M1683. Intensificación en Saberes Técnicos: Diseño y Gestión de Bases de Datos Territoriales. Geoestadística I

Domingo Rasilla

01 enero, 2022

Contents

Esta sesión estará orientada a los pasos necesarios para efectuar un análisis geoestadístico, utilizando para ello diferentes datos

- *swiss_ppt.zip*. Contiene un conjunto de observaciones puntuales sobre precipitación en Suiza.
- *meuse.zip*. Contiene shapefiles correspondientes a un conjunto de datos con las muestras de suelo procedentes de una llanura aluvial próxima al río Mosa, en los Países Bajos, así como los bordes del río.
- *ne_50m_admin_0_countries.zip*. Contiene un shapefile de contornos de todos los paises del mundo.
- *oregon.zip*: Conjunto de datos de temperatura de las estaciones meteorológicas del estado de Oregon, en USA.

Como en ocasiones anteriores, deberá crearse una carpeta en el directorio de trabajo donde almacenar estos ficheros.

También será necesario trabajr con tres paquetes de R: **sf**, **stars** y **gstat**. Si no se instalado previamente, debe ejecutarse el siguiente comando; si ya se han instalado en una ocasión anterior debe saltarse dicho comando

```
pkgs <- c("ggplot2",
          "gstat",
          "RColorBrewer",
          "stars",
```

```
          "sf",
          "viridis")

install.packages(pkgs)
```

```
library(ggplot2)
library(gstat)
library(RColorBrewer)
library(sf)
library(stars)
library(viridis)
```

1. IMPORTACIÓN DE LOS DATOS EN R

La sesión comienza con la importación de los diferentes conjuntos de datos. Téngase en cuenta las diferentes características de cada uno de ellos:

1.1 Precipitación en Suiza

Este conjunto de datos aparece en 3 ficheros: el primero contiene los datos de precipitación de cada estación meteorológica en formato csv (*swiss_ppt.csv*). Comenzaremos importando este fichero y convirtiéndolo en un objeto `sf`, en el que incluiremos una proyección Lambert (código EPSG 2056) como sistema de coordenadas de referencia .

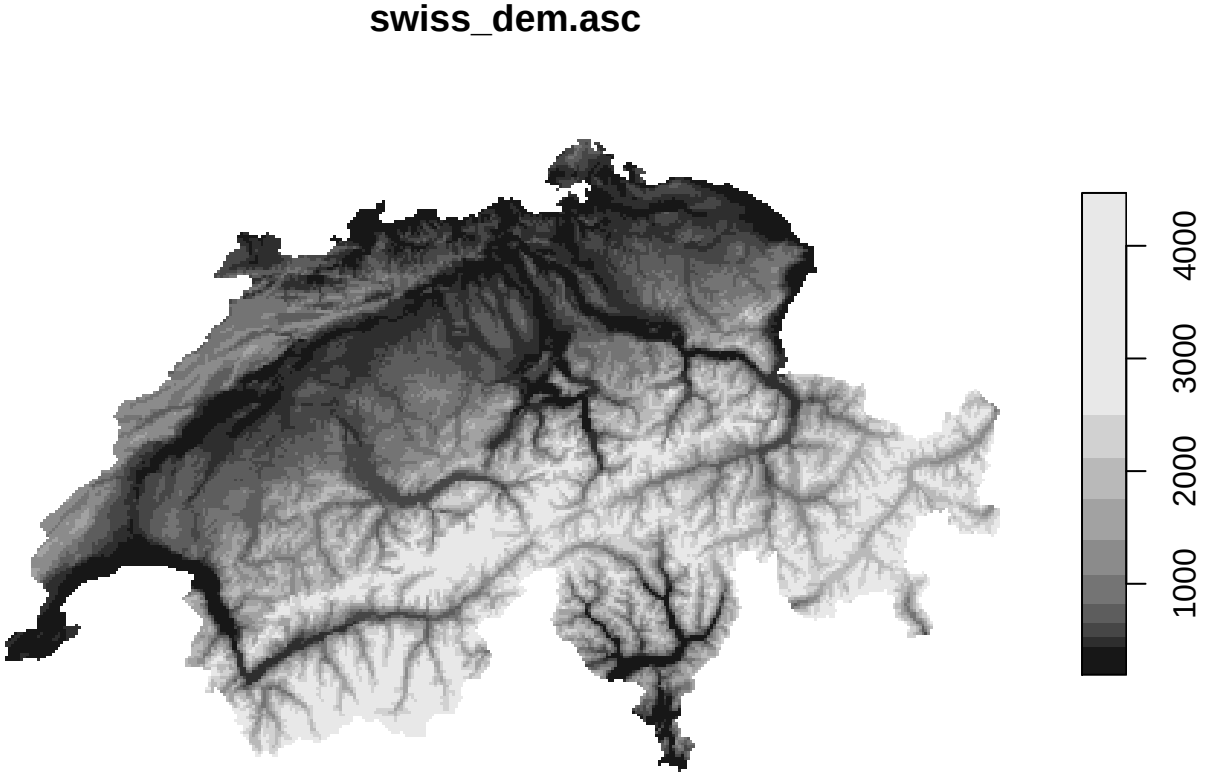
```
swiss <- read.csv("D:/Docencia_Master_2021/GEOG6000/sesion05/swiss_ppt/swiss_ppt.csv")
head(swiss)
```

```
##    id       x       y  ppt elev
## 1 287 2688674 1292361 18.4  754
## 2 292 2692432 1289049 12.1  516
## 3 259 2678227 1288974 13.8  561
## 4 319 2701430 1285778 12.6  412
## 5 257 2676752 1283409 15.6  419
## 6 302 2696966 1282408 10.0  423
```

```
swiss.sf <- st_as_sf(swiss,
                     coords = c("x", "y"),
                     crs = 2056)
```

En segundo lugar, cargaremos un dem correspondiente a Suiza como objeto `raster` (*swiss_dem.asc*)

```
swiss.dem <- read_stars("D:/Docencia_Master_2021/GEOG6000/sesion05/swiss_ppt/swiss_dem.asc")
st_crs(swiss.dem) <- 2056

plot(swiss.dem)
```



Finalmente, el último fichero a cargar es un shapefile con los límites administrativos del país (*ne_50m_admin_0_countries.shp*). Aunque el fichero original contiene los límites de todos los países del mundo, se extraerá sólo los correspondientes a Suiza:

```
countries <- st_read("D:/Docencia_Master_2021/GEOG6000/sesion05/ne_50m_admin_0_countries/ne_50m_admin_0_
```

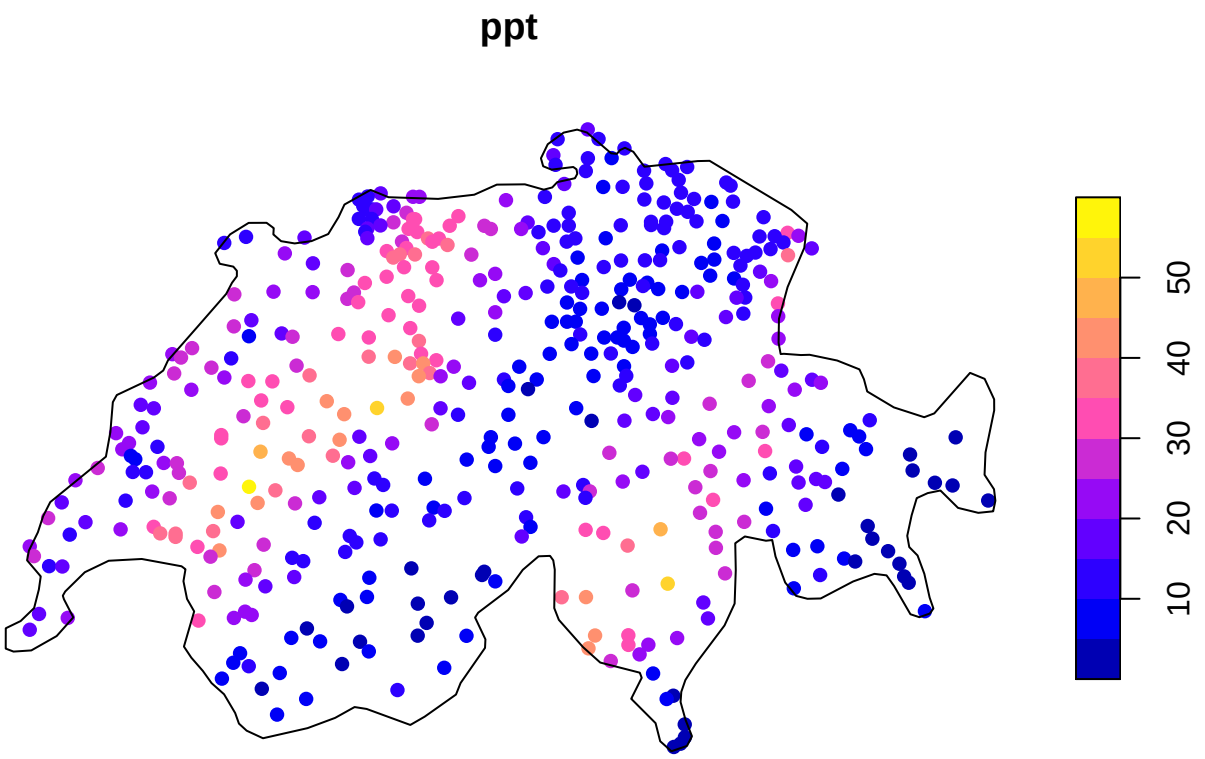
```
## Reading layer 'ne_50m_admin_0_countries' from data source
##   'D:\Docencia_Master_2021\GEOG6000\sesion05\ne_50m_admin_0_countries\ne_50m_admin_0_countries.shp'
##   using driver 'ESRI Shapefile'
## Simple feature collection with 241 features and 94 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -180 ymin: -89.99893 xmax: 180 ymax: 83.59961
## Geodetic CRS:  WGS 84
```

```
swiss.bord <- subset(countries, NAME == "Switzerland")
swiss.bord <- st_transform(swiss.bord, 2056)
```

A continuación, cartografiaremos los datos de precipitación en forma de puntos coloreados y los límites administrativos del país.

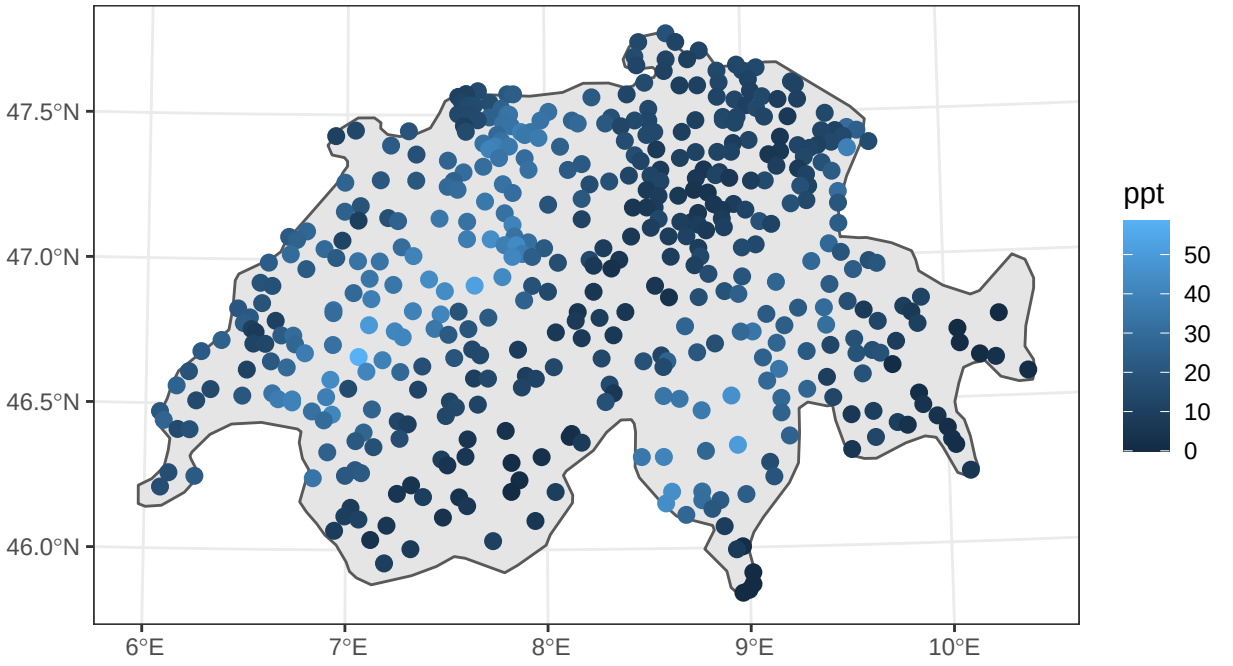
```
plot(swiss.sf["ppt"],
     reset = FALSE,
     pch = 16)
```

```
plot(st_geometry(swiss.bord), add = TRUE)
```



Esto mismo puede realizarse con **ggplot2**:

```
ggplot() +
  geom_sf(data = swiss.bord) +
  geom_sf(data = swiss.sf, aes(col = ppt), size = 2.5) +
  theme_bw()
```



1.2 Datos de suelos del Mosa

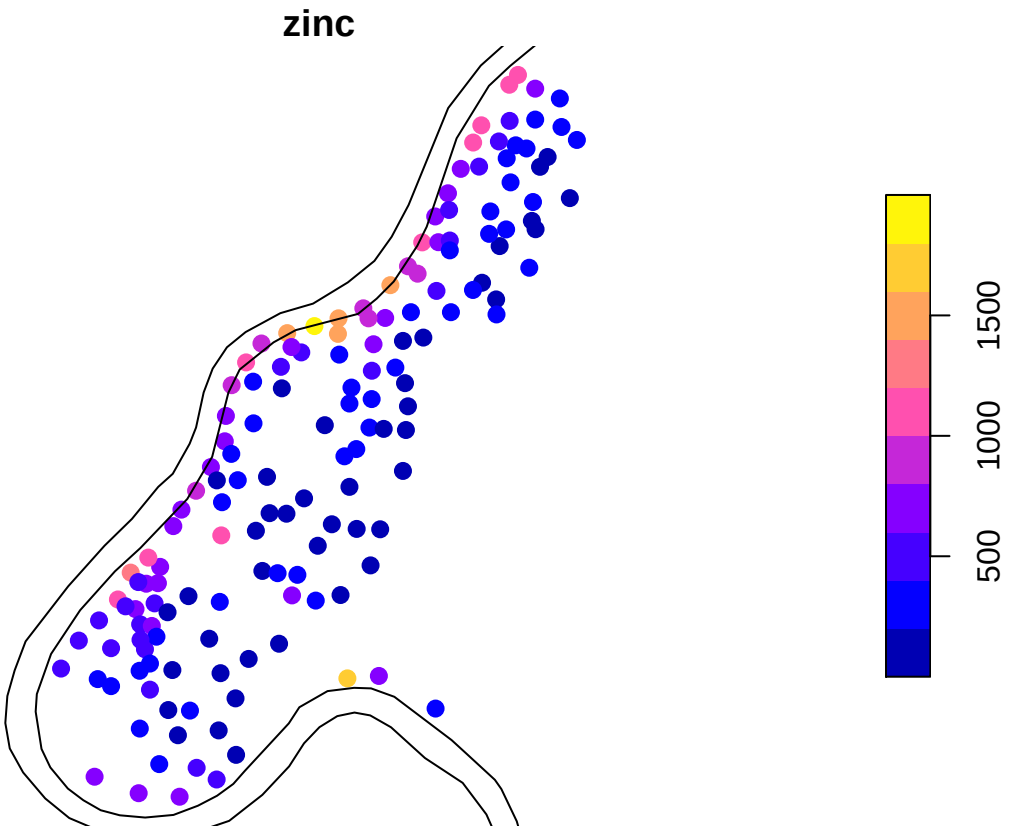
En este caso, sólo es necesario importar el shapefile con los puntos de observación (*meuse.shp*) y el fichero correspondiente al trazado del río (*meuseriv.shp*).

```
meuse <- st_read("D:/Docencia_Master_2021/GEOG6000/sesion05/meuse/meuse.shp", quiet = TRUE)

meuseriv <- st_read("D:/Docencia_Master_2021/GEOG6000/sesion05/meuse/meuseriv.shp", quiet = TRUE)
```

2. VARIOGRAMAS

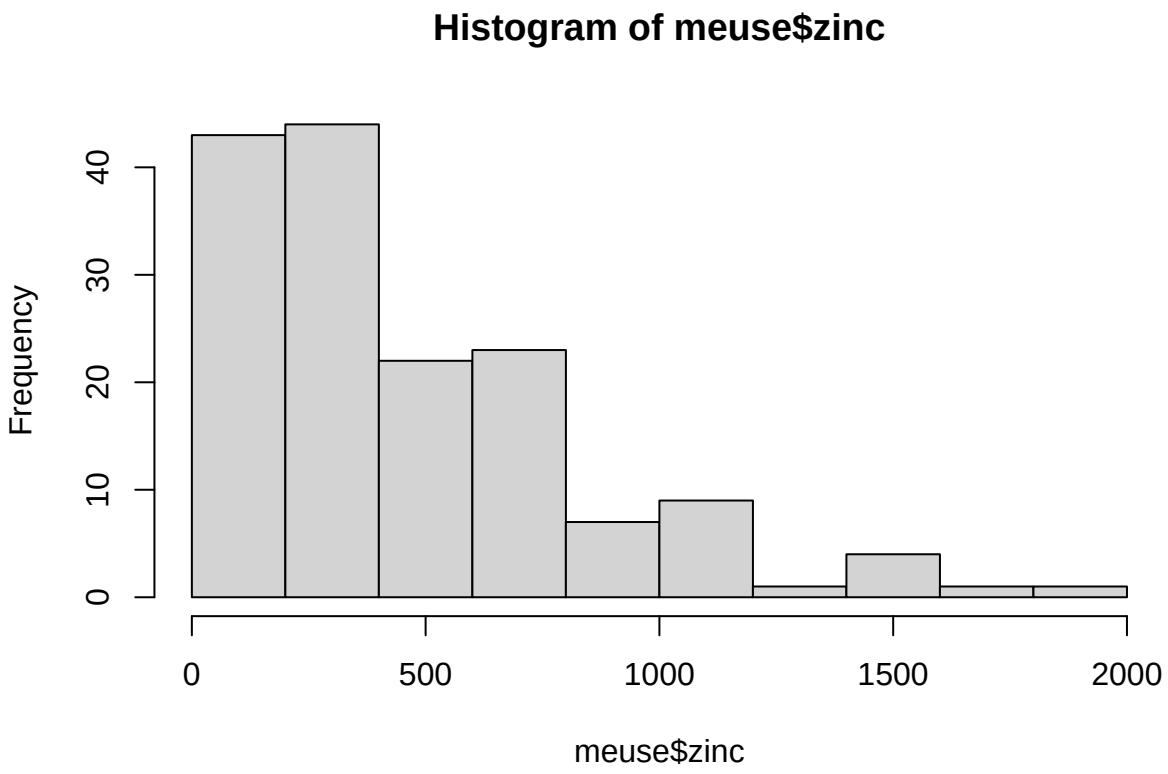
Usaremos el conjunto de datos del Mosa como ejemplo de cómo realizar un análisis de variogramas. Comenzaremos con una simple gráfica de las concentraciónes de Zinc usando la función `plot ()`:

```
plot(meuse["zinc"],
     pch = 16,
     cex = 1.25,
     reset = FALSE)

plot(meuseriv,
     add = TRUE,
     col = NA)
```



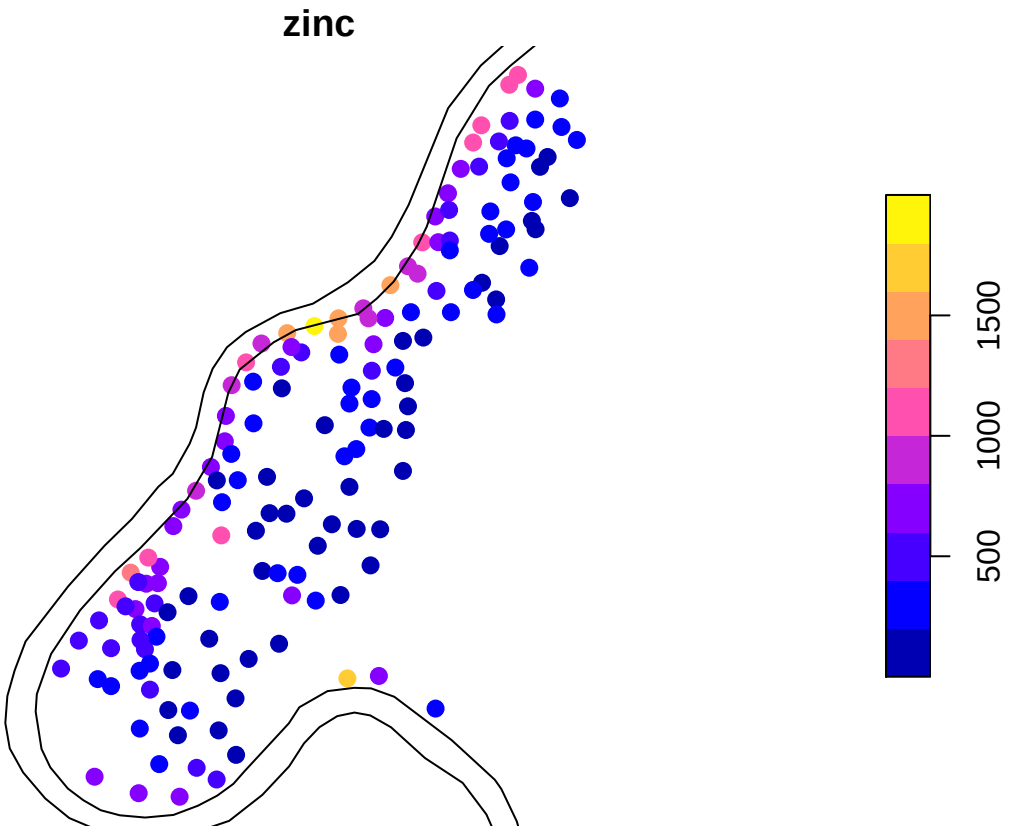
Al elaborar un histograma con los valores de ese metal aparece un claro sesgo a la derecha, con algunos valores muy elevados.

```
hist(meuse$zinc)
```



En consecuencia, y dado que la mayoría de los métodos que revisaremos asumen datos siguiendo una distribución normal, es conveniente transformarlos en sus logarítmos. Al border a cartografiarlos aparece claramente un patrón espacial, en el que las concetraciones más altas se ubican a lo largo del borde del río.

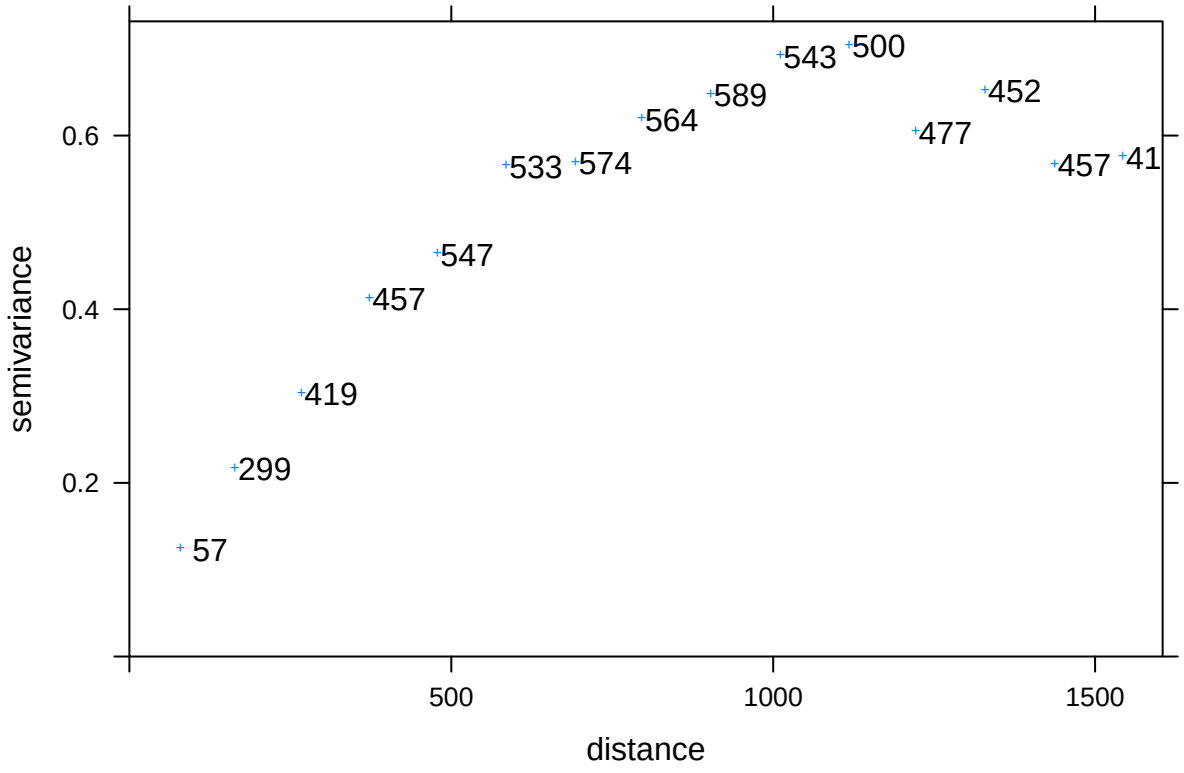
```
meuse$lzinc <- log(meuse$zinc)

plot(meuse["zinc"],
     pch = 16,
     cex = 1.25,
     reset = FALSE)

plot(meuseriv,
     add = TRUE,
     col = NA)
```



2.2 Variograma de la muestra

La función `variogram ()` del paquete **gstat** se construye un variograma con los datos transformados logarítmicamente. Al principio sólo usaremos la fórmula `lzinc ~ 1`, que indica que estamos asumiendo que el valor medio del logaritmo de zinc no varía en nuestra región. Más adelante podremos incluir covariables. Finalmente trazamos el variograma, agregando un argumento para mostrar el número de pares de puntos usados para calcular cada punto

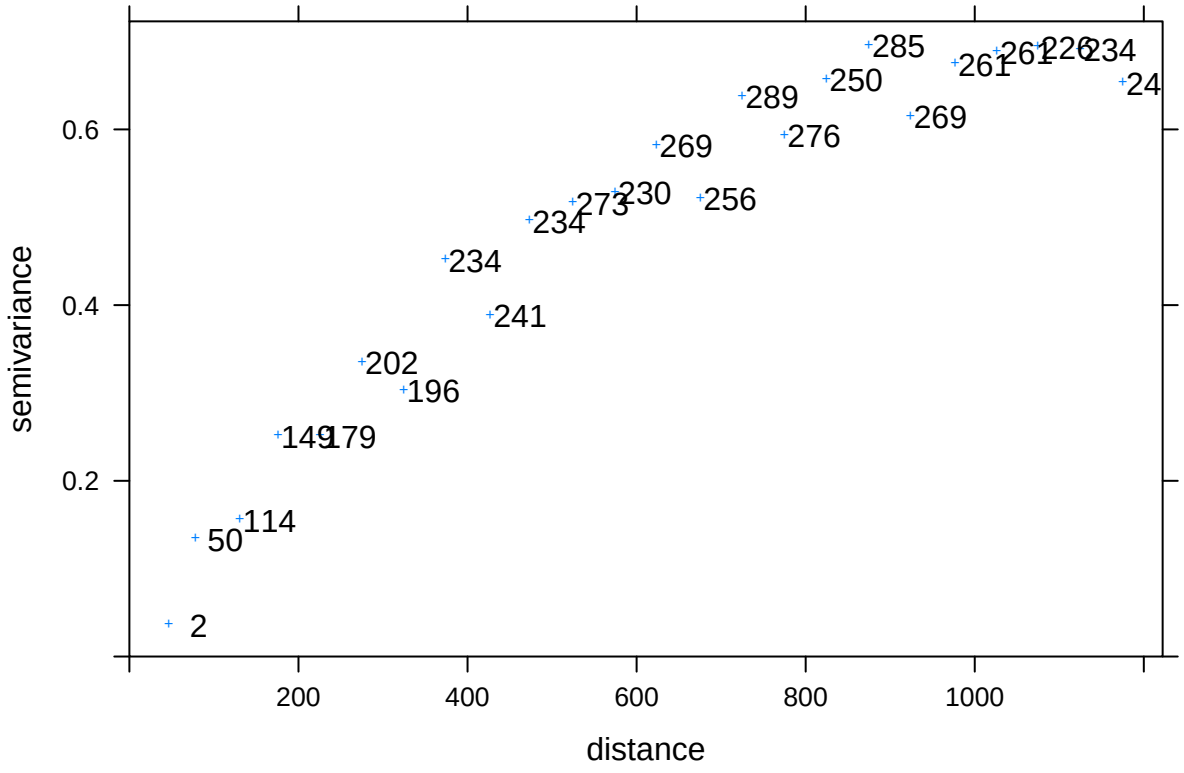
```
mzinc.var <- variogram(lzinc ~ 1, data = meuse)

plot(mzinc.var, plot.numbers = TRUE, pch = '+')
```



Este primer análisis puede enriquecerse especificando el número de retardos que se incluirán en el análisis mediante el uso de dos parámetros, lo que nos permite buscar dependencias espaciales en un rango de distancias mayor o menor. - El primero de ellos, `cutoff`, especifica la distancia máxima sobre la que consideraremos las diferencias por pares entre puntos. - El segundo, `width`, especifica el tamaño de cada retardo.

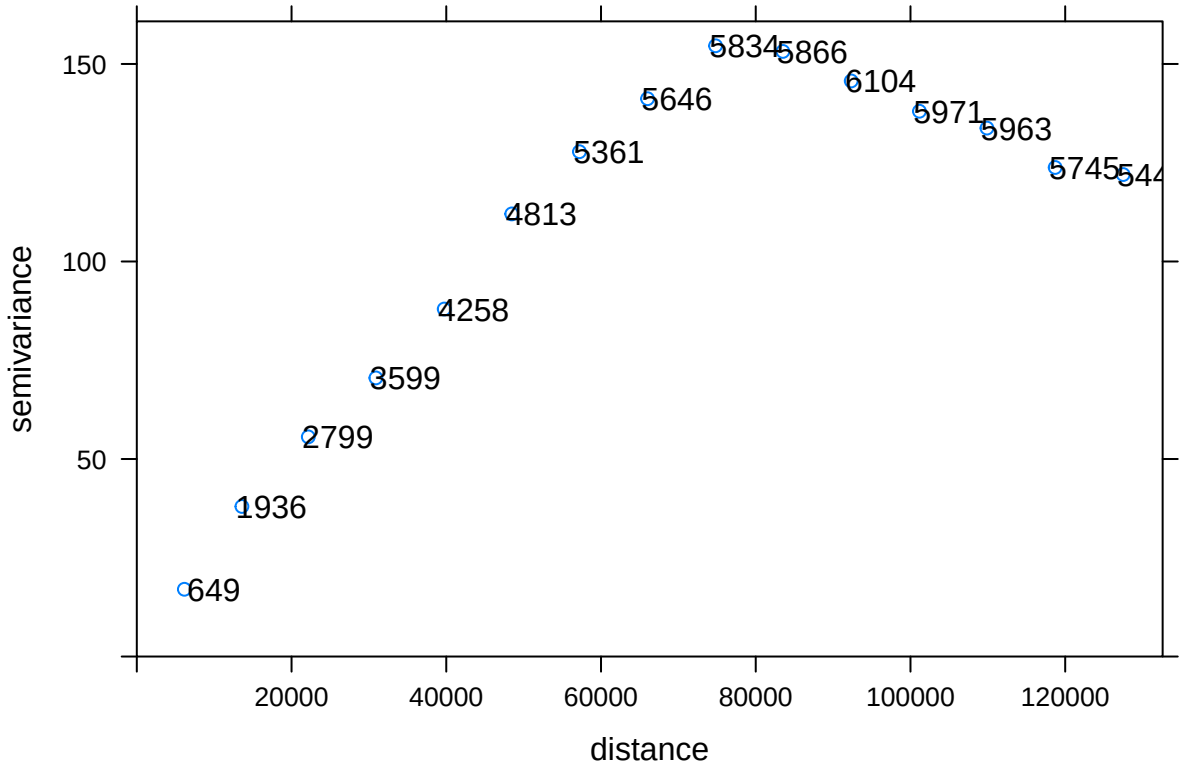
Sus valores se han establecido de forma arbitraria, por lo es importante experimentar con estos parámetros para obtener el variograma óptimo.

```
mzinc.var2 <- variogram(lzinc ~ 1,
                        data = meuse,
                        cutoff = 1200,
                        width = 50)

plot(mzinc.var2, plot.numbers = TRUE, pch = '+')
```



De la misma manera, podemos usar la función `variogram()` para construir un variograma con los datos de precipitación de Suiza.

```
ppt.var <- variogram(ppt ~ 1, swiss.sf)

plot(ppt.var, plot.numbers = TRUE)
```



2.2 Modelización de variograma

Una vez obtenido el variograma correspondiente a la muestra de datos reales, es posible ajustar un variograma modelo a esos datos. Existe un conjunto de modelos paramétricos estandar que pueden ser utilizados; para conoccer los que incluye el paquete **gstat** teclee

```
vgm()
```

```
##    short                                     long
## 1    Nug                              Nug (nugget)
## 2    Exp                         Exp (exponential)
## 3    Sph                           Sph (spherical)
## 4    Gau                            Gau (gaussian)
## 5    Exc        Exclass (Exponential class/stable)
## 6    Mat                               Mat (Matern)
## 7    Ste Mat (Matern, M. Stein's parameterization)
## 8    Cir                             Cir (circular)
## 9    Lin                               Lin (linear)
## 10   Bes                               Bes (bessel)
## 11   Pen                      Pen (pentaspherical)
## 12   Per                             Per (periodic)
## 13   Wav                                 Wav (wave)
## 14   Hol                                 Hol (hole)
## 15   Log                        Log (logarithmic)
```

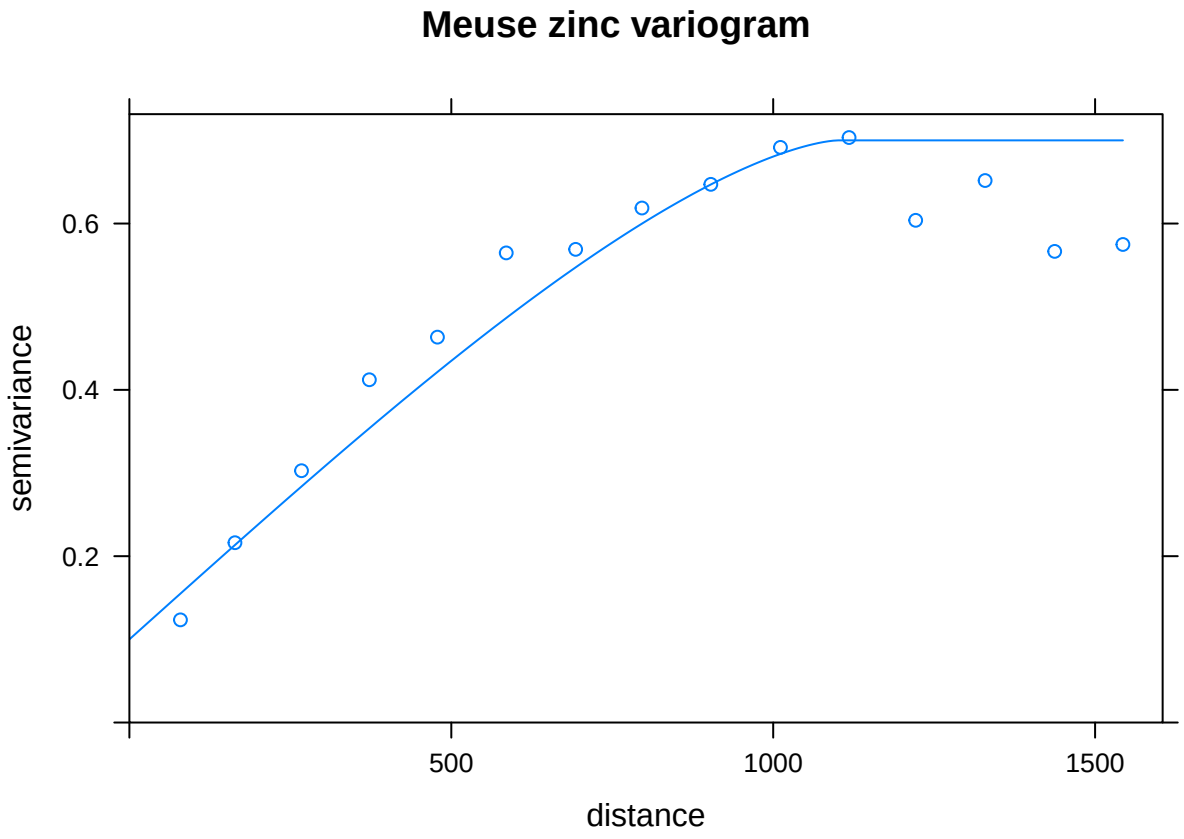
```
## 16   Pow                              Pow (power)
## 17   Spl                            Spl (spline)
## 18   Leg                          Leg (Legendre)
## 19   Err                Err (Measurement error)
## 20   Int                         Int (Intercept)
```

Para ajustar un modelo, primero es necesario crear el modelo manualmente; posteriormente, con la función `fit.variogram ()`, ajusta ese modelo al variograma de la muestra mediante un método de mínimos cuadrados ponderados. Para obtener el primer modelo es necesario aportar la siguiente información en forma de argumentos:

- la forma modelo
- el valor de la pepita o `nugget`, que es la intersección con el eje Y)
- el rango del modelo (la distancia a la que el variograma de muestra se vuelve plano)
- el umbral o 'sill´, el valor de semivariancia (eje y) del rango

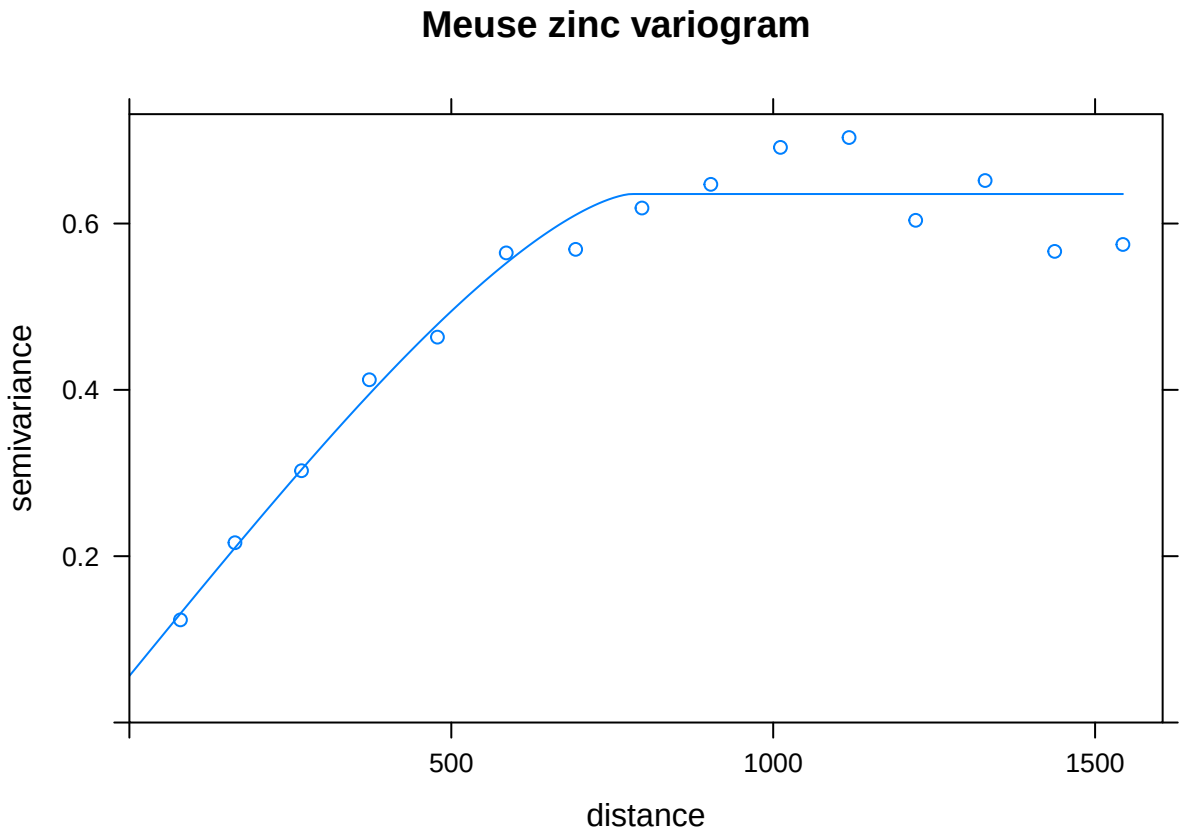
En la sintaxis de ejemplo, las variables serán especificadas una a una y por separado; posteriormente, la función `vgm ()` constuirá ese modelo. El código contiene unos valores arbitrarios, por lo que vale la pena volver a trazar el variograma original para ver cómo se comparan estos valores con la variable de muestra.

```
modNugget <- 0.1
modRange <- 1100
modSill <- 0.6

mzinc.vgm1 <- vgm(psill = modSill,
                  model = "Cir",
                  range = modRange,
                  nugget = modNugget)

plot(mzinc.var, mzinc.vgm1, main = "Meuse zinc variogram")
```



Dado que el ajuste de nuestro modelo al variograma de muestra no es totalmente correcto, es posible usar un método iterativo de mínimos cuadrados ponderados (`fit.variogram ()`) para ajustar mejor el variograma del modelo al variograma de muestra.

```
mzinc.vgm2 <- fit.variogram(mzinc.var, mzinc.vgm1)

plot(mzinc.var, mzinc.vgm2, main = "Meuse zinc variogram")
```



Puede comprobarse los parámetros finales del variograma escribiendo el nombre del objeto que corresponde al modelo ajustado:

```
mzinc.vgm2
```

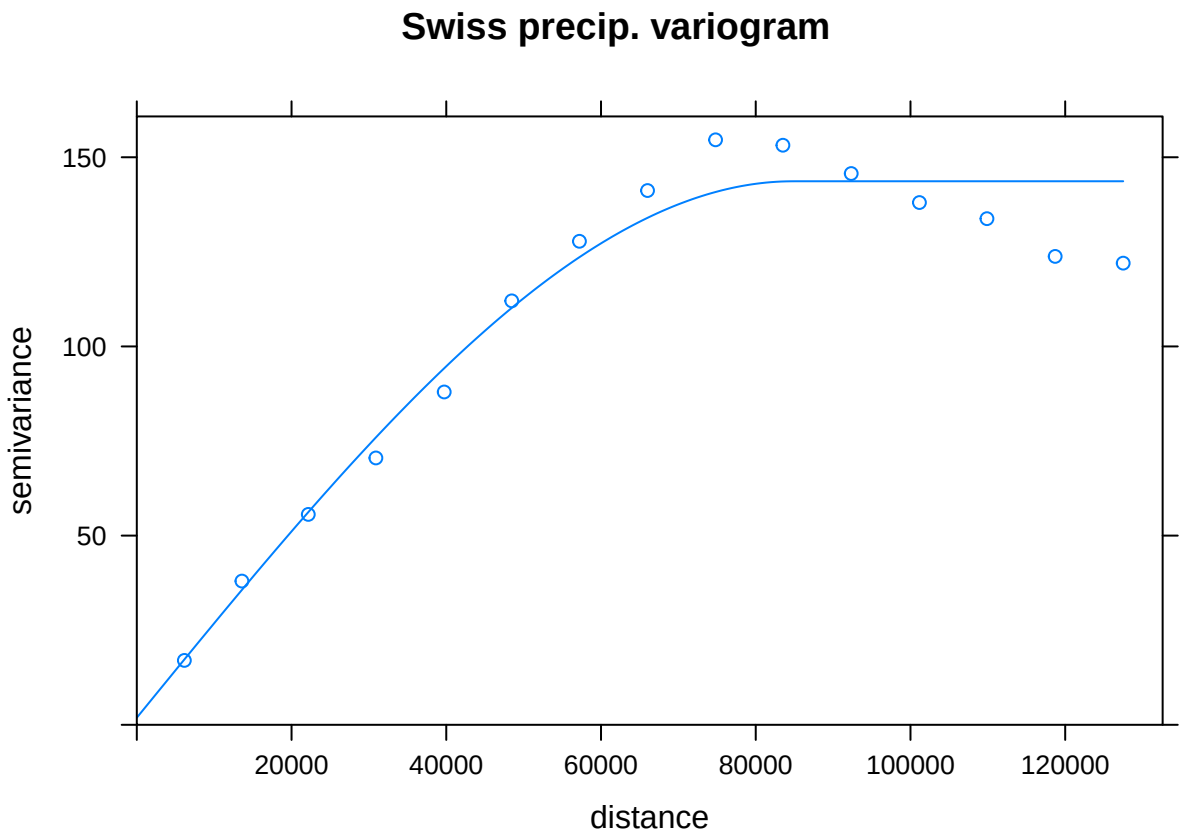
```
##   model      psill    range
## 1   Nug 0.05605498   0.0000
## 2   Cir 0.57945087 779.3562
```

La rutina de ajuste es generalmente bastante robusta y se ajustará a un modelo inicial que puede ser algo diferente del variograma de muestra, pero siempre vale la pena trazar el nuevo modelo para estar seguro. Es conveniente también rehacer los cálculos utilizando otros modelos, por ejemplo, el esférico.

De la misma manera, podemos proceder para obtener un modelo de variograma con los datos de precipitación. Recuerde revisar el variograma de muestra del paso anterior para estimar visualmente los tres parámetros necesarios.

```
modNugget <- 10
modRange <- 75000
modSill <- 140

ppt.vgm1 <- vgm(psill = modSill,
                model = "Sph",
                range = modRange,
                nugget = modNugget)
```

```
ppt.vgm2 <- fit.variogram(ppt.var, ppt.vgm1)

plot(ppt.var, ppt.vgm2, main = "Swiss precip. variogram")
```



3. PREDICCIÓN ESPACIAL

3.1 Kriging ordinario

Una vez obtenido ese modelo ajustado, puede utilizarse para predecir los valores de una determinada variable sobre un espacio, por ejemplo la precipitación en Suiza.

La función `krige ()` realiza esa predicción espacial, utilizando kriging ordinario como método predeterminado. Esto requiere el siguiente conjunto de argumentos:

- Una fórmula que especifica la variable a predecir (que puede incluir covariables).
- El objeto Spatial* con los valores observados de la variable.
- Un objeto Spatial* con las coordenadas que se utilizarán para la predicción.
- El modelo de variograma ajustado
- Un parámetro opcional que limita el número de puntos que se utilizarán para predecir los valores en una ubicación determinada.

Consulte `? Krige` para conocer otros parámetros.

```
ppt.pred.ok <- krige(ppt ~ 1,
                     locations = swiss.sf,
                     newdata = swiss.dem,
                     model = ppt.vgm2,
                     nmax = 40)
```

```
## [using ordinary kriging]
```

3.1.1 Cartografía de los resultados

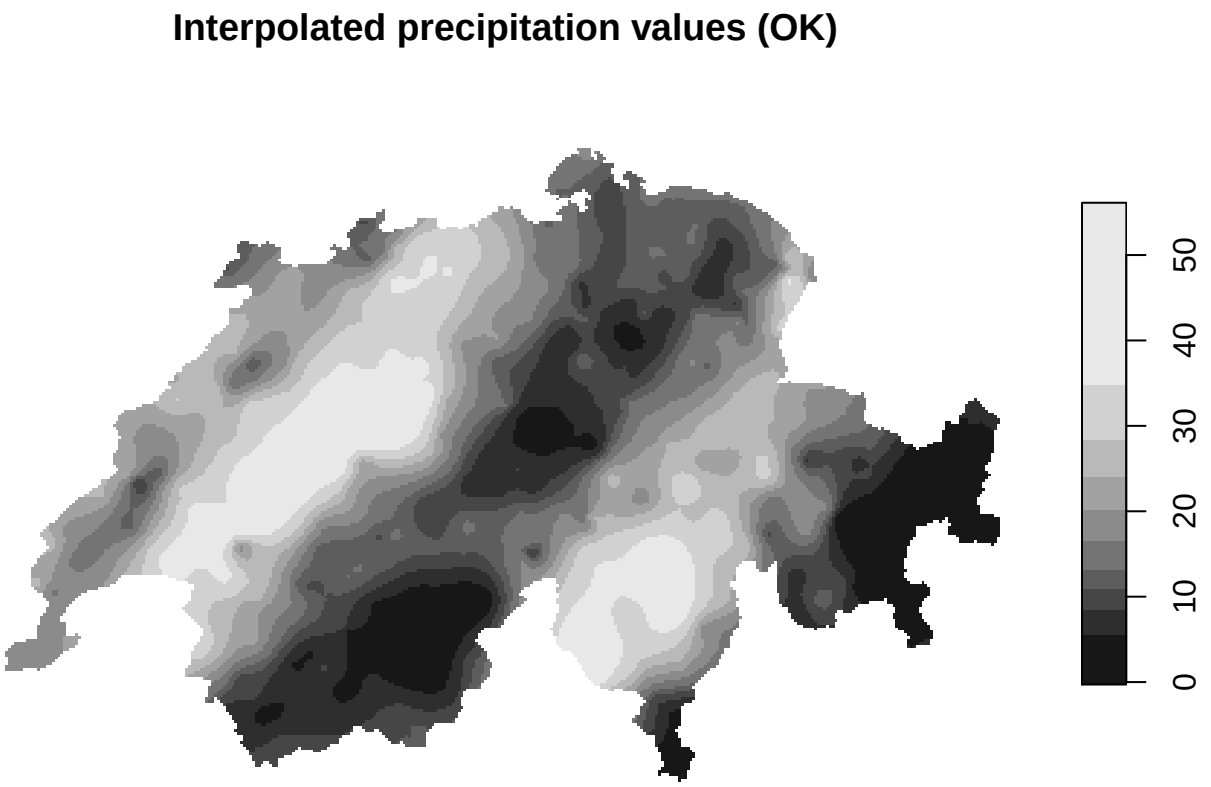
La salida de la función `krige ()` es un SpatialPixelsDataFrame (un formato espacial algo antiguo), que tiene un espacio llamado `data`, que contiene predicciones en una variable llamada`var1.pred` y errores de predicción en `var1.var`:

```
names(ppt.pred.ok)
```

```
## [1] "var1.pred" "var1.var"
```

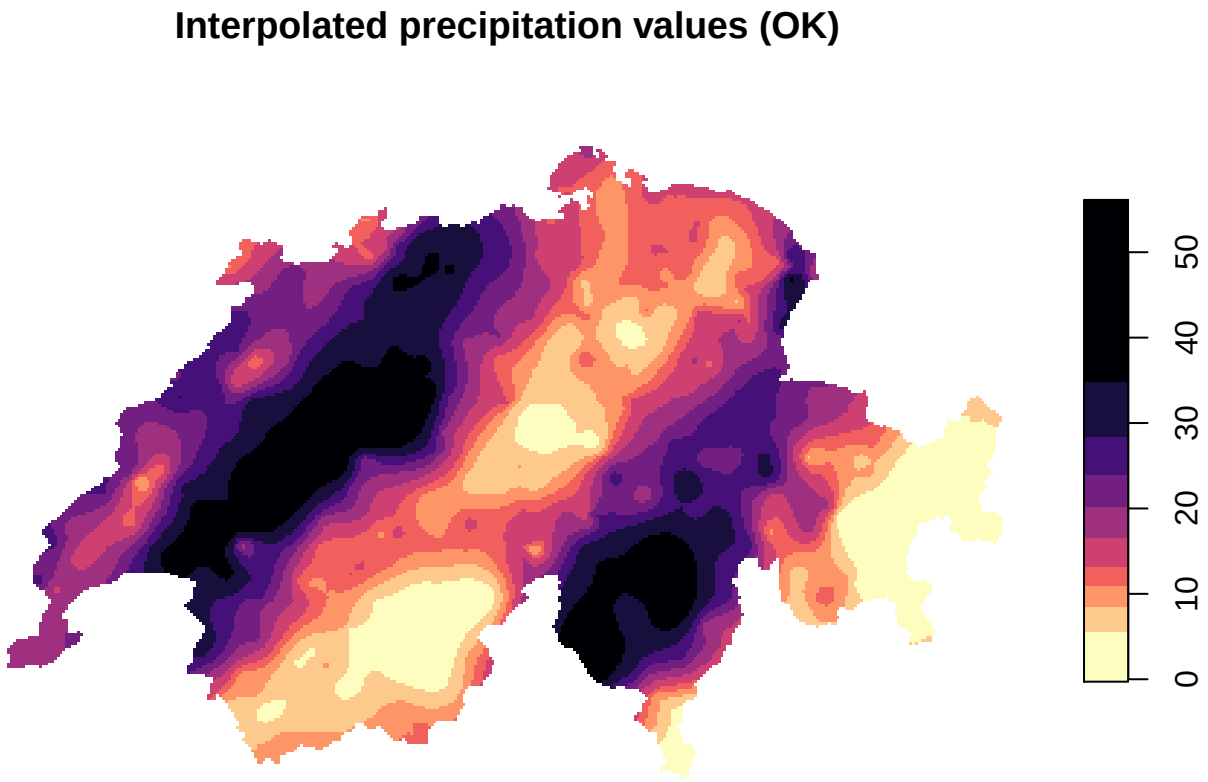
Para comprobar los valores predichos, simplemente cartografiamos la primera variable del objeto:

```
plot(ppt.pred.ok, main = "Interpolated precipitation values (OK)")
```



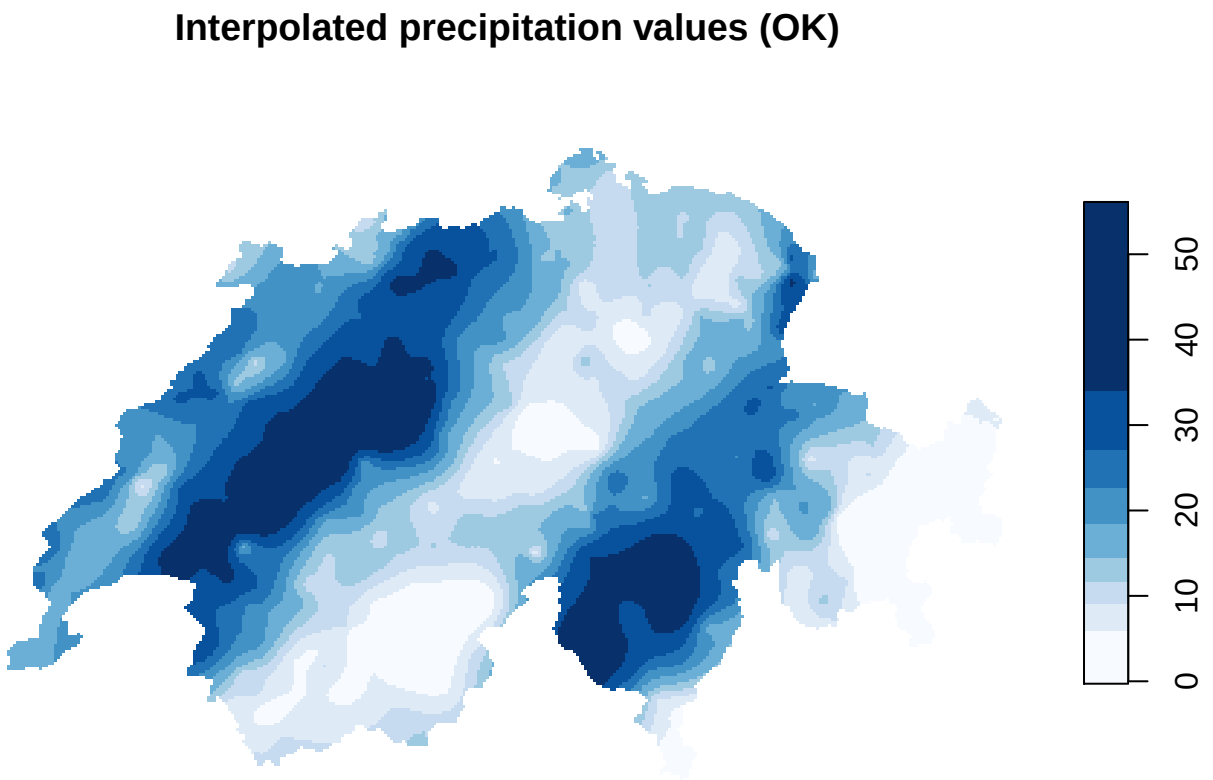
Se puede mejorar sustancialmente este mapa añadiendo algo de color usando algunos paquetes específicos que merece la pena instalar por defecto. Un ejemplo usado una de las plaetas del paquete **viridis** [palettes][virID]:

```
nbreaks <- 11
my.pal <- rev(magma(10))

plot(ppt.pred.ok,
     col = my.pal,
     main = "Interpolated precipitation values (OK)")
```



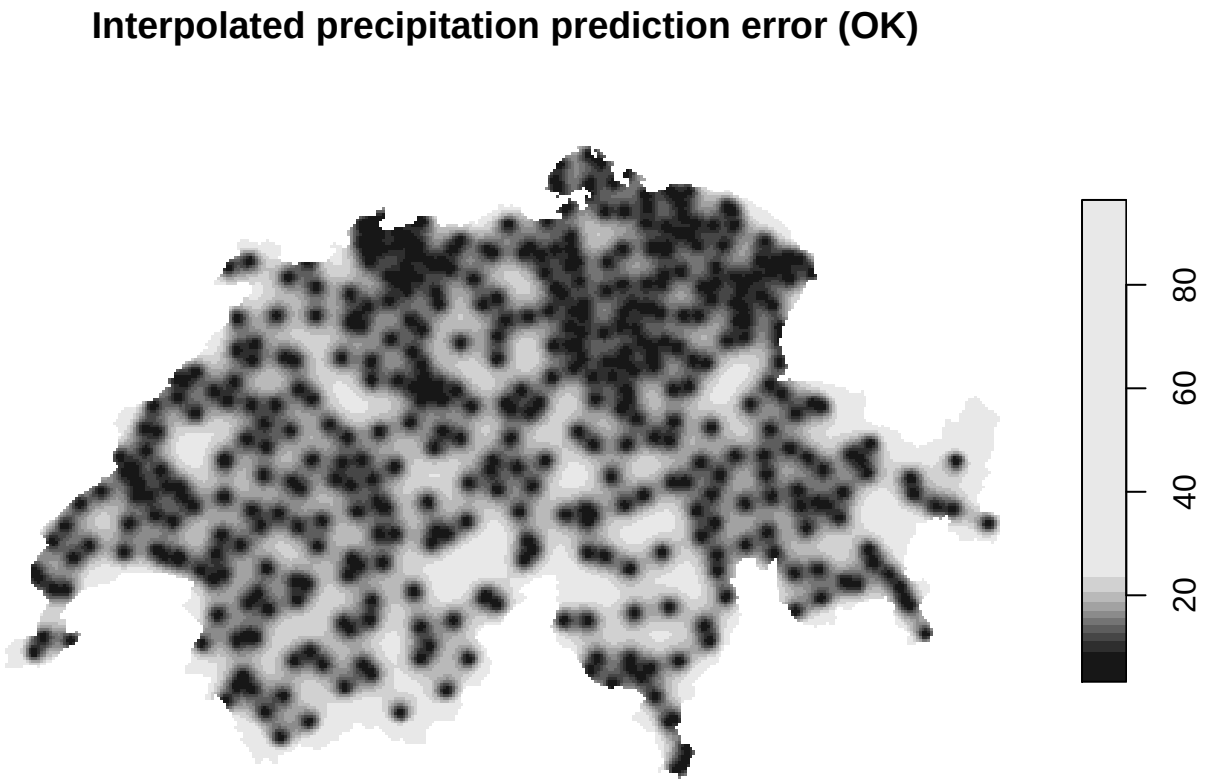
Aquí se puede comprobar otro ejemplo usando el paquete [RColorBrewer][rcbID] (para conocer todo el set de paletas, escriba`display.brewer.all()`):

```
my.pal <- brewer.pal(9, "Blues")

plot(ppt.pred.ok, col = my.pal,
     main = "Interpolated precipitation values (OK)")
```



Y para cartografíar los errores de la predicción:

```
plot(ppt.pred.ok["var1.var"],
     main = "Interpolated precipitation prediction error (OK)")
```



3.2 Evaluación de la calidad del modelo.

Para evaluar el rendimiento de un modelo siguiendo el procedimiento de kriging, se puede utilizar una validación cruzada con n veces (también llamada k veces). Este procedimiento divide los datos en n subconjuntos y luego predice iterativamente cada subconjunto a partir de los restantes $n-1$ conjuntos (cojunto de prueba). La función `krige.cv ()` realiza esta validación cruzada: toma los mismos argumentos que la función `krige ()`, pero deja fuera un objeto con coordenadas para nuevas predicciones, y, mediante el argumento `nfold`, el número de subconjuntos que se utilizarán. El procedimiento emite algunas advertencias sobre la proyección, pero pueden ser ignoradas.

```
ppt.cv.ok <- krige.cv(ppt ~ 1,
                      locations = swiss.sf,
                      model = ppt.vgm2,
                      nmax = 40,
                      nfold = 5)

head(ppt.cv.ok)
```

```
## Simple feature collection with 6 features and 6 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 2676752 ymin: 1282408 xmax: 2701430 ymax: 1292361
## Projected CRS: CH1903+ / LV95
##   var1.pred var1.var observed  residual     zscore fold                geometry
```

```
## 1  12.68214 21.97131     18.4  5.717864  1.2198486    4 POINT (2688674 1292361)
## 2  14.91461 16.68816     12.1 -2.814610 -0.6889916    2 POINT (2692432 1289049)
## 3  17.00452 23.56597     13.8 -3.204522 -0.6601166    5 POINT (2678227 1288974)
## 4  11.17502 22.75744     12.6  1.424983  0.2987087    4 POINT (2701430 1285778)
## 5  13.27573 13.76713     15.6  2.324265  0.6264176    1 POINT (2676752 1283409)
## 6  11.59967 17.41110     10.0 -1.599668 -0.3833687    5 POINT (2696966 1282408)
```

La salida de esta función es un objeto espacial con las siguientes variables:

- `var1.pred`: la predicción, mediante validación cruzada, de la variable en cada lugar (cuando ésta forma parte del conjunto de prueba)
- `var1.var`: el error de predicción, obtenido mediante validación cruzada, en cada lugar.
- `observado`: el valor observado en el sitio
- `residual`: la diferencia entre el valor predicho y observado.
- `z-score`: una puntuación de z calculada como el residuo dividido por el error
- `fold`: el 'fold' o iteración cuando el lugar formaba parte del conjunto de prueba

Además, ofrece dos estadísticos: el error cuadrático medio de predicción (RMSEP) y los R^2_P de predicción:

```
## RMSEP
sqrt(mean(ppt.cv.ok$residual^2))
```

```
## [1] 4.876485
```

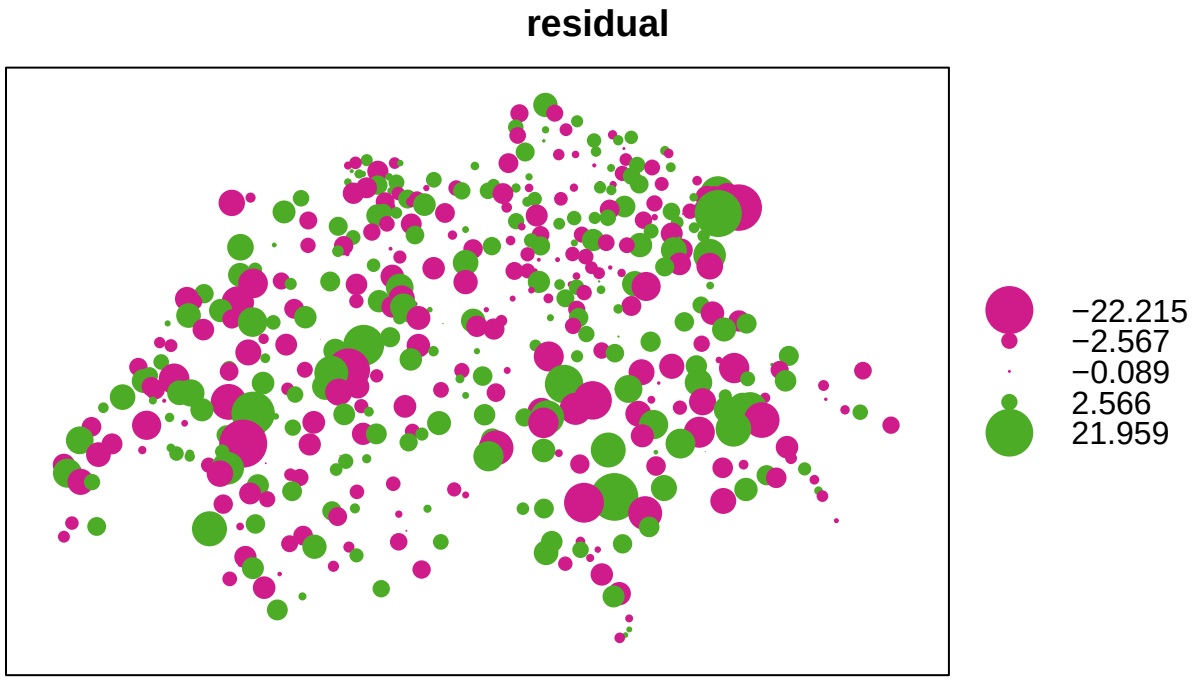
```
##R2P
cor(ppt.cv.ok$observed, ppt.cv.ok$var1.pred)^2
```

```
## [1] 0.8120919
```

El primero de ellos (RMSEP) proporciona el error promedio que podría esperarse al hacer una predicción, mientras que el segundo (R^2_P) proporciona la cantidad de varianza del conjunto de datos de prueba predicho por el modelo.

La salida de la función `krige.cv ()` contiene, para cada observación, el valor predicho cuando esa ubicación se omitió del modelo, el S.E., el valor observado y el residual (predicho - observado). Podemos usar la función `bubble ()` para trazarlos, que muestra tanto el tamaño como la dirección del residuo. Tenga en cuenta que para usar esto necesitamos convertir el objeto `sf` a la clase`sp` anterior usando la función `as_Spatial ()`:

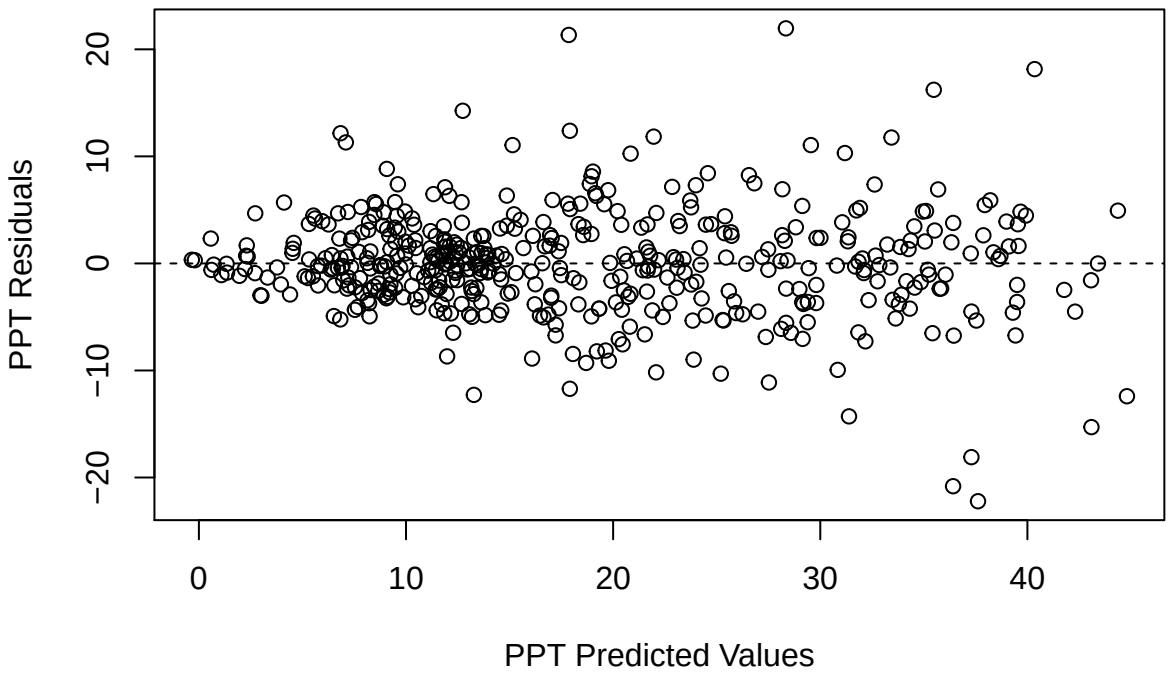
```
sp::bubble(as_Spatial(ppt.cv.ok)[,"residual"], pch = 16)
```



El mapa no muestra patrón espacial definido alguno, lo cual es un buen resultado: cualquier subestimación o sobreestimación sistemática sugeriría que existe una tendencia u otro componente estructural que no está siendo capturado por el modelo.

Finalmente, se puede elaborar un gráfico de residuos contra valores predichos para buscar cualquier sesgo en las predicciones del modelo. De nuevo, los residuos no muestran ningún sesgo, aunque sí cierta heteroscedacidad (forma de abanico = varianza no constante) en valores más altos.

```
plot(ppt.cv.ok$var1.pred, ppt.cv.ok$residual,
     xlab = 'PPT Predicted Values',
     ylab = 'PPT Residuals')

abline(h = 0, lty = 2)
```


PPT Residuals
20
10
0
−10
−20
0
10
20
30
40
PPT Predicted Values