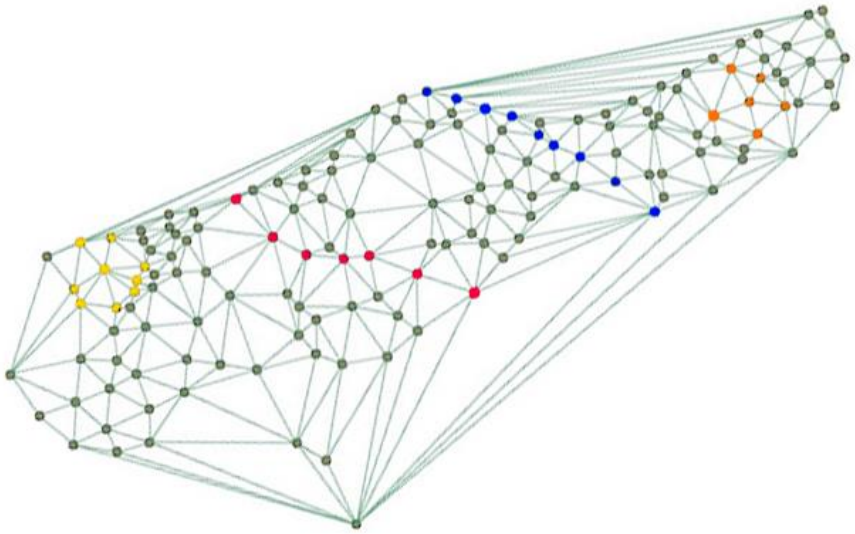


MANUAL DE USUARIO:

Análisis estadístico de datos espaciales con R y QGis

Unidad IV

Introducción al análisis de datos espaciales de tipo continuo
Geo-estadística y datos espaciales agregados o regionales

Autor: **Ligdamis A. Gutiérrez E. PhD.**

Versión 1.0

Managua, Febrero de 2020

ÍNDICE

Unidad IV: Introducción al análisis de datos espaciales de tipo continuo Geo-estadística y datos espaciales agregados o regionales

1) Generación de mapas

Uno de los aspectos que resulta muy útil en el análisis de datos es la generación de mapas. En los mapas, se puedes representar posteriormente puntos espaciales provenientes de datos y a los que se les pueden aplicar variados análisis estadísticos. Tanto en R como en QGis se pueden crear mapas de una forma relativamente simple mediante funciones predefinidas y estableciendo parámetros concretos en cada situación, veamos cómo se realiza la generación de mapas básicos.

- Mapas en R

R permite realizar un mapa mediante las funciones "**adarea**()" y "contenida en el paquete "mapq". Dicha función puede crear un mapa de cualquier parte del mundo con una alta resolución.
El paquete "mapq" se encuentra en la carpeta de datos del capítulo bajo el nombre de "mapq.zip". Para instalarlo hay que ir al menú ya sea de R o de RStudio y en la sección de instalación de paquetes (Paquete/instlar paquete desde archivos locales) en R y en RStudio desde el Menú "Tools/Instalar paquetes desde archivos locales (zip, tar)" .

Ahora hay que acceder a la librería mediante "**library**(mapq)" y cargar los datos globales de los mapas mediante "**data**(World)".

```
> library(mapq)
> data(World)
```

Seguidamente, se utiliza la función "**adarea**()", para llamar a una región geográfica, se utiliza el argumento "**Area**=" y a continuación colocar entre comillas el nombre de la zona que se desea representar. Por ejemplo tenemos el caso de Nicaragua.

```
> adarea(Area="Nicaragua")
```

Se presentará el mapa de Nicaragua en el área de gráficos de RStudio o en una ventana aparte en R.

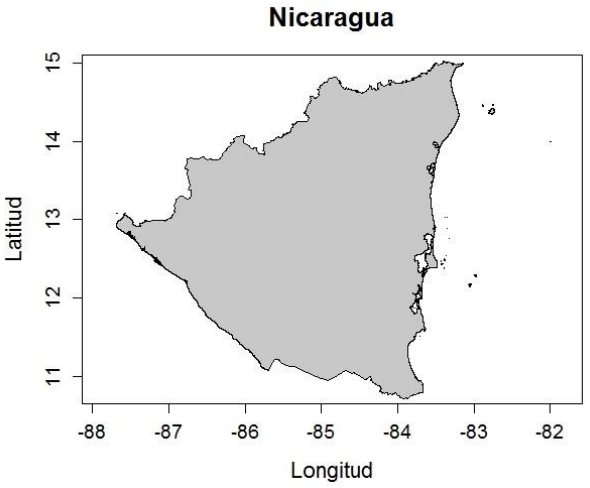


Fig.1. Mapa de Nicaragua mediante "mapq"

En caso de querer otro país por ejemplo "Guatemala", la instrucción simplemente cambia el nombre de la regios en "**Area**="

```
> adarea(Area="Guatemala")
```

El mapa generado se presentará en la zona de gráficos o en una ventana exterior en el caso de R.

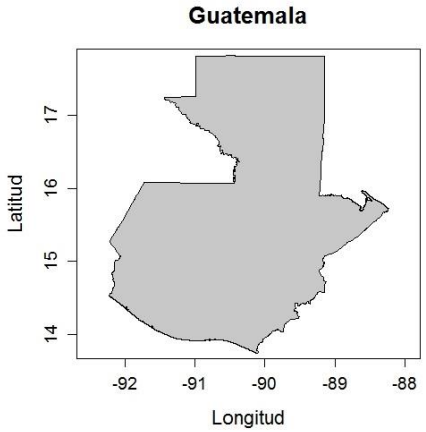


Fig.2. Mapa de Guatemala mediante "mapq"

Se pueden representar varias áreas administrativas a la vez, colocando las áreas en un vector mediante la función "**c**()" y separando cada área mediante una coma, como por ejemplo.

```
> adarea(Area = c("Nicaragua", "Honduras"))
```

Lo que dibujará las áreas de Nicaragua y Honduras en el mapa

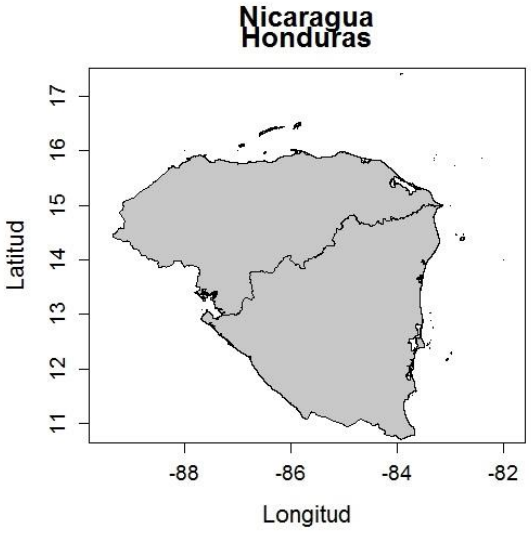


Fig.3. Mapa de combinado de Nicaragua y Honduras mediante "mapq"

Nota: Los nombres de las área administrativas están en inglés como por ejemplo España es "Spain" y no llevan asteriscos como el caso de México, es "Mexico"

Ejercicio: Cargar diferentes áreas administrativas y visualizar los mapas generados.

Para representar un mapa mundial, si se quiere ver en mayor tamaño, se debe de cambiar la visualización de la ventana a la siguiente configuración: "**windows**(18,9)". Esto abrirá una nueva ventana exterior con ese tamaño. Las instrucciones para presentar el mapa mundial son las siguientes:

```
> windows(18,9)
> adarea(Area = c("World"))
```

El mapa resultante es el que se muestra a continuación.

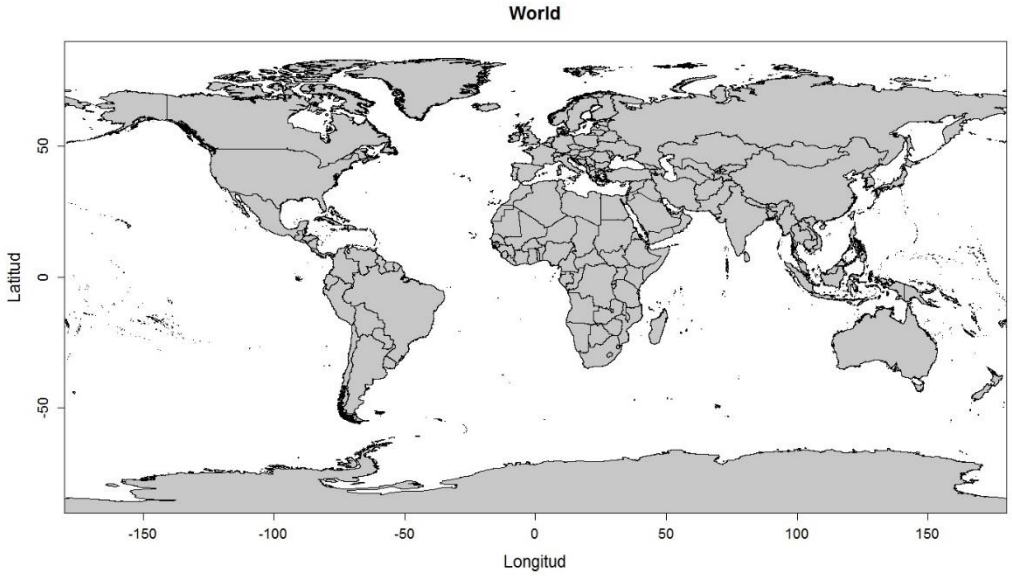


Fig.4. Mapa Mundial mediante "mapq"

TRUCO: Para guardar los mapas, hay que realizar una copia del anterior y sobrescribir el generado con la ventana grande.

- Modificar los parámetros de los mapas.

Es posible modificar el color de fondo del mapa, tanto en el fondo del mar por medio del parámetro "**colbg**", como el color de la zona terrestre con el parámetro "**colcon**". Hay que graficarlos en la ventana grande anteriormente ejemplificada, para que el vector resultante pueda caber. Se pueden utilizar cualquiera de los parámetros de color o utilizar la función "**rgb**()" que asigna tres indicadores, uno para rojo, otro para verde y un tercero para el azul. Por ejemplo, para cambiar los parámetros del color en el mapa de Nicaragua tenemos el siguiente código.

```
> windows(18,9)
> adarea(Area=c("Nicaragua"), colbg="blue", colcon="green")
```

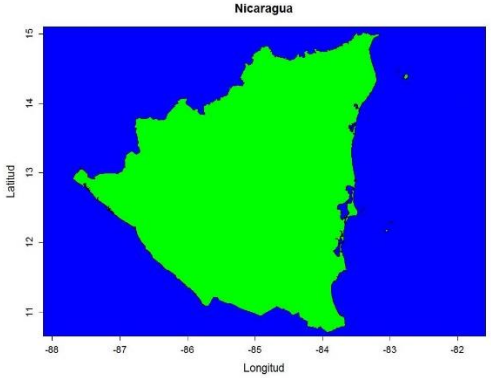


Fig.5. Cambio de los parámetros de color en el mapa de Nicaragua

Y mediante la función “rgb()”

```
> windows(18,9)
> adarea(Area=c("Nicaragua"), colbg=rgb(175,255,255, maxColorValue = 2
55), colcon = rgb(200,225,150,maxColorValue = 255))
```

La figura resultante es la siguiente:

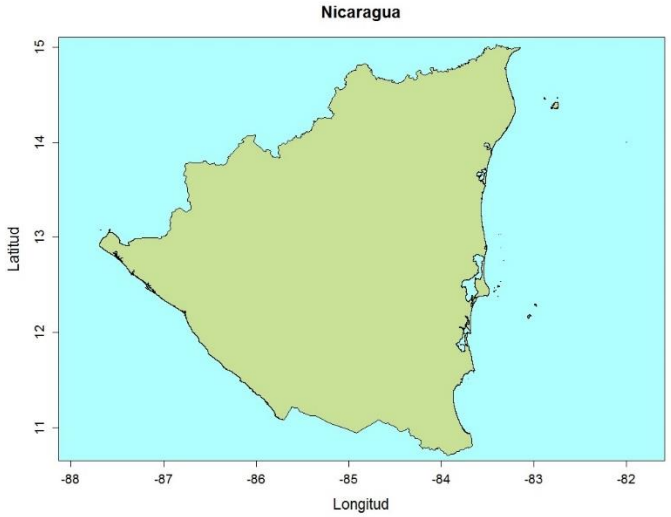


Fig.6. Cambio de los parámetros de color mediante “rgb()”, en el mapa de Nicaragua

Si observamos el mapa, se presenta tanto la latitud, como la longitud de las coordenas. Esto nos puede servir para presentar una zona más reducida de dicho mapa mediante esas coordenadas. Se tienen que indicar la mínima y máxima en cada caso, es decir, desde donde hasta donde se construirá el mapa. Por ejemplo, vamos a seleccionar una zona de Nicaragua.

```
> windows(18,9)
> adarea(Area=c("Nicaragua"), colbg=rgb(175,255,255, maxColorValue = 2
55), colcon = rgb(200,225,150,maxColorValue = 255), minLon = -86.7, ma
xLon = -86, minLat = 11.6, maxLat=12)
```

Hay que tomar en cuenta que la mínima longitud se encuentra hacia la izquierda o sea, hacia el oeste y la máxima longitud se encuentra hacia la derecha, es decir hacia el este. El mapa resultante es:

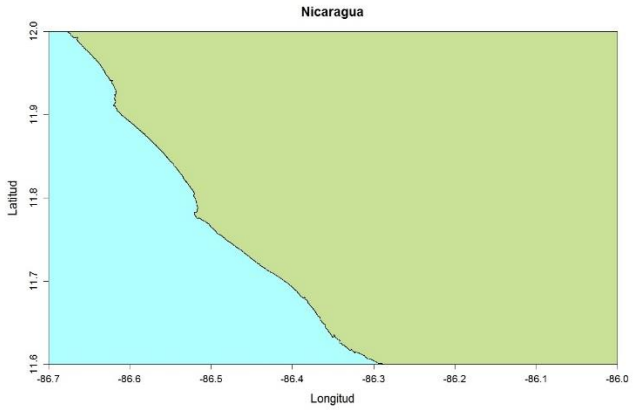


Fig.7. Selección de una zona de latitud y longitud en el mapa de Nicaragua

En caso de que se seleccione el mapa mundial mediante (**Area**="World") y se quiera mostrar un área mediante el rango de latitud y longitud, se puede mediante el argumento "**exclude**" especificar que unos determinados países se pongan de un color de fondo, que se especifica con el argumento "**colexc**", para así resaltar solamente el área que se dese, como por ejemplo si se quiere representar el área de Centro América. La instrucción sería la siguiente:

```
> windows(18,9)
> adarea(Area="World", minLon = -120, maxLon= -75, minLat=5, maxLat= 35,
+        exclude=c("Cuba","Jamaica","United States", "Mexico", "Colombia", "Bahamas"),
+        colexc="ivory1", main="Centro América", colcon="khaki", cex.main=1.4)
```

El mapa resultante es el siguiente:

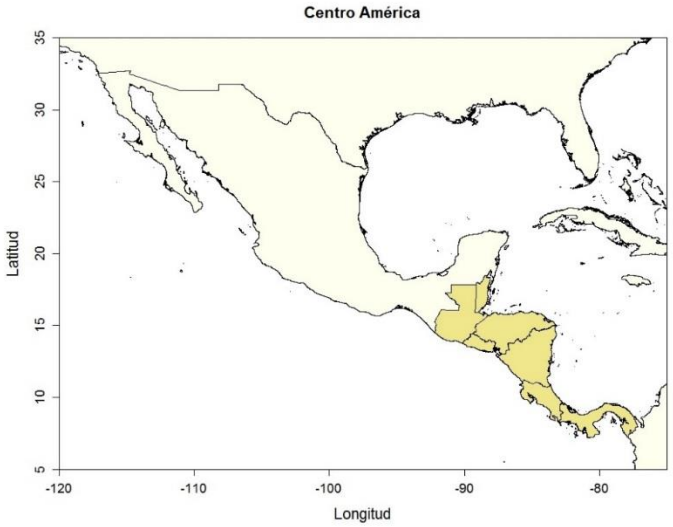


Fig.8. Selección de una zona de latitud y longitud, excluyendo países en el mapa mundial

- Establecer parámetros gráficos dentro de los mapas

Se puede añadir información adicional a un mapa, como la distribución de elementos, ciudades, zonas en donde se producen eventos, encuestas, datos poblacionales, datos de campañas de salud, etc. La función "**adarea**()" el paquete "**mapq**", que se ha indicado anteriormente, permite incluir en el mapa cualquier tipo de información a partir de un archivo, que simplemente necesita indicar las coordenadas de latitud y longitud.

Por ejemplo, en el paquete "**qmap**" viene una pequeña base de datos llamada "*Flamingo*", en donde se encuentran las coordenadas de tres especies de flamenco del género "Phoenicopterus" en Bolivia (Tabajo por Hurlbert y Keith, 1979). Las primeras instrucciones, cargan la librería, el conjunto de datos del mapa mundial y el archivo de datos.

```
> library(mapq)
> data(World)
> data(Flamingo)
> Flamingo  # Para ver el contenido del archivo de datos de R
                  Especie      Lon      Lat
1    Phoenicopterus jamesi -67.7833 -22.2833
2    Phoenicopterus jamesi -67.4312 -20.9999
3    Phoenicopterus jamesi -67.1456 -18.0648
4    Phoenicopterus jamesi -67.1044 -18.0935
5    Phoenicopterus jamesi -69.0910 -16.1741
6   Phoenicopterus andinus -67.1108 -17.9752
7 Phoenicopterus chilensis -68.3946 -18.9199
```

```
8 Phoenicopterus chilensis -66.7913 -19.2021
9 Phoenicopterus chilensis -68.1440 -18.7411
```

Como se puede observar, el contenido del archivo es muy simple, un conjunto de nueve datos estructurados por un campo para el nombre de la especie, un campo para longitud y otro para latitud. En este caso está incluido en la librería, pero puede ser cargado desde un archivo "svc" como ya hemos venido realizándolo en las pasadas unidades y puede contener, las localizaciones de escuelas, centros de salud, zonas de observaciones, etc.

Vamos a construir el mapa de Bolivia, para ello también utilizamos la función "**par**()", que permite establecer y configurar parámetros gráficos. Los gráficos se pueden establecer especificándolos como argumentos o como una lista de valores.

```
> # Creación del Mapa de Bolivia
> par(bg="grey95",lwd=1,fg="black")
+ # bg = color para el fondo de la region, lwd = ancho de linea, fg = color del primer plano
> adarea(Area=c("Bolivia"), main="Distribución de especies de flamenco
s en Bolivia", boxf="n")
```

Hay que recordar que se puede solicitar la ayuda para poder ver por ejemplo los parámetros de las funciones. Por ejemplo teclear "?par", o teclear "?adarea". Así veremos que el parámetro "**boxf**" indica el marco a construir al colocar el valor a "n", representa que no se va a construir ningún marco en el gráfico. El gráfico generado es el siguiente.

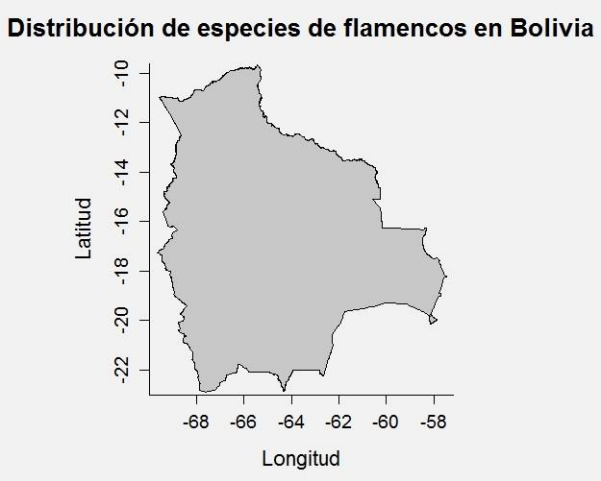


Fig.9. Construcción del mapa de Bolivia con los parámetros indicados en el argumento "par"

Ahora vamos a incluir los puntos de la base de datos en el mapa, configurando sus parámetros para darles color e incluyendo una leyenda.

```
> #Generación de puntos de la Distribución de las especies

> #Primeros puntos para la especie (Phoenicopterus jamesi) en color azul, pch=19 (punto)
> points(Flamingo[Flamingo$Especie=="Phoenicopterus jamesi", "Lon"], Flamingo[Flamingo$E
specie=="Phoenicopterus jamesi", "Lat"],pch=19,col="blue",cex=2)

> #Segundos puntos para la especie (Phoenicopterus andinus) en color verde
> points(Flamingo[Flamingo$Especie=="Phoenicopterus andinus", "Lon"], Flamingo[Flamingo$
Especie=="Phoenicopterus andinus", "Lat"],pch=19,col="green",cex=2)

> #Terceros puntos para la especie (Phoenicopterus chilensis) en color rojo
```

```
> points(Flamingo[Flamingo$Especie=="Phoenicopterus chilensis", "Lon"], Flamingo[Flaming
o$Especie=="Phoenicopterus chilensis", "Lat"],pch=19,col="red",cex=2)
> par(font=3)

> #Inclusión de la leyenda en el gráfico. Siguiendo el orden de color (1-azul, 2-rojo y 3-verde)
> legend("topright",c("P. jamesi","P. andinus","P. chilensis"),pch=c(19,19,19),col=c("bl
ue","green","red"))
```

Como Se observa en el código, la función "points()", construye los puntos en base a los datos del archivo "Flamingo" que ha sido cargado, en los primeros el valor del campo "Especie" se asigna a "Phonicopterus jamesi" que es el valor de dicho campo, con las coordenadas de latitud y longitud dela tabla. Así se presentará dicho punto en el mapa, los parámetros "pch", "col" y "cex", indican el tipo de símbolo, el color y su tamaño. Esto mismo se realiza con cada una de las especies incluidas en la base de datos o tabla.

La leyenda será incluida en la parte superior derecha, se le indica con un vector el nombre, los símbolos y el color, en el orden establecido. El gráfico resultante es el siguiente.

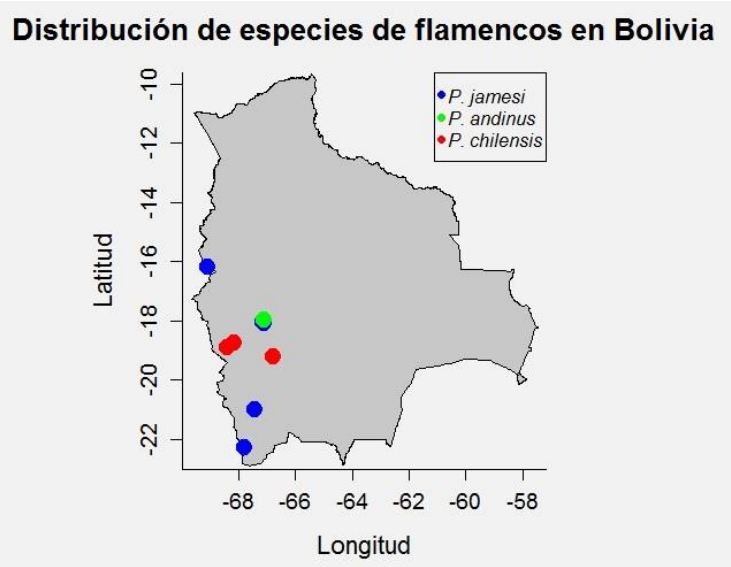


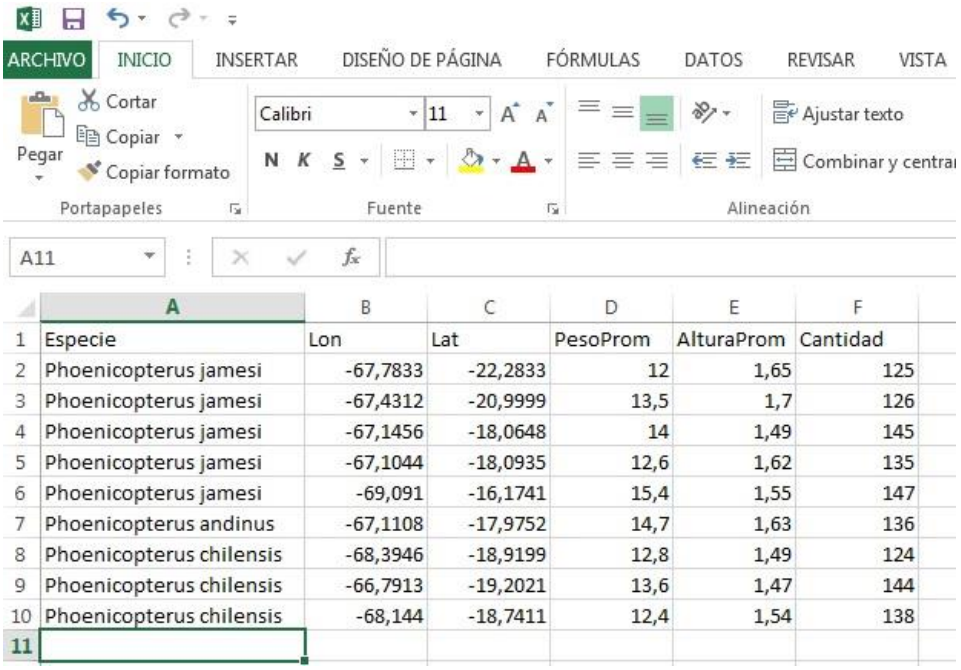
Fig.10. Inclusión en el mapa de Bolivia los puntos con los datos de la tabla "Flamingo"

De esta forma, se pueden realizar todo tipo de mapas con los puntos de las coordenadas de una tabla, los mismo impera si en dicha tabla existieran datos de observaciones, también se podrían analizar los valores estadísticos de estos datos.

- Inclusión de datos mediante una tabla

Como hemos visto en las unidades anteriores, podemos crear nuestras propias tablas de datos, mediante archivos "csv" que contengan la información. Por ejemplo vamos a crear una tabla en Excel con los datos anteriores, añadiendo las columnas de Peso promedio (PesoProm), altura promedio (AlturaProm) y Número de ejemplares en cada zona. Hay que tomar en cuenta que en Excel, a los números de latitud y longitud, hay que darles un formato general con cuatro decimales e indicar el decimal mediante la coma ",".

Posteriormente, al finalizar las entradas, guardamos el archivo en la carpeta "datos" al que llamaremos "Flamingos2", con el formato delimitado por comas "csv". La tabla será similar a la siguiente.



	A	B	C	D	E	F
1	Especie	Lon	Lat	PesoProm	AlturaProm	Cantidad
2	Phoenicopterus jamesi	-67,7833	-22,2833	12	1,65	125
3	Phoenicopterus jamesi	-67,4312	-20,9999	13,5	1,7	126
4	Phoenicopterus jamesi	-67,1456	-18,0648	14	1,49	145
5	Phoenicopterus jamesi	-67,1044	-18,0935	12,6	1,62	135
6	Phoenicopterus jamesi	-69,091	-16,1741	15,4	1,55	147
7	Phoenicopterus andinus	-67,1108	-17,9752	14,7	1,63	136
8	Phoenicopterus chilensis	-68,3946	-18,9199	12,8	1,49	124
9	Phoenicopterus chilensis	-66,7913	-19,2021	13,6	1,47	144
10	Phoenicopterus chilensis	-68,144	-18,7411	12,4	1,54	138
11						

Fig.11. Tabla de Excel “Flamingos2”

Se observa como en la tabla se han añadido los valores de Peso promedio, Altura promedio y cantidad. Estos valores pueden servirnos para construir análisis estadísticos de los datos representados en el mapa.

Limpiamos el escritorio y la zona de gráficos, para generar un nuevo script en el que guardaremos los comandos. Y construimos las líneas de código para representar el mismo mapa anterior, solo que esta vez con nuestros propios datos, desde el archivo de datos que hemos creado. Lo primero es llamar al archivo desde el sitio de la carpeta datos, mediante la función “readcsv2()”: La secuencia completa de los códigos para representar el mapa con sus puntos es la siguiente.

```
> Flamingo2 <- read.csv2(file = "./datos/Flamingos2.csv", header = TRUE, dec = ",")
> Flamingo2
                  Especie      Lon      Lat PesoProm AlturaProm Cantidad
1    Phoenicopterus jamesi -67.7833 -22.2833     12.0       1.65      125
2    Phoenicopterus jamesi -67.4312 -20.9999     13.5       1.70      126
3    Phoenicopterus jamesi -67.1456 -18.0648     14.0       1.49      145
4    Phoenicopterus jamesi -67.1044 -18.0935     12.6       1.62      135
5    Phoenicopterus jamesi -69.0910 -16.1741     15.4       1.55      147
6   Phoenicopterus andinus -67.1108 -17.9752     14.7       1.63      136
7 Phoenicopterus chilensis -68.3946 -18.9199     12.8       1.49      124
8 Phoenicopterus chilensis -66.7913 -19.2021     13.6       1.47      144
9 Phoenicopterus chilensis -68.1440 -18.7411     12.4       1.54      138
```

Se puede observar la lectura del archive mediante “read.csv2()”, se indica la ruta y el nombre, así como que se lean los encabezados mediante “header = TRUE”, y se indica que el separador de decimales es la coma (dec = “,”) . Así se presenta correctamente la tabla con el contenido de datos. Ahora se construye el mapa de Bolivia, hasta ahora todo igual.

```
> # Creación del Mapa de Bolivia
> par(bg="grey95",lwd=1,fg="black")   # bg = color para el fondo de la region, lwd = ancho de linea,
fg = color del primer plano
> adarea(Area=c("Bolivia"), main="Distribución de especies de flamencos en Bolivia", boxf="n")
```

Ahora, para construir los puntos, se llama al archivo “csv” que hemos cargado “Flamingos2” y recordemos el campo en donde se encuentra el dato por medio del símbolo “$”. Lo demás es todo igual.

```
> #Generacion de puntos mediante la tabla
> #Primeros puntos para la especie (Phoenicopterus jamesi) en color azul, pch=19 que corresponde a p
unto
> points(Flamingo2[Flamingo2$Especie=="Phoenicopterus jamesi", "Lon"], Flamingo2[Flamingo2$Especie==
"Phoenicopterus jamesi", "Lat"],pch=19,col="blue",cex=2)
> #Segundos puntos para la especie (Phoenicopterus andinus) en color verde
> points(Flamingo2[Flamingo2$Especie=="Phoenicopterus andinus", "Lon"], Flamingo2[Flamingo$Especie==
"Phoenicopterus andinus", "Lat"],pch=19,col="green",cex=2)
> #Terceros puntos para la especie (Phoenicopterus chilensis) en color rojo
> points(Flamingo2[Flamingo2$Especie=="Phoenicopterus chilensis", "Lon"], Flamingo2[Flamingo$Especie
=="Phoenicopterus chilensis", "Lat"],pch=19,col="red",cex=2)
> par(font=3)
> #Inclusión de la leyenda en el gráfico. Siguiendo el orden de color (1-azul, 2-rojo y 3-verde)
> legend("topright",c("P. jamesi","P. andinus","P. chilensis"),pch=c(19,19,19),col=c("blue","green",
"red"))
```

El mapa generado debe ser exactamente igual al anterior, solo que este es creado a partir de nuestro archive de datos.

- Cálculo de probabilidades con los datos

Se pueden realizar en R cálculos de probabilidades de forma sencilla. Vamos a realizar los cálculos de las probabilidades: Binomial, Distribución de Poisson, Distribución Geométrica y Distribución Hipergeométrica con los valores de la tabla. Las funciones generadas son las siguientes:

a) Distribución Binomial: función “pbinom(x,n,p), para obtener el valor de distribución en x de la binomial (n,p)
b) Distribución Binomial de masa: función “dbinom(x,n,p)”, para obtener el valor de la función de masa en x de la binomial (n,p)
c) Distribución de Poisson: unción “ppois(x,a)”, para obtener el valor de la función de distribución en x de la Poisson (a)
d) Distribución de Poisson de masa: función “dpois(x,a)”, para obtener el valor de la función de masa en x de la Poisson (a)
e) Distribución Geométrica: función “pgeom(x,p)”, para obtener el valor de distribución en x de la geométrica (p)
f) Distribución Geométrica de masa: función “dgeom(x,p)”, para obtener el valor de masa en x de la geométrica (p)
g) Distribución Hipergeométrica: función “phyper(x,D,N-D,n), para obtener el valor en x de la Hipergeométrica (D,N,n)
h) Distribución Hipergeométrica de masa: función “rhyper(x,D,N-D,n)”, para obtener el valor de masa en x de la Hipergeométrica (D,N,n).

En la función Hipergeométrica:
x: son los datos o elementos a que pertenece la categoría
D: es el número de elementos en la población original que pertenecen a la categoría deseada
(N-D): La muestra
N: es el tamaño de población
n: es el tamaño de la muestra extraída

Las expresiones de cálculo de ejemplo con los valores de la tabla son las siguientes

```
> # Cálculo de la distribución Binomial

> DistBino <-pbinom(c(Flamingo2$AlturaProm,Flamingo2$PesoProm), 2, 0.5)  # Distribución
en x de la binomial (n,p)
> DistBino
 [1] 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 0.75 1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.0
0 1.00

> DistBinoMasa <- dbinom(c(Flamingo2$Especie=="Phoenicopterus jamesi", lon ==-67.7833, l
at ==-22.2833),2,0.5)
  # Cálculo de masa en x de la binomial (n,p)
> DistBinoMasa
 [1] 0.50 0.50 0.50 0.50 0.50 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.2
5 0.25 0.25 0.25 0.25 0.25 0.25
[24] 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25

> # Cálculo de la distribución de Poisson

> DistPoison <-ppois(c(Flamingo2$AlturaProm,Flamingo2$PesoProm), 2)  # Distribución en x
de la Poisson (a)
> DistPoison
 [1] 0.4060058 0.4060058 0.4060058 0.4060058 0.4060058 0.4060058 0.4060058 0.4060058 0.4
060058 0.9999998 1.0000000
[12] 1.0000000 0.9999998 1.0000000 1.0000000 0.9999998 1.0000000 0.9999998

> DistPoisonMasa <-dpois(c(Flamingo2$Especie=="Phoenicopterus jamesi", lon ==-67.7833, l
at ==-22.2833),2)  # Cálculo de masa en x de la Poisson (a)
> DistPoisonMasa
 [1] 0.2706706 0.2706706 0.2706706 0.2706706 0.2706706 0.1353353 0.1353353 0.1353353 0.1
353353 0.1353353 0.1353353
[12] 0.1353353 0.1353353 0.1353353 0.1353353 0.1353353 0.1353353 0.1353353 0.1353353 0.1
353353 0.1353353 0.1353353
[23] 0.1353353 0.1353353 0.1353353 0.1353353 0.1353353 0.1353353 0.1353353 0.1353353 0.1
353353 0.1353353 0.1353353

> # Cálculo de la distribución Geométrica

> DistGeom <-pgeom(c(Flamingo2$AlturaProm,Flamingo2$PesoProm), 0.5)  # Distribución en x
de la geométrica (p)
> DistGeom
 [1] 0.7500000 0.7500000 0.7500000 0.7500000 0.7500000 0.7500000 0.7500000 0.7500000 0.7
500000 0.9998779 0.9999390 0.9999695 0.9998779
[14] 0.9999847 0.9999695 0.9998779 0.9999390 0.9998779

> DistMasa <-dgeom(c(Flamingo2$Especie=="Phoenicopterus jamesi", lon ==-67.7833, lat ==-
22.2833),0.5)  # Cálculo de masa en x de la geométrica (p)
> DistMasa
 [1] 0.25 0.25 0.25 0.25 0.25 0.50 0.50 0.50 0.50 0.50 0.50 0.50 0.50 0.50 0.50 0.50 0.5
0 0.50 0.50 0.50 0.50 0.50 0.50 0.50 0.50 0.50 0.50 0.50
[28] 0.50 0.50 0.50 0.50 0.50 0.50

> # Cálculo de la distribución Hipergeométrica

> DistHiperGeom <-phyper(c(Flamingo2$Especie=="Phoenicopterus jamesi", lon ==-67.7833, l
at ==-22.2833), 125, 1220, 80, 15)  # Distribución en x de la Hipergeométrica (D,N,n)
> #(x,D,N-D,n), Donde
> # x son los datos o elementos a que pertenece la categoría
> # D es el número de elementos en la población original que pertenecen a la categoría deseada
> # (N-D) La muestra
```

```
> # N es el tamaño de población
> # n es el tamaño de la muestra extraída
> DistHiperGeom
 [1] 0.0031013801 0.0031013801 0.0031013801 0.0031013801 0.0031013801 0.0003176263 0.000
3176263 0.0003176263 0.0003176263 0.0003176263
[11] 0.0003176263 0.0003176263 0.0003176263 0.0003176263 0.0003176263 0.0003176263 0.000
3176263 0.0003176263 0.0003176263 0.0003176263
[21] 0.0003176263 0.0003176263 0.0003176263 0.0003176263 0.0003176263 0.0003176263 0.000
3176263 0.0003176263 0.0003176263 0.0003176263
[31] 0.0003176263 0.0003176263 0.0003176263

> DistHiperGeomMasa <-dhyper(c(Flamingo2$Especie=="Phoenicopterus jamesi", lon ==-67.783
3, lat ==-22.2833), 125, 1220, 80, 15)  # Distribución en x de la Hipergeométrica (D,N,n)
> DistHiperGeomMasa
 [1] -5.883955 -5.883955 -5.883955 -5.883955 -5.883955 -8.054635 -8.054635 -8.054635 -8.
054635 -8.054635 -8.054635 -8.054635 -8.054635
[14] -8.054635 -8.054635 -8.054635 -8.054635 -8.054635 -8.054635 -8.054635 -8.054635 -8.
054635 -8.054635 -8.054635 -8.054635 -8.054635
[27] -8.054635 -8.054635 -8.054635 -8.054635 -8.054635 -8.054635 -8.054635
```

Valores estadísticos que se pueden calcular entre otros muchos con R son:

1) Distribución Uniforme: mediante las funciones “punif(x,a,b)”, que calcula la distribución de x en la uniforme (a,b)
2) Distribución uniforme: función “dunif(x,a,b)”, que calcula la densidad en x de la uniforme (a,b).
3) Distribución Beta: función “pbeta(x,a,b)”, calcula la función de distribución en x de la beta (a,b)
4) Distribución Beta: función “dbeta(x,a,b)”, que calcula la función de densidad en x de la beta (a,b)
5) Distribución Gamma: función “pgama(x,a,b)”, que calcula la función de distribución en x de la gama (a,b).
6) Distribución Gamma: función “dgamma(x,a,b)”, que calcula la función de densidad en x de la gamma (a,b)
7) Distribución de Cauchy: función “pcauchy(x,a,b)”, calcula la función de distribución en x de la Cauchy (a,b)
8) Distribución de Cauchy: función “dcauchy(x,a,b)”, calcula la función de densidad en x de la Cauchy (a,b)
9) Distribución Exponencial: función “pexp(x,a)”, calcula la función de distribución en x de la Exponencial (a)
10) Distribución Exponencial: función “dexp(x,a)”, calcula la función de densidad en x de la Exponencial (a)
11) Distribución “t” de Student: función “pt(x,n)”, calcula la función de distribución en x de la t-Student con n grados
12) Distribución “t” de Student: función “dt(x,n)”, calcula la función de densidad en x de la t-Student con n grados.

En todas colocando de primera letra, los valores “q” y “r” se obtienen el cuantil de orden p de los valores (a,b) y el valor “r” al inicio, calcula una muestra aleatoria de tamaño “n” de los valores (a,b). Por ejemplo, tomemos una muestra aleatoria de tamaño 1000, con un valor de n de 50 y un valor p de 0.5. Recordemos que el valor de n tiene que ser mayor o igual a 1 y el valor p debe de estar en el intervalo entre 0 y 1, sin incluirlos., construiremos un vector de accisas y calcularemos la densidad normal sobre el histograma anterior. Si se calcula y representa varias veces, cada vez se obtendrán muestras distintas, ya que se estarán seleccionando 1000 muestras al azar de una binominal de (50,0.5), por lo que los gráficos resultantes serán diferentes. >Los valores a calcular serán los siguientes:

```
> # Ejemplo de distribución binomial con aproximación normal

> Nmuestra <- rbinom(1000, 50,0.5)    #El tamaño m de la muestra 1000,  n = 50 y p = 0.5
> hist(Nmuestra, prob="T", col=5, main ="Histograma de la Muestra")
> y<-seq(0,50,len=100)      #construcción del vector de abscisas
> lines(y,dnorm(y,25,3.535),col=2) #línea de representación de las abscisas de
la densidad normal sobre el histograma.
```

El gráfico resultante será el siguiente:

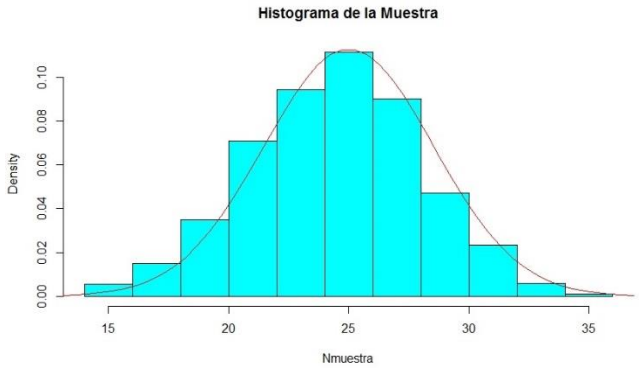


Fig.12. Histograma de datos binomiales con aproximación normal.

Y si utilizamos la muestra de datos gamma (1,1) con una aproximación normal tendríamos.

```
> # Creación de un histograma de datos gamma (1,1) con aproximación normal

> y1=rgamma(5000,1,1)      # Se crean una muestra de 5000 datos procedentes de una gamma
(1,1)
> y2<-matrix(y1,ncol=4)  # Se crea una matriz de 5000/4 = 1250 muestras de una gamma (1,
1)
> y3<-rowMeans(y2)       # Se crea un vector de 1250 medias muestrales de tamaño n=4 de
una gamma (1,1)
> # Creación del Gráfico
> hist(y3,prob=T,col=5)    # crea el histograma del vector creado de medias muestrales
> y<-seq(0,2.5,len=100)    # se crea el vector de abscisas
> lines(y,dnorm(y,1,0.5),col =2)  #Se construye la línea del vector de abscisas con la a
proximación normal
```

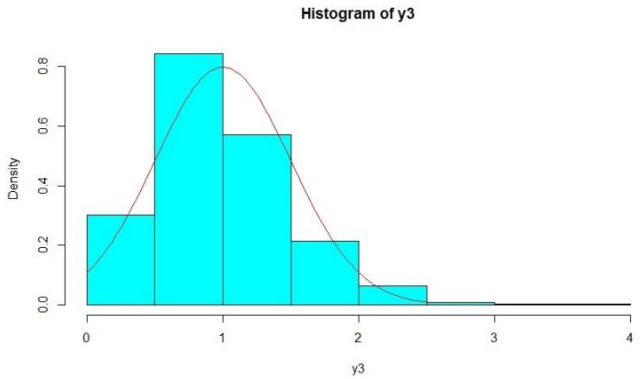


Fig.13. Histograma de datos Gamma (1,1) con aproximación normal.

Si modificamos los valores en los parámetros anteriores, haciendo el tamaño de las muestras igual a 10, n= 10, y modificando los parámetros de la normal a aproximar, se podría obtener una buena aproximación. El código sería el siguiente:

```
> # Modificar los parámetros de la normal a aproximar

> y1=rgamma(5000,1,1)      # Se crean una muestra de 5000 datos procedentes de una gamma (1,1)
> y2<-matrix(y1,ncol=10)  # Se crea una matriz de 5000/10 = 500 muestras de una gamma (1,1)
> y3<-rowMeans(y2)        # Se crea un vector de 500 medias muestrales de tamaño n=4 de una gamma
(1,1)
> # Creación del Gráfico
> hist(y3,prob=T,col=5, main ="Histograma de la Muestra")   # crea el histograma del vec
tor creado de medias muestrales
> y<-seq(0,2.5,len=100)   # se crea el vector de abscisas
> lines(y,dnorm(y,1,0.316),col =2)  #Se construye la línea del vector de abscisas con la
aproximación normal
```

El gráfico resultante sería el siguiente.

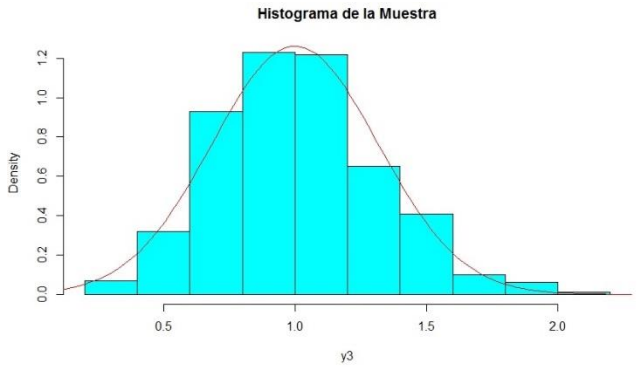


Fig.14. Histograma mejorado de datos Gamma (1,1) con aproximación normal.

Como se puede razonar, todos estos cálculos se pueden aplicar a los valores de los datos ingresados en los archivos que representan los puntos espaciales a nalizar.

- Mapas con polígonos

También es posible y sumamente sencillo representar datos mediante polígonos en lugar de puntos.

Para ello se utiliza la función ya vista "**adarea()**" y la función "**polygon()**". Es muy importante a tomar en cuenta que el archivo de datos que vamos a utilizar, para el caso del ejemplo "Gorila", la primera y la última coordenada de cada especie deben de ser iguales (lo mismo si se utiliza un archivo de datos externo creado por nosotros), esto es para que los polígonos puedan ser cerrados. Adicionalmente, se debe de dejar una línea en blanco entre los conjuntos de coordenadas en cada especie o elemento a representar.

Ejemplo 2:

```
> library(mapq)   # Carga de la librería mapq
> data(World)    # Carga de los datos para los mapas de los países
> data(Gorilla)  # Carga de los datos del archivo Gorilla

> Gorilla  # Visualizar la tabla de los datos de Gorila
        Lon       Lat          Species
1  28.335128 -2.670354 Gorilla beringei
2  28.762184 -1.583872 Gorilla beringei
3  27.761002 -0.933704 Gorilla beringei
4  26.452522 -0.691816 Gorilla beringei
```

```
5  26.588612 -1.494967 Gorilla beringei
6  26.884179 -2.191474 Gorilla beringei
7  27.860464 -2.683385 Gorilla beringei
8  28.457960 -3.779664 Gorilla beringei
9  28.780923 -3.979196 Gorilla beringei
10 28.943000 -3.464235 Gorilla beringei
11 28.669220 -2.548450 Gorilla beringei
12 28.335128 -2.670354 Gorilla beringei
13        NA        NA
14  9.904367  3.218027  Gorilla gorilla
15 10.155314  3.186957  Gorilla gorilla
16  9.700000  1.100000  Gorilla gorilla
17 10.000000  0.300000  Gorilla gorilla
18  9.026660 -0.979119  Gorilla gorilla
19  9.801311 -2.265638  Gorilla gorilla
20 11.666649 -4.351979  Gorilla gorilla
21 12.628917 -5.728361  Gorilla gorilla
22 13.061675 -5.600404  Gorilla gorilla
23 11.861444 -3.466939  Gorilla gorilla
24 13.778651 -3.124081  Gorilla gorilla
25 14.152868 -1.589775  Gorilla gorilla
26 16.096338 -2.546000  Gorilla gorilla
27 16.517247  3.087374  Gorilla gorilla
28 18.595060  3.473410  Gorilla gorilla
29 15.151944  3.462198  Gorilla gorilla
30 13.544109  4.914733  Gorilla gorilla
31  9.904367  3.218027  Gorilla gorilla
```

En la tabla se observa la línea en blanco “datos (NA)”, como separación para las especies o los elementos en cada polígono.

También se observa como las coordenadas de longitud y latitud inicial y final deben de ser las mismas para que el polígono se cierre.

La construcción del gráfico implica, cambiar el tamaño de la ventana. Las instrucciones son las siguientes.

```
> # Cambio de la ventana
> windows(12,12)
> # Construcción del Gráfico
> adarea(Area="World", minLon=2, maxLon=31, minLat=-8, maxLat=5, colcon="khaki", colbg =
rgb(175, 225, 255, maxColorValue= 255),main="Distribución de especies de gorilas")
> # Construcción de los polígonos
> polygon(Gorilla$Lon,Gorilla$Lat,col=c("red","green"))   #construcción de los polígonos
> par(font=3)  # Parámetros gráficos, tipo de letra 3
> # Leyenda
> legend(24,4,c("G. beringei","G. gorilla"),pch=15,col=c("red","green"),bty="n")
```

El gráfico resultante se observa en la siguiente imagen:

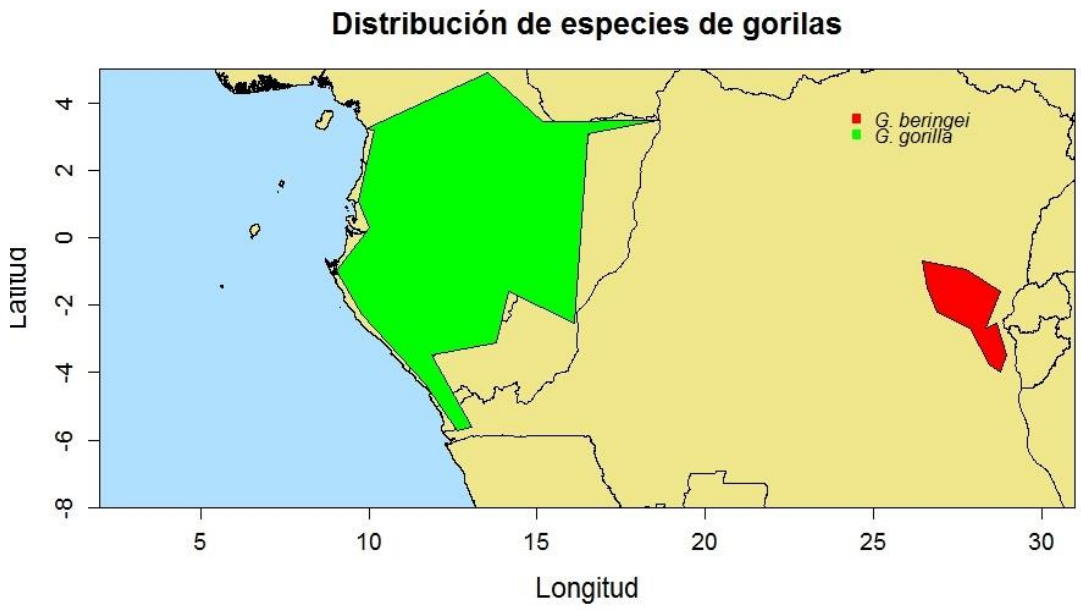


Fig.15. Gráfico de la distribución de especies de gorila mediante polígonos.

Mediante esta técnica se pueden representar los datos a través de polígonos en un mapa.

- Mapas de Vectores.

En los mapas también es posible añadir información como vectores de dirección e intensidad, que pueden resultar útiles como por ejemplo para representar vientos o líneas de trayectorias de movilizaciones, etc. Estas líneas se construyen mediante la función "**vectorField()**" del paquete "**plotrix**", para lo cual hay que llamar a la librería "**library(plotix)**". Vamos atrabajar con el archivo de datos "Viento.csv" que se encuentra en la carpeta del ejercicio 3 de esta unidad. Dicho archivo hay que copiarlo al directorio donde se están cargando los datos o indicar la ruta donde seencuentra al cargarlo.

Ejemplo 3: Creamos un nuevo script en RStudio para ingresar los datos

```
> library(mapq)     # Cargar la librería mapq
> library(plotrix) # Cargar la librería plotrix
> data(World)       # Cargar el conjunto de datos mundial para mapas
> windows(12,7)     # Modificar el tamaño de la ventana
```

Se presenta una nueva ventana con el tamaño indicado, ahora procederemos a crear el mapa

```
> # Creación del mapa geográfico de Panamá
> adarea(Area=c("Panama"), colbg=rgb(175,225,255, maxColorValue=255), colcon="
khaki")
```

Se presentará el mapa de Panamá. Ahora vamos a cargar el archivo de datos (csv) en donde se encuentran las coordenadas de los vectores a representar.

```
> # Lectura de los datos de viento a través del archivo (csv)
>
> datos2<-read.csv2(file = "./datos/Viento.csv", header=TRUE, encoding= "latin1")
```

Ahora para ver el contenido de dicho archivo, lo llamamos mediante su nombre.

```
> datos2    # visualizar contenido del archivo
   Longitud Latitud DirViento IntViento  X X.1 X9.3
1    -79.80    7.60        90       2.0 NA  NA   NA
2    -79.90    7.70        89       2.0 NA  NA   NA
3    -79.70    7.80        92       2.0 NA  NA   NA
4    -79.80    7.90        90       2.0 NA  NA   NA
5    -79.90    8.00        82       1.9 NA  NA   NA
6    -79.80    8.10        74       1.8 NA  NA   NA
7    -79.90    8.30        66       1.7 NA  NA   NA
8    -79.70    8.40        58       1.6 NA  NA   NA
9    -79.80    8.50        50       1.5 NA  NA   NA
10   -79.60    8.57        42       1.5 NA  NA   NA
11   -79.70    8.45        34       1.5 NA  NA   NA
12   -79.50    8.50        26       1.5 NA  NA   NA
13   -79.40    8.60        10       1.6 NA  NA   NA
14   -79.40    8.70        23       1.6 NA  NA   NA
15   -79.30    8.70        25       1.6 NA  NA   NA
16   -79.50    8.70        14       1.5 NA  NA   NA
17   -79.50    8.80        26       1.4 NA  NA   NA
18   -79.40    8.80        21       1.4 NA  NA   NA
19   -79.60    8.80        18       1.4 NA  NA   NA
20   -79.50    8.90        12       1.4 NA  NA   NA
21   -79.50    8.70         6       1.5 NA  NA   NA
22   -79.10    8.70       -18       1.5 NA  NA   NA
23   -79.00    8.80       -24       1.6 NA  NA   NA
24   -78.90    8.70       -30       1.3 NA  NA   NA
25   -78.90    8.80       -30       1.5 NA  NA   NA
26   -78.80    8.70       -35       1.5 NA  NA   NA
27   -78.85    8.60       -40       1.5 NA  NA   NA
28   -78.55    8.50       -55       1.6 NA  NA   NA
29   -78.65    8.40       -50       1.3 NA  NA   NA
30   -78.65    8.30       -55       1.7 NA  NA   NA
31   -79.30    8.80        15       2.0 NA  NA   NA
32   -79.30    8.40       -50       2.0 NA  NA   NA
33   -79.20    8.20       -55       2.0 NA  NA   NA
34   -79.00    8.10       -55       2.0 NA  NA   NA
35   -77.80    9.30         0       2.0 NA  NA   NA
```

El contenido muestra las coordenadas de longitud y latitud, la dirección e intensidad del viento, así como tres columnas de valores que no contienen datos.

Mediante la función “par(), establecemos los parámetros gráficos de dichos vectores

```
> par(fg="white",lwd=2)    # parametros gráficos tipo de linea de color blaca
y tamaño 2
```

Ahora utilizamos la función “vectorField()”, para la construcción de las líneas mediante los datos de la tabla.

```
 > # Uso de la función (vectorField) para construcción de las líneas
 > vectorField(datos2$DirViento, datos2$IntViento, datos2$Longitud,
 + datos2$ Latitud, scale=0.2, vecspec="deg", headspan=0.05)
```

Se presentan las líneas en blanco en el mapa. Ahora colocamos el texto de las líneas en ellas, mediante la función “text()”, estableciendo como parámetros las coordenadas y un subtítulo.

```
> # Construcción del texto en las líneas
> text(-77.8,9.45,substitute("20 km h"^-1),cex=1.2)
```

El gráfico resultante es el siguiente:

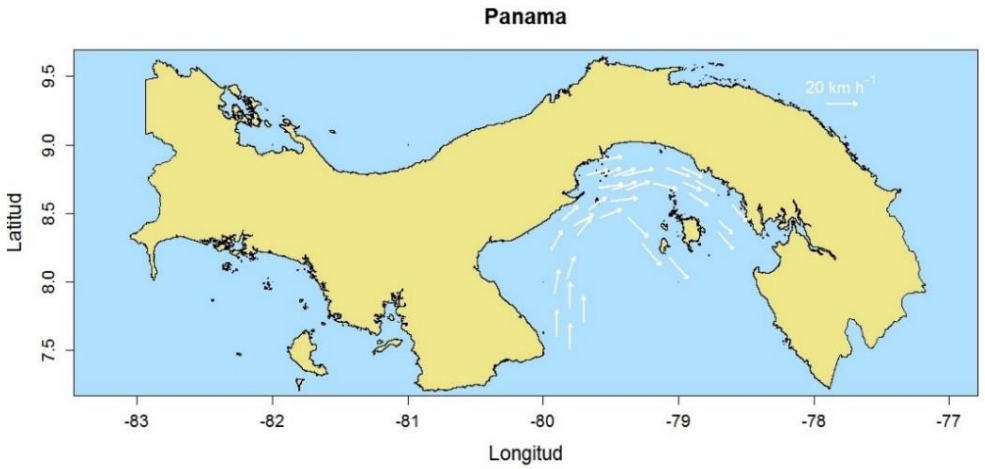


Fig.16. Mapa de Panamá con vectores que representan líneas de viento.

Hay que recordar que para poder ver los parámetros de las funciones solo basta con dar un vistazo a la ayuda en la documentación de cada función. Por ejemplo para la función "vectorField", hay que teclear "?vectorField", con lo que aparecerá en el panel de ayuda la documentación.

Ejercicio: Crear dos mapas y configurar sus parámetros, crear una tabla en Excel con datos y convertirla a un archivo de datos "csv", incorporar puntos sobre uno de los mapas y polígonos en el otro. Realizar algunos análisis estadísticos sobre los datos de los puntos y polígonos añadidos.

- Mapas en QGis

Cerramos RStudio (recordar salvar la sesión) y abrimos de nuevo QGis, para poder presentar una forma de incluir mapas en capas.

Existen muchas maneras de incluir mapas vectoriales y raster en QGis, algunas formas son gratuitas, otras incluyen tener que realizar un registro o acceder mediante cuotas de pago. Aquí veremos las más simples que nos ayudarán a presentar nuestros datos en mapas que se encuentran de forma de libre acceso. La manera más simple es realizarlo mediante el uso del plugin "**QuickMapServices**", a través de la herramienta "**XYZ Tiles**".

Para instalar el plugin, hay que ir al Menú "Complementos/Administrar e instalar complementos" y teclear "Quick" con lo que saldrá el complemento e instalarlo.

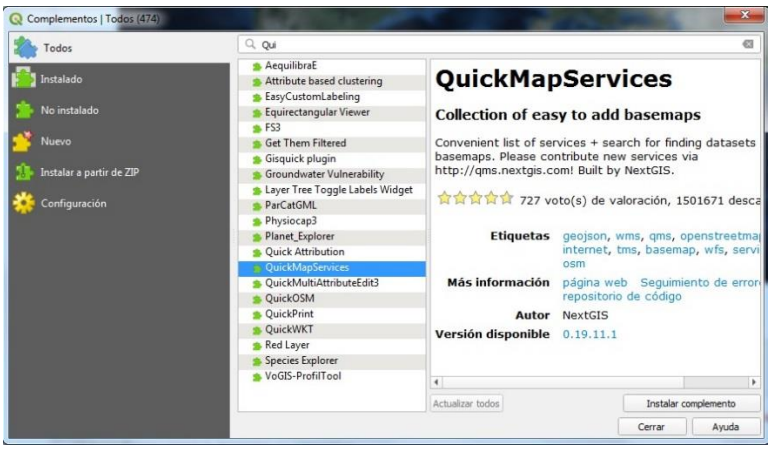


Fig.17. Instalación del complemento "QuickMapServices" en QGis

Una vez instalado ahora habrá que ir al menú “Capa/Administrados de fuente de datos” y ahí en el navegador “Browser”, se encuentra en la lista la herramienta “XYZ Tiles”. Hacemos clic con el botón derecho del ratón y presentamos “Conexión Nueva”, para que se abra la ventana de diálogo que establecerá nuevas conexiones. Ahí se indicará el nombre de la conexión, la “URL” o dirección de internet en donde se encuentra la capa a importar. El otro parámetro que hay que asignar es establecer el nivel de zoom máximo en 19. La ventana de ejemplo es la siguiente.

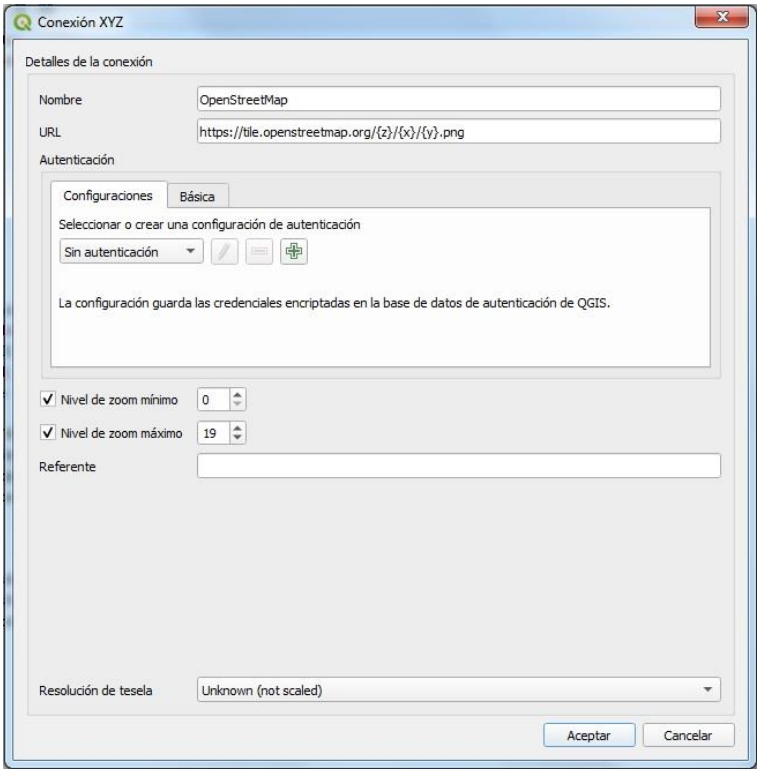


Fig.18. Ventana de Conexión XYZ

En el ejemplo, se va a establecer la conexión a OpenStripMap, mediante la dirección URL

https://tile.openstreetmap.org/{z}/{x}/{y}.png

Una vez establecidos los parámetros se da clic a “Aceptar”, para que esta capa se incorpore a QGis.

Algunas de las direcciones que se pueden incorporar de Google se encuentran en el Web de Google siguiente.

https://wiki.openstreetmap.org/wiki/Tiles

Otra fuente de las direcciones para las conexiones “xyz” se encuentran disponibles en el siguiente enlace Web. Es un script de Klas Karlsson y que además se puede ejecutar desde la consola de Python. Copiando el enlace y dando “Enter”

https://raw.githubusercontent.com/klakar/QGIS_resources/master/collections/Geosupportsystem/python/qgis_basemaps.py

Ahora, para ingresar una capa XYZ ya creada, vemos por ejemplo en la parte de menú "Capa/Administrados de fuente de datos" y ahí en el navegador "Browser", la capas XYZ que tenemos instaladas

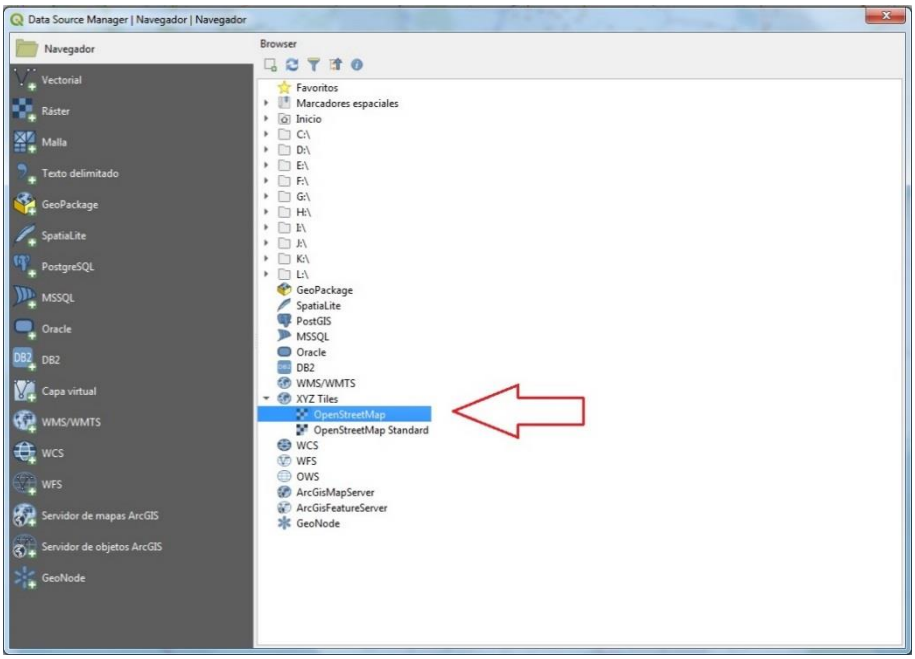


Fig.19. Ventana de Browser con la conexión XYZ Tiles.

En esta ventana damos doble clic, por ejemplo a "OpenStreetMap", y se presentará en mapa en la ventana de gráficos y en la zona de capas se indicará la capa. Podemos hacer un acercamiento ya sea con el ratón moviendo la rueda o con el zoom, para seleccionar un área determinada. Aquí tenemos un ejemplo.

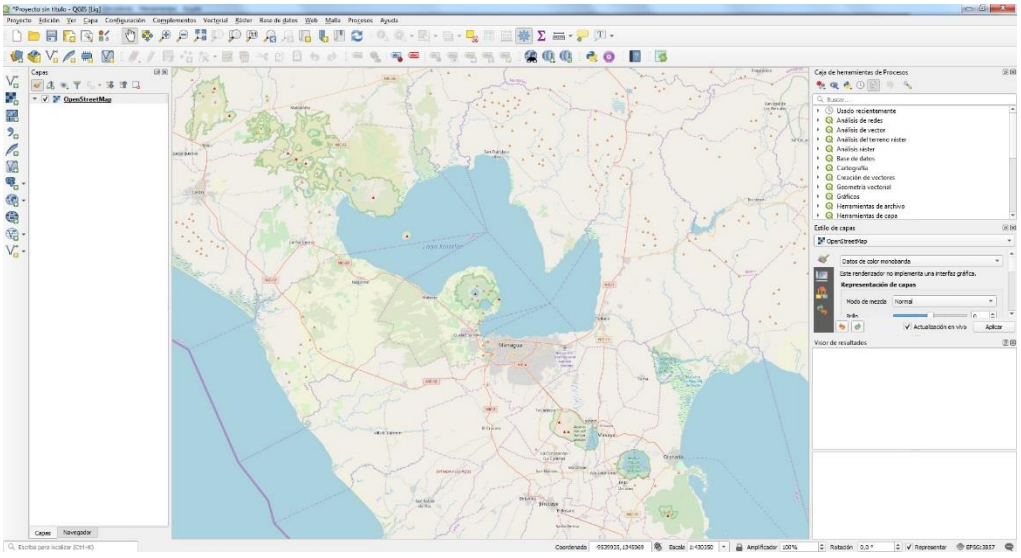


Fig.20. Ventana de OpenStreetMap con acercamiento a Nicaragua.

En el mapa nos hemos acercado a la zona del pacífico de Nicaragua. Este mapa ya se puede exportar como imagen o guardarlo como plantilla para poder utilizarlo posteriormente.

Otro aspecto interesante es poder incorporar una capa raster que brinda el servicio ESRI (*Environmental Systems Research Institute*), quienes en su página Web disponen de una gran

variedad de recursos y mapas. Sin embargo, se requiere una suscripción a dichos servicios. El enlace a la página es el siguiente:

https://www.esri.com/es-es/arcgis/products/arcgis-online/overview

A pesar de ello, se dispone de un enlace que a través de la consola de Python en QGis permite colocar una capa raster satelital sobre la imagen de la capa por ejemplo de OpenStreetMap que hemos añadido. Basta con abrir la consola de Python, en el menú "Complementos/Consola de Python", y en la parte inferior escribir el siguiente código y dar "Enter".

```
qgis.utils.iface.addRasterLayer('http://server.arcgisonline.com/arcgis/rest/services/ESRI_Imagery_World_2D/MapServer?f=json&pretty=true','raster')
```

Se presentará una imagen de la capa raster satelital del mapa como a continuación se observa

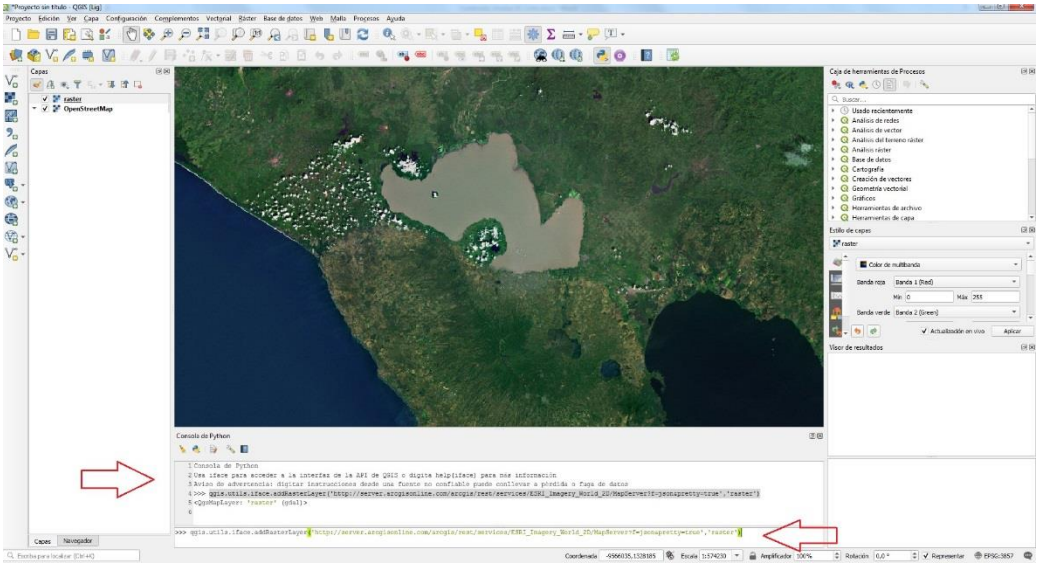


Fig.21. Ventana de capa raster satelital de Nicaragua.

Un ejemplo de una paca de la zona de Granada en España y de su capa raster es la siguiente:

Fig.22. Ventana de OpenStreetMap de Granada en España.

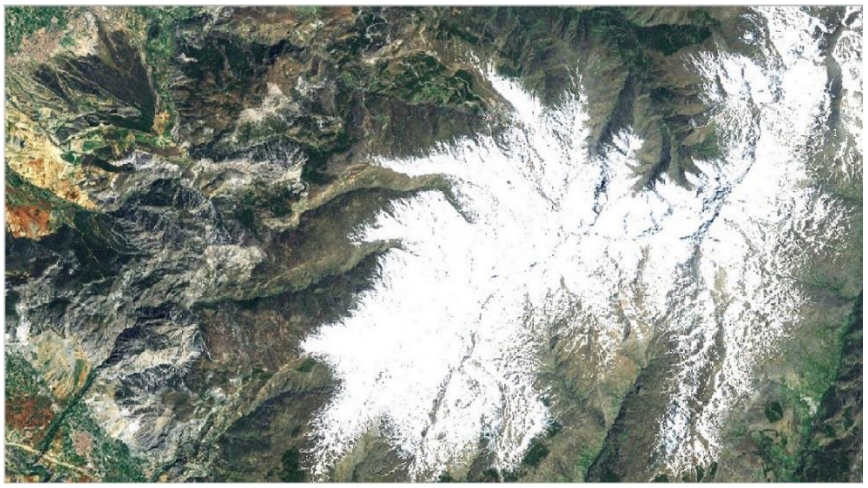


Fig.22. Ventana de capa raster de Granada en España.

Podemos guardar estas capas, incorporar datos o puntos en ellas mediante archivos o tablas de atributos creadas mediante la creación de puntos espaciales en capas vectoriales y realizar análisis con sus contenidos.

A la lista de direcciones URL, XYZ Tiles que hemos incluido en los enlaces, podemos destacar también entre muchas las siguientes, que se encuentran disponibles en Internet.

1) **OpenStreetMap Mapnick:** http://tile.openstreetmap.org/{z}/{x}/{y}.png
2) **OSM Cycle Map:** http://tile.thunderforest.com/cycle/{z}/{x}/{y}.png
3) **OSM Black and White:** http://tiles.wmflabs.org/bw-mapnik/{z}/{x}/{y}.png
4) **OSM2World/3D (D,A,CH):** http://tiles.osm2world.org/osm/pngtiles/n/{z}/{x}/{y}.png
5) **Google Maps**: https://mt1.google.com/vt/lyrs=r&x={x}&y={y}&z={z}
6) **GoogleSatellite:** http://www.google.cn/maps/vt?lyrs=s@189&gl=cn&x={x}&y={y}&z={z}
7) **Google Hybrid:** https://mt1.google.com/vt/lyrs=y&x={x}&y={y}&z={z}
8) **Google Terrain:** https://mt1.google.com/vt/lyrs=t&x={x}&y={y}&z={z}
9) **GoogleTraffic:** https://mt1.google.com/vt?lyrs=h@159000000,traffic|seconds_into_week:-1&style=3&x={x}&y={y}&z={z}
10) **Google Roads:** https://mt1.google.com/vt/lyrs=h&x={x}&y={y}&z={z}
11) **Bing maps:** http://ecn.dynamic.t0.tiles.virtualearth.net/comp/CompositionHandler/{q}?mkt=en-us&it=G,VE,BX,L,LA&shading=hill
12) **Bing Satélite**: http://ecn.t3.tiles.virtualearth.net/tiles/a{q}.jpeg?g=0&dir=dir_n'
13) **ESRI Imagery/Satellite:** https://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer/tile/{z}/{y}/{x}
14) **ESRI National Geographic**: http://services.arcgisonline.com/ArcGIS/rest/services/NatGeo_World_Map/MapServer/tile/{z}/{y}/{x}
15) **ESRI Physical:** https://server.arcgisonline.com/ArcGIS/rest/services/World_Physical_Map/MapServer/tile/{z}/{y}/{x}
16) **ESRI Streets:** https://server.arcgisonline.com/ArcGIS/rest/services/World_Street_Map/MapServer/tile/{z}/{y}/{x}
17) **ESRI Terrain:** https://server.arcgisonline.com/ArcGIS/rest/services/World_Terrain_Base/MapServer/tile/{z}/{y}/{x}
18) **ESRI Topo:** https://server.arcgisonline.com/ArcGIS/rest/services/World_Topo_Map/MapServer/tile/{z}/{y}/{x}
19) **ESRI Transportation:** https://server.arcgisonline.com/ArcGIS/rest/services/Reference/World_Transportation/MapServer/tile/{z}/{y}/{x}

Ejercicio: añadir direcciones y crear conexiones XYZ Tiles. Incorporar mapas, crear una tabla en Excel con datos y convertirla a un archivo de datos "csv" e incorporar una capa vectorial de puntos sobre el mapa y si es posible, realizar algunos análisis estadísticos de los puntos espaciales añadidos.

2) Introducción a datos espaciales de tipo continuo. Geoestadística

El término de Geoestadística, se refiere generalmente al análisis de datos espaciales $\{Z(s_1),\ldots,Z(s_n)\}$ en donde las localizaciones $\{s_1,\ldots,s_n\}$ no son aleatorias sino fijas.

Es muy habitual no tener observaciones de "Z" en todas las localizaciones deseadas, siendo uno de los propósitos de la Geoestadística, realizar "predicciones" (generalmente a través de interpolaciones) de "Z" en algunas localizaciones "s_0", en donde su valor es desconocido. Otrto aspecto de interés a conocer es la media de "Z" en alguna zona determinada "B_0".

Los anteriores dos objeticos suelen ser conseguidos realizando la estimación y modelizando la correlación espacial (covarianza o semivarianza) y analizando la estacionariedad, tanto en las localizaciones concretas como las situadas en redes (Grids). Trabajamos ahora con RStudio, por lo que cerramos QGis y procedemos a abrir el programa RStudio.

- Variograma

En los objetivos anteriormente descritos, existe una función determinante; se trata del "Variograma". De acuerdo a la definición más generalizada utilizada, indica como: "***variograma*** *o* ***semivariograma*** *a una herramienta que permite analizar el comportamiento espacial de una variable sobre un área definida, obteniendo como resultado un Variograma experimental que refleja la distancia máxima y la forma en que un punto tiene influencia sobre otro punto a diferentes distancias.*". Además se indica que: "*El resultado de este análisis no puede ser aplicado directamente en los diferentes métodos de interpolación que lo ocupan como información base, es por esto que una vez calculado el Variograma experimental, debe ser realizado un modelo matemático que modele de la mejor forma posible al Variograma experimental, el cual es conocido como Variograma teórico*[1]".

En cuanto a sus usos, se dice que "*El Variograma permite conocer el alcance, es decir, la distancia máxima a la que una muestra tiene influencia sobre otra muestra, una aplicación de esta información es conocer la vecindad en que se pueden buscar muestras para estimar el valor de un punto específico*".

Las características que permiten elegir un buen Variograma se definen como:

a) El número de pares
b) La suavidad de la curva debe ser suave
c) Los datos deben acercarse al modelo matemático

Para construir un Variograma en R se utiliza la función "variogram()", que viene incluida en el paquete "gstat", por lo que hay que cargar dicho paquete.

Se procede a instalar el paquete en RStudio, ya sea mediante el menú "Tools/Install Packages", o mediante la consola con la instrucción.

```
> install.packages("gstat")
```

Ahora cargar la librería, a través de "library(gstat)". Si se desea ayuda teclear en la consola "?gstat" y "?variogram".

[1] Fuente: Wikipedia

Ejemplo 4: (Abrimos un nuevo script en el que copiaremos los códigos)

Vamos a utilizar los datos del archivo "meuse", que se encuentran en la librería "sp" o el archivo "meuse10.txt" que vimos en unidades pasadas. En este conjunto o matriz de datos se encontraban entre otras variables, los niveles de "Zinc". Vamos a cargar la librería y los datos.

```
> library(sp)

> data(meuse)     # Carga los datos de la matriz meuse
```

Ahora, para recordar, vamos a presentar este conjunto de datos espaciales en una cuadricula regular, recordemos que se hacía mediante el comando "meuse.grid" y presentamos un resumen estadístico de estos datos espaciales mediante la función "summary()"

```
> data(meuse.grid)    # Presenta los datos espaciales en una cuadricula regular

> summary(meuse.grid) # Presenta un resumen de análisis estadístico de la rejilla
```

x	y	part.a	part.b	dist	soil	ffreq
Min. :178460	Min. :329620	Min. :0.0000	Min. :0.0000	Min. :0.0000	1:1665	1: 779
1st Qu.:179420	1st Qu.:330460	1st Qu.:0.0000	1st Qu.:0.0000	1st Qu.:0.1193	2:1084	2:1335
Median :179980	Median :331220	Median :0.0000	Median :1.0000	Median :0.2715	3: 354	3: 989
Mean :179985	Mean :331348	Mean :0.3986	Mean :0.6014	Mean :0.2971		
3rd Qu.:180580	3rd Qu.:332140	3rd Qu.:1.0000	3rd Qu.:1.0000	3rd Qu.:0.4402		
Max. :181540	Max. :333740	Max. :1.0000	Max. :1.0000	Max. :0.9926		

En la imagen que obteníamos en el capítulo I, se podía intuir mediante un análisis descriptivo que las mediciones de metales comparadas con las distancias al río evidenciaban que las concentraciones mayores se encontraban cercanas al río.

Esta relación se puede linealizar transformando logarítmicamente las concentraciones en este caso de "Zinc" y tomando la raíz cuadrada de la distancia al río mediante el siguiente comando.

Linear1<- plot((log(meuse$zinc)~sqrt(meuse$dist))), o también se puede escribir
Linear2<- plot((log(zinc)~sqrt(dist), meuse)

```
> Linear1<- plot((log(meuse$zinc)~sqrt(meuse$dist)))   # linealizar logarítmicamente la
concentración de Zinc

> Linear2<- plot(log(zinc)~sqrt(dist),meuse)     # linealizar logarítmicamente la concen
tración de Zinc
```

Recordemos que el símbolo "~", se encuentra mediante la combinación de teclas (Alt + 126)
Cualquiera de las dos instrucciones anteriores nos dará como resultado el mismo gráfico. Ahora, para colocar la recta de regresión, podemos utilizar la función "abline()", que introduce líneas rectas al gráfico y la función "lm()" que hemos visto que presenta un ajuste de modelo lineal. Y que calcula la recta de regresión como habíamos mencionado en la unidad I

```
#introducimos una recta de ajuste, cálculo de la recta de regresión

> abline(lm(log(zinc)~sqrt(dist), meuse))
```

El resultado es la gráfica de los puntos espaciales de las muestras de observación de "Zinc" con el modelo de ajuste mediante la recta de regresión.

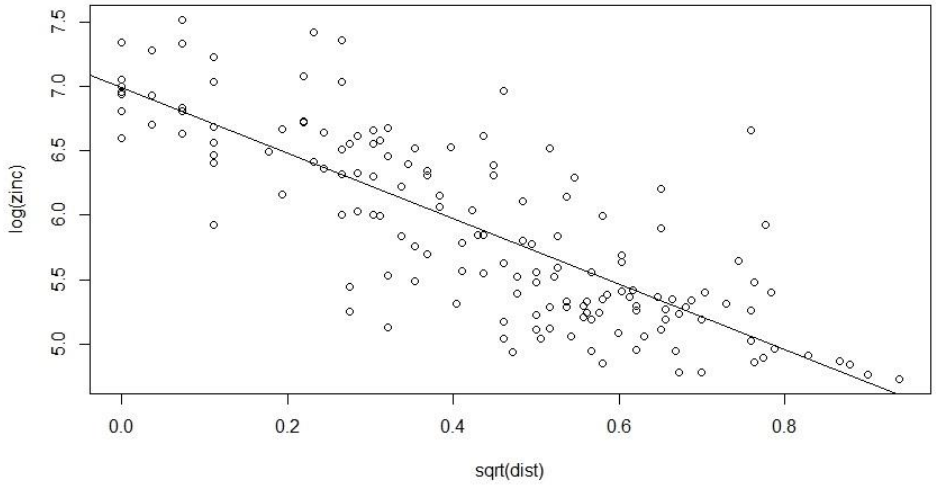


Fig.23. Gráfico de la concentración de Zinc con recta de regresión

Para el análisis de la Varianza, podemos ver en la imagen que se observa presenta una dispersión bastante grande y al que la recta de regresión no suministra un ajuste aceptable. En este caso se puede admitir una tendencia constante o fija con una ordenada al origen, lo que podrá construirse mediante un modelo de tendencia fija al origen con "log(zinc)~1". Pero lo primero es transformar la matriz de datos "meuse" a objeto de la clase SpatialPoints-DataFrame mediante la función "coordinates()" de la siguiente manera:

```
# Ajuste del modelo al origen mediante log(zinc)~1
# Se convierten en un objeto de la clase SpatialPoints-DataFrame
> coordinates(meuse) = ~x+y
```

Ahora se define el ajuste al modelo

```
# Se asume una tendencia constante para la variable log(zinc).

> TendenciaFija = variogram(log(meuse$zinc)~1, data=meuse)
```

El primer argumento de la función "variogram()" toma "log(zinc)~1", lo que significa que se asume una tendencia constante para la variable "meuse$zinc". Si observamos el contenido de los valores obtenidos en "TendenciaFija", veremos lo siguiente:

```
> TendenciaFija
    np       dist     gamma dir.hor dir.ver   id
1   57   79.29244 0.1234479       0       0 var1
2  299  163.97367 0.2162185       0       0 var1
3  419  267.36483 0.3027859       0       0 var1
4  457  372.73542 0.4121448       0       0 var1
5  547  478.47670 0.4634128       0       0 var1
6  533  585.34058 0.5646933       0       0 var1
7  574  693.14526 0.5689683       0       0 var1
8  564  796.18365 0.6186769       0       0 var1
9  589  903.14650 0.6471479       0       0 var1
10 543 1011.29177 0.6915705       0       0 var1
11 500 1117.86235 0.7033984       0       0 var1
12 477 1221.32810 0.6038770       0       0 var1
13 452 1329.16407 0.6517158       0       0 var1
14 457 1437.25620 0.5665318       0       0 var1
15 415 1543.20248 0.5748227       0       0 var1
```

Ahora mediante la función "fit.variogram()", vamos a ajustar el modelo del Variograma a un modelo de Variograma de muestra. El modelo se indica mediante el argumento "vgm", el cual genera un modelo de Variograma (ver fit.variogram()" y "¿vgm").

```
> AjusteModelo = fit.variogram(TendenciaFija, model = vgm(1, "Sph", 900, 1))
```

La función "fit.variogram" lleva como argumentos

i) El objeto que es el Variograma calculado "TendenciaFija"
j) El modelo Variograma de muestra a construir mediante la función "vgm()"

En la función "vgm()" se indica: *vgm(psill, model, range, nugget)*

i) psill: El umbral de la ventana en 1
ii) model: El modelo es "Sph" es de pepitas cilíndricas
iii) range: El rango del componente de modelo de Variograma, lo hemos establecido hasta 900 por norma general, en caso de anisotropía[2] se debe de establecer un rango mayor.
iv) nugget: tipo de componente de pepita del Variograma, asignado a 1. En caso de que falte, se omite este componente.

La función de ajuste al modelo de Variograma de muestra es la siguiente:

```
> AjusteModelo = fit.variogram(TendenciaFija, model = vgm(1, "Sph", 900, 1))
```

Si presentamos este modelo de ajuste tendremos:

```
> AjusteModelo
  model      psill    range
1   Nug 0.05066243   0.0000
2   Sph 0.59060780 897.0209
```

Presentamos la gráfica del modelo Variograma con el modelo Variograma maestral. El argumento "pl =T" presenta los valores de los puntos en el gráfico.

```
> plot(TendenciaFija, AjusteModelo, main="Variograma Muestral", col =c("red","
blue"), pl = T)
```

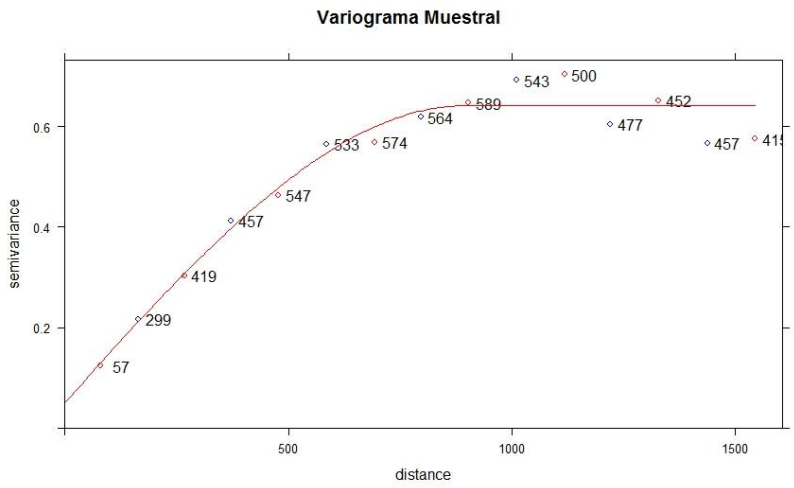


Fig.24. Gráfico del Variograma muestral.

[2] Se habla de anisotropía cuando se produce cualquier cambio de escala de una figura o un cuerpo, como en un gráfico x - y, con factores distintos (o en dependencia de una función) en cada coordenada.

En la figura se observa que a menos de distancia de 900 apenas existen datos y que la mayor correlación se encuentra en una distancia algo mayores a 1000, entre 1000 y 1500.

También, en lugar de optar por la constante al origen representada por (~1), se puede especificar una función media, utilizando la raíz cuadrada de la distancia como variable predictiva, siendo "~sqrt(dist)", vamos a realizar algunos cambios al modelo y observar los valores y el gráfico resultante.

```
> FuncionMedia = variogram(log(zinc)~sqrt(dist), meuse)

> AjusteModelo2 = fit.variogram(FuncionMedia, model = vgm(1, "Exp", 300, 1))

> AjusteModelo2
  model      psill    range
1   Nug 0.05712231   0.0000
2   Exp 0.17641559 340.3201

> plot(FuncionMedia, AjusteModelo2, main="Variograma Muestral 2 Ajuste media",
col =c("red","blue"), pl = T )
```

El gráfico resultante es:

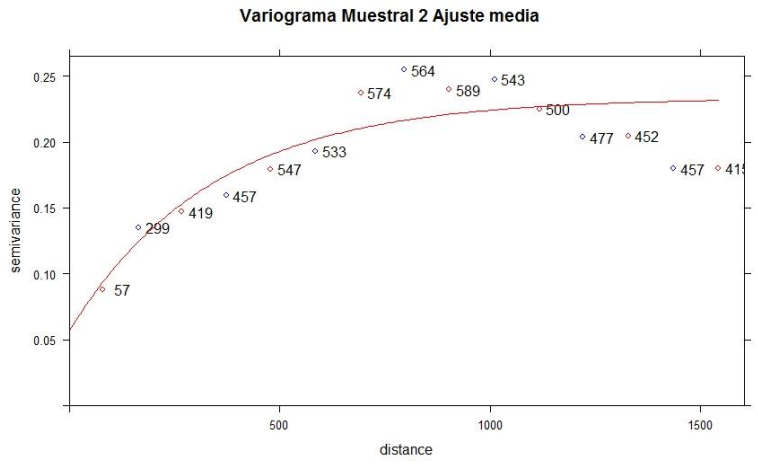


Fig.25. Gráfico del Variograma muestral con ajuste predictivo de media

El gráfico que se observa, presenta el Variograma de los residuos con respecto a la función media ajustada. Los residuos se han calculado utilizando mínimos cuadrados ordinarios.

Por otro lado, también es posible representar diagramas de dispersión de pares ordenados de observaciones de los datos de un archivo, en este caso de "meuse", definidos a distancias por ejemplo en intervalos de (0,50), (50,100), ((100,150), (150,200), (200,250), y (250 a 300).

Ejemplo 5: (Abrimos un nuevo script en el que copiaremos los códigos)

Para iniciar este proceso, primero cargamos la librería "sp", a continuación cargamos los datos mediante "data(meuse)", y convertimos en un objeto dela clase SpatialPoints-DataFrame mediante la función "coordinates(meuse)", que determinan las coordenadas sobre cuyos valores se establecerán los grupos o clases de valores de distancias menores que las ahí establecidas, es decir, nos vamos a referir a distancias Euclídeas en el plano de dos puntos $s_i - s_j$. Para ello necesitamos de la librería "gstat" y de la función "hscat()" que pertenece a esta librería. Por lo tanto cargamos la librería y construimos la función. Veamos hasta aquí este proceso.

```
> # Carga de librería y datos
> library(sp)
> data(meuse)

> # Conversión en objetos de tipo SpatialPoints-DataFrame
> coordinates(meuse) = ~x+y

> # Carga de librería (gstat)
> library(gstat)

> # Diagrama de dispersión por intervalos de distancia entre puntos mediante "hscat"
> hscat(log(zinc)~1, data = meuse, breaks = c(0,50,100,150,200,250,300))
```

El gráfico resultante con los intervalos es el siguiente.

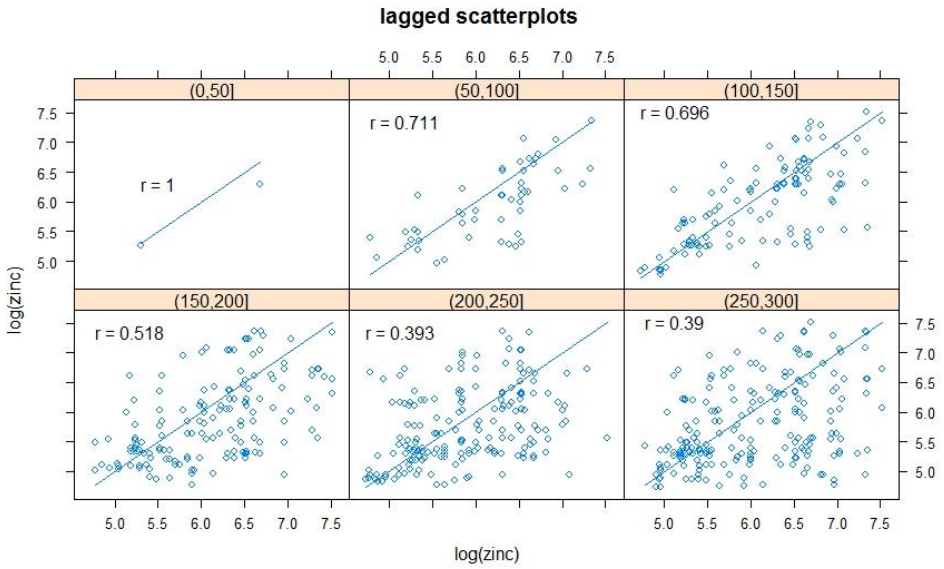


Fig.26. Gráfico del diagrama de dispersión por intervalos de distancias entre puntos.

En la función anterior "hscat()", el primer argumento representa los datos "log(zinc) que siguen como hemos presentado una tendencia constante con coordenada en el origen el segundo argumento, representa los datos, mediante "data=" que en este caso son los de "meuse", como podrían ser cualquier archivo de datos por ejemplo de tipo "csv", y por último, se indican los puntos de corte mediante el parámetro "breaks" con el vector indicando los intervalos de distancias, en base a las coordenadas proporcionadas con "coordinate(meuse).

En el gráfico se observa que apenas existen datos a menos de 50 y que la mayor correlación se encuentra en una distancia entre 50 y 100.

Sin embargo, la limitante de este gráfico es que para que pueda ser interpretable, no pueden utilizarse muchos datos ni un rango muy grande de distancias. Se puede representar la nube de Variograma a través de la siguiente sentencia.

```
> # Nube de Variograma
> plot(variogram(log(zinc)~1, meuse,cloud = T), main ="Nube de Variograma")
```

En donde el argumento "cloud" se designa a "TUE" o verdadero. El gráfico resultante es el siguiente.

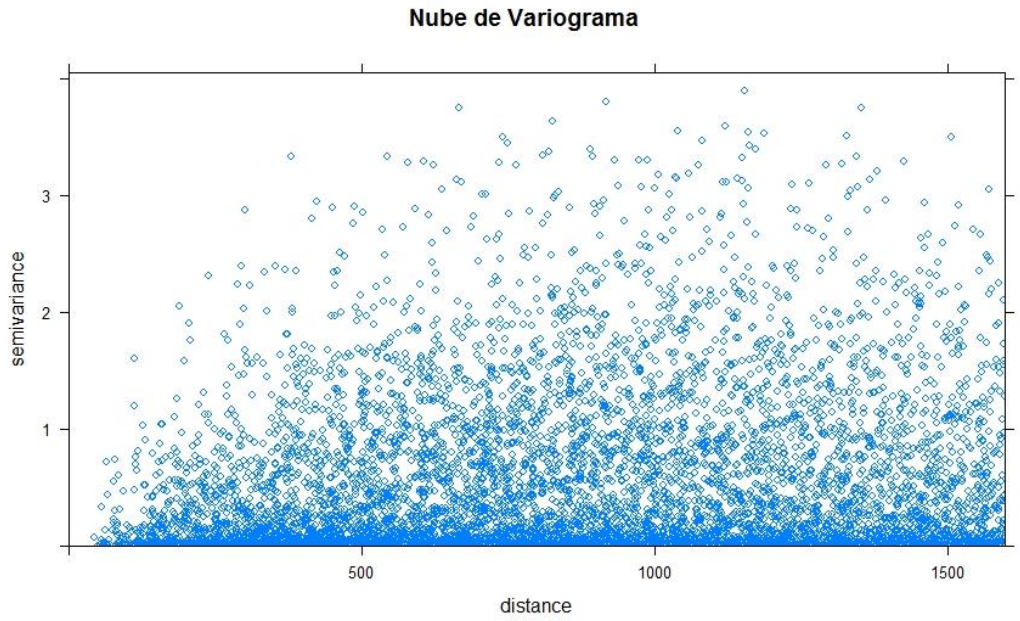


Fig.27. Gráfico Nube de Variograma.

En la figura se observa que los valores de la izquierda del eje de abscisas (h pequeños) y arriba del de ordenadas (diferencias grandes) son los que muestran, por lo tanto, una mayor pendiente y serán valores anómalos "outliers[3]" locales.

Ahora bien, en caso de querer identificar cuáles son estos puntos de entre los datos iniciales, utilizamos en la función "plot()", el argumento "digitize = T", lo que permite marcar con el ratón los límites de un polígono.

Una vez ejecutada la siguiente orden, se verá que el cursor del ratón cambia a una cruz "+", y al dar clic en un punto se presenta un símbolo de marca en color azul. Hay que tener cuidado de seleccionar todas las esquinas y cerrar el polígono, una vez finalizada las marcas se pulsa la tecla escape "Esc" del teclado y se verá el polígono generado.

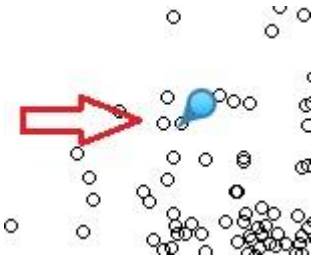


Fig.28. Marcas de la construcción del polígono en el gráfico Nube de Variograma.

```
 > #Variograma con polígonos de datos extremos
> VarPoligonos <- plot(variogram(log(zinc)~1,meuse,cloud=T),digitize =T)
[1] "mouse-left digitizes, mouse-right closes polygon"
```

Se observa, como al finalizar la construcción del polígono se cierra el polígono. La figura generada es la siguiente:

[3] Valor atípico, es una observación que es numéricamente distante del resto de los datos

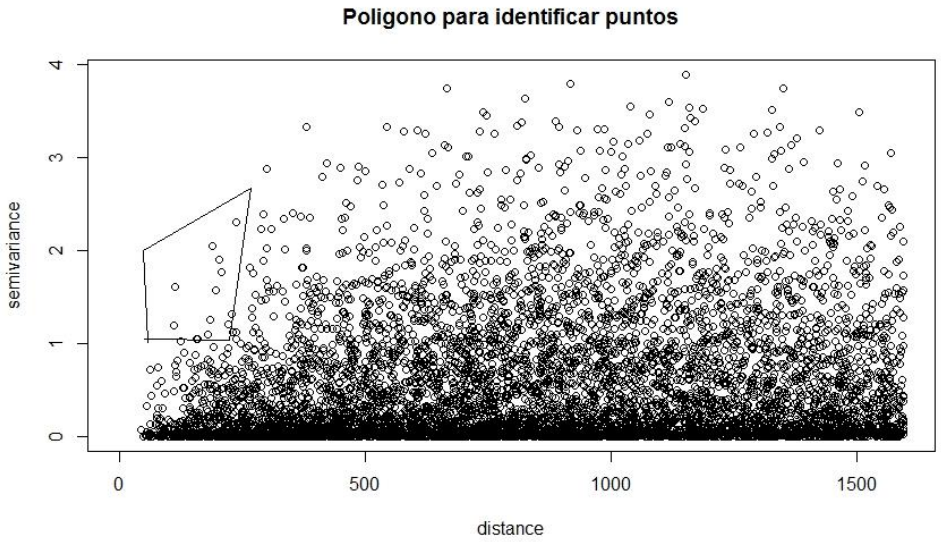


Fig.29. Nube de Variograma con Polígono de datos extremos.

Ahora se pueden observar cuales son los datos originales dentro del polígono generado, ejecutando un "plot()", con argumentos del polígono generado y los datos originales de "meuse".

```
> plot(VarPoligonos,meuse, main = "Datos originales de dentro del polígono", c
ol = "blue")
```

El gráfico resultante es el siguiente.

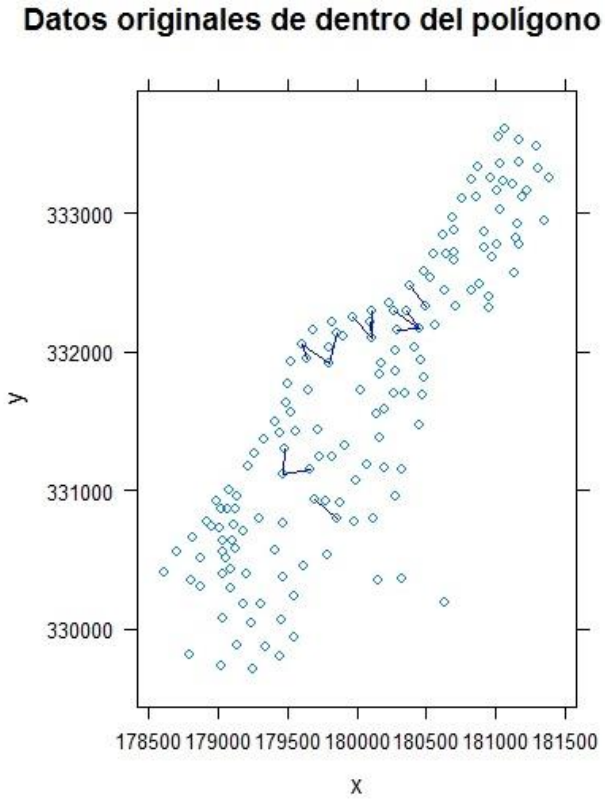


Fig.30. Datos originales de dentro del polígono de datos extremos.

3) Introducción a datos espaciales agregados o regionales

En muchas ocasiones los datos espaciales observados representan áreas o zonas, datos que se denominan como datos agregados o regionales. Estas áreas van a ser las habitualmente designadas como unidades de investigación, dicho de otro modo, los datos observados, los cuales deben de poseer límites bien definidos. Sin embargo, por su propia definición, varias de dichas áreas pueden unirse y formar un nuevo dato, por lo que interesa analizar si existe autocorrelación espacial entre varias unidades, debido a que en muchas ocasiones, lo que pase en una determinada zona se encuentra relacionado con lo que suceda en una zona adyacente. Este problema es conocido como el problema de "Galton" y consiste en establecer cuantas observaciones (zonas) independientes existen en la muestra cuando se han utilizado límites arbitrarios en la definición de las áreas de estudio. Así que el primer paso es estudiar cómo definir pesos a las áreas o zonas consideradas.

- Entornos y pesos de Áreas

Asignar pesos a las zonas de estudio es necesario, si lo que queremos realizar es un análisis de datos espaciales supuesto que estas sean Áreas, fundamentalmente con el propósito de establecer residuos totalmente independientes (sin autocorrelación espacial) lo que será necesario cuando se realice el ajuste de un modelo a este tipo de datos.

El tener que determinar las zonas (o clusters) a considerar ya en si es un problema, aunque se supone que las zonas ya se encuentran definidas. Posteriormente, a la hora de fijar pesos a dichas zonas se recomienda asignar un peso igual a uno (1) a las zonas limítrofes y un peso igual a cero (0) a las zonas que no sean limítrofes a una zona dada. Es decir, en caso de que una zona A solo tenga una zona como vecina, el peso de A será de 1, pero si A posee dos zonas vecinas, su peso será de dos. Esta forma de fijar pesos se denomina "binaria" y con ella, si una zona tiene a su alrededor2 zonas, su peso será el doble que el de otra zona con sólo una zona limítrofe.

La forma natural de contar cuantos vecinos tiene una zona es unir los centroides de las zonas. El número de conexiones entre los centroides nos va dar su peso.

La forma binaria de asignar pesos hará que algunas áreas tengan peso 2 y otras pesos por ejemplo 7. Otra forma de asignar pesos es la forma "ponderada", en donde los pesos de cada área son estandarizados de forma que todos los pesos sumen 1.

En R la función "nb2listw()" de la librería "spdep", es la que calcula los pesos de las zonas de las dos maneras mencionadas (binaria y ponderada),las cuales se indican mediante el argumento "style", asignando el valor "W" en el caso de ser de la forma ponderada y por medio del argumento "B" cuando se realiza de una forma binaria.

- Contraste global de autocorrelación espacial: Estadístico I de Moran

El estadístico (índice) de I Moran se define como el cociente del producto de la variable de interés y su retardo (lag), dela siguiente forma

$$I = \frac{n}{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}} \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\,(y_{i-}\,\bar{y})(y_{j-}\,\bar{y})}{\sum_{i=1}^{n}(y_{i-}\,\bar{y})^2}$$

Este índice toma un valor comprendido entre (-1 y 1). Si I 0 0 se interpreta que los datos se encuentran distribuidos al azar (del estilo del CSR); si es positivo habrá concentración y, si en cambio toma un valor negativo, se entiende que existe una dispersión mayor de la que se tendría si los datos se distribuyeran al azar.

Habitualmente, se calculan los valores de este índice, de su media, de su varianza, de la desviación estándar, así como del p-valor del test de la hipótesis nula de ausencia de autocorrelación espacial. Veamos un ejemplo de esto.

Ejemplo 6: (Abrimos un nuevo script en el que copiaremos los códigos)

En este ejemplo, vamos a suponer que se tiene interés en estudiar el estado norteamericano de Nueva York, que es un estado compuesto por varios condados. En el directorio de datos del ejemplo se encuentran los tres archivos Shapefiles asociados a un GIS (shp, shx y dbf), los tres con el nombre "NY". Copiamos los archivos al directorio de datos para que de ahí se puedan leer.

La matriz de datos se refiere a casos de leucemia en este estado y se encuentra formada por 281 individuos y 12 variables. En QGis se puede importar directamente el archivo "NY.shp", sabiendo que se encuentra en coordenadas UTM zona 18N. Pero aquí lo vamos a representar mediante RStudio.

Lo primero es cargar las librerías (spdep) y (rgdal), que vamos a utilizar, en caso de que no se encuentran los paquetes, deben de ser instalados y posteriormente llamados por "library()". A continuación, cargaremos los datos desde el archivo de datos utilizando la función de lectura para este tipo de archivos "readOGR()" (Nota: es la letra o no el número cero). La instalación desde el Menú "Tools" o desde la línea de comandos:

```
 > # instalación de paquete spdep
> install.packages("spdep")

> # Carga de librería sp
> library(sp)

# Instalación de paquete rgdal
> install.packages("rgdal")

> # Lectura del archivo NY
> NuevaY <- readOGR(dsn="./datos", layer="NY")
OGR data source with driver: ESRI Shapefile
Source: "C:\Users\Lig\Documents\Programas_R\datos", layer: "NY"
with 281 features
It has 17 fields
Integer64 fields read as strings:  CLAVEAREA

> # Dibujo de grafico en color de borde azul border=4 y con groso lwd=2
> plot(NuevaY, border = 4, lwd =2, main = "Mapa del Estado de Nueva York")
```

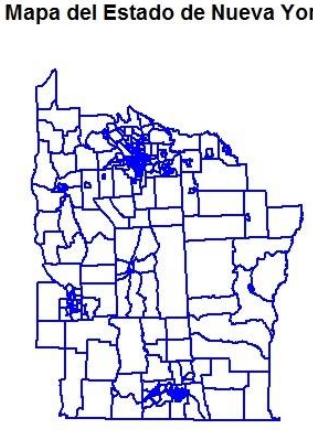


Fig.31. Gráfico con el mapa del estado de Nueva York, leído desde un archivo Shapefile.

4) Generación de gráficos varios para ayuda de análisis estadísticos en R

- Gráficos demográficos

Ejemplo 7: (Abrimos un nuevo script en el que copiaremos los códigos)

La estructura de la edad de una población se puede representar mediante la función "pyramid.plot()" del paquete "plotrix". En este archivo de datos, sepresenta el porcentaje de hombres y mujeres españolas, junto con el de hombres y mujeres extranjeros. Los datos han sido publicados por el Instituto Nacional de Estadística de España (www.ine.es)

Se procede a instalar y cargar el paquete en el sistema. Posteriormente se calcula el porcentaje de hombres y mujeres españoles y extranjeros durante 1991. En el primer gráfico sed representa a la población de españoles y en el segundo a la de extranjeros. El argumento "gap" en la función "pyramid.plot()" sirve para establecer el espacio entre la pirámide de hombres y la de mujeres. Mediante el argumento lógico "show.values=T" se especifica que se muestren los valores de las barras y con el argumento "ndig" se define el número de decimales a colocar. Finalmente, en el segundo gráfico se debe de especificar el argumento "add=T", para que se presente o pinte sobre el anterior y no sea un gráfico nuevo.

Las instrucciones son las siguientes

```
> #Instalacion paquete plotrix
> install.packages("plotrix")
Installing package into 'C:/Users/Lig/Documents/R/win-library/3.6'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.6/plotrix_3.7-7.zip
'
Content type 'application/zip' length 1132324 bytes (1.1 MB)
downloaded 1.1 MB

package 'plotrix' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\Lig\AppData\Local\Temp\RtmpkvgRpy\downloaded_packages
> # Carga de paquete en el sistema
> library(plotrix)

> # Lectura del archivo de datos csv
> Poblacion1<-read.csv2(file= "./datos/Población.csv", header=TRUE, encoding="
latin1")

> #Se adjuntan los datos
> attach(Poblacion1)

> # Se visualizan los datos
> Poblacion1
             Edad  M.1900  H.1900  M.1991  H.1991 ME.1991 HE.1991
1        < 5 1070030 1091022  978088 1031838   65532   70165
2     5 a 9 1047335 1065722 1187252 1249688   97355  108723
3   10 a 14  971953  987686 1505300 1580502  177625  188080
4   15 a 19  804024  754471 1631351 1708221  205550  220361
5   20 a 24  819834  737866 1586109 1651254  247433  260898
6   25 a 29  736258  678726 1533448 1570881  282154  293755
7   30 a 34  665382  622601 1425248 1437258  253694  265893
8   35 a 39  583438  553103 1251522 1255807  220268  233580
9   40 a 44  601506  552430 1205572 1199423  183247  225492
10  45 a 49  491138  450703 1103166 1089911  142308  170026
11  50 a 54  499550  455312 1008239  964961  108890  119655
12  55 a 59  358105  337679 1153216 1086317   98023   98855
```

```
13   60 a 64  364305  329435 1105315 1002129   87320   82175
14   65 a 69  204716  191785  989769  844266   60376   63320
15   70 a 74  160469  146761  774254  561392   40261   37613
16   75 a 79   75874   72784  641737  410966   20536   14795
17   80 a 84   47903   36533  445807  252288    8470    7316
18      > 85   18554   12395  310429  139344    4656    3484
```

```
> #Los datos se transforman a porcentajes
> s<-sum(H.1991) + sum(M.1991)+ sum(HE.1991)+ sum(ME.1991)
> M1991<-M.1991*100/s; H1991<-H.1991*100/s
> ME1991<-ME.1991*100/s; HE1991<-HE.1991*100/s

> # Se construye el Primer gráfico
> pyramid.plot(H1991,M1991,labels=Edad,
+              main="Población española y extranjera en España en 1991",lxcol=
"blue",rxcol="pink",labelcex=1.2, unit="%",
+              gap=1,show.values=T, top.labels=c("Hombres","Edad","Mujeres"),n
dig=1,xlim=c(5,5))
[1] 5.1 4.1 4.1 2.1

> # Se construye el Segundo gráfico
> pyramid.plot(HE1991,ME1991,lxcol="red",rxcol="red",
+              labelcex=0, gap=1,show.values=F, ndig=2,xlim=c(5,5),add=T)
[1] 4 2 4 2
```

El gráfico resultante es el siguiente:

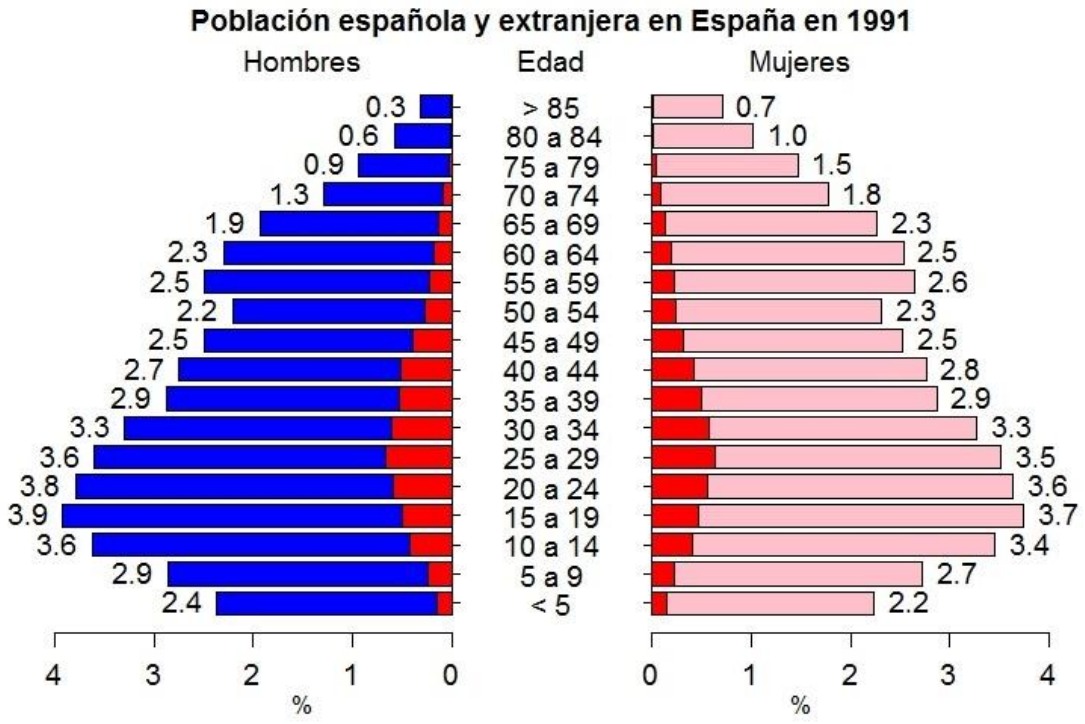


Fig.32. Gráfico demográfico dela población española y extranjera durante 1991.

Estos gráficos son bastante útiles para representar los movimientos por ejemplo de estudiantes en hombres y mujeres en centros escolares, campañas, etc.

- Gráficos de Burbujas

En el mismo paquete “plotrix” se puede utilizarla función “size_n_color()” para representar un gráfico de burbujas.

Ejemplo 8: (Abrimos un nuevo script en el que copiaremos los códigos)

Para este ejemplo, vamos a utilizar el archivo “csv” de datos llamado “AARotiferos.csv”, que muestra la abundancia y concentración de aminoácidos por individuo, de la especie de rotífero “Brachionus angularis, en las lagunas del parque Nacional de Doñana (España).

El gráfico de abundancia de los individuos se va a representar mediante el tamaño de los círculos y el gradiente de colores dentro de los círculos representa a su vez, la concentración de los aminoácidos. La escala de la variable que se representará con los tamaños de los círculos (en este caso la abundancia), debe de tener un rango similar a la escala de la variable del eje x (en este caso la temperatura). Por esta razón, en primer lugar la variable abundancia se divide por 100.

A través de la función “color.scale()” se define el gradiente del color dentro de los círculos y la función de la matriz de datos que se utiliza para los cálculos.

El último número puede ser cero (0) (escala verdes) o uno (1) (escala azules). Si solamente se desea representar la variable abundancia y no la concentración de aminoácidos, mediante el argumento “col” se define el color de los círculos y no sería entonces necesaria la función “color.scale()”, debido a que todos los círculos tendrían el mismo color.

Los argumentos “xat” e “yat” se utilizan para ubicar las marcas y etiquetas de los ejes. La leyenda del tamaño de los círculos se representa por medio de las funciones “points()” y “text()”. En cuanto a la leyenda del gradiente de colores se representa con la función “color.leyend()”, en donde se especifican las coordenadas de los extremos del rectángulo de la leyenda (cuatro coordenadas), los límites dela secuencia de colores y el intervalo, en caso de que se presenten en posición vertical, se indica con el argumento “gradient=y”, y si es en posición horizontal, mediante “gradient=x”.

La ubicación de las etiquetas, si son a la derecha mediante el argumento “align=rb” o a la izquierda “align=lt”. Con el argumento “rect=col” se pone el mismo intervalo para la secuencia de colores y el mismo gradiente de colores que se utilizó en la función “size_n_color()” con el argumento “color.scale”. Para poder ver la ayuda y los argumentos de las funciones, recordemos utilizar (¿size_n_color).

Las instrucciones son las siguientes:

```
> #Carga de librería
> library(plotrix)

> #Lectura de los datos desde el archivo csv
> Burbuja1<-read.csv2(file= "./datos/AARotíferos.csv", header=TRUE, encoding="
latin1")

> # Se cargan los datos
> attach(Burbuja1)
```

```
> # Se visualiza el contenido de los datos
> Burbuja1
           Abundancia   AA Temperatura Conductividad
         1          0 4952        23.9         1.034
         2         81 5230        27.5         0.905
         3          0 4952        32.6         0.217
         4          6 5789        32.8         1.535
         5         22 6629        32.3         1.162
         6          0 4952        26.4         0.419
         7          0 4952        31.3         0.371
         8         21 6785        28.5         1.563
         9          0 4952        28.4         0.747
         10       104 4952        25.6         0.670
         11        92 4560        24.6         0.957
         12        14 5203        32.7         0.556
         13         8 5568        31.4         0.978
         14         6 5780        30.9         1.110
         15        34 5202        30.1         0.735
         16        53 5304        29.1         1.300
         17        66 5003        27.3         1.190
         18         5 5789        32.0         1.490
         19         7 5789        32.6         1.540

> # La variable abundancia se divide por 100
> Ab<-Burbuja1$Abundancia/100
> # Contenido de la variable abundancia
> Ab
 [1] 0.00 0.81 0.00 0.06 0.22 0.00 0.00 0.21 0.00 1.04 0.92 0.14 0.08 0.06 0.3
4 0.53 0.66 0.05 0.07

> # Construcción del Gráfico con los circulos mediante points()
> size_n_color(Temperatura, Conductividad, Ab, sizefun="sqrt",cex.main= 1.6,
+              color.scale (AA,c(0.8,0),c(0.8,1),1), pch=16, xlab="", ylab="",
+              main="Abundancia y concentración de aminoácidos", xat=seq(20,35,by=5),
+              yat=seq(0,2,by=0.5), xlim = c(20,35), ylim=c(0,2))

> # Construcción de los puntos y leyendas
> points(21,1.9,cex=10);text(22,1.9,"100 Individuos por litro",cex=1.2, pos=4)
> points(21,1.685,cex=2.5);text(22.8,1.685,"  25",cex=1.2)
> points(21,1.580,cex=1);text(22.8,1.58,"  10",cex=1.2)
> color.legend(20,0,21,0.50,seq(4500,6500,by=500),gradient="y",align="rb", rect.col=colo
r.scale(seq(4500,6500,by=500),c(0.8,0),c(0.8,1),1))
> text(23.8,0.28,"Picomoles", cex=1.2)
> text(23.8,0.195,substitute("individuo"^-1),cex=1.2)
> mtext(expression(paste("Conductividad (mScm"^-2,")")), 2, line=2.4, font = 1, cex=1.4)
> mtext(expression(paste("Temperatura (°C)")), 1, line=2.8, font = 1, cex=1.4)
```

El gráfico generado es el siguiente:

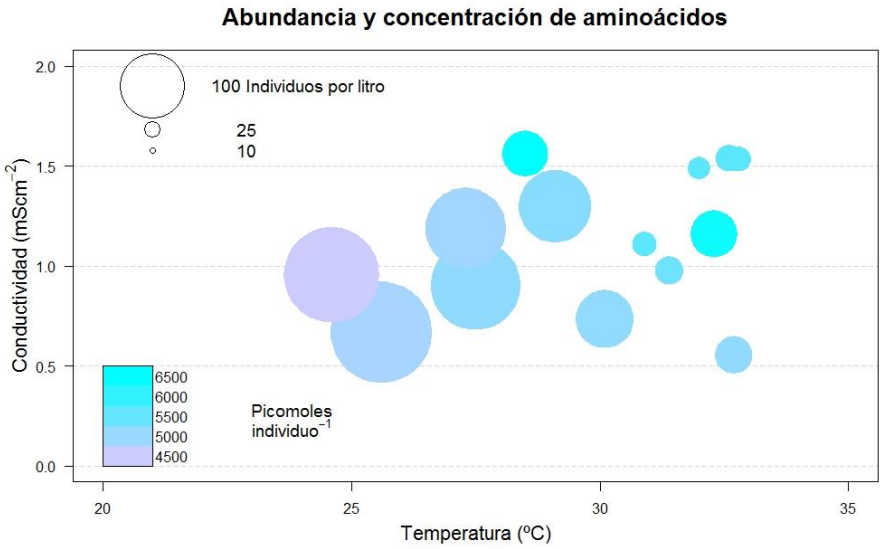


Fig.33. Gráfico de Burbujas de la Abundancia y concentración de aminoácidos.

Ejercicios: Crear dos conjuntos de datos en una tabla de Excel y guárdelos como un archivo “csv”, mediante dichos archivos de datos cree un gráfico demográfico y otro de burbujas. Pruebe a variar los parámetros delos gráficos para presentar mejor los resultados.

- **Bibliografía**:

1. García Alfonso. (2008) Estadística aplicada con R. Madrid. Librería UNED
2. García Alfonso. (2008) Ejercicios de estadística aplicada. Madrid, Librería UNED
3. Guisande González Castor, Vaamonde Liste Antonio (2012) Gráficos estadísticos y mapas con R, Madrid, Ediciones Díaz de Santos
4. García Alfonso. (2014) La interpretación de los datos Una introducción a la estadística aplicada. Madrid, Librería UNED
5. Cabrero Yolanda, García Alfonso. (2015) Análisis estadístico de datos espaciales con QGis y R. Madrid, Librería UNED.

- **Enlaces de Interés.**

1.- Páginas Web de R
https://rstudio.com/
https://www.r-project.org/

2.- Página Web de RStudio
https://rstudio.com/products/rstudio/

3.- Página Web de QGis

https://www.qgis.org/en/site/index.html

4.- Páginas Web para documentación QGis

https://docs.qgis.org/3.4/es/docs/user_manual/ (Manual de usuario)
https://docs.qgis.org/3.4/es/docs/training_manual/ (Manual de aprendizaje)
https://docs.qgis.org/3.4/es/docs/gentle_gis_introduction/ (Introducción a GIS)