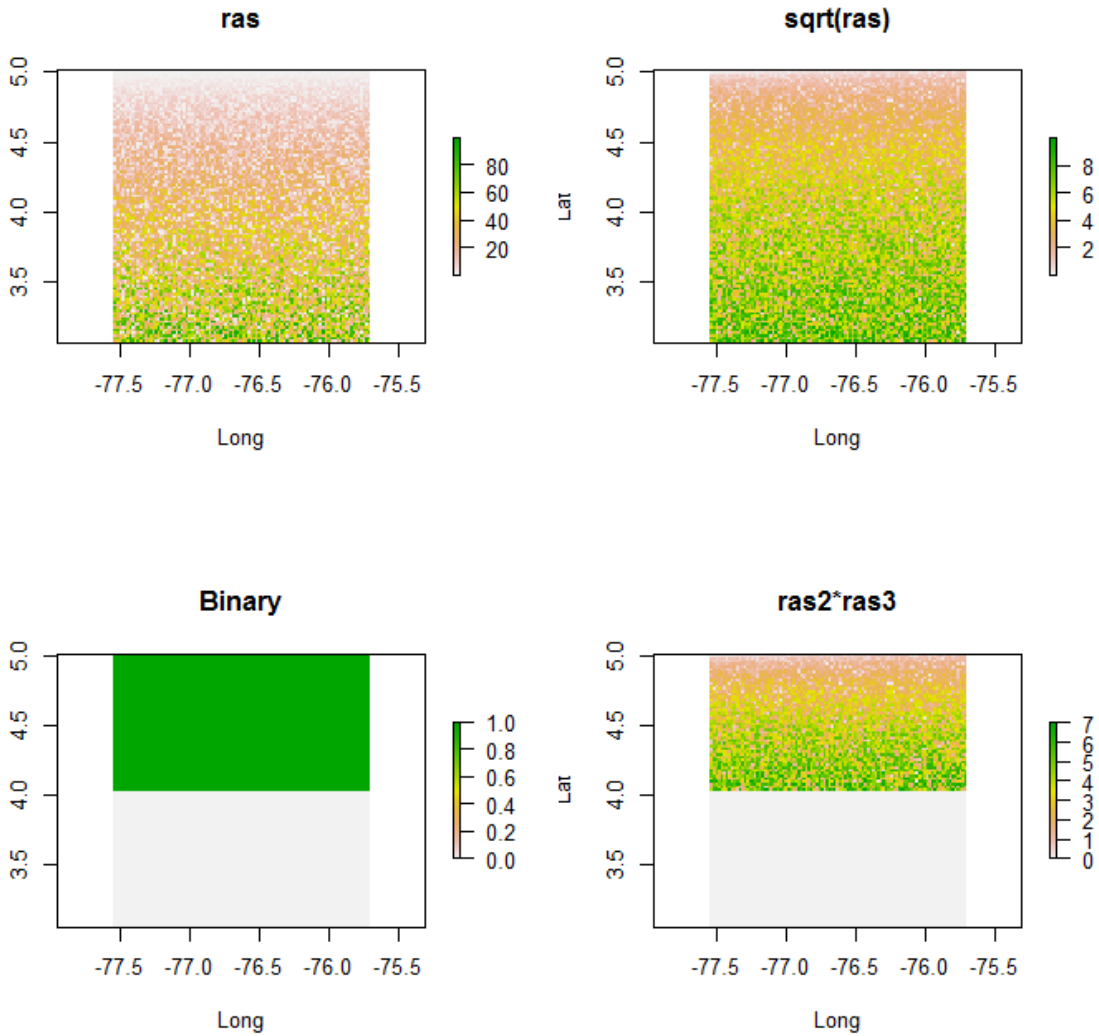


Crear un objeto Raster

- Especificando todos los parámetros anteriores se puede crear un Raster vacío de manera muy sencilla.
- Luego el Raster se puede llenar asignándole un vector con tantas entradas como pixeles, o haciéndole **operaciones algebráicas a otros Rasters**.

```
Console C:/Users/USER/Copy/Mineriallegal/
> ras<-raster(nrow=100,ncol=100,extent(valle.sf))
> vector<-1:ncell(ras)
> ras[]<-(1/100)*runif(vector,max=vector)
> ras2<-sqrt(ras)
> ras3<-raster(nrow=100,ncol=100,extent(valle.sf))
> ras3[]<-0
> ras3[1:(0.5*ncell(ras3))]<-1
> ras4<-ras2*ras3
```

Crear un objeto Raster



Dibujar SpatialPolygons encima de Rasters

La ventaja de tener objetos georreferenciados es que se pueden yuxtaponer. El paquete **ggmap** nos permite bajar mapas de google y graficarles encima los objetos que deseemos.
Esta función es de particular importancia pues queremos dibujar las minas del censo encima de las imágenes satelitales.

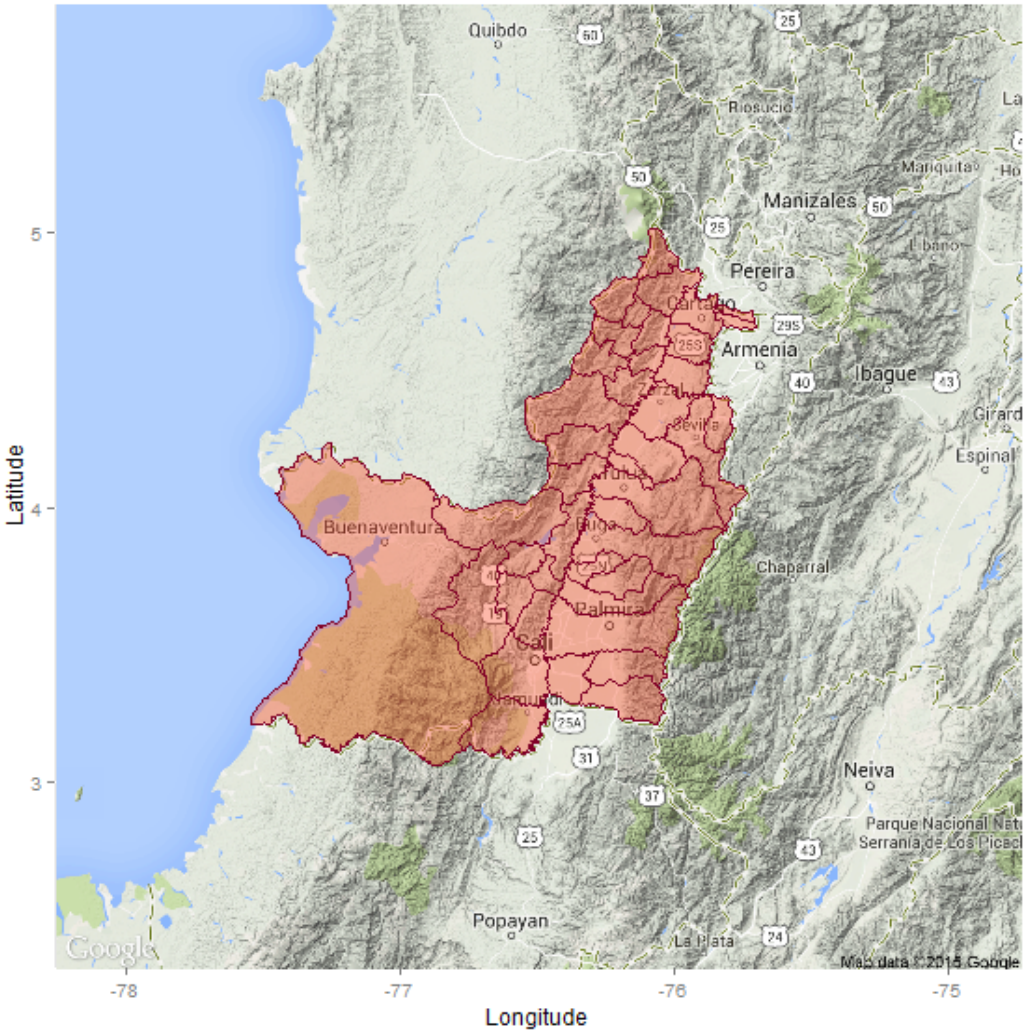
Plotear municipios del Valle sobre mapa

```
#Escogemos los polígonos que se van a dibujar
area<-valle.sf
#Se escoge un color
colors <- brewer.pal(9, "YlorRd")
#ggmap se mete a google y baja el mapa que uno le diga
mapImage <- get_map(location = c(lon = -76.5, lat = 4.08), color
                    source = "google", maptype = "terrain",zoom

#Y ahora hagamos el plot
ggmap(mapImage) +
  geom_polygon(aes(x = long,
                   y = lat,
                   group = group),
               data = area,
               color = colors[9],
               fill = colors[6],
               alpha = 0.4) +
  labs(x = "Longitude",
       y = "Latitude")

```

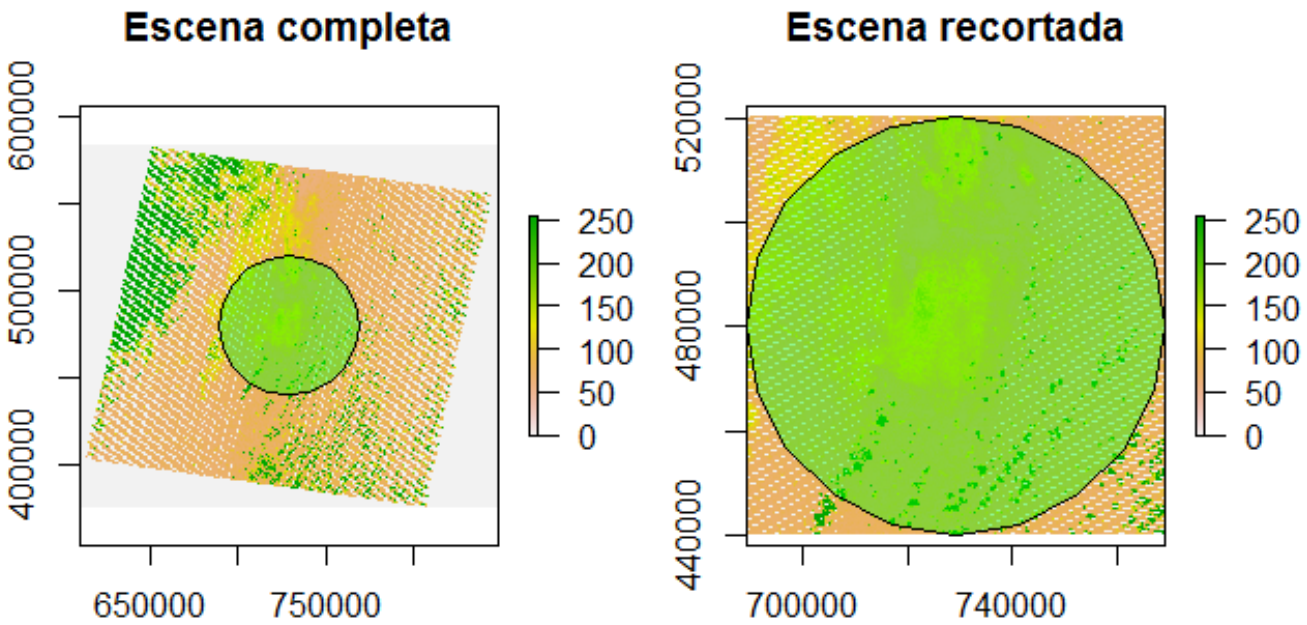
Plotear municipios del Valle sobre mapa



La función Crop

- Puede resultar conveniente restringir el análisis gráfico a cierta área definida por un **Bounding Box**.
- Las coordenadas de los bounding box de objetos 'SpatialPolygons' y de objetos 'Raster' se extraen con el comando **extent()**.
- Los objetos 'Raster' y 'Spatial Polygons' se pueden recortar de acuerdo a un extent dado.
- Esto es particularmente útil para **ahorrar tiempos de ejecución**.

Recortar un Raster y un SpatialPolygons

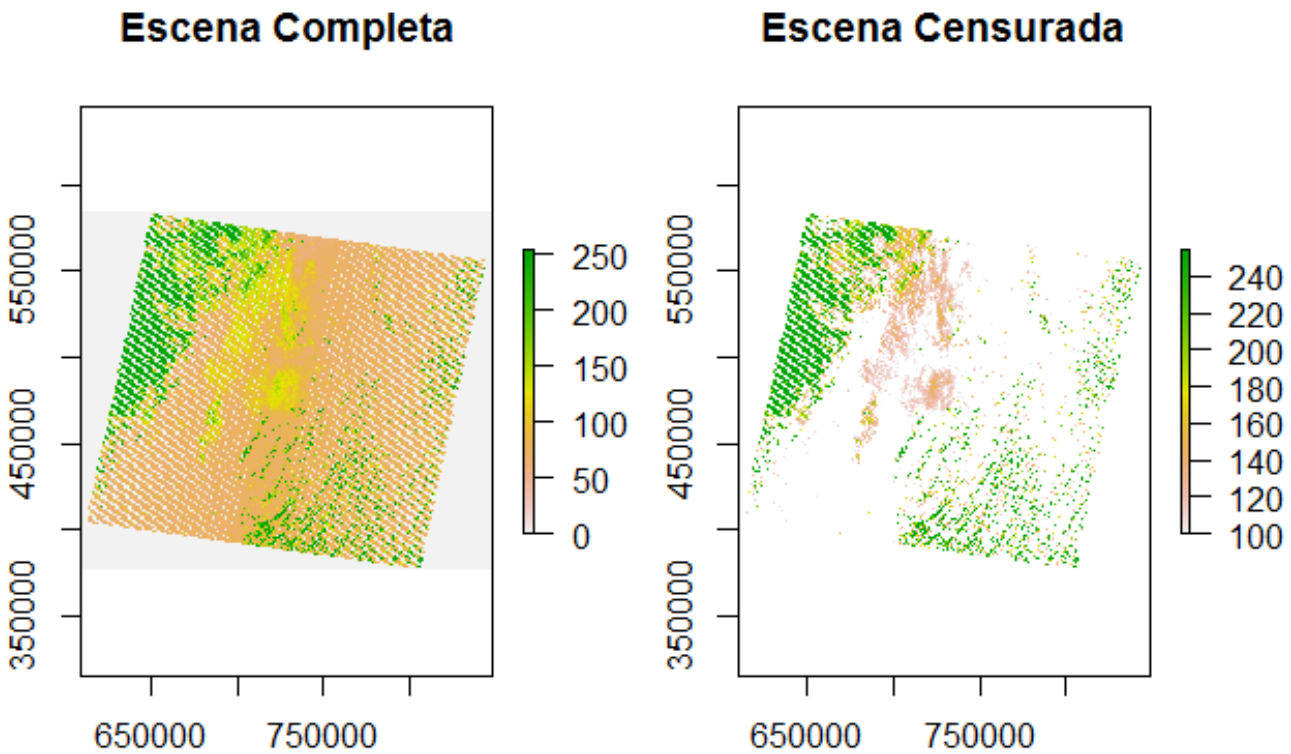


Modificar valores de un Raster para análisis

- Suponga que identifica que en las escenas de Landsat7 las partes cubiertas de agua toman valores entre 0 y 100. (No es el caso)
- Usando proposiciones lógicas puede modificar los valores en un Raster.
- Se puede pedir que todos los valores entre 0 y 100 en un Raster se vuelvan **NA**.

Basta con ejecutar el comando $raster[raster < 100] < -NA$.

Censuramiento de un Raster

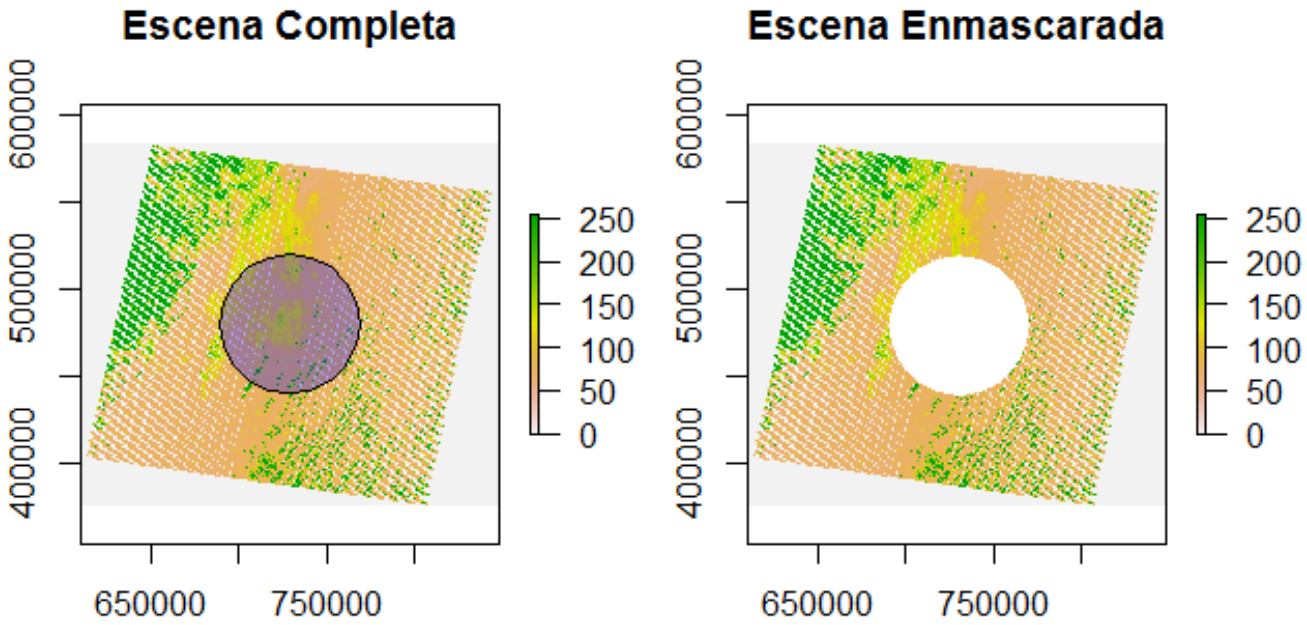


Enmascaramiento

- Una función más avanzada que la función **crop()** es la función **mask()**.
- Si se quiere restringir el análisis de un Raster **exactamente** al área cubierta por cierto polígono,(por ejemplo, fuera de una nube) se usa la función **mask()**
- Se demora mucho más que **crop**, pues debe preguntarle a cada celda del objeto 'raster' si su centroide está dentro del área del polígono.

Usamos el comando **mask(raster,polygon)**.

Enmascaramiento

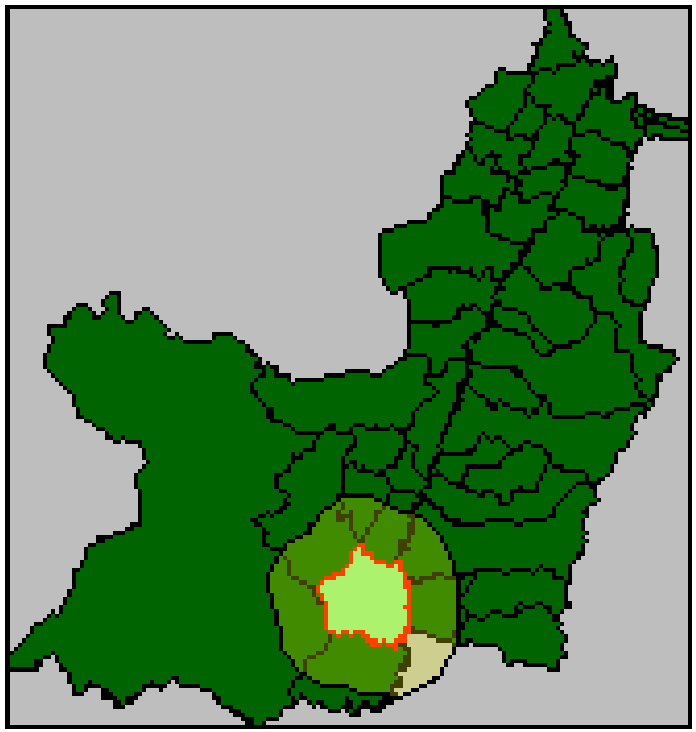


Buffer, una herramienta muy útil

- A veces se necesita, además de la ubicación de un polígono, una vecindad alrededor de él.
- **R** nos permite dibujar buffers alrededor de líneas, puntos y polígonos con la función **gBuffer** del paquete 'Rgeos'.

Buffer, una herramienta muy útil

Cali con Buffer de 15km



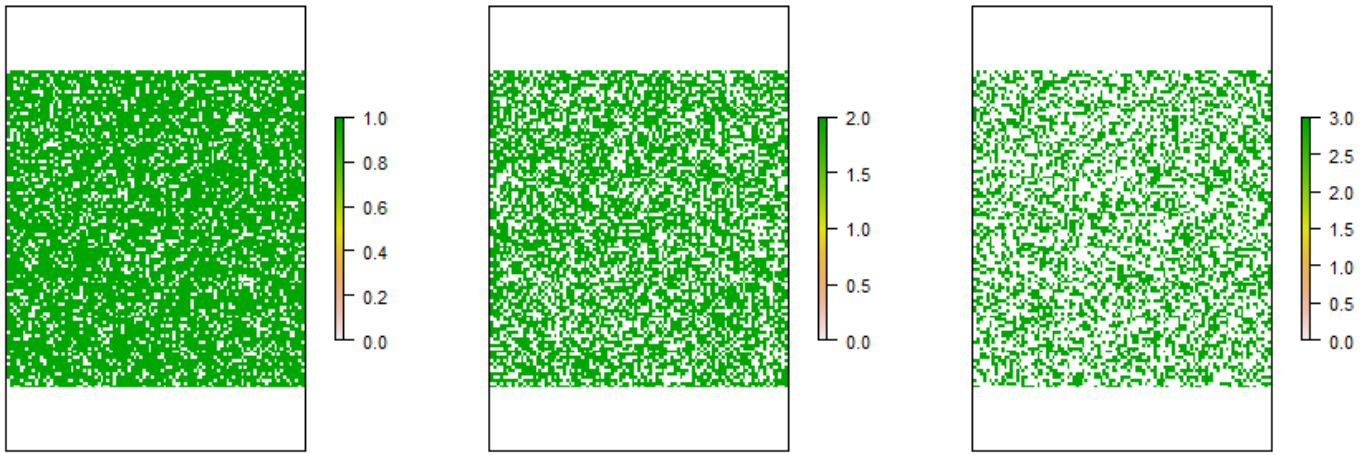
Rasterizar un polígono

- A veces se necesita tener los objetos 'SpatialPolygons' como objetos 'raster'. Por ejemplo, en un problema de clasificación.
- Esto se puede lograr fácilmente enmascarando un 'raster' dummy.
- La función **rasterize()** hace esto automáticamente.

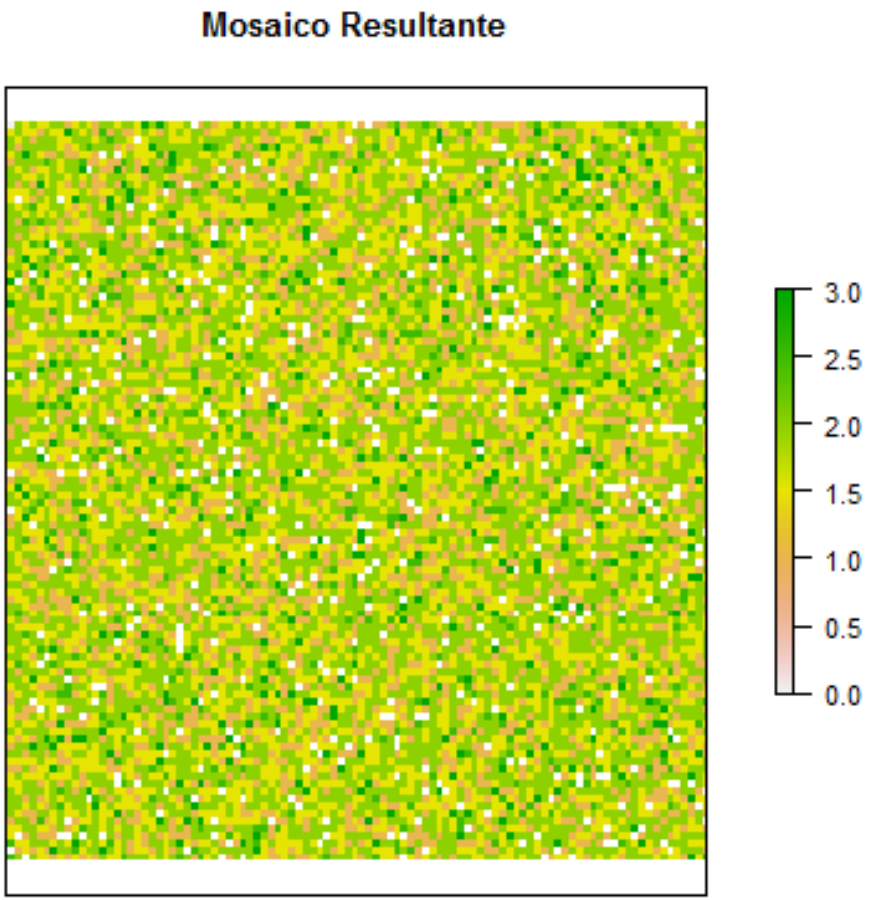
Merge y Mosaic

- A veces se quieren unir varios Rasters que pueden coincidir en algunos pixeles.
- Por ejemplo si se quiere armar un solo Raster grande de un mismo departamento.
- Las funciones **merge()** y **mosaic()** sirven para combinar rasters en un solo objeto.
- En los pixeles donde se yuxtapongan, la función pone un valor acorde a una función escogida por el usuario.
- Estas funciones son muy útiles para "Tapar Huecos".

Merge de Rasters con huecos



Merge de Rasters con huecos



Aplicaciones

- Con estas herramientas tan simples podemos procesar muchos datos espaciales como lo haríamos en ArcGis.
- Podemos manipular la extensión de mapas, lidiar con cobertura de nubes, hacer análisis "Por departamento" e , idealmente, problemas de clasificación.
- Veamos algunas de las funciones que vimos aplicadas a problemas reales

Estratificar el acceso a un río

Con el polígono referenciado de un río, podemos usar la función **gDistance** para construir tres niveles de cercanía a un río

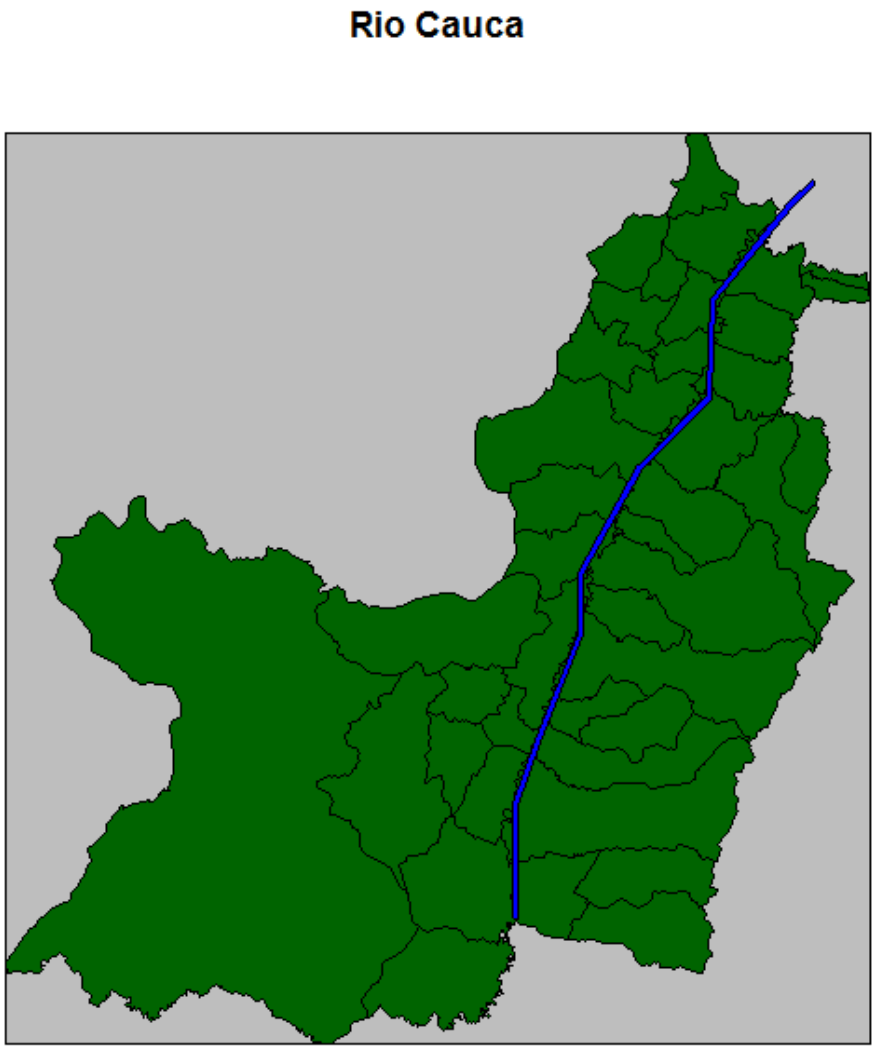
- Supongamos que tenemos el siguiente Raster, y esta linea diagonal es un río que pasa por ella.
- Podemos construir una función que para cada celda del Raster devuelva la distancia de su centroide al polígono.

Estratificar el acceso a un río

Con ayuda de la función **locator(n)**, definimos un "Rio Cauca" dándole click a 12 puntos en el polígono del Valle del Cauca.

```
#Hagamos un río
plot(valle.bb,col="gray",main="Rio Cauca")
plot(valle.sf,add=TRUE,col="dark green")
coord_rio<-locator(12)
RioCauca<-SpatialLines(list(Lines(Line(coord_rio), ID="a")))
Riecito<-gBuffer(RioCauca,byid=TRUE,width=700)
plot(Riecillo,col="blue",add=TRUE)
```

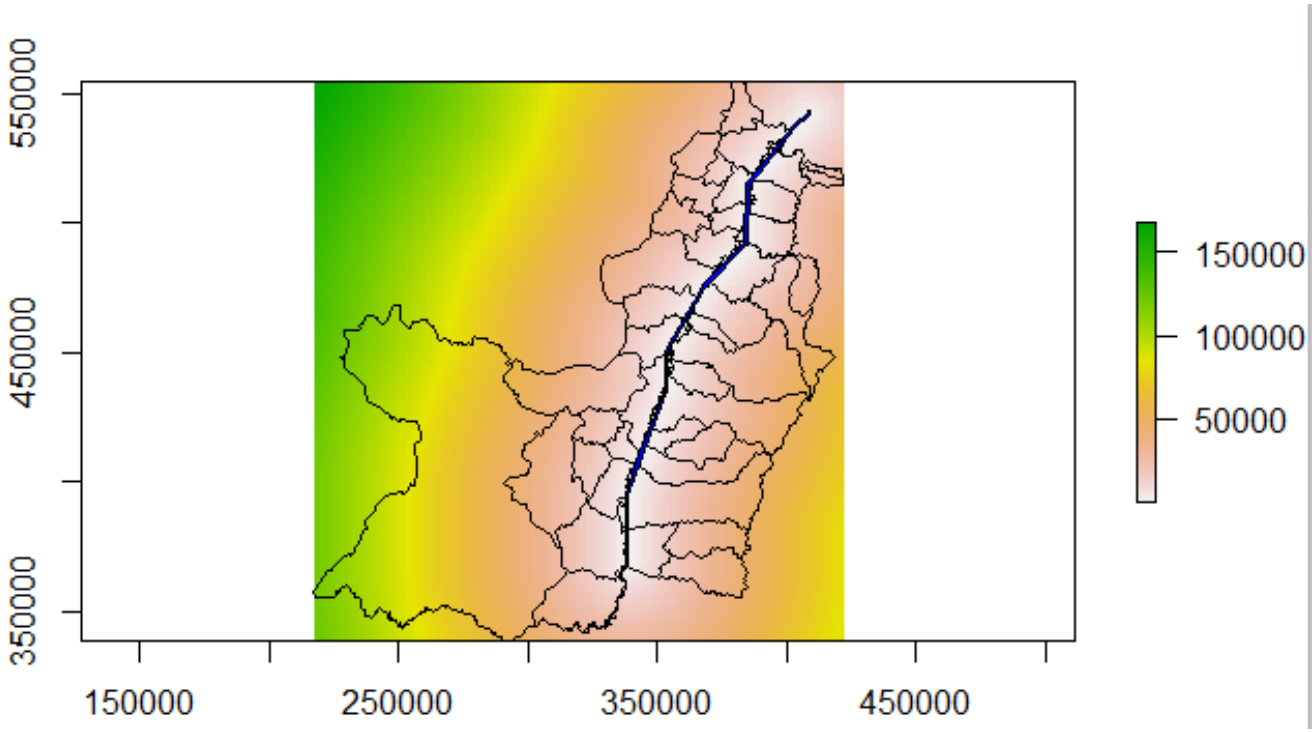
Un rio bastante decente



Raster: distancia al río

```
#Ahora hagamos un raster del tamaño del valle de 30x30
ras<-raster(nrow=100,ncol=100,extent(valle.sf))
distCauca<-function(n){
  pt<-xyFromCell(ras,cell=n,spatial=TRUE)
  proj4string(pt)<-CRS(proj4string(RioCauca))
  return(as.vector(gDistance(pt,RioCauca,byid=TRUE)))
}
ras[]<-distCauca(1:ncell(ras))
plot(ras)
plot(valle.sf,add=TRUE)
plot(Riecito,add=TRUE,col="blue")
```

Raster: distancia al río

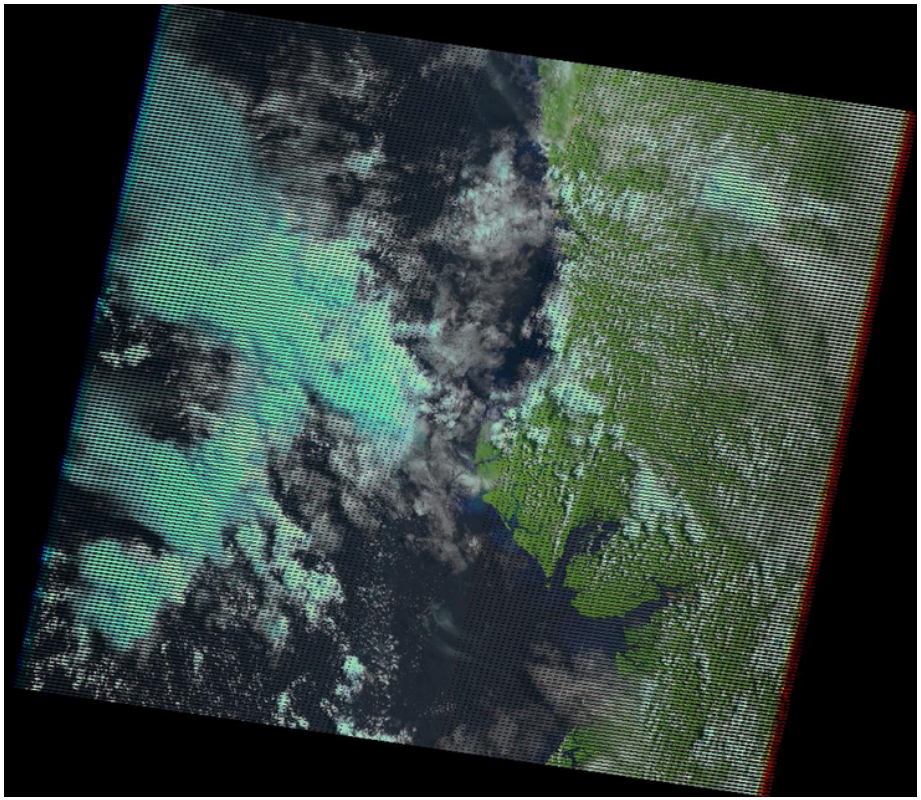


Remover nubes en una imagen satelital

- Las nubes son un inconveniente para cualquier estrategia que pretenda "minar" los colores de los pixeles del terreno en una imagen satelital.
- Afortunadamente tenemos las herramientas necesarias para eliminarlas.
- La función **clouds()** del paquete 'Landsat' crea una máscara con las nubes de una escena.
- Con la máscara podemos enmascarar toda la escena y quedarnos con los pixeles que caen fuera de las nubes.

Nubes inconvenientes

Las nubes pueden tapar una parte importante de la escena:



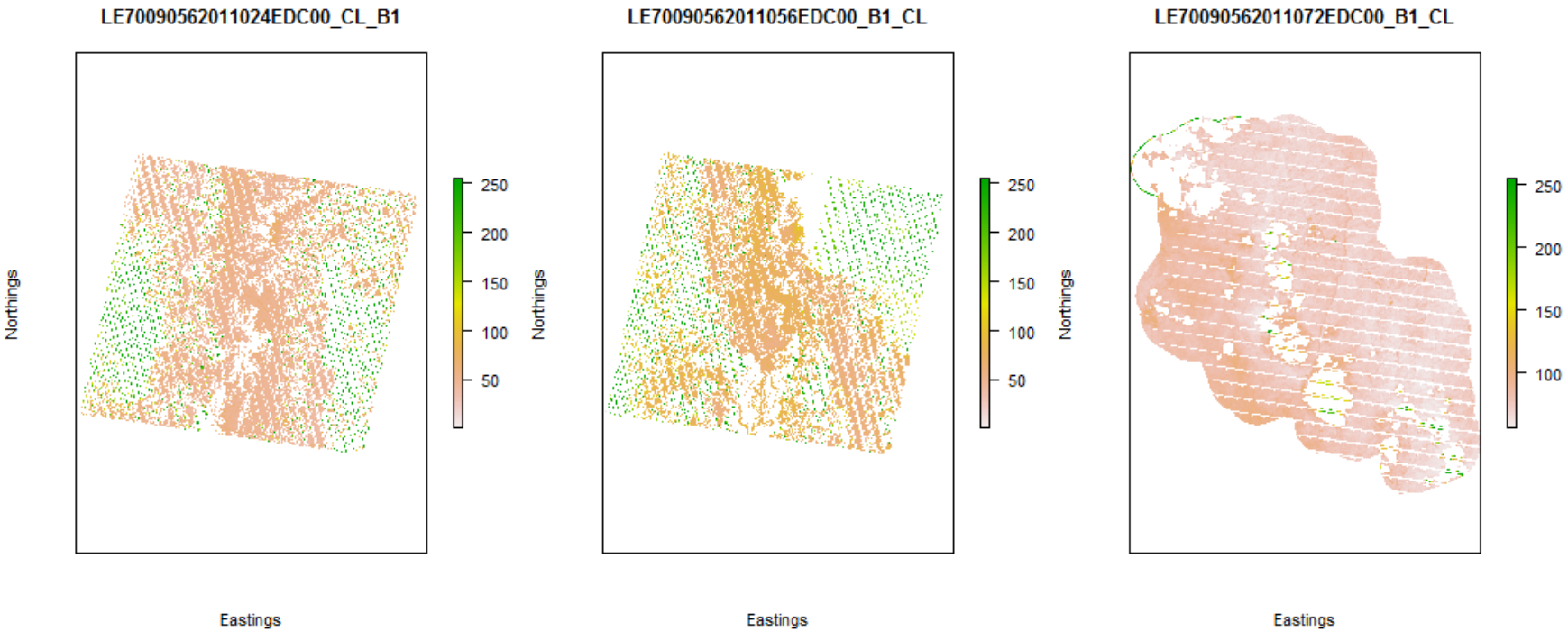
Cloud Masking

Figure: Con Nubes

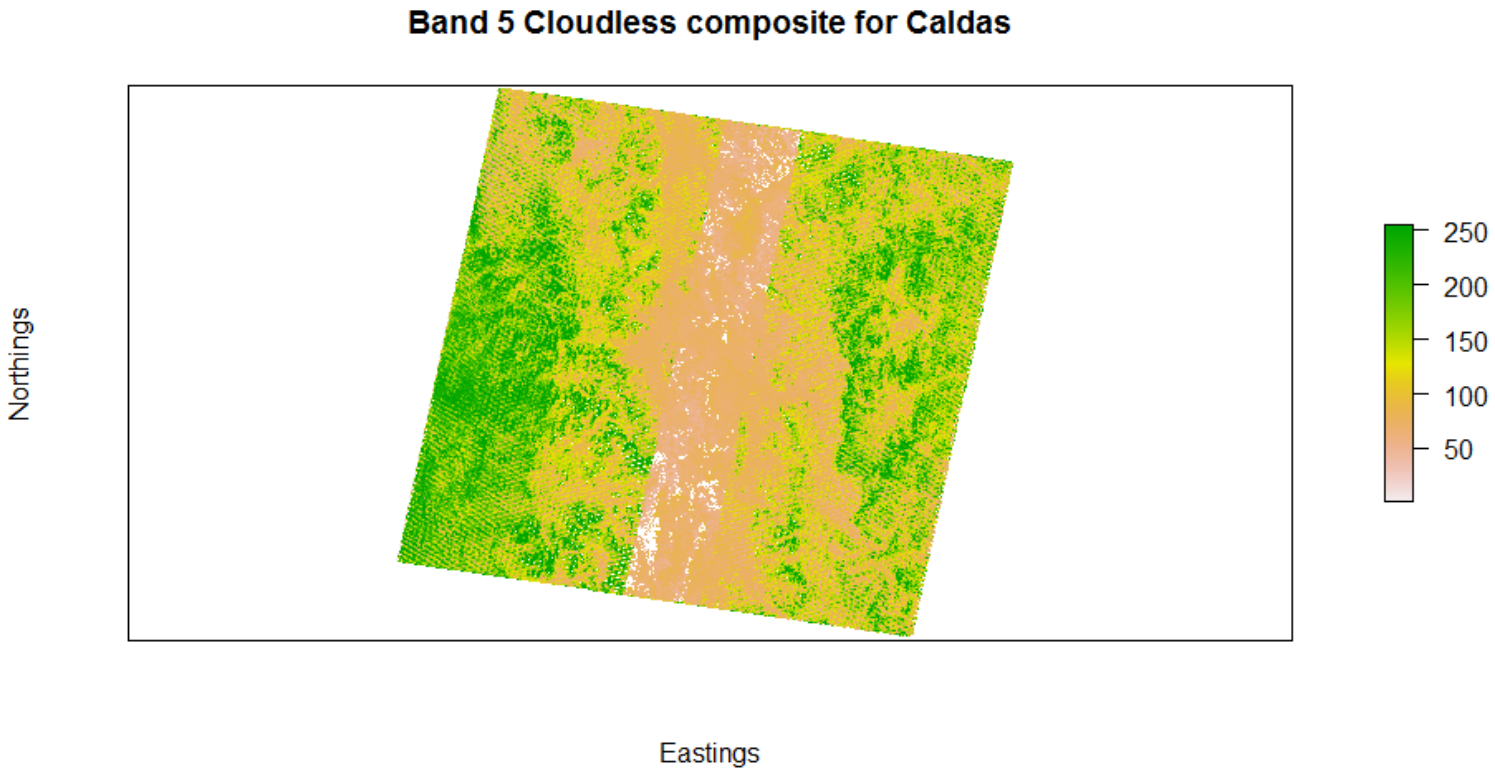
Mask of clouds (yellow) and their projection in the surface (orange)

Figure: Nubes Enmascaradas

Escenas de Caldas sin nubes



Mosaico de imagen sin nubes



Point in Polygon

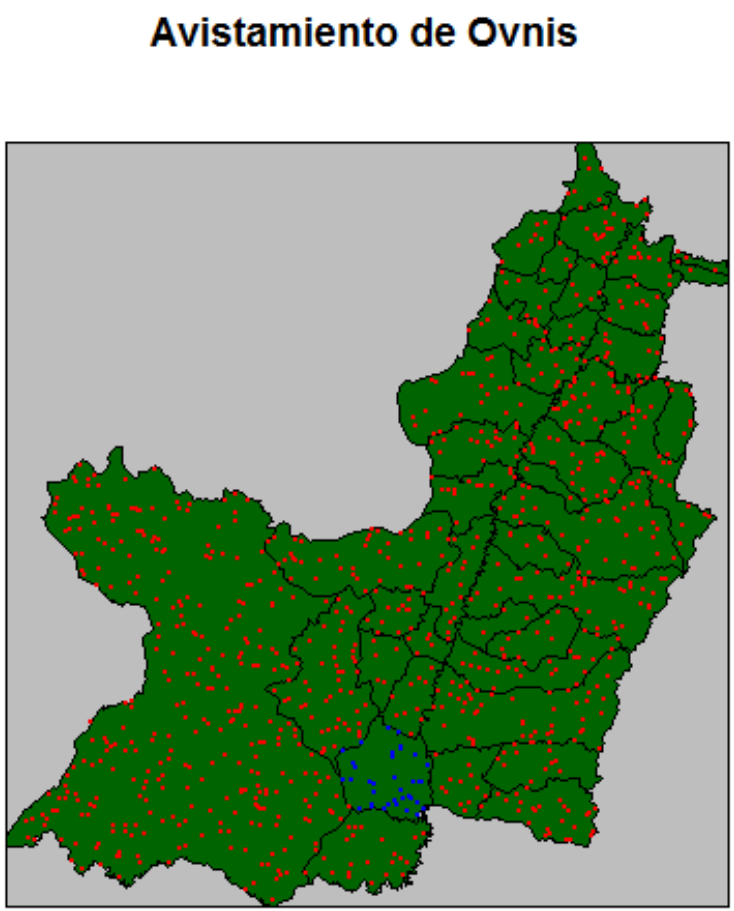
- Otra aplicación interesante es el del problema Punto-En-Polígono.
- Supongamos que tengo avistamientos de OVNI en todo el Valle del Cauca.
- ¿Cuántos avistamientos de OVNI están dentro de Cali?

Point in Polygon

- Con la función **spSample()** puedo samplear 50000 observaciones en el bounding-box del Valle del Cauca.
- Luego con la función **gContains()** obtengo un vector de booleanas que me indican para cada punto si pertenece a Cali o no.

```
> #Point in Polygon Analysis de Cali
> plot(valle.bb,col="gray",main="Avistamiento de Ovnis")
> plot(valle.sf,add=TRUE,col="dark green")
> ovnis<-spsample(valle.sf,1000,type="random")
> ovnis_cali<-ovnis[gWithin(ovnis,cali.sf,byid=TRUE)]
> length(ovnis_cali)
[1] 37
```

Point in Polygon



Crear una capa de clasificación y extraer

- Para poder hacer clasificación supervisada se necesita una matriz con valores para cada pixel, y valores de una **variable dependiente**.
- Esto se puede lograr rasterizando las minas con un Buffer alrededor, adjuntando el resultado como un Layer extra y usando la función **extract**
- Se explicará en más detalle en las siguientes diapositivas.