

MANUAL DE USUARIO:

Análisis estadístico de datos espaciales con R y QGIS

Unidad II

Introducción a la estadística descriptiva e Inferencial y uso básico
con R y sus interfaces



Autor: Ligdamis A. Gutiérrez E. PhD.

Versión 1.0

ÍNDICE

Unidad II: Introducción a la estadística descriptiva e Inferencial y uso básico con R y sus interfaces

1) Introducción a la estadística descriptiva.....	3
2) Datos unidimensionales	3
○ Representaciones gráficas	3
▪ Datos de tipo cualitativo	3
▪ Datos de tipo cuantitativo	3
○ Medidas de posición	4
▪ Media aritmética	4
▪ Mediana	4
○ Medidas de dispersión	4
• Varianza	4
• Desviación Típica	4
3) Datos Bidimensionales.....	18
○ Tablas de datos	18
○ Ajustes por mínimos cuadrados	18
○ Precisión del ajuste por mínimos cuadrados	22
○ Gráficos dentro de gráficos	29
4) RCommander para análisis de datos estadísticos	35
5) Introducción a los datos espaciales de tipo discreto	43
○ Introducción	43
○ Datos espaciales y su representación	44
○ Representación en puntos y Polígonos	43
▪ Generación de puntos espaciales básicos	45
▪ Generación de puntos espaciales de tipo Polígono.....	48
▪ Generación básica de líneas con datos espaciales	52
○ Generación de datos espaciales mediante un archivo de datos	53
▪ Representación en puntos y resumen estadístico de los datos espaciales mediante un archivo de datos	55
▪ Representación por medio de líneas	57
▪ Representación por medio de Redes (Grids)	58
6) Introducción a los Procesos puntuales Espaciales.....	62
○ Análisis de la distribución espacial	63
○ Procesos Puntuales y Aleatoriedad Espacial Completa (CSR)	66
○ Análisis de la CRS a partir de datos creados	71
7) Bibliografía y Enlaces de Interés	62

1) Introducción a la Estadística Descriptiva

En la estadística, los datos son el elemento más importante, por lo que se les debe de dar un correcto tratamiento, en esta unidad se verá como resumirlos con una media de posición, la media o la mediana y analizar lo concentrados que se encuentran alrededor de la media con una medida de dispersión, la varianza o la desviación típica. Estos tres aspectos es lo que se denomina, “Estadística descriptiva”, y que se dedica a analizar y representar los datos. Primero vamos a considerar datos unidimensionales y posteriormente datos bidimensionales, tales como, por ejemplo, peso y altura o Edad y nivel de educación entre un conjunto de datos. Los datos por lo tanto no son más que el resultado de observar una o varias variables unidimensionales, como peso, edad, educación, etc., en el conjunto que forman la muestra, entendido como un grupo de elementos elegidos al azar de la población denominada “Inferencia Estadística”. Con la Estadística descriptiva se deja que los datos hablen por sí mismos, dando una representación fija de la población de la que se desea sacar conclusiones mediante la Inferencia Estadística.

En la primera unidad, ya hemos hecho una introducción de los conceptos básicos que intervienen en la estadística descriptiva. En esta unidad se profundizará aún más sobre el papel del análisis estadístico de los datos a través de la estadística descriptiva y sus representaciones gráficas

2) Datos Unidimensionales

Con los datos unidimensionales se pueden analizar tres aspectos: su representación gráfica, el cálculo de medidas de posición y el cálculo de medidas de dispersión.

○ Representaciones Gráficas

Los datos unidimensionales pueden ser de dos tipos: Proceden de la observación de una variable de tipo “cualitativa”, como el estado civil, el sexo, cuyos valores no son numéricos (Casado, mujer), o bien proceden de una variable de tipo “cuantitativo”, que proporcionan valores numéricos como el peso, la altura. De esta forma, la representación gráfica de dichos datos, depende de la clase que a la que pertenezcan.

• Datos de tipo cualitativo.

Para los datos de tipo cualitativo, las dos representaciones gráficas son el “Diagrama de sectores”, que se establece con la función “pie()” o también conocido como diagrama de pastel, que consiste como hemos visto en dividir un círculo en tantos sectores como valores posea la variable cualitativa, asignando a cada sector un tamaño (ángulo) proporcional al número de elementos que presenten dicho valor, número que se conoce como “frecuencia absoluta de valor”. El segundo es el diagrama de Rectángulos, el cual se obtiene mediante la función “barplot()”.

Los datos que provienen de observaciones de una variable de esta clase, vendrán contenidos en una tabla (función “table()”) , en donde se presenta el recuento de elementos que presentan los diferentes valores de la variable.

• Datos de tipo cuantitativo

El caso de los datos de este tipo es numérico, habitualmente los datos corresponden a un carácter cuantitativo sin agrupar en intervalos. Las representaciones gráficas más generales para presentar este tipo de datos son el “Diagrama de barras” representado por la función “barplot()”, que se emplea cuando los valores distintos son pocos y el histograma, que se representa mediante la función “hist()” y que se utiliza cuando los valores distintos son muchos. También, cuando se presenta el caso de frecuencias acumuladas, la representación gráfica será el “Diagrama de

Frecuencias Acumuladas”, llamado “Función de distribución empírica”, y si las frecuencias acumuladas a representar son las relativas.

- Medidas de Posición

Las principales medidas de posición se dividen en dos grupos

- a) Medidas de Tendencia Central (Media y Mediana):

- La Media Aritmética: se calcula mediante la función “mean()”,
- La mediana, que se obtiene a través de la función “median()”

- b) Medidas de posición de Tendencia no central (los cuantiles), que se calculan mediante la función “quantile()”

- Medidas de Dispersión:

- Varianza
- Desviación Típica

La cuasi-varianza y la cuasi-desviación típica, se calculan mediante las funciones “var()” y “sd()”. En caso de necesitar la varianza o la desviación típica, basta con multiplicar el resultado de las funciones *var* y *sd* por $(n - 1)/n$, siendo *n* el número total de datos con el que se está trabajando.

- Varianza: En R se realiza el cálculo de cuasi-varianza mediante la función “var()”
A su vez, la varianza = $(n-1)/n * var(x)$
- Desviación Estándar: La cuasi-desviación típica se calcula con “sd()” y a su vez la desviación estándar = $\sqrt{varianza}$

Ahora, veamos unos ejemplos prácticos de este tipo de datos, su representación gráfica y el análisis estadístico de las medidas de Posición, Relativas y de Forma de los datos

Ejemplo1.

En un estudio sobre las razones por las cuales no fue programado un curso de formación profesional, designado para trabajadores sin empleo, se obtuvieron los datos por la siguiente distribución de frecuencias absolutas.

Motivos	n_i
No presentaron títulos	45
No cumplían con la edad	5
No presentaron identificación	3
No cumplían con el perfil académico	14
Solicitado en la convocatoria	
Otras causas	2
	69

Para realizar un análisis descriptivo de este caso, se va a utilizar el diagrama de sectores. Para analizar, lo primero es crear con los datos de entrada un vector.

```
> Causas <- c(45,5,3,14,2)
```

Los títulos de los motivos pueden contenerse en otro vector.

```
> Motivos <- c("No títulos", "No Edad", "No ID", "No PA", "Otro")
```

Ahora a través de RStudio, de forma sencilla se puede construir el diagrama, aplicando lo ya aprendido en la unidad anterior. La expresión será la siguiente:

```
> pie(Causas, clockwise = TRUE, main = "Motivos de No Programar Curso",  
, col =c(2:6), labels = Motivos)
```

Siendo la representación gráfica resultante.

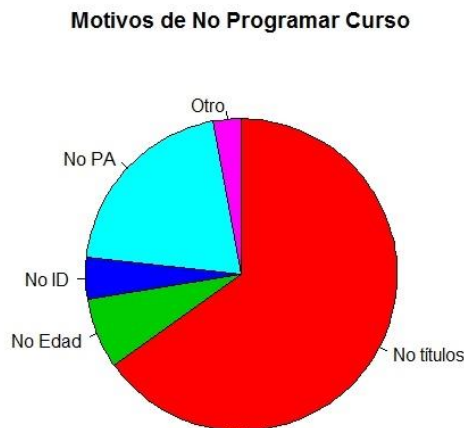


Fig. 1 Diagrama de sectores de Motivos

En cuanto al diagrama de rectángulos, se obtiene de forma sencilla mediante la función “barplot()”. La expresión será la siguiente

```
> barplot(Causas, names = Motivos, col = c(2:6), main = "Motivos de No  
Programar Curso", xlab = "Motivos", ylab = "Número", legend.text= Moti  
vos)
```

La representación gráfica resultantes es:

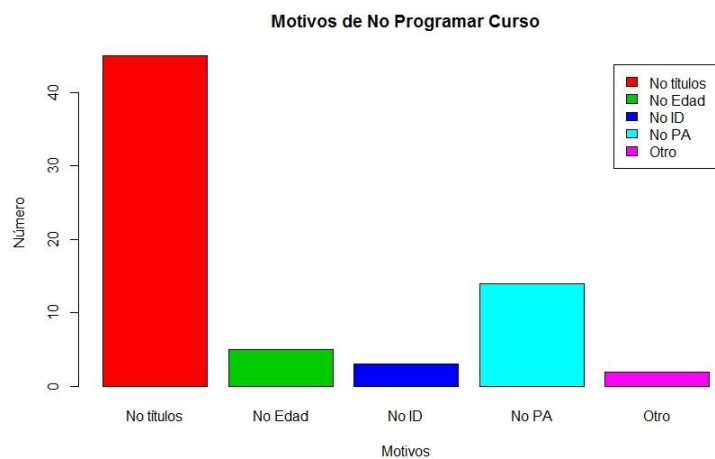


Fig. 2 Diagrama de Rectángulos de los motivos

El análisis estadístico de las Medidas de Posición, Relativas y de Forma de los datos es:

I) Cálculo de las Medidas de posición:

a) Medidas de Tendencia Central (Media y Mediana)

```
> (media_Aritmetica = mean(Causas))    # Cálculo de la Media Aritmética
[1] 13.8

> (Mediana = median(Causas))           # cálculo de la Mediana
[1] 5
```

b) Medidas de posición de Tendencia no central (los cuantiles)

```
> quantile(Causas, probs = c(0.25, 0.75))    # Cálculo de los cuantiles
indicando probabilidad

25% 75%
3   14

> quantile(Causas, probs = seq(0, 1, 0.25), na.rm = FALSE)
0% 25% 50% 75% 100%
2   3   5   14  45
```

II) Cálculo de las Medidas de Dispersión (varianza, desviación estándar e IQR)

```
> (VarianzaMuestral = var(Causas))    # cálculo de la Cuasi-Varianza
[1] 326.7

> (Varianza <- 4/5 * var(Causas))      # cálculo de la Varianza
[1] 261.36

> (DesvTipicaMuestral = sd(Causas))    # cálculo de la cuasi-Desv. Típica
[1] 18.07484

> (Desvt_Causas <- sqrt(Varianza))    # Cálculo de la Desviación Estándar
[1] 16.16663

> IQR(Causas)                          # Cálculo del Rango Intercuartílico
[1] 11
```

El resumen estadístico se obtiene mediante la función “summary()”.

```
> summary(Causas)                      # Resumen Estadístico
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2.0    3.0    5.0   13.8   14.0   45.0
```

III) Cálculo de las Medidas de Forma

Las medidas de forma se centran en el estudio de la forma que presenta una distribución, mediante el análisis de la simetría y la Kurtosis o el apuntamiento de la distribución en cuestión.

Para determinar la simetría de una distribución en R se utiliza la función “`skewness()`”, que viene incluida en el paquete “`moments`”. Por lo tanto, para poder utilizar esta función, es necesario instalar el paquete y, posteriormente cargarlo.

Para instalar el paquete “`moments`”, recordemos que hay que seleccionar en el menú principal Tools/Install Packages, de esta forma, se presentará la siguiente ventana donde escribimos “`moments`”.

```
> install.packages("moments")
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/moments_0
.14.zip'
Content type 'application/zip' length 56559 bytes (55 KB)
downloaded 55 KB
```

package ‘`moments`’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
C:\Users\Lig\AppData\Local\Temp\RtmpMZiIzR\downloaded_packages

A continuación, hay que cargarlo en el Sistema para poder utilizarlo

```
> library("moments")
```

Ahora se utilizará la función “`skewness()`”, cuya sintaxis es la siguiente

`skewness(x, na.rm = FALSE)` # donde

`x`: es el vector que incluye los valores de la variable

`na.rm`: es un argumento lógico que indica si hay que eliminar los valores faltantes del conjunto de datos.

```
> skewness(Causas)
[1] 1.267936
```

Ahora, para estudiar la curtosis (de una distribución univariante), de un conjunto de datos, se utiliza la función “`kurtosis()`”, con los mismos parámetros que la anterior

```
> kurtosis(Causas)
[1] 2.888563
```

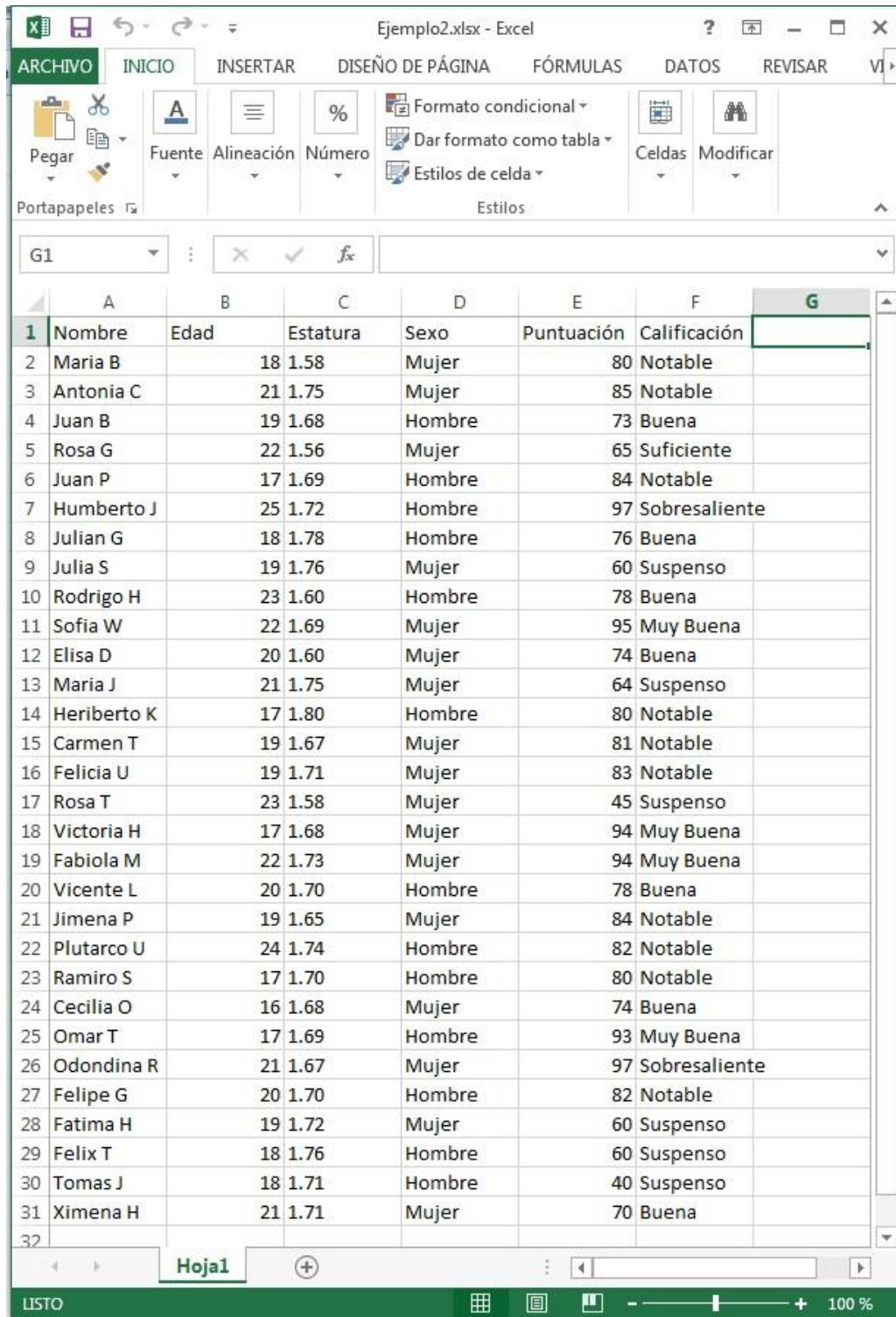
De la misma forma que en la función **`skewness`**, se puede utilizar el parámetro lógico “**`na.rm`**” que indica si hay que eliminar los parámetros faltantes denominados (NA) por defecto a FALSO. La sintaxis completa es:

`kurtosis(x, na.rm = FALSE, type = 3)`

Actividad_Ejemplo1: Crear un script llamado Ejemplo1_Unidad_II, en el que se coloquen los comandos o expresiones generadas en este ejercicio, guardarlo en una carpeta de datos para la unidad y probar sus resultados.

Ejemplo No. 2

En una facultad de una Universidad, se tiene en una hoja electrónica de cálculo de Excel, un conjunto de datos que contiene la información sobre la edad, la estatura, el sexo, la puntuación y la calificación de 30 estudiantes de cuarto semestre. De forma siguiente:



	A	B	C	D	E	F	G
1	Nombre	Edad	Estatura	Sexo	Puntuación	Calificación	
2	Maria B	18	1.58	Mujer	80	Notable	
3	Antonia C	21	1.75	Mujer	85	Notable	
4	Juan B	19	1.68	Hombre	73	Buena	
5	Rosa G	22	1.56	Mujer	65	Suficiente	
6	Juan P	17	1.69	Hombre	84	Notable	
7	Humberto J	25	1.72	Hombre	97	Sobresaliente	
8	Julian G	18	1.78	Hombre	76	Buena	
9	Julia S	19	1.76	Mujer	60	Suspenso	
10	Rodrigo H	23	1.60	Hombre	78	Buena	
11	Sofia W	22	1.69	Mujer	95	Muy Buena	
12	Elisa D	20	1.60	Mujer	74	Buena	
13	Maria J	21	1.75	Mujer	64	Suspenso	
14	Heriberto K	17	1.80	Hombre	80	Notable	
15	Carmen T	19	1.67	Mujer	81	Notable	
16	Felicia U	19	1.71	Mujer	83	Notable	
17	Rosa T	23	1.58	Mujer	45	Suspenso	
18	Victoria H	17	1.68	Mujer	94	Muy Buena	
19	Fabiola M	22	1.73	Mujer	94	Muy Buena	
20	Vicente L	20	1.70	Hombre	78	Buena	
21	Jimena P	19	1.65	Mujer	84	Notable	
22	Plutarco U	24	1.74	Hombre	82	Notable	
23	Ramiro S	17	1.70	Hombre	80	Notable	
24	Cecilia O	16	1.68	Mujer	74	Buena	
25	Omar T	17	1.69	Hombre	93	Muy Buena	
26	Odondina R	21	1.67	Mujer	97	Sobresaliente	
27	Felipe G	20	1.70	Hombre	82	Notable	
28	Fatima H	19	1.72	Mujer	60	Suspenso	
29	Felix T	18	1.76	Hombre	60	Suspenso	
30	Tomas J	18	1.71	Hombre	40	Suspenso	
31	Ximena H	21	1.71	Mujer	70	Buena	
32							

Fig. 3 Libro de Excel con los datos de la Facultad

- 1) Crear el archivo en Excel, llamarle “Ejemplo2.xlsx” y guardarlo en una carpeta.
- 2) Cargar el archivo de Excel “Ejemplo2.xlsx” en RStudio.

Para cargar el archivo en RStudio, se puede realizar ya sea desde la función “read.table()” o desde el menú de RStudio: “File/Import Dataset/ From Excel...” o por medio de la ventana de Medio ambiente pulsando el icono de “Import Dataset”. Hay que cargar en memoria la librería de Excel “xlsx” del paquete que se descargó en la unidad anterior.

```
> library(readxl)
```

```
> Datos_Facultad <- read_excel("datos/Ejemplo2.xlsx")
```

Con esta instrucción se crea un objeto “Datos_Facultad”, que contiene a la tabla

```
> attach(Datos_Facultad)
```

Con el menú o la ventana de medio ambiente, al importar se presenta la pantalla

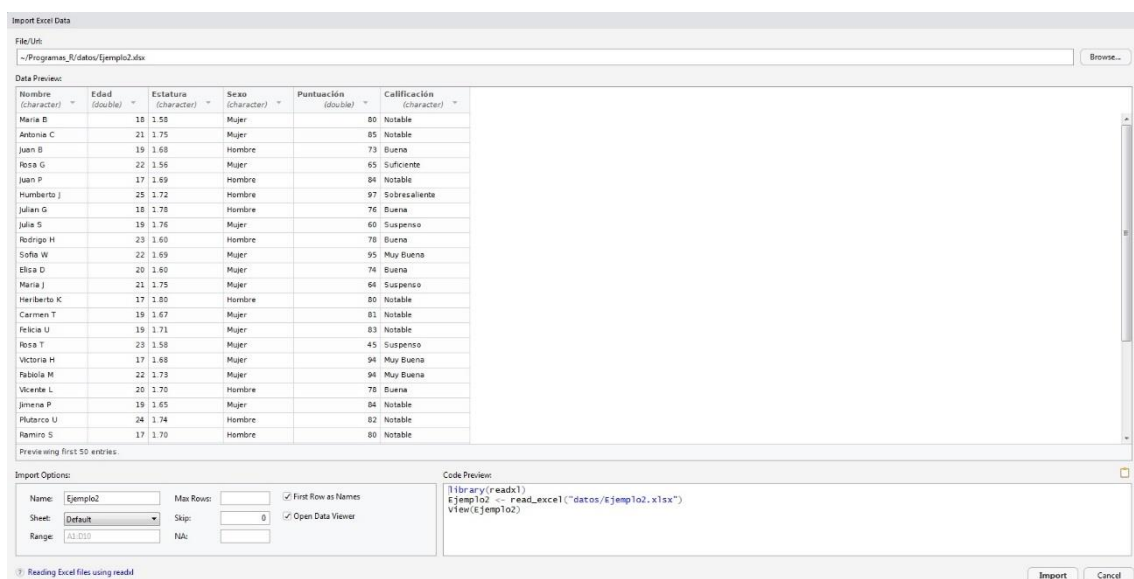


Fig. 4 Pantalla de Import Dataset

Al dar clic en “Import”, se importa la tabla, y se convierte en un objeto con el nombre original “Ejemplo2”. Se pueden importar los datos de las dos formas, así trabajar con cualquiera de los dos objetos creados.

- 3) Una vez importado el “Dataset” del Ejemplo2: Realizar un análisis estadístico descriptivo de los datos, a través de los siguientes puntos:
 - a) Comprobar el modo y la clase de “Ejemplo2”

```
> mode(Ejemplo2)
[1] "list"
```

```
> class(Ejemplo2)
[1] "tbl_df"      "tbl"        "data.frame"
```

- b) Crear una tabla para las frecuencias absolutas y otra para relativas para las variables: Sexo y Calificación y comprobar su contenido

Frecuencias Absolutas

```
> Frec_abs_Calif <- table(Ejemplo2$Calificación)
```

```
> Frec_abs_Calif
```

	Buena	Muy Buena	Notable	Sobresaliente	Suficiente
Suspense	7	4	10	2	1

```
> Frec_abs_Sexo <- table(Ejemplo2$Sexo)
```

```
> Frec_abs_Sexo
```

Hombre	Mujer
13	17

Frecuencias Relativas

Para calcular la frecuencia relativa se utiliza la función “prop.table()”

```
> Frec_rel_Calif <- prop.table(Frec_abs_Calif)
```

```
> Frec_rel_Calif
```

	Buena	Muy Buena	Notable	Sobresaliente	Suficiente
Suspense	0.23333333	0.13333333	0.33333333	0.06666667	0.03333333

```
> Frec_rel_Sexo <- prop.table(Frec_abs_Sexo)
```

```
> Frec_rel_Sexo
```

Hombre	Mujer
0.4333333	0.5666667

- c) Presentar las variables *Calificación* y *Sexo*, mediante diagrama de sectores y diagramas de barras.

Diagrama de Sectores

Para el Diagrama de sectores, lo primero es crear los vectores con el contenido de las variables “Calificación” y “Sexo”.

```
> cal1 <- table(Ejemplo2$Calificación)
```

```
> sex1 <- table(Ejemplo2$Sexo)
```

Ahora recordemos que para incluir las cantidades en los sectores, se crea un vector de la tabla al que llamamos “Cantidad_Cal” para la variable Calificación y “Cantidad_Sex” para la variable “Sexo”, por medio de la función “round()”, se redondea los números en caso de haber decimales.

```
> Cantidad_Cal <- round(cal1)
```

```
> Cantidad_Sex <- round(sex1)
```

Ahora se crean los vectores etiquetas, en el que se juntan, el nombre de las etiquetas con las cantidades, por medio de la función “paste()”, se hace uso de “rownames()” para obtener el nombre de las filas de la tabla.

```
> Etiquetas_Cal <- paste(rownames(cal1), Cantidad_Cal)
```

```
> Etiquetas_Sex <- paste(rownames(sex1), Cantidad_Sex)
```

Ahora, se incluyen estas etiquetas a través de “labels” en las funciones de las gráficas.

Para la Variable Calificación

```
> pie(cal1, clockwise = TRUE, main = "Calificaciones de Curso", col =  
(1:20), labels = Etiquetas_Cal)
```

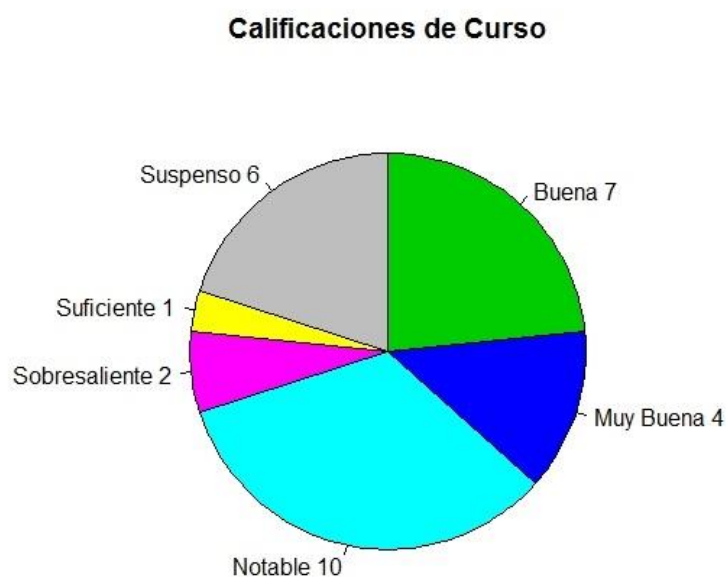


Fig. 5 Gráfico de sectores de la variable “Calificación”

Para la Variable “Sexo”

```
> pie(Sex1, clockwise = TRUE, main = "Sexo de Alumnos del Curso", col = c("steelblue4", "orangered"), labels = Etiquetas_Sex)
```

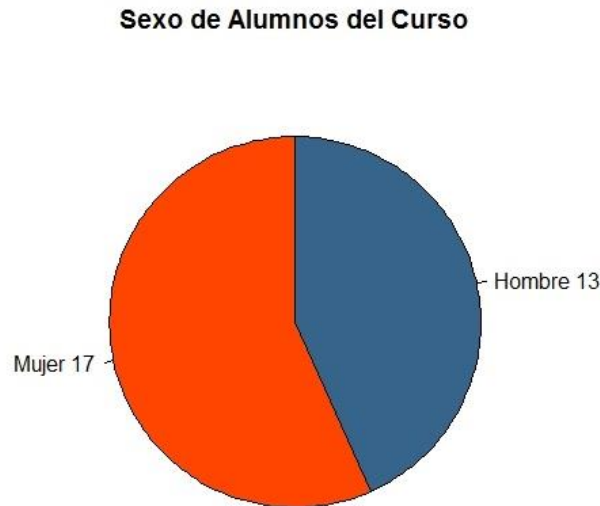


Fig. 6 Gráfico de sectores de la variable “Sexo”

Diagrama de Barras de la Variable Calificación

Al presentar un diagrama de barras, posiblemente el orden de las variables no esté en el que se desea que se presenten. Para esto se debe de crear un vector mediante la instrucción “factor()” en el que se estipulen los niveles mediante la función “levels()”, indicando en el vector, el orden a estipular. Para la variable Calificación, el orden de los niveles va desde el menor al mayor de la siguiente manera.

```
> Ejemplo2$Calificación <- factor(Ejemplo2$Calificación, levels = c('S  
uspense', 'Suficiente', 'Buena', 'Muy Buena', 'Notable', 'Sobresaliente'))
```

En base a este orden, hay que volver a crear el vector con el contenido de la variable “Calificación”

```
> call <- table(Ejemplo2$Calificación)  
> call
```

	Suspense	Suficiente	Buena	Muy Buena	Notable
Sobresaliente	6	1	7	4	10

Ahora creamos un vector llamado “Barras_Calif”, que contenga el gráfico

```
> Barras_Calif <- barplot(table(Ejemplo2$Calificación), space = 0, col =  
c(2:20), xlab="Calificaciones", ylab="Frecuencias absolutas", main="D  
iagrama de barras para la Variable Edad")
```

El argumento “space = 0”, es para que no exista espacio entre las barras

Para incluir las cantidades en las barras, se utiliza la función “text” con el vector anterior

```
> text(Barras_Calif[,1],Cal1,labels=Cal1, pos=3,cex=0.80)
```

El argumento “pos”, indica la posición del texto

El argumento “cex”, indica el tamaño de la letra o fuente.

El Gráfico que se presenta es el siguiente:

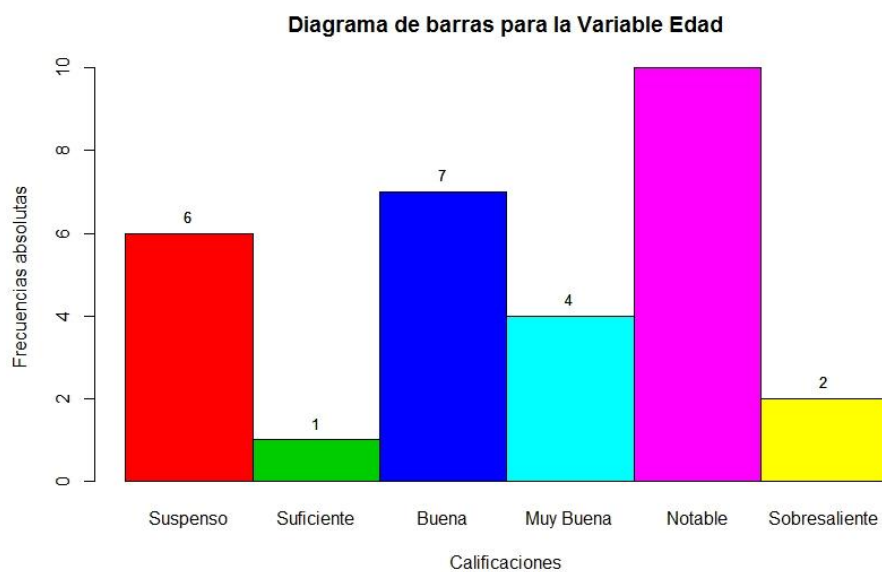


Fig. 7 Gráfico de Frecuencias Absolutas de la variable Calificación

Diagrama de Barras de la Variable Sexo

Similar al tratamiento de la anterior variable, se debe de incluir las etiquetas de las cantidades en las barras, tanto de Hombres como de Mujeres en el gráfico.

```
> Ejemplo2$Sexo <- factor(Ejemplo2$Sexo, levels = c('Hombre','Mujer'))
```

```
> sex2 <- table(Ejemplo2$Sexo)
```

```
> Barras_Sex <- barplot(table(Ejemplo2$Sexo), space = 0, col=c(21,71),  
xlab="Sexo", ylab="Frecuencias absolutas", main = "Diagrama de barras p  
ara la variable Sexo")
```

Y para el texto

```
> text(Barras_Sex[,1],sex2,labels=sex2, pos=3,cex=1)
```

El gráfico resultante es el siguiente

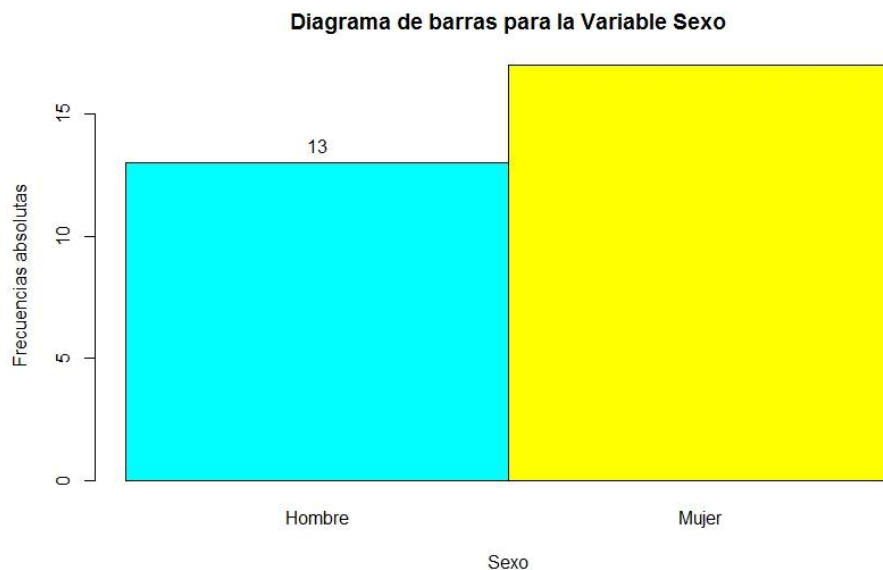


Fig. 7 Gráfico de Frecuencias Absolutas de la variable Sexo

Ahora si se necesita combinar dos variables, por ejemplo en este caso saber la calificación por sexos, tenemos que crear un vector que incluya ambas variables como el siguiente

```
> Comb1 <- table(Ejemplo2$Calificación,Ejemplo2$Sexo)
> Comb1
```

	Hombre	Mujer
Suspenso	2	4
Suficiente	0	1
Buena	4	3
Muy Buena	1	3
Notable	5	5
Sobresaliente	1	1

Para graficar este vector

```
> barplot(Comb1, beside = F, col = c(3:10), ylim = c(0,20)) # ylim estable el límite para el eje y
```

Y para colocar una leyenda encima del gráfico orientada arriba a la derecha, estableciendo el orden de los niveles

```
> legend("topright", title="Calificaciones", legend=levels(Ejemplo2$Calificación), fill=c(3:10), cex=.75, border=F, horiz=T, text.col="blue")
```

El gráfico que se genera es un gráfico de barras apiladas, esto lo realiza el argumento a Falso de “beside” de la forma siguiente:

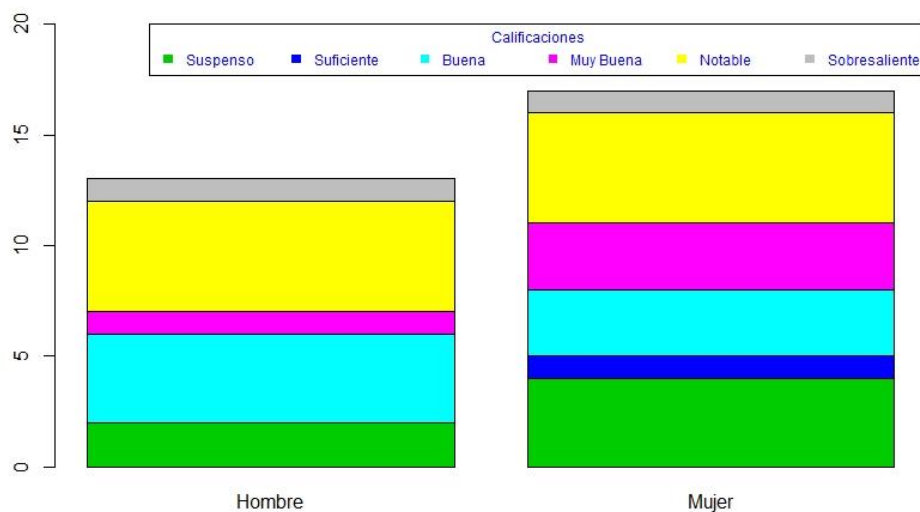


Fig. 8 Gráfico de Frecuencias Absolutas apilado de Calificación x Sexo

En el caso de presentar la gráfica no apilada de la calificación por hombres y mujeres, el argumento “beside” se coloca a verdadero y tenemos lo siguiente:

```
> barplot(Comb1, beside = T, col=rainbow(3), main="Calificaciones por
Sexo", xlab= "Sexo", ylab= "Cantidad", legend=levels(Ejemplo2$Calificac
ión), names.arg =levels(Ejemplo2$Sexo), ylim=c(0,10))
```

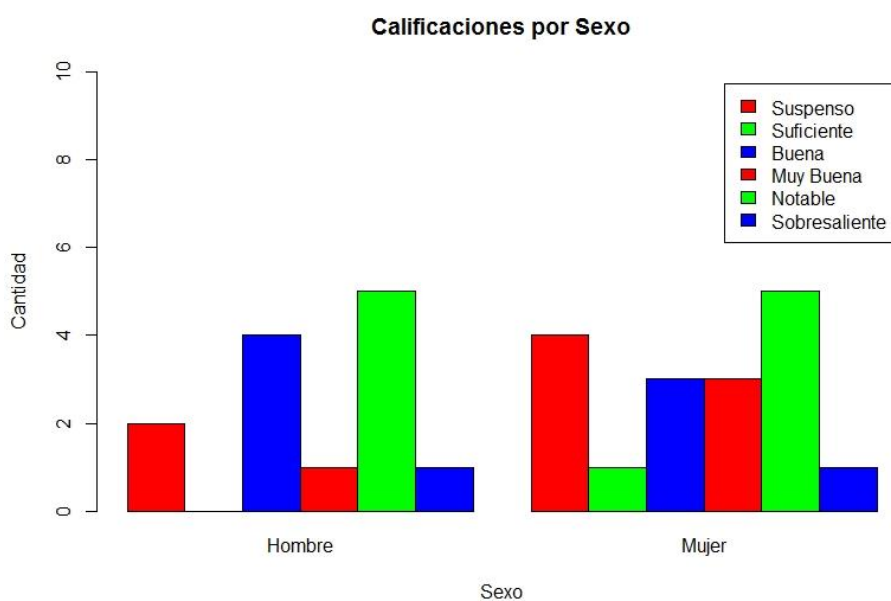


Fig. 8 Gráfico de Frecuencias Absolutas no apilado de Calificación x Sexo

- d) Revisión de los modos de las variables y cambio si es necesario

Al trabajar con tablas, probablemente nos encontremos con que determinadas variables no son del modo que se necesitan, por ejemplo: Hemos querido que la variable “Estatura” se presente como carácter y no como debería de ser de modo numérico. Esto se podría comprobar con la función “mode()”

```
> mode(Ejemplo2$Estatura)
[1] "character"
```

Esto traería un problema a la hora de trabajar con los datos. Para evitar esto y convertir el modo de una variable se utiliza la función “as.numeric()”, que lo que hace es convertir a numérico los valores datos como argumento. Así por ejemplo:

```
> Ejemplo2$Estatura <- as.numeric(Ejemplo2$Estatura)
```

Cambiaría el modo de la variable. Si se comprueba de nuevo, el resultado es diferente.

```
> mode(Ejemplo2$Estatura)
[1] "numeric"
```

Así los datos de una tabla o dataframe, pueden ser incluso modificados en su modo, de acuerdo a las necesidades.

- e) Realizar el análisis estadístico de las Variables Estatura y Edad con las Medidas de Posición, Relativas y de Forma:

Medidas de Posición

- i) Medidas de Tendencia Central (Media y Mediana)

```
> (media_Aritmetica_Esta = mean(Ejemplo2$Estatura)) # Cálculo de la M
edia Aritmética de la Estatura
[1] 1.692
```

```
> (media_Aritmetica_Edad = round(mean(Ejemplo2$Edad))) # Cálculo
de la Media Aritmética de la Edad
[1] 20
```

```
> (Mediana_Estatura = median(Ejemplo2$Estatura)) # Cálculo
de la Mediana de la Estatura
[1] 1.7
```

```
> (Mediana_Edad = median(Ejemplo2$Edad)) # Cálculo de la M
ediana de la Edad
[1] 19
```

- ii) Medidas de posición de Tendencia no central (los cuantiles)

```
> quantile(Ejemplo2$Estatura, probs = seq(0, 1, 0.25), na.rm = FALSE)
 0%   25%   50%   75%  100%
1.5600 1.6725 1.7000 1.7275 1.8000
```

```
> quantile(Ejemplo2$Edad, probs = seq(0, 1, 0.25), na.rm = FALSE)
 0%  25%  50%  75% 100%

```

16 18 19 21 25

iii) Cálculo de las Medidas de Dispersión (varianza, desviación estándar e IQR)

```
> (VarianzaMuestra1_Estatura = var(Ejemplo2$Estatura))      # cálculo  
de la Cuasi-Varianza de la Estatura  
[1] 0.003602759
```

```
> (VarianzaMuestra1_Edad = var(Ejemplo2$Edad))              # cálculo de la C  
uasi-Varianza de la Edad  
[1] 5.374713
```

```
> (Varianza_Estatura <- 29/30 * var(Ejemplo2$Estatura))     # cálculo  
de la Varianza de la Estatura  
[1] 0.003482667
```

```
> (Varianza_Edad <- 29/30 * var(Ejemplo2$Edad))             # cálculo de la v  
arianza de la Edad  
[1] 5.195556
```

```
> (DesvTipica_Altura = sd(Ejemplo2$Estatura)) # Cálculo de la cuasi-De  
sv. Típica de la Estatura  
[1] 0.06002298
```

```
> (DesvTipica_Edad = sd(Ejemplo2$Edad))      # cálculo de la cuasi-De  
sv. Típica de la Edad  
[1] 2.318343
```

```
> (Desvt_Estatura <- sqrt(Varianza_Estatura)) # Cálculo de la Desviac  
ión Estándar de la Estatura  
[1] 0.05901412
```

```
> (Desvt_Edad <- sqrt(Varianza_Edad)) # Cálculo de la Desviación Está  
ndar de la Edad  
[1] 2.279376
```

```
> IQR(Ejemplo2$Estatura)      # cálculo del Rango Intercuartílico de la  
Estatura  
[1] 0.055
```

```
> IQR(Ejemplo2$Edad)      # cálculo del Rango Intercuartílico de la Edad  
[1] 3
```

f) Realizar un resumen estadístico de las variables Estatura y Edad

```
> summary(Ejemplo2$Estatura)      # Resumen estadístico de la variabl  
e Estatura  
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.   
 1.560  1.673   1.700   1.692  1.728   1.800
```

```
> summary(Ejemplo2$Edad)          # Resumen estadístico de la variable Ed  
ad  
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.   
 16.00  18.00   19.00   19.73  21.00   25.00
```

Pregunta: ¿Coinciden estos valores con los del cálculo individual? R: Si

g) Indicar que variable entre la Altura y la Edad es más homogénea.

Para determinar la homogeneidad de una variable (o, lo que es lo mismo, la representatividad de su media), se calcula el llamado “Coeficiente de Variación de Pearson”, para cada una de ellas, el cual se define como el cociente entre la desviación estándar y la media de la variable. Al haber calculado estos valores, procedemos a verificar cuál de estas variables es más homogénea.

El coeficiente de variación de la variable Estatura es:

```
> (Coef_Variacion_Altura <- Desvt_Estatura/media_Aritmetica_Esta) #  
Coeficiente Variación Estatura  
[1] 0.03487832
```

El coeficiente de variación de la variable Edad es:

```
> (Coef_Variacion_Edad <- Desvt_Edad/media_Aritmetica_Edad) # Coeficie  
nte Variación Edad  
[1] 0.1139688
```

Por lo tanto, la variable más homogénea es la Altura, debido a que presenta un coeficiente de variación más cercano a cero.

3) Datos Bidimensionales

Los datos bidimensionales se encuentran conformados por pares de datos (x,y), para realizar una representación gráfica de ellos se utiliza el “Diagrama de dispersión”, conocido también por “Nube de puntos”, que consiste en representar en un sistema de ejes coordenados de dos dimensiones tantos puntos como datos, asignando a cada dato (x_i , y_j) el punto de coordenadas (x_i , y_j). En R la representación gráfica se establece a través de la función “plot()”. Analizando alguna de sus múltiples posibilidades, así como el representar sobre dicha gráfica la recta de regresión o mínimos cuadrados.

Una de las principales características que permite observar la bondad del ajuste de esta recta a los datos que forman la nube de puntos es el llamado “Coeficiente de correlación lineal de Pearson (r)”, el cual se calcula con la función “cor()”, que se verá más adelante.

- Tablas de datos

En todos los casos de datos bidimensionales los datos vendrán en forma de tabla de doble entrada donde los valores de las dos variables definen las filas y las columnas, aglomerando en dicha tabla el número de elementos de la muestra que representan a la vez un valor y otro de ambas variables.

- Ajustes por mínimos cuadrados

Un elemento asociado a los datos bidimensionales es la denominada recta de mínimos cuadrados o de regresión. En R se puede obtener fácilmente al utilizar la función “lm(linear models)”. La recta permite minimizar las diferencias entre los valores observados y los teóricos que produce, haciendo la recta más próxima a la nube de puntos. El coeficiente de determinación (es decir, el coeficiente de correlación al cuadrado), mide la bondad del ajuste de la recta a los datos.

Ejemplo3:

Tras preguntar a 18 docentes de un instituto sobre las horas que impartían en los cursos de un período escolar y las horas semanales que en término medio destinaban a la preparación de los cursos se obtuvieron los siguientes resultados.

Horas Docentes:	30	25	15	21	28	43	32	50	19
Horas PrepClases:	15.6	9.8	13	14.5	16.4	18.3	17.2	16.5	14.8
Horas Docentes:	18	23	21	17	20	25	14	31	19
Horas PrepClases:	19.4	21.8	14.7	15	13.2	12.4	9.7	11.5	9.6

Se requiere analizar estos pares de datos (x,y) por medio de una representación gráfica, obtener la recta de mínimos cuadrados.

- a) Se debe crear el par de datos (x,y), por lo tanto, lo primero es crear los vectores de entrada de los datos, para la variable (x = Horas Docentes) y la variable (y = Horas PrepClases)

```
> XHoraD <- c(30,25,15,21,28,43,32,50,19,18,23,21,17,20,25,14,31,19)
> YHoraP <- c(15.6,9.8,13,14.5,16.4,18.3,17.2,16.5,14.8,19.4,21.8,14.7,
,15,13.2,12.4,9.7,11.5,9.6)
```

La representación gráfica de la nube de puntos por medio de la función “plot()” es la siguiente:

```
> plot(XHoraD,YHoraP,main = "Horas Trabajadas y Horas Preparación Doce
nte", pch =c(9,15), col = c("blue","red"), xlab ="Horas Clase Docente"
,ylab ="Horas Preparación Clase")
```

Para calcular la recta de mínimos cuadrados o de regresión se utiliza la función “lm(y ~x)” entre los valores de X y de Y de la forma siguiente:

```
> Recta_ajuste <- lm(YHoraP ~ XHoraD) # lm(y ~x): El signo de equival
encia ~ = Alt + 126
```

El vector-objeto “Recta_ajuste” es una lista que contiene toda la información relevante sobre el análisis estadístico.

Para visualizar la recta de mínimos cuadrados calculada tecleamos:

```
> Recta_ajuste
```

```
Call:
lm(formula = YHoraP ~ XHoraD)
```

```

Coefficients:
(Intercept)      XHorad
      11.7185         0.1163

```

Se observa que la ordenada en el origen es 11.72 y la pendiente 0.12

Por lo que la ecuación de la recta de ajuste será:

$$\text{Horas de preparación} = 11.72 + 0.12 \text{ Horas docencia}$$

También, se puede obtener un resumen de los principales resultados del análisis, al utilizar como se ha visto, la función “summary()”

```
> summary(Recta_ajuste)
```

```

Call:
lm(formula = YHoraP ~ XHorad)

```

```

Residuals:
    Min       1Q   Median       3Q      Max
-4.827 -1.929  0.365   1.394   7.406

```

```

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  11.71851     2.24524   5.219 8.43e-05 ***
XHorad        0.11633     0.08408   1.384  0.185
---

```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```

Residual standard error: 3.294 on 16 degrees of freedom
Multiple R-squared:  0.1069,    Adjusted R-squared:  0.05104
F-statistic: 1.914 on 1 and 16 DF,  p-value: 0.1855

```

Para añadir esta recta al gráfico se utiliza la función “abline()”. La función “abline()”, permite dibujar líneas a partir de su intercepto y pendiente. Existen dos parámetros además del color que se pueden manipular, estos son: “lty” que controla el tipo de línea y “lwd” que controla el ancho de línea y que debe de ser un número mayor a cero en un intervalo de 1 a 8. Los tipos de línea se representan en la siguiente imagen:

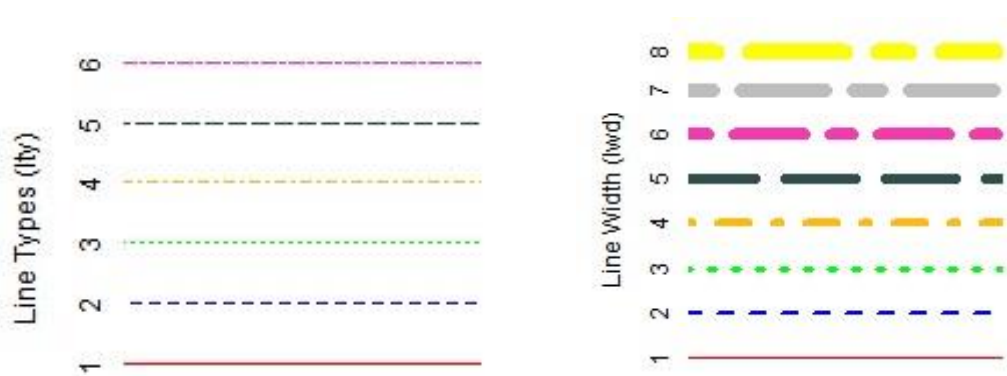


Fig. 9 Parámetros “lty”, y “lwd” de abline

```
> abline(Recta_ajuste, lty = 2, lwd = 2, col = "red")
```

Se observa que se ha elegido un tipo de línea 2 que es punteada y un grosor de 2, con un color en rojo.

Si se desea colocar una leyenda sobre el gráfico, recordar que hay que utilizar la función “legend()” de la siguiente forma

```
> legend(26.8,21.8, c("Recta de Mínimos Cuadrados"),lty=c(2), lwd=c(2), col= "red")
```

Los primeros dos parámetros son las coordenadas (x,y) donde se presentará la caja de la leyenda, el siguiente es el título a colocar y los parámetros “lty”, “lwd” y color deben de ser los mismos que en “abline()”.

El resultado final del gráfico al presentar la función “plot()”, más “abline()” y “legend()” es el siguiente.

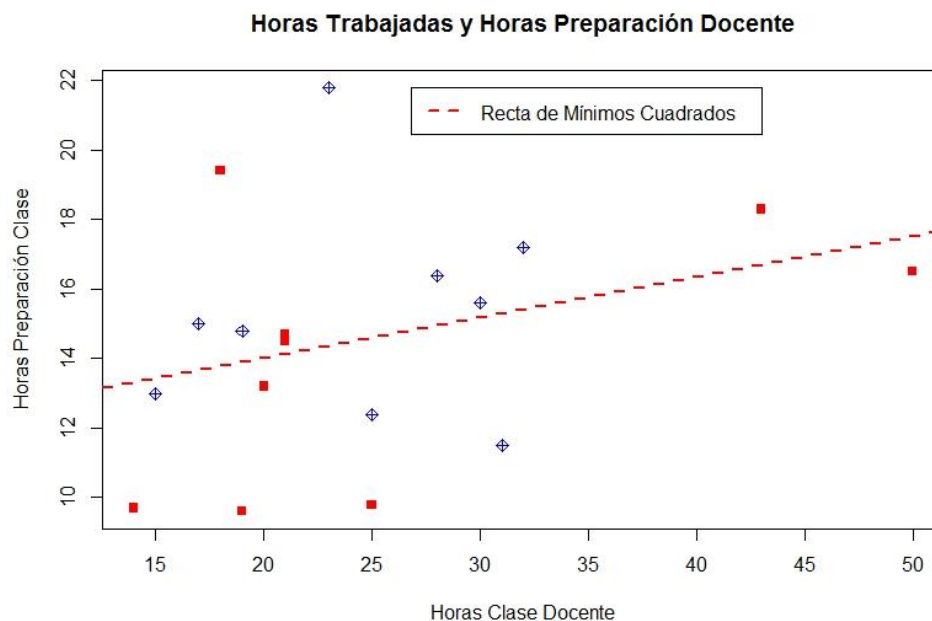


Fig. 10 Gráfico de Horas trabajadas x Horas preparación, con Recta de mínimos cuadrados

Recta de tendencia no lineal, función de suavizado

Otra opción es dibujar una línea que se ajuste a los datos, pero denominada de tendencia no lineal, es decir, no forzar que sea una línea recta, para realizar este ajuste se utiliza la función “supsmu()”. El parámetro “bass” controla el suavizado de la recta, valores de 0 hasta 10 indican una suavidad creciente.

La expresión a considerar para una línea de suavizado en la gráfica anterior es la siguiente:

```
> lines(supsmu(XHoraD, YHoraP, bass=2), col="blue") # de 0 a 10, controla el nivel de "smoothness o suavizado" de la línea
```

Para la leyenda del suavizado, la expresión es la siguiente:

```
> legend(26.8,19.8, c("Recta de Suavizado"),lty=c(1), lwd=c(1), col= "blue")
```

Hay que recordar que además de colocar coordenadas, la leyenda se puede ubicar en: "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" y "center". El gráfico resultante se observa en la siguiente imagen.

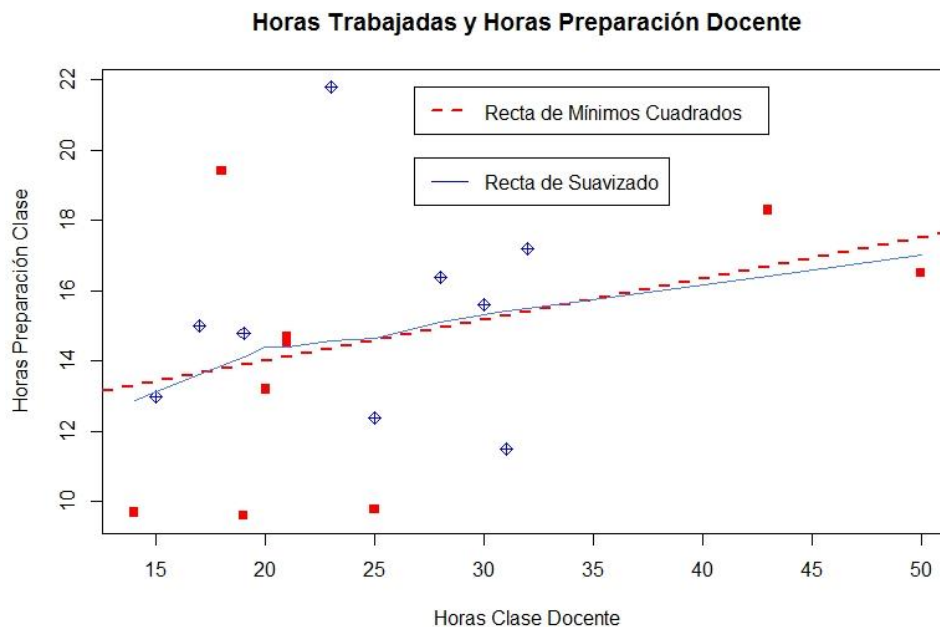


Fig. 11 Gráfico de Horas trabajados x Horas preparación, con Recta de mínimos cuadrados y una línea de tendencia no lineal

- Precisión del ajuste por mínimos cuadrados

Como se observa en la gráfica, la nube de puntos parece menos concentrada alrededor de su recta de ajuste, lo que llevaría a deducir que las horas de preparación de clase que se obtendrían para preparar 60 horas de docencia, no sería muy fiable. La causa de esta falta de concentración de los valores observados alrededor de la recta puede deberse a que ambas variables no se encuentren relacionadas linealmente, lo que lleva a deducir que para este tipo de dato se podría ajustar a otro tipo de función.

Se necesita entonces, un valor que proporcione una medida lo más próxima que está la función que se ha ajustado (sea o no una recta) a la nube de puntos de los datos; es decir, una medida de la bondad del ajuste. A este valor se le conoce con el nombre de “Varianza Residual”.

Aunque al comparar el ajuste de los datos de dos funciones se puede utilizar la variación residual, eligiendo la que presente una varianza menor, es conveniente utilizar otro valor que permita decidir si un ajuste es o no adecuado en sí mismo porque pueda que uno sea mejor que otro o que ambos sean malos. En este punto es donde se presenta el concepto de “Coeficiente de Determinación (R^2)”. Este coeficiente se encuentra como se sabe comprendido entre 0 y 1, en donde un buen ajuste se dará en los casos en donde dicho coeficiente se encuentre cercano a 1 y de un mal ajuste en los que esté cercano a cero. La valoración de lo que se considere como cercano o lejos, es tema de la “Inferencia Estadística”.

Un valor que se relacione con el ajuste a una recta es el “Coeficiente de Correlación de Pearson (r)”. Este coeficiente toma valores entre (-1 y 1), siendo $R^2 = (r)^2$, para calcular este valor en R se utiliza la función “cor()”.

Ejemplo4:

Calcular el coeficiente de correlación lineal de Pearson para los datos del ejemplo 3.

Para realizar este cálculo, se toman los datos de los vectores de entrada (x,y) en este caso “XHorad” y “YHorap”, como parámetros en la función “cor()”.

De esta forma el cálculo del coeficiente de correlación de Pearson “r”, se realiza así:

```
> cor(XHorad,YHorap)
[1] 0.3268932
```

Y para obtener el Coeficiente de Determinación R^2 , el cálculo es

```
> cor(XHorad,YHorap)^2
[1] 0.1068592
```

Por lo que se determina que el Coeficiente de determinación es: $R^2 = 0.1068592$

Como conclusión se puede decir que no es un buen ajuste, pues la recta determinada permite explicar las horas de planificación mediante las horas de docencia con solo un 10.7% de fiabilidad.

Ejemplo5:

Los siguientes datos corresponden a un estudio realizado en una guardería infantil, en el que se midió el tamaño del vocabulario, es decir, el número de palabras que manejaban los niños de edades comprendidas de 1 a 6 años (Estudio realizado por Weiner¹ en 1977).

Edad	1	1.5	2	2.5	3	3.5	4	4.5	5	6
No. Palabras	3	22	272	446	896	1222	1540	1870	2072	2562

Se pide determinar la recta de regresión, la de suavizado y analizar la bondad del ajuste con el Coeficiente de Determinación:

Lo primero es construir los vectores que se utilizarán en la construcción de diagrama y el análisis, para posteriormente realizar el cálculo de la recta de mínimos, graficar y añadir las rectas de regresión y suavizado, para finalizar realizando el análisis de la bondad del ajuste.

La secuencia de los cálculos es la siguiente:

```
> VEdad <- c(1,1.5,2,2.5,3,3.5,4,4.5,5,6)
> VPalabras <- c(3,22,272,446,896,1222,1540,1870,2070,2562)
```

¹ Weiner, B. (1977). Discovering Psychology, Chicago: Science Research Association, 97

```
> RectaRegresión <- lm(VPalabras~VEdad) # Cálculo de la Recta de Regresión
> RectaRegresión
```

```
Call:
lm(formula = VPalabras ~ VEdad)
```

```
Coefficients:
(Intercept)      VEdad
    -763.6         561.8
```

Se observa que la ordenada en el origen es -763.6 y la pendiente 561.8

Por lo que la ecuación de la recta de regresión será:

$$\text{Número de palabras} = -763.6 + 561.8 \text{ Edad}$$

La representación gráfica y las líneas de regresión y suavizado se calculan con las siguientes expresiones:

```
> plot(VEdad,VPalabras, pch = 17, xlab="Edad de los niños", ylab="Número de Palabras", main = "Tamaño Vocabulario, niños de 1 a 6 años", col = "red")

> abline(RectaRegresión, col = "blue", lty= 1, lwd =2, )

> legend("topleft", c("Recta de Mínimos Cuadrados"),lty=c(1), lwd=c(2), col= "Blue")

> lines(supsmu(VEdad, VPalabras, bass=10), lty=c(2), lwd=c(2), col="red") # de 0 a 10, controla el nivel de "smoothness o suavizado" de la línea

> legend("left", c("Recta de Suavizado"),lty=c(2), lwd=c(2), col= "red")
```

La gráfica resultante es la siguiente:

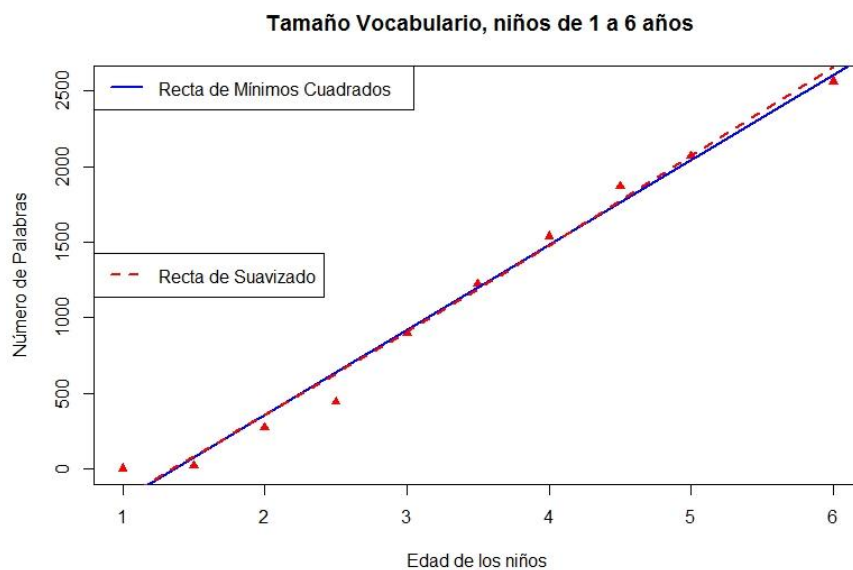


Fig. 12 Gráfico de dispersión y recta de regresión y suavizado

Para analizar la bondad del ajuste mediante el Coeficiente de Determinación, tenemos:

```
> cor(vEdad,vPalabras)^2      # cálculo del coeficiente de Determinación
[1] 0.985278
```

El valor resultante (0.985278) parece indicar un buen ajuste, debido a que la recta determinada permite explicar el número de palabras mediante la Edad con un 98.5% de fiabilidad.

Ejemplos con tablas de doble entrada.

En ocasiones, los datos bidimensionales se encuentran contenidos en tablas de doble entrada. Debido a que las variables toman unos pocos valores distintos y el número total de datos es muy superior.

Ejemplo 6:

Se han clasificado un grupo de 176 profesores de una región de acuerdo a su sexo y estado civil, obteniéndose la siguiente tabla.

Estado Civil Sexo	Soltero	Casado	Viudo	Sep/Div
Masculino	20	40	5	11
Femenino	29	38	11	20

Este tipo de datos como se observa, no pueden ser tratados como un vector, sino que tienen que ser tratados mediante una matriz, por lo que hay que construir la Matriz de los profesores de acuerdo a la siguiente expresión.

```
> Profesores <- matrix(c(20,29,40,38,5,11,11,20), ncol = 4, dimnames =  
list(c("Masculino","Femenino"), c("Soltero","Casado","Viudo", "Sep/Div"  
)))
```

Para comprobar el contenido de la matriz:

```
> Profesores
      Soltero Casado Viudo Sep/Div
Masculino    20    40     5     11
Femenino     29    38    11     20

> mode(Profesores)
[1] "numeric"

> class(Profesores)
[1] "matrix"
```

Ha sido correctamente creada, con el tipo y clase de datos apropiado.

Ahora para realizar el análisis de los datos, de esta distribución bidimensional de frecuencias absolutas se puede obtener con mucha facilidad la distribución bidimensional de frecuencias relativas mediante la función “prop.table()”, vista anteriormente.

```
> prop.table(Profesores)
      Soltero   Casado   Viudo   Sep/Div
Masculino 0.1149425 0.2298851 0.02873563 0.06321839
Femenino  0.1666667 0.2183908 0.06321839 0.11494253
```

Ahora bien, las distribuciones marginales (absolutas y relativas) se calculan mediante la función “margin.table()”. En donde el primer argumento de la función representa la tabla de doble entrada, en este caso la matriz y el segundo sirve para indicar si se desea que se represente por filas (colocando un número 1) o si se quiere que se represente por columnas (marcando un número 2). Veamos esto a continuación.

```
> margin.table(Profesores,1)      # Presentado por filas (Masculino y Femenino)
Masculino  Femenino
      76      98
```

```
> margin.table(Profesores,2)      # Presentado por Columnas (Soltero, Casado, Viudo, Sep/Div)
Soltero  Casado  Viudo Sep/Div
      49      78      16      31
```

Por lo tanto, las distribuciones marginales relativas de los datos se pueden obtener de dos maneras: Obteniendo primero las frecuencias relativas y luego las marginales o por el contrario, obteniendo primero las marginales y posteriormente las relativas.

```
> margin.table(prop.table(Profesores),1)
Masculino  Femenino
0.4367816 0.5632184

> margin.table(prop.table(Profesores),2)
      Soltero   Casado   Viudo   Sep/Div
0.28160920 0.44827586 0.09195402 0.17816092
```

Como las variables que forman la tabla son de tipo cualitativo y de tipo cuantitativo pero, en este caso, con valores aislados, la representación gráfica de los datos que es la más adecuada sería el diagrama de barras o rectángulos, que en R se representa mediante la función “barplot()”. La función “barplot()” trata de una matriz a representar y construye un diagrama para cada uno de los valores de las filas, agrupados por los valores de la variable que aparece en las columnas de la matriz. También, se puede elegir que los represente sobrepuestos unos a otros (por defecto), o que los haga pegados, en este caso se utiliza el argumento “beside=T”.

De acuerdo a la matriz de frecuencias que se utilice (absolutas o relativas), el gráfico será de unas u otras.

a) Opción de barras pegadas

La expresión para construir el diagrama es la siguiente:

```
> barplot(Profesores, beside=T, col=c("red","blue"), xlab ="Estado Civil", ylab ="Cantidad", main ="Clasificación Profesores", legend.text = T)
```

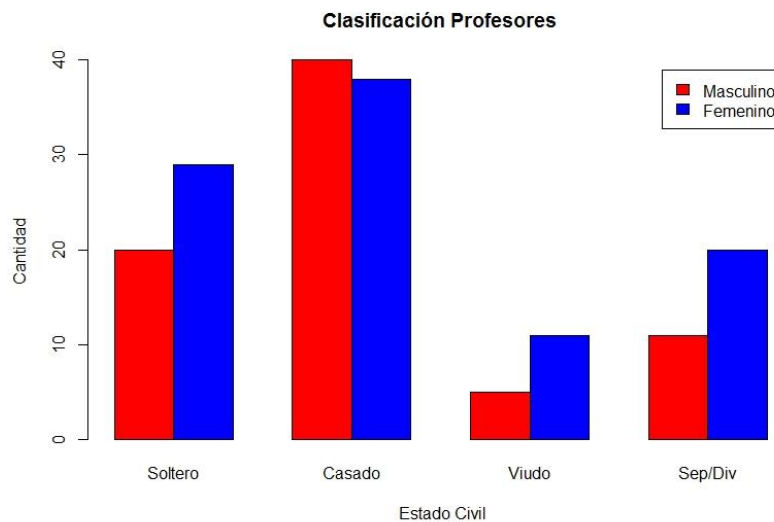


Fig. 13 Gráfico de barras pegadas

b) Opción Barras sobrepuestas

```
> barplot(Profesores, beside=F, col=c("red","blue"), xlab ="Estado Civil", ylab ="Cantidad", main ="Clasificación Profesores", legend.text = T)
```

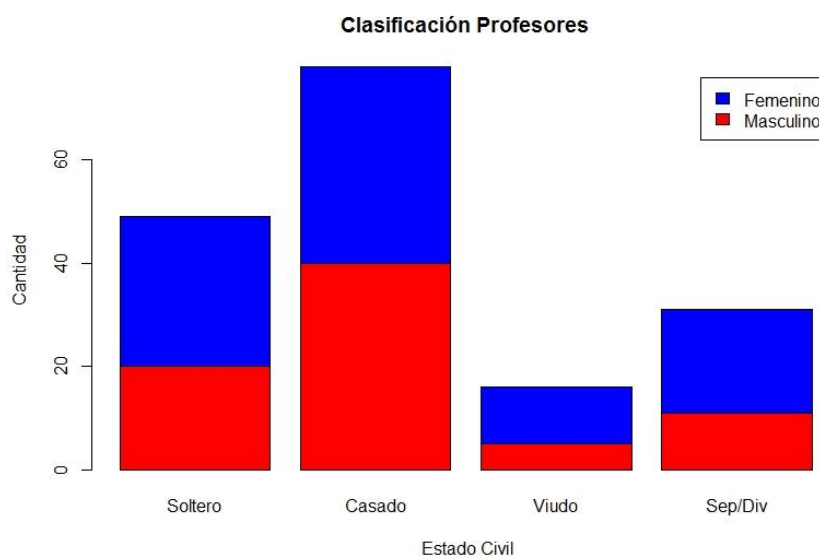


Fig. 14 Gráfico de barras sobrepuestas

Si lo que se quiere representar son los valores de las columnas agrupados por los valores de la variable que contiene los valores de las filas, se debe de transponer la matriz que se está utilizando como primer argumento de la función “plot()”

Para transponer la matriz se utiliza

```
> prop.table(t(Profesores)) # La “t” indica transponer la matriz
```

Esta instrucción se coloca directamente dentro de la expresión de la función “plot()”, quedando de la siguiente manera:

```
> barplot(prop.table(t(Profesores)), beside=T, col=c("red","blue"), xlab="Sexo", ylab="Cantidad", main="Clasificación Profesores", legend.text = T)
```

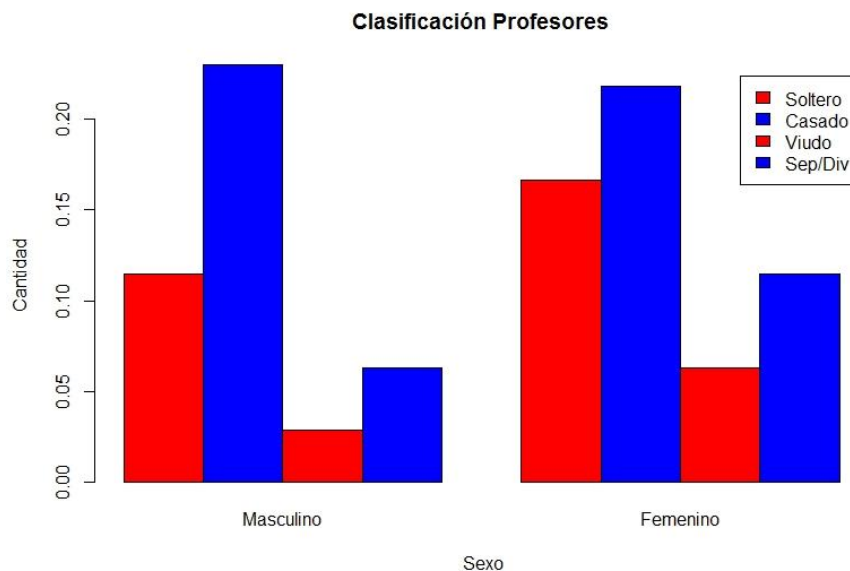


Fig. 15 Gráfico de barras de la matriz traspuesta

Adicionalmente, también se puede dividir la ventana de gráficos para representar más de un gráfico, mediante la función “par(mfrow=c(a,b))”, en donde se permite dividir la pantalla gráfica en una matriz de gráficos de dimensión “a x b”.

```
> par(mfrow =c(1,2)) # Divide la ventana de gráficos en dimensión 1 x 2
> barplot(prop.table(t(Profesores),2), beside=T, col=c("red","blue"), xlab="Sexo", ylab="Cantidad", main="Clasificación Profesores", legend.text = T)
> barplot(prop.table((Profesores),2), beside=T, col=c("red","blue"), xlab="Sexo", ylab="Cantidad", main="Clasificación Profesores", legend.text = T)
```

Se divide la pantalla en una dimensión (1 x 2), el primer diagrama será la matriz traspuesta y el segundo la normal. Para recuperar la pantalla a su forma normal se debe de ejecutar la instrucción.

```
> par(mfrow=c(1,1))
```

La gráfica que resulta de la división de la pantalla es la siguiente:

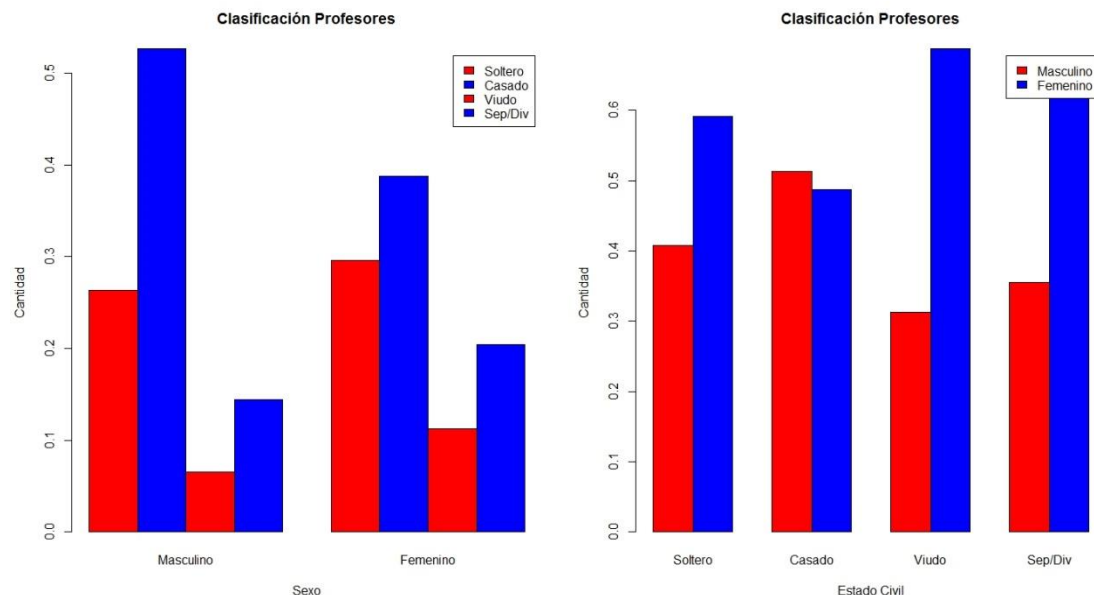


Fig. 16 Gráfico de barras Combinadas

- Gráficos dentro de gráficos

Una de las opciones más útiles para insertar gráficos dentro de otros gráficos es utilizar la función “subplot()” que está contenida en el paquete “Hmisc”, de Harrell², 2012.

Para poder utilizar este paquete, hay que cargarlo primero, por cualquiera de las opciones que R y RSudio permiten. Aquí lo vamos a hacer de la siguiente manera.

```
> library(Hmisc)
Loading required package: lattice
Loading required package: survival
Loading required package: Formula
Loading required package: ggplot2

Attaching package: 'Hmisc'

The following objects are masked from 'package:base':

    format.pval, units
```

² Harrell FE Jr. (2012) Package ‘Hmisc’. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria.

Ejemplo 7:

Vamos a trabajar con los datos de un archivo externo llamado “Temperatura.csv”, y que corresponden a datos de las temperaturas en un lago tomadas cada media hora durante varios meses. Por lo tanto, el siguiente paso es leer los datos desde el archivo con la función “read()”.

```
> Datos_Temp <- read.csv2(file = "./datos/Temperatura.csv", header = TRUE, encoding="latin1")
```

Mediante el argumento “encoding” se estipula que el tipo codificado de carácter a leer será “latin1”, que es el más usual, esto para evitar algún posible error de lectura en los caracteres.

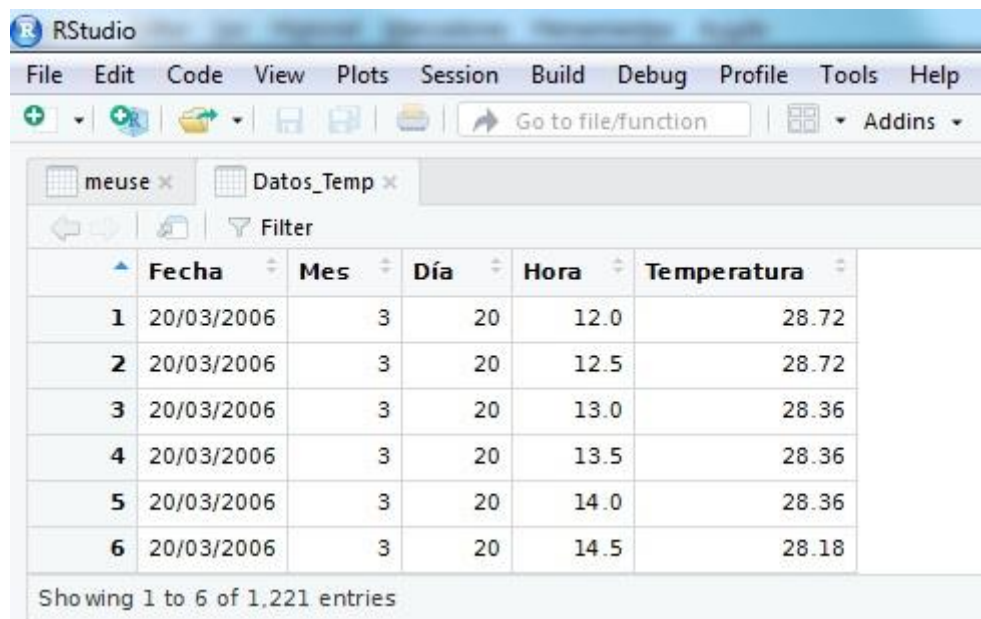
En la lectura de archivos es útil utilizar la función “na.exclude”, porque permite tratar con datos de tipo “NA”, o datos vacíos, lo que ocurre cuando en alguna celda del archivo de datos está se encuentre vacía. Por lo tanto, para evitar que se envíe algún error si a la lectura de los datos existe este tipo de situación se utiliza dicha función.

```
> Datos_Temp<-na.exclude(Datos_Temp)
```

Se observa que se ha incluido la función para prevenir en la nueva asignación del archivo. Ahora hay que “Adjuntar” o cargar los datos con la función “attach()”.

```
> attach(Datos_Temp)
```

Podemos ver el contenido de los datos, con la función “View()”, el cual presenta dicho contenido en la ventana de archivos de RSudio o en una ventana externa en R



	Fecha	Mes	Día	Hora	Temperatura
1	20/03/2006	3	20	12.0	28.72
2	20/03/2006	3	20	12.5	28.72
3	20/03/2006	3	20	13.0	28.36
4	20/03/2006	3	20	13.5	28.36
5	20/03/2006	3	20	14.0	28.36
6	20/03/2006	3	20	14.5	28.18

Showing 1 to 6 of 1,221 entries

Fig. 17 Contenido de “Datos_Temp”

Para visualizar las características de dichos datos utilizamos las funciones “mode()”, “class()” y “length()”

```
> mode(Datos_Temp)           # Es una lista de datos
[1] "list"

> class(Datos_Temp)          # De clase estructura de datos o data.frame
[1] "data.frame"

> length(Datos_Temp)         # y de tamaño 5, con 5 variables
[1] 5
```

De estos datos, vamos a calcular los valores medios de la temperatura para cada día y ordenarlos por fechas. Primero vamos a realizar el cálculo de los valores medios de la temperatura por cada día y lo asignamos a una variable llamada “Datos_Temp1”.

```
> Datos_Temp1<-aggregate(Datos_Temp[, c("Temperatura")], na.rm=TRUE,
by= list(Fecha=Fecha), mean)
```

Mediante la función “aggregate()”, divide los datos en subconjuntos, calcula estadísticas de resumen para cada uno y devuelve el resultado en una forma conveniente. El parámetro a subdividir se establece en los valores de “Temperatura”. Hay que recordar que el parámetro: “na.rm”, pertenece al cálculo de la estimación de densidad KERNEL, es un valor lógico, si es verdadero (TRUE), los valores faltantes se eliminan de x. Si FALSE cualquier valor faltante causa un error. Al final, calculamos la media, mediante la función “mean()”

Ahora ordenamos los datos por fechas.

```
> Datos_Temp2<-Datos_Temp1[order(as.Date(Datos_Temp1$Fecha, format =
"%d/%m/%Y")),]
```

Esta nueva variable que hemos llamado “Datos_Temp2”, vamos a proceder a ordenar los datos ya calculados contenidos en “Datos_Temp1”, mediante la función “order()”, como se observa, la función “order()”, indica mediante “as.Date”, que los datos van a ser ordenados por la variable “Fecha”, el parámetro que le sigue, “format” especifica el formato de salida de fecha que existe en la matriz de datos.

Ahora a esta matriz que se ha creado, se le van a añadir las etiquetas de las columnas mediante la función “colnames()”

```
> colnames(Datos_Temp2)<-c("Fecha", "Temperatura")
```

Los nombres de las columnas van a ser “Fecha” y “Temperatura”. Ahora vamos a eliminar la primera fila que se crea por defecto, al realizar este tipo de datos, mediante la siguiente instrucción.

```
> rownames(Datos_Temp2)<-NULL
```

Ahora, presentemos este conjunto de datos para ver el resultado.

```
> Datos_Temp2
      Fecha Temperatura
```

```

1 20/03/2006      28.56250
2 21/03/2006      28.57000
3 22/03/2006      28.51375
4 23/03/2006      28.50824
5 24/03/2006      28.67200    # Se llega hasta la línea 55

```

Muy bien, ya se ha finalizado con el tratamiento de los datos. Ahora vamos a proceder a visualizarlos. Vamos a crear una nueva ventana para visualizar mejor los gráficos, debido a que vamos a incluir más de uno en ella. Esto se realiza mediante la función “windows()”

```
> windows(11,7)    # Ancho y altura de la Ventana en pulgadas, por defecto 7
```

Lo siguiente, es generar el gráfico principal de las temperaturas. Veamos la expresión, que aunque un poco larga es sumamente fácil de entender.

```
> plot(Datos_Temp2$Temperatura,type="b",pch=16,col="blue",cex=1.5,xaxt="n", cex.main=1.6,
+ xlab="Días",cex.lab=2, cex.axis=1.5,ylab="",main="Temperatura media diaria en el lago Yahuarkaka (Leticia, Amazonas)")
```

Hay que recordar que R coloca un signo “+”, cuando las expresiones son largas y continúan en la siguiente línea.

En la función, mediante “plot()”, tomamos el conjunto de datos de temperatura (signo \$), y hemos introducido el tipo de datos que deseamos que se presente y que será de un tipo “b”, los tipos disponibles para graficar con “plot()” y que es conveniente conocer son:

- "p" Para puntos,
- "l" Para líneas,
- "b" Para ambos, puntos y líneas (cuando son pares de datos a graficar),
- "c" Para que las líneas estén solas de "b",
- "o" Para ambos sobrepuestos,
- "h" Para histogramas como (o ‘alta-densidad) líneas verticales,
- "s" Para escalones,
- "S" Para otros tipos,
- "n" Para no hacer trazos, no dibujar.

El tipo de símbolo es 16 (puntos de color sólido), el color será azul, El parámetro “cex”, sirve para modificar el tamaño de los símbolos, por defecto es “cex=1”, en la gráfica hemos determinado que los símbolos tengan un tamaño de 1.5, algo más grandes que el normal. En el gráfico no nos interesa presentar los argumentos que por defecto se visualizan en el eje “x”, porque vamos a colocar ahí los intervalos que se han calculado, por lo que se utiliza el argumento (xaxt=”n”), para que no se presenten los intervalos del eje “x”. El parámetro “cex.main”, indica el tamaño del título, el parámetro cex.lab indica el tamaño de las etiquetas.

Ahora mediante la función “mtext()”, vamos a escribir texto en los márgenes del gráfico

```
> mtext("Temperatura",2, line=2.5,cex=2)
```

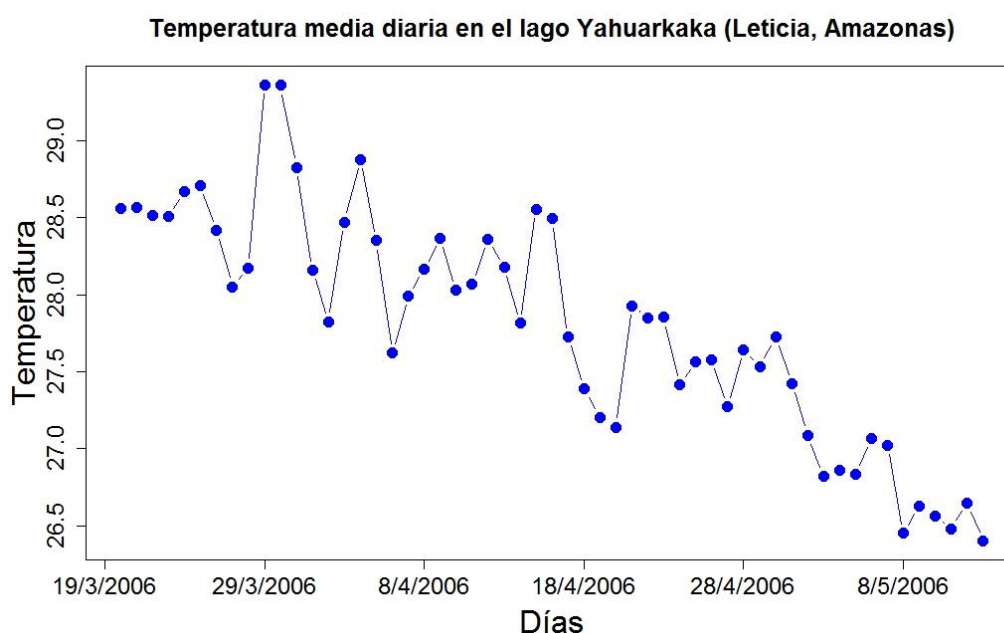
Después del texto el número indica el “side” o en qué lado se va a escribir (1 = abajo, 2 = izquierda, 3 = arriba, 4 = derecha), El parámetro “line” indica en qué línea del margen se va a escribir, comenzando por cero, y contando hacia afuera y de tamaño “cex” = 2.

La siguiente expresión a crear es a través de la función “axis()”, que agrega un eje al diagrama. Recordemos que quitamos el eje x, por lo que ahora lo agregamos pero con nuestros propios parámetros:

```
> axis(1,at=seq(0,50,by=10),labels=c("19/3/2006","29/3/2006","8/4/2006",
", "18/4/2006", "28/4/2006", "8/5/2006"),cex.axis=1.5)
```

Mediante la función “axis()”, se añade un eje “x”. El primer parámetro que es un entero indica la posición del eje (1 = abajo, 2 = izquierda, 3 = arriba, 4 = derecha). El parámetro “at”, indica los puntos en que se dibujan las marcas de graduación, aquí se establece por un valor a 1 y con una secuencia “seq(0,50, by=10)”, luego van las etiquetas a escribir y el tamaño del eje.

El resultado de todo esto, es el siguiente gráfico.



```
> subplot(plot(G1$Hora,G1$Temperatura,type="b",pch=17,col="red",cex=1,
ylab="",
+ xlab="",bty="l",cex.axis=1,xaxp=c(0,30,5)), x=10,y=27.35,size=c(1.5,
1.5), vadj=0.5, hadj=0.5)
```

Dentro de la función “subplot()” se especifican las coordenadas donde se ubicará el gráfico interno, con “x” e “y”

El parámetro “bty” sirve para modificar la caja. Los posibles valores para este parámetro son: 'o' valor por defecto para obtener la caja usual; se pueden usar también los símbolos 'l', '7', 'c', 'u' o 'j', el resultado será una caja con la forma del símbolo; se puede usar 'n' para suprimir la caja. Con el parámetro “size” se indica el tamaño del gráfico interior y los argumentos “vadj” y “hadj” permiten definir el ajuste vertical y horizontal del gráfico, respectivamente. Un valor “0” significa que se ajusta a la izquierda y un valor “1” arriba. En este caso, el valor “vadj” designado a “0.5” indica que está centrado.

Ahora se incluyen los textos de los ejes.

```
> text(10,26.55,"Hora del día")
> text(2.4,27.3,"Temperatura",srt=90) # str = ángulo del texto
```

El gráfico resultante es el siguiente:

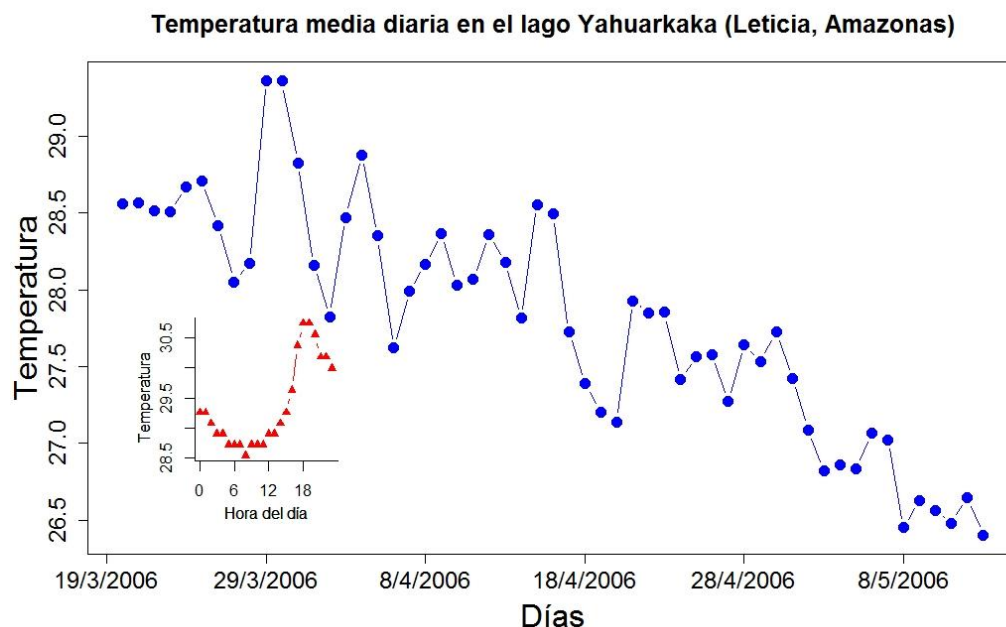


Fig. 18 Gráfico principal de las temperaturas media, más el gráfico de las temperaturas del ciclo del día 29/03/2006

Ejercicio: Construya e incluya dos gráficos, G2 y G3, con el ciclo de temperaturas de los días “25/04/2006” y “09/05/2006”. Los parámetros de la localización de los ejes x e y, además de los textos de los ejes son los siguientes.

Día 25/04/2006: (ejes) x = 37, y = 28.9 ; (textos) (37, 28.1), (29.4, 28.85)

Día 09/05/2006: (ejes) x = 51, y = 28; (textos) (51, 27.2) , (43.4, 27.95)

4) RCommander para análisis de datos estadísticos

Una vez que hemos visto la construcción y análisis estadístico de gráficos con R y RStudio, vamos a ver brevemente la interfaz gráfica de usuario (GUI) de Rcommander (Rcmdr) la cual veremos que facilita considerablemente el uso de R. En la instalación hemos podido observar que contiene un menú que permite de forma similar a otros programas estadísticos realizar de modo sencillo la mayoría de las operaciones más comunes y separa las instrucciones y resultados en dos ventanas diferentes.

La principal ventaja de Rcmdr, radica en que el usuario puede realizar casi todas las operaciones estadísticas de análisis de datos, sin tener un conocimiento a profundidad de un lenguaje de programación. Rcmdr se encarga de construir las órdenes a partir del menú, como lo hacen otros programas estadísticos como SPSS, SAS, Statistica, etc., e incluso quienes manejen un lenguaje de programación, les resulta mucho más cómodo y rápido interactuar con Rcmdr. Sin embargo, lo que se ha hecho hasta ahora, de manejar los comando de R en R y RStudio, permite multiplicar la potencia de Rcmdr y mejorar los resultados, debido a que se pueden modificar convenientemente las ordenes generadas en un inicio por Rcmdr. Hay que recordar que tanto RStudio como Rcmdr trabajan bajo la base de R.

Hemos visto que para iniciar Rcmdr se teclea la siguiente instrucción en la consola de R (también es válido en RStudio).

> library(Rcmdr) # Rcmdr responderá indicando lo siguiente:

```
Loading required package: splines
Loading required package: RcmdrMisc
Loading required package: car
Loading required package: carData
Loading required package: sandwich
Loading required package: effects
Registered S3 methods overwritten by 'lme4':
  method          from
cooks.distance.influence.merMod car
influence.merMod   car
dfbeta.influence.merMod car
dfbetas.influence.merMod car
lattice theme set by effectsTheme()
See ?effectsTheme for details.
Loading required package: sp
```

```
Versión del Rcmdr 2.6-2
Attaching package: 'Rcmdr'
```

A continuación se abrirá la ventana gráfica de Rcmdr. En la ventana superior se pueden teclear todos los comandos e instrucciones que hasta ahora se han realizado igual que en R y RStudio. La diferencia es que si se ejecuta dentro de RStudio, la pantalla de salida de la ejecución de los comandos es a través de la consola de RStudio. Probemos con ejecutar en la pantalla de Script la siguiente sentencia. Primero en el Cmdr de R y luego con RStudio.

```
(Vector_Prueba <- mean(c(13,14,17,12,15)))
```

Ejemplo 8:

Pero ahora veremos un caso más práctico de cómo utilizar RCmdr para el análisis de datos estadísticos. Vamos a utilizar un archivo externo de Excel en donde se encuentran los datos de pigmentos de algas. Este archivo que se encuentra en la carpeta de datos se llama “PigAlgas.xls”.

Ahora seleccionamos en el menú de la parte superior “Datos”, vamos a importar datos y desde ahí indicamos que esos datos son de Excel. Como se observa en la siguiente imagen.

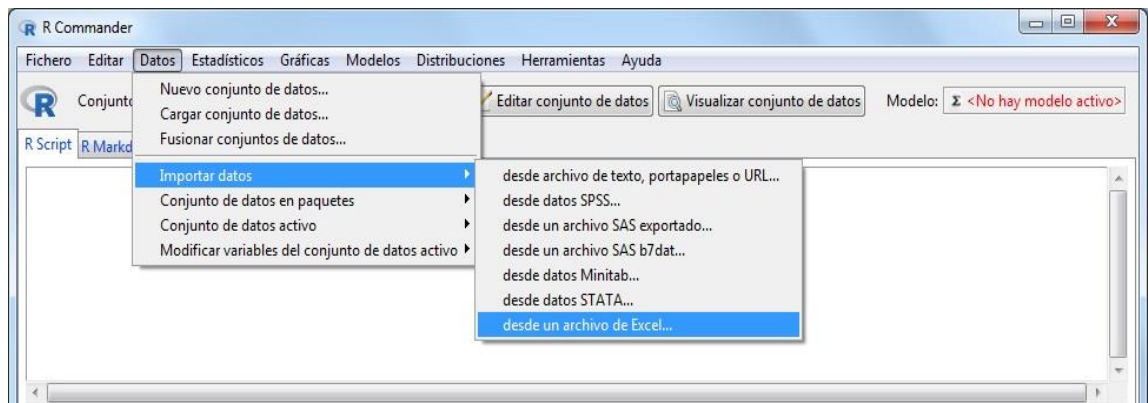


Fig. 19 Ventana de Importar datos de Excel en RCmdr

A continuación se presenta una pequeña ventana o cuadro de diálogo, que nos indica el introducir el nombre del conjunto de datos, o sea, darle nombre al conjunto de datos que vamos a importar, por defecto el nombre es “Dataset”, aquí le damos un nombre particular, como el conjunto de datos es de análisis de algas, vamos a decir que se llamará “DatosAlgas” y damos clic a “aceptar”.

Se abrirá la ventana del explorador de Windows, para seleccionar el archivo a importar, seleccionamos la carpeta y buscamos el archivo “PigAlgas.xls” y damos clic a “Abrir”.

Si observamos en la parte superior de la ventana de RCmdr se encuentra un Script, que indica los comandos que se van ejecutando y estos se presentan en la salida, situada en la parte inferior. El comando que se ha generado será similar al siguiente:

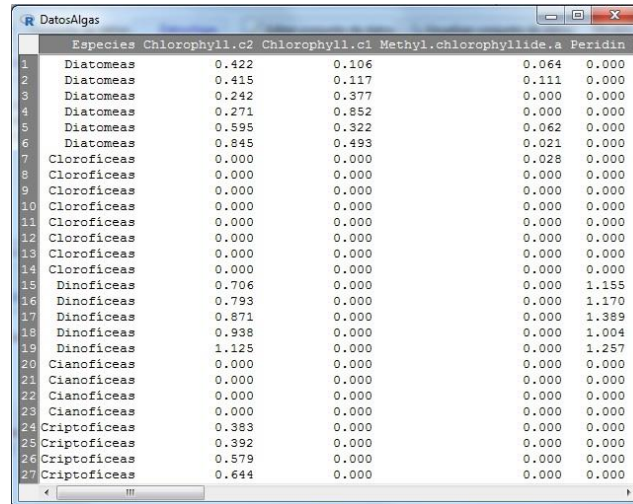
```
> DatosAlgas <- readXL("C:/Users/Lig/Documents/Programas_R/datos/PigAlgas.xls",  
rownames=FALSE, header=TRUE,  
+ na="", sheet="PigAlgas", stringsAsFactors=TRUE)
```

Si vemos la sintaxis, nos parecerá muy familiar, en efecto es igual a la que hasta ahora hemos estado tecleando. En la parte inferior, en la ventana de mensajes, se muestra una primera información del archivo que indica que “El conjunto de datos DatosAlgas contiene 27 filas y 20 columnas. Podemos en la parte de script teclear por ejemplo “mode(DatosAlgas)”, dar clic al botón “Ejecutar” y veremos que en la salida se presentará el modo de estos datos de la siguiente manera

```
> mode(DatosAlgas)  
[1] "list"
```


Bien, ahora vamos a visualizar los datos, para esto, damos clic, en el botón situado en la parte superior, donde indica “Visualizar conjunto de datos”. Para poder observar y analizar los datos, ya que es importante que cuando se presentan, no existan en dicho conjunto líneas vacías, que contengan el código “NA”, si existen hay que eliminarlas.

El contenido de este archivo se presenta en una pantalla exterior y es similar al siguiente:



	Especies	Chlorophyll.c2	Chlorophyll.c1	Methyl.chlorophyllide.a	Peridin
1	Diatomeas	0.422	0.106	0.064	0.000
2	Diatomeas	0.415	0.117	0.111	0.000
3	Diatomeas	0.242	0.377	0.000	0.000
4	Diatomeas	0.271	0.852	0.000	0.000
5	Diatomeas	0.595	0.322	0.062	0.000
6	Diatomeas	0.845	0.493	0.021	0.000
7	Cloroficeas	0.000	0.000	0.028	0.000
8	Cloroficeas	0.000	0.000	0.000	0.000
9	Cloroficeas	0.000	0.000	0.000	0.000
10	Cloroficeas	0.000	0.000	0.000	0.000
11	Cloroficeas	0.000	0.000	0.000	0.000
12	Cloroficeas	0.000	0.000	0.000	0.000
13	Cloroficeas	0.000	0.000	0.000	0.000
14	Cloroficeas	0.000	0.000	0.000	0.000
15	Dinoficeas	0.706	0.000	0.000	1.155
16	Dinoficeas	0.793	0.000	0.000	1.170
17	Dinoficeas	0.871	0.000	0.000	1.389
18	Dinoficeas	0.938	0.000	0.000	1.004
19	Dinoficeas	1.125	0.000	0.000	1.257
20	Cianoficeas	0.000	0.000	0.000	0.000
21	Cianoficeas	0.000	0.000	0.000	0.000
22	Cianoficeas	0.000	0.000	0.000	0.000
23	Cianoficeas	0.000	0.000	0.000	0.000
24	Criptoficeas	0.383	0.000	0.000	0.000
25	Criptoficeas	0.392	0.000	0.000	0.000
26	Criptoficeas	0.579	0.000	0.000	0.000
27	Criptoficeas	0.644	0.000	0.000	0.000

Fig. 20 Ventana emergente del contenido de DatosAlgas

Bien, ya tenemos los datos cargados y listos para trabajar con ellos. Ahora vamos a utilizar los gráficos estadísticos. En el menú de gráficos, hay varios tipos que son los más básicos y cuyo uso es más frecuente, que son los que hemos estado utilizando hasta ahora y algunos más, ahí están los secuenciales (para series temporales), histogramas (para variables continuas), de tallo y hojas, diagramas de cajas o bigotes, diagramas de dispersión (para ver la relación entre dos variables cuantitativas), gráficas de líneas, gráficas de barras, gráficas de sectores o pie (para variables cualitativas), y además gráficos 3D, etc. Si vemos en la parte superior de la pantalla de RCmdr, donde dice “Conjunto de datos”, se presenta en color azul el nombre de los datos cargados que es “DatosAlgas”

Vamos ahora a construir un gráfico con un diagrama de caja. Así que seleccionamos en el Menú de gráficos el diagrama de caja como en la figura siguiente.

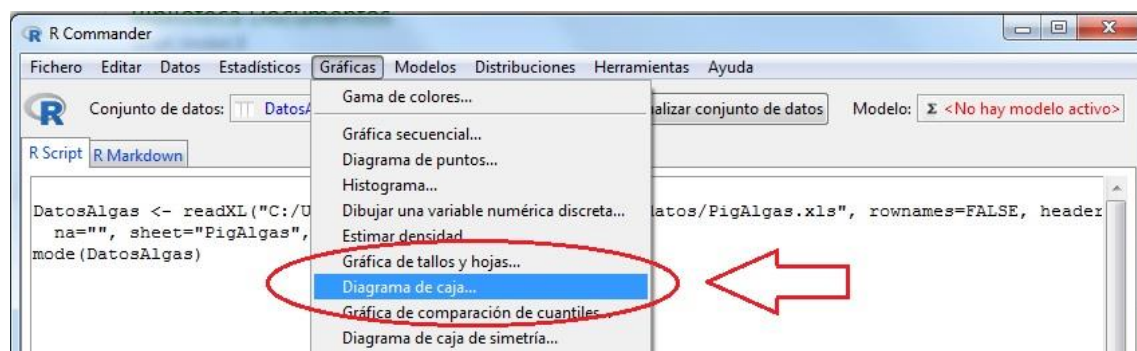


Fig. 21 Ventana de selección del gráfico de diagrama de caja

A continuación, se presenta una ventana que nos permite seleccionar la variable a trabajar, se indica seleccionar una. Para nuestro ejemplo, vamos a seleccionar la variable “Chlophyll.a.epimer” y damos clic al botón “Gráfica por grupos”.

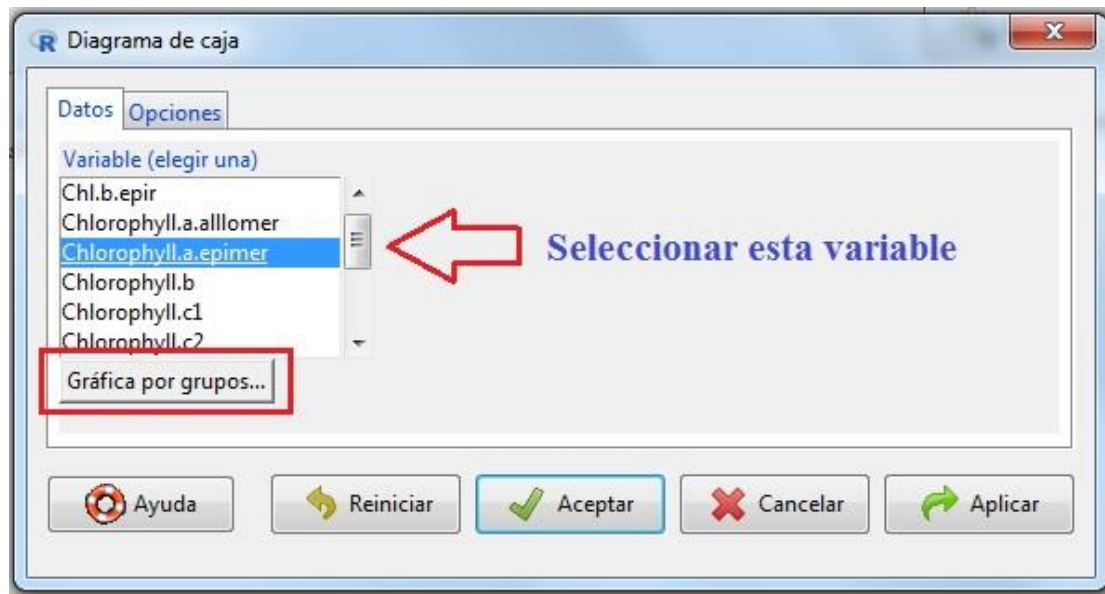


Fig. 22 Ventana de selección de la variable Chlophyll.a.epimer

Cuando se da clic a “Gráfica por grupos”, se presenta la ventana de la elección de grupos.

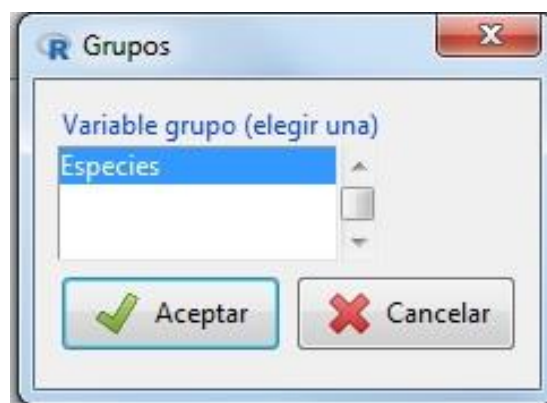


Fig. 23 Ventana de selección de grupos

En este caso solo existe un grupo llamado “Especies”, así que se da clic a “Aceptar”, esta acción retorna a la pantalla anterior, en donde se observa que “gráfica por grupos” ha cambiado de color, indicando que ya está seleccionado el grupo, por lo que se debe de dar clic de nuevo a “Aceptar” en esta pantalla y eso es todo. El gráfico ha sido creado con los datos y se presentará en una ventana exterior de la siguiente manera.

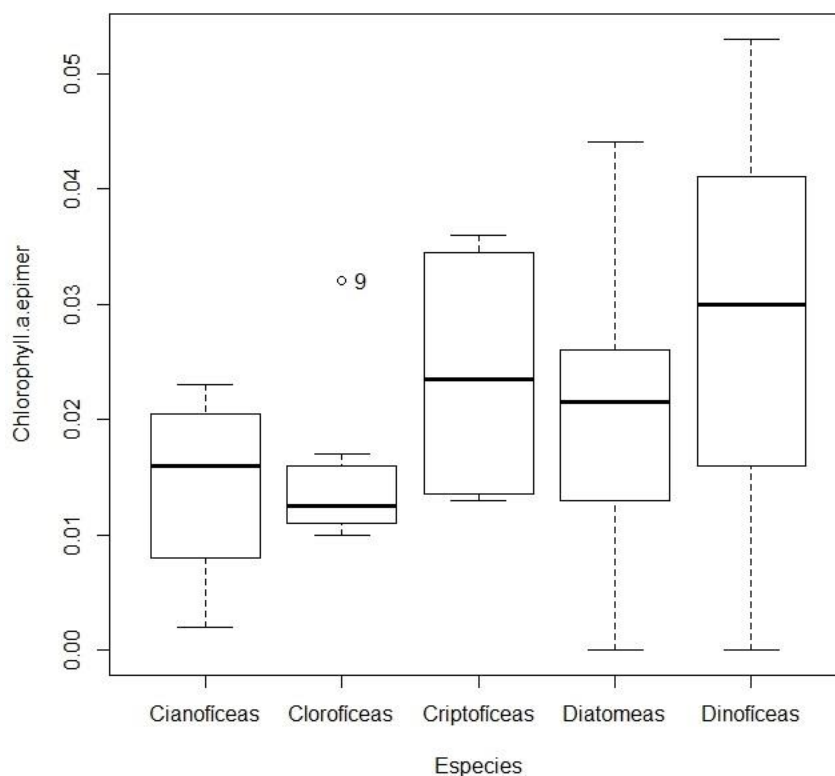


Fig. 24 Ventana del gráfico del análisis de la variable Chlophyll.a.epimer en el grupo de Especies

Para guardar este gráfico, en la pantalla gráfica se encuentra el menú de archivo, ahí se selecciona “Guardar cómo”, se presentan una serie de formatos en los que se puede almacenar el gráfico, de ellos se elige “jpeg”, para imagen y se selecciona la calidad, decimos que un 100%. La pantalla es similar a la siguiente. Un vez elegido el tipo y calidad se abre la pantalla del explorador de Windows, para seleccionar la carpeta donde guardar el gráfico. La guardamos en la carpeta “Gráficos” indicando un nombre.

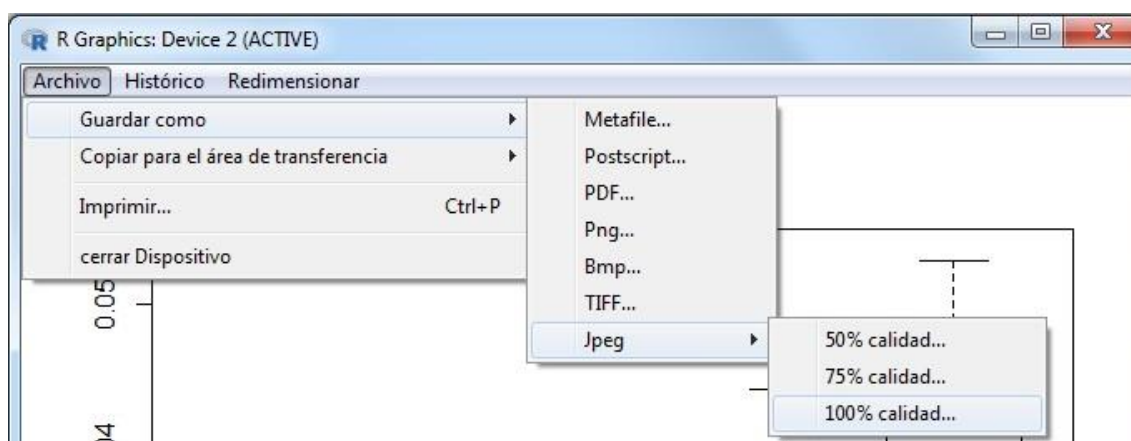


Fig. 25 Ventana del menú Guardar como, para almacenar el gráfico.

Podemos comprobar que el proceso para generar el gráfico es rápido y muy cómodo, aunque el gráfico que se presenta es muy sencillo. Pero como hemos mencionado, el gráfico se puede mejorar.

Observemos en la pantalla de Rcmdr. En la parte de Script, la expresión que ha generado el gráfico.

```
Boxplot(Chlorophyll.a.epimer~Especies, data=DatosAlgas, id=list(method="y"))
```

Ahí está la función “boxplot()” y su sintaxis, tal y como se hubiera tecleado en R y RStudio para generar el resultado. Rcmdr lo ha hecho por nosotros. Aquí es donde se puede mejorar, añadiendo instrucciones, o parámetros a la función. Por ejemplo, queremos que las cajas sean de color, basta con agregar a la línea de comandos, el parámetro “col=orange”, y le damos un título con “main”, por ejemplo. Una vez hecho esto, seleccionar la línea (pulsando doble clic para que se resalte) y se da clic al botón de “Ejecutar”, veremos el cambio en el gráfico. Bien, hacemos los siguientes cambios en la función en la parte de “Script”

```
Boxplot(Chlorophyll.a.epimer~Especies, data=DatosAlgas, id=list(method="y"), col = "orange", main = "Datos de Algas")
```

Damos clic a Ejecutar y el gráfico resultante ya mejorado es el siguiente:

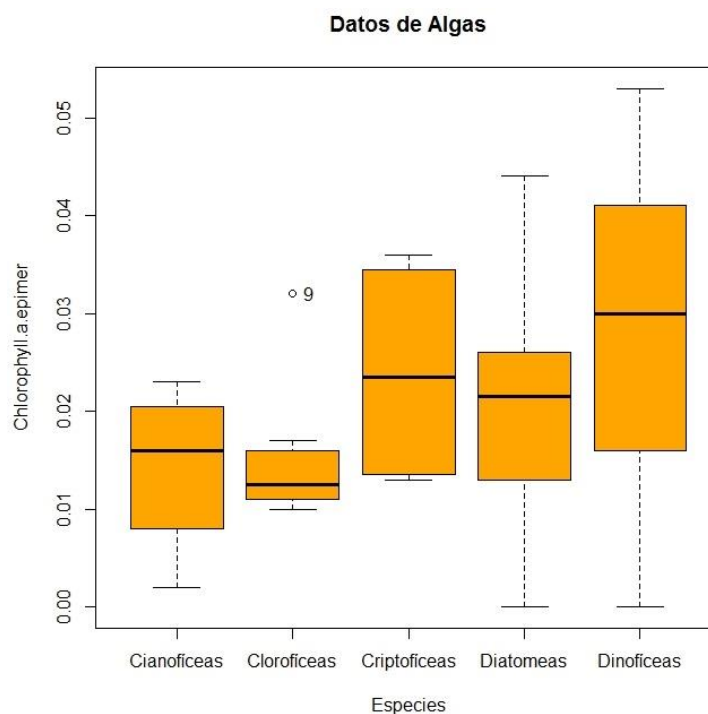


Fig. 26 Ventana con los cambios realizados en el gráfico.

La ventana de gráfico no tiene memoria, así que cada vez que le demos a guardar, hay que buscar la carpeta y dar el nombre del gráfico para almacenarlo. También, se puede modificar la gama de colores que presentan algunos tipos de gráficos, por defecto están en blanco y negro, pero seleccionado en el menú de gráficos en la parte superior “Gama

de colores”, se puede elegir la gama de colores que se prefiera, pero esto no está disponible para todos los gráficos, por lo que si se desea modificar el color, se debe de realizar desde la ventana Script, al realizar la edición de la función gráfica.



Fig. 26 Ventana de Gama de colores

Si queremos limpiar las ventanas, en el menú “Editar”, se selecciona en la parte inferior “Limpiar Ventana”, y se borra el contenido de la ventana activa, es decir, ya sea si el cursor se encuentra en “Script” o en “Salida”.

En el menú “gráficos”, se encuentra además el diagrama de tallos y hojas (Stem-and-Leaf Diagram). Este diagrama es útil, debido a que permite obtener simultáneamente una distribución de frecuencias de la variable y su representación gráfica. En RCmdr, el gráfico de tallos y hojas se presenta en la pantalla de salida. Para la variable “Chlorophyll.a.epimer” será el siguiente: Si se observa, para crearlo se utiliza la función “stem.leaf()”

```
> with(DatosAlgas, stem.leaf(Chlorophyll.a.epimer, na.rm=TRUE))
1 | 2: represents 0.012
leaf unit: 0.001
n: 27
 3  0* | 002
    0. |
12  1* | 011233344
(5)  1. | 56778
10  2* | 3
 9  2. | 66
 7  3* | 023
 4  3. | 6
 3  4* | 14
    4. |
 1  5* | 3
```

Como ejercicio, realizar la construcción de las gráficas que se detallan, Observe los resultados gráficos y el código que ha generado dicho resultado:

- Construir gráficos con la misma variable y el mismo grupo por medio de “histogramas”, “diagramas de puntos”, “Gráfica de comparación de cuantiles”, “Diagrama de caja de simetría”.
- Construir un diagrama de tallos y hojas para otras dos variables y observar su resultado.
- Construir “Diagrama de dispersión entre las variables “Chlorophyll.a.allomer y Chlorophyll.a.epimer”, según el grupo de especies.

5) Introducción a los datos espaciales de tipo discreto

○ Introducción.

Una vez que nos hemos familiarizado con el análisis de los datos por medio de R, RStudio y RCmdr, vamos a introducir el análisis de los datos espaciales mediante R. Para ellos vamos a utilizar muchas de las técnicas y sobre todo la sintaxis ya aprendida durante los ejercicios con RStudio. Para analizar y trabajar con los ejemplos propuestos desde aquí en adelante, se van a utilizar archivos almacenados en una carpeta de datos y que corresponden a observaciones reales generadas.

La localización geográfica de los lugares en donde se producen los acontecimientos observados es muy importante. De hecho, en la actualidad el análisis de aspectos tan vitales como el cambio climático y sus efectos y consecuencias, pueden condicionar la toma de decisiones, por ejemplo a la hora de establecer escuelas, centros de salud o distribuir recursos, etc.

Formalmente, los datos que son analizados mediante este tipo de técnicas consisten en un par de elementos:

- a) En primer lugar se establecen las localizaciones $\{s_1, \dots, s_n\}$, sobre una superficie, generalmente la tierra, o dicho de otra forma, pares de puntos (x_i, y_i) , como (Latitud, Longitud), o (Menor distancia a un punto, Menor distancia a una línea imaginaria paralela a un punto), puesto que en este sistema de referencia puede resultar el más adecuado en la modelización del fenómeno (Venables y Dichmont, 2004)³.
- b) Los datos $\{Z(s_1), \dots, Z(s_n)\}$, observados sobre esas localizaciones, como podrían ser por ejemplo precipitaciones de lluvia, sequía, polución aérea, etc. En este caso se supone que los datos son el resultado de la observación de una variable “Z”, unidimensional o multidimensional.

Según el tipo de localización “s” que se considere, los datos espaciales se denominan y analizan de forma diferente. Si las localizaciones $\{s_1, \dots, s_n\}$, son fijas pero valores cualquiera de la superficie considerada, es decir, matemáticamente valores cualesquiera de \mathbb{R}^k (generalmente $k = 2$ o $k = 3$), se trata en este caso de Geoestadística. En general, el valor de $Z(s)$ es conocido en algunos puntos y se desea conocer (interpolar) el valor de Z en localizaciones diferentes en donde su valor es desconocido. En este tipo de datos las localizaciones se mueven de forma continua en \mathbb{R}^k .

Si las localizaciones son valores aislados y no son fijas sino que además son aleatorias (pero independientes de Z) se está refiriendo a “Procesos Puntuales (Point patterns)”, como por ejemplo, los árboles en un bosque.

Para concluir podemos decir que los datos pueden venir agregados, formando grupos de pequeños rectángulos, conformando lo que se conoce como “Datos Reticulares o Regionales (lattice data)”.

³ Venables, W.N. y Dichmont, C.M. (2004). A generalised linear model for catch allocation: an example from Australia's northern prawn fishery, Fisheries Research, 70, 409-426

Por lo tanto, si el índice espacial “s” es discreto estaremos hablando de “Procesos Puntuales” y si el índice es continuo se estará refiriendo al campo de la Geoestadística y para terminar, si son rejillas o agregados se está mencionando que son datos Regionales.

Lo habitual en todos estos casos es que la variable “Z” no se considere o más bien se tome como una constante y que como mucho, se le añada una marca a los datos como por ejemplo que pertenecen a una clase o a otra, que son de un área determinada o población o de otra, de manera que el interés en este tipo de datos se centra en las localizaciones teniendo como objetivo los siguientes:

- a) Analizar la distribución que presentan los datos espaciales, como por ejemplo si se encuentran o no igualmente espaciados.
- b) Analizar las marcas que presentan estas localizaciones, para por ejemplo poder comparar un par de elementos.
- c) Analizar la densidad de las localizaciones, es decir, el número de elementos por cada unidad de área.

Por otra parte, también es posible considerar distinto el índice de localización de un posible índice temporal “t”, esto ocurre si se quiere considerar datos espaciales a lo largo del tiempo, como por ejemplo el análisis de sequías, inundaciones o terremotos a lo largo del tiempo, en este caso se hablará de modelos espacio-temporales.

- Datos espaciales y su representación

Generalmente, la matriz de datos espaciales se encuentra formada por columnas en donde aparecerán localizaciones, además de las columnas correspondientes a valores de las variables en esas localizaciones. En R, los métodos y clases para el manejo de datos espaciales, se encuentran incluidos en la librería “sp (*Classes and Methods for Spatial Data*)”. Los datos pueden ser tomados desde cero es decir creados o desde un archivo de datos.

Los datos espaciales, se implementan por medio de cuatro tipos diferentes de clases:

- 1) Puntos
- 2) Líneas
- 3) Polígonos y
- 4) Redes (grids)

El paquete “sp”, posee diferentes funciones o clases para utilizar con cada tipo de dato espacial. Por ejemplo para “puntos”, se tiene la clase “SpatialPoints”, para líneas las clases “Line”, “Lines” y “SpatialLines”, para “polígonos”, las clases “Polygon”, “Polygons” y “SpatialPolygons” y para “Redes (grids)”, la clase “SpatialGrid”, entre muchas otras.

- Representación en Puntos y Polígonos

En R es habitual representar “Puntos” que conforman una nube de puntos generalmente sin marco ni ejes coordenados como sucede en los mapas, para esto se utiliza la función “plot()” con los argumentos que dicha función contiene y de los cuales se han visto algunos ejemplos. Previamente se debe de extraer las localizaciones de la matriz de datos.

A lo largo de los siguientes enunciados, vamos a ir describiendo los métodos de construcción y manejo de datos espaciales más generales.

- Generación de puntos espaciales básicos

Se pueden construir datos espaciales y su representación desde cero, o sea a través de datos que se van a ingresar por medio de vectores y posteriormente convertirlos a matrices, listas o data.frame. El siguiente ejemplo presenta esta manera de construir de forma básica estos datos espaciales.

Ejemplo 9:

Se van a crear dos vectores con 8 observaciones, que contienen la información de puntos espaciales, dichas observaciones pueden ser datos de coordenadas (latitud – longitud), de localizaciones de: escuelas, sitios de observación, lugares de toma de muestra, etc., a los que se les nombrará, "Punto1" y "Punto2" de la siguiente forma:

```
> Punto1 = c(25,38.5,12.4,37.4,15.6,46.8,24.6,37.8)
> Punto2 <- c(-25.6,-45.8,-32.6,-41.2,-23.3,-26.4,-22.8,-21.7)
```

Mediante la función "SpatialPoints" se van a crear los objetos espaciales de dichos puntos y graficarlos.

Creación del primer Grupo de puntos espaciales con cbind

```
> Objeto1 <- SpatialPoints(cbind(Punto1,Punto2)) # Con 'cbind' se combinan los objetos por filas o columnas
```

Crear datos de objetos espaciales de tipo punto con la función "SpatialPoints", es bastante sencillo, como se observa, se crea un Objeto1, que contienen la colección de datos espaciales combinados mediante la función "cbind()". Dicha función, toma una secuencia de vectores, matrices, o data.frame y las combina por filas o columnas. El contenido se puede visualizar de la siguiente forma.

```
> Objeto1 # Presenta el contenido del objeto creado
SpatialPoints:
  Punto1 Punto2
[1,] 25.0 -25.6
[2,] 38.5 -45.8
[3,] 12.4 -32.6
[4,] 37.4 -41.2
[5,] 15.6 -23.3
[6,] 46.8 -26.4
[7,] 24.6 -22.8
[8,] 37.8 -21.7
Coordinate Reference System (CRS) arguments: NA

> class(Objeto1) # Presenta la clase del objeto
[1] "SpatialPoints"
attr(,"package")
[1] "sp"
> mode(Objeto1) # Presenta el tipo de modo de almacenamiento de un Objeto1
[1] "S4"
```


NOTA: Al ser R un lenguaje orientado a objetos, los objetos se definen en clases y R posee tres sistemas orientados a objetos-: (OO): [[S3]], [[S4]] y [[R5]] En R, los objetos básicos tenían una clase implícita definida y la más simple, que era la S3, dicha clase para algunos ya está obsoleta y se ha cambiado por S4, porque posee más ventajas como por ejemplo, define formalmente la representación y la herencia para cada clase. Así los objetos creados con “SpatialPoints”, pertenecen a esta clase.

Bien, ahora que ya se encuentran creados los objetos, ya solo queda representar los puntos espaciales por medio de una representación gráfica por medio de la función “plot()”

```
> plot(Objeto1, pch = 15, col = "red") # Tipo de puntos 15 y color rojo
```

El gráfico generado es el siguiente

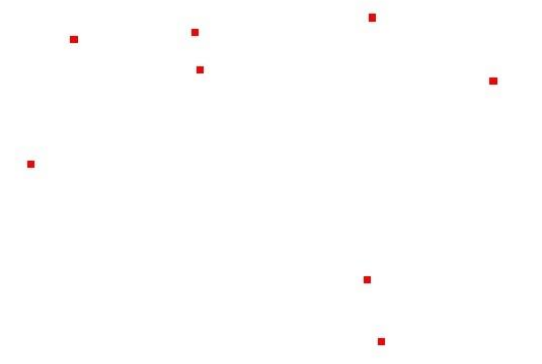


Fig. 27 Gráfico de puntos espaciales mediante “cbind()”

Creación del Segundo Grupo de puntos espaciales con “list()”

También se puede crear los objetos a partir de listas, colocando como argumento de “SpatialPoints()” a la función “list()”.

```
> Objeto2 <- SpatialPoints(list(Punto1,Punto2)) # Con list se crea un objeto de tipo lista
```

Al presentar el Objeto2 creado, este va a diferir del anterior en cuanto a su contenido, mas no en su clase ni modo, como se aprecia en la siguiente descripción.

```
> Objeto2
SpatialPoints:
      c.25.6...38.5...12.4...37.4...15.6...46.8...24.6...37.8. c..25.6...45.8...
32.6...41.2...23.3...26.4...22.8...21.7.
[1,]                                25.0
-25.6
[2,]                                38.5
-45.8
[3,]                                12.4
-32.6
[4,]                                37.4
-41.2
```

```

[5,] 15.6
-23.3
[6,] 46.8
-26.4
[7,] 24.6
-22.8
[8,] 37.8
-21.7
Coordinate Reference System (CRS) arguments: NA
> class(Objeto2)
[1] "SpatialPoints"
attr(,"package")
[1] "sp"
> mode(Objeto2)
[1] "s4"

```

Si se construyera la representación gráfica de este objeto con “plot()”, la gráfica resultante sería exactamente igual a la del “Objeto1”, pues son los mismos puntos. Solo que estos contenidos en una lista.

```
> plot(Objeto2, pch = 15, col = "red") #Tipo de puntos 15 y color rojo
```

Creación del Tercer Grupo de puntos espaciales con “data.frame()”

Ahora, también estos puntos espaciales, pueden ser también convertidos a una estructura de datos o “data.frame”, a través de la función “SpatialPoints()”. Vamos a proceder a crear un tercer objeto “Objeto3” mediante esta premisa.

```
> Objeto3 <- SpatialPoints(data.frame(Punto1,Punto2)) # Se crean los
objetos espaciales de tipo estructura de datos.
```

La estructura es similar a la utilizada con

```

> mode(Objeto3)
[1] "s4"
> Objeto3
SpatialPoints:
      Punto1 Punto2
[1,]  25.0  -25.6
[2,]  38.5  -45.8
[3,]  12.4  -32.6
[4,]  37.4  -41.2
[5,]  15.6  -23.3
[6,]  46.8  -26.4
[7,]  24.6  -22.8
[8,]  37.8  -21.7
Coordinate Reference System (CRS) arguments: NA
>
> class(Objeto3)
[1] "SpatialPoints"
attr(,"package")
[1] "sp"
>
> mode(Objeto3)
[1] "s4"

```

La presentación gráfica de estos datos espaciales de tipo punto, obviamente sería igual a las anteriores.

- Generación de puntos espaciales de tipo polígonos

Ejemplo 10:

Se va a proceder a crear datos espaciales de tipo polígono mediante la función “Polygon()”, Procedemos a crear tres vectores para tres polígonos. Mediante la función “Poligon()” se crean tres polígonos cuyos vértices están dados por los valores en (x,y), establecidos en este caso por los vectores.

Cuando se establecen o se conocen las coordenadas de cada polígono, la función **cbind()**, toma una secuencia de vectores, matrices, o data.frame y las combina por filas o columnas en base a un arreglo matricial para crear un objeto de tipo **Polygon** donde se permite además especificar si los polígonos son “agujeros” (hole = TRUE), para nuestro ejemplo al no incluirlo se establece a FALSO y no lo presenta. De esta forma, los objetos de tipo **Polygon** representan el primer paso a la construcción de los objetos de tipo **Polygons**. Estos últimos aceptan una lista de objetos **Polygon** y la asignación de un identificador id. Bajo esta premisa, lo primero es crear los objetos tipo **Polygon**.

```
> Pol1 = Polygon(cbind(c(2,4,4,1,2),c(2,3,5,4,2)))
> Pol2 = Polygon(cbind(c(5,4,2,5),c(2,3,2,2)))
> Pol3 = Polygon(cbind(c(4,4,5,10,4),c(5,3,2,5,5)))
```

Hasta aquí se han creado los objetos de clase polígono, sin embargo, como se ha mencionado, hay que crear los objetos de la clase **Polygons**, con función “Polygons()” con la función “list()”, para efectuar la lista de objetos con un argumento único para cada polígono de la siguiente manera.

```
> Lpol1 = Polygons(list(Pol1), "s1")
> Lpol2 = Polygons(list(Pol2), "s2")
> Lpol3 = Polygons(list(Pol3), "s3")
```

Una vez que se dispone de los objetos de clase **Polygons** con un identificador. Ahora ya se pueden crear los objetos espaciales **SpatialPolygons** de la clase polígonos con la función “SpatialPolygons()”, y que corresponden a una lista de **Polygons** indicando el orden en el que se van a representar.

```
> ObjetEspaPol = SpatialPolygons(list(Lpol1,Lpol2,Lpol3), 1:3)
```

Como se observa, se ha creado, ahora se puede visualizar la representación gráfica.

```
> plot(ObjetEspaPol, col = 3:6)
```

El gráfico que se genera, se observa en la siguiente figura. También pueden dibujarse los ejes, colocando el argumento “axes = T” en el gráfico.

```
> plot(ObjetEspaPol, col = 3:6, axes = T)      # Presenta los ejes
```

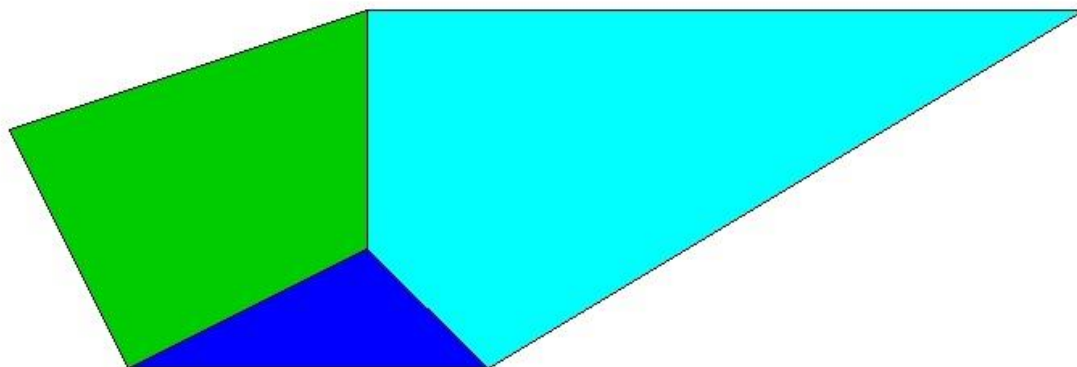


Fig. 29 Gráfico de polígonos espaciales solidos sin agujero

Si se desea incluir el agujero (hole), se debe de colocar a true el valor y se designa en cuál de los polígonos se va a incluir el agujero con las coordenadas. Al código anterior solo hay que modificarlo algunos puntos. La secuencia completa de las expresiones es la siguiente:

```
> # Presentar agujero

> Pol1 = Polygon(cbind(c(2,4,4,1,2),c(2,3,5,4,2))) # Polygon crea un
n objeto de clase espacial
> Pol2 = Polygon(cbind(c(5,4,2,5),c(2,3,2,2)))
> Pol3 = Polygon(cbind(c(4,4,5,10,4),c(5,3,2,5,5)))
> Pol4 = Polygon(cbind(c(5,6,6,5,5),c(4,4,3,3,4)), hole = TRUE) # cr
eación del polígono agujero

> Lpol1 = Polygons(list(Pol1), "s1")
> Lpol2 = Polygons(list(Pol2), "s2")
> Lpol3b = Polygons(list(Pol3, Pol4), "s3/4")

> ObjetEspaPol_Aguj = SpatialPolygons(list(Lpol1,Lpol2,Lpol3b), 1:3)
> plot(ObjetEspaPol_Aguj, col = 3:6, pbg = "white", axes = T)
```

Como se observa se ha creado un cuarto objeto polígono de tipo **Polygon**, indicando las coordenadas y con el valor del argumento “hole = TRUE”.

Ahora se incluye en la tercera expresión, junto con el objeto polígono “Pol3”, o sea, va a ir dentro de ese polígono. “s3/4” es el identificador.

Ahora en la lista de creación del objeto espacial polígono con **SpatialPolygons**, solo se incluye.

El resultado gráfico, incluyendo los ejes es el siguiente:

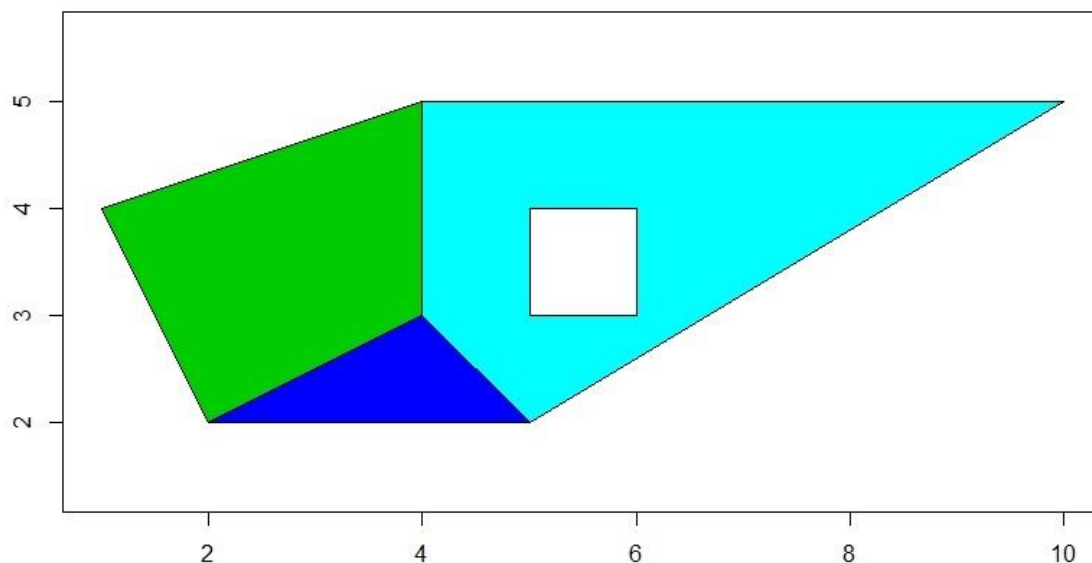


Fig. 29 Gráfico de polígonos espaciales solidos con agujero

Ejercicio: Separe cada polígono y presente su gráfico (pruebe a cambiar el color), identifique cada uno de los polígonos generados

Ahora si se desean presentar los argumentos hay que crear un **Data Frame** de acuerdo a los objetos creados y se incluye en la función que crea el objeto llamada **SpatialPolygonsDataFrame**, La secuencia de expresiones es la siguiente:

```
> #Colocar atributos en los polígonos sin agujero
> AtribPol = data.frame(a=1:3, b=3:1, row.names=c("s3", "s2", "s1"))
> Spa_DataPol = SpatialPolygonsDataFrame(ObjetEspaPol, AtribPol)
> as(Spa_DataPol, "data.frame")
  a b
s1 3 1
s2 2 2
s3 1 3
> spplot(Spa_DataPol)
```

El **Data Frame** crea dos objetos, (a y b), esto indica que se van a presentar dos diagramas. Se construye un objeto espacial de datos mediante **SpatialPolygonsDataFrame**, con el objeto de atributos **Data Frame** creado y el objeto que contiene los polígonos espaciales antes creados "**ObjetEspaPol**", la función "**as()**" fuerza al objeto "**Spa_DataPol**" a pertenecer a la clase "**Data.Frame**" y en la secuencia anterior se observa la salida de esta función. Por último, la función "**spplot()**", dibuja o traza gráficos para datos espaciales con atributos. El resultado del gráfico con atributos sin agujero es el siguiente:

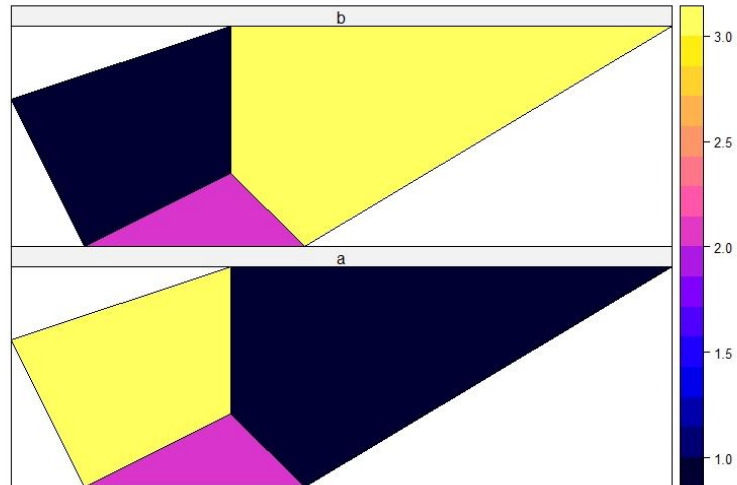


Fig. 30 Gráfico de polígonos espaciales solidos con atributos sin agujero

Para incluir el agujero, se hacen las modificaciones siguientes:

```
> #Colocar atributos en los poligonos con agujero
>
> AtribPol2 = data.frame(a=1:3, b=3:1, row.names=c("s3/4", "s2", "s1"))
>
> Spa_DataPol2 = SpatialPolygonsDataFrame(ObjetEspaPol_Aguj, AtribPol2)
>
> as(Spa_DataPol2, "data.frame")
      a b
s1    3 1
s2    2 2
s3/4  1 3
> splot(Spa_DataPol2)
```

Y el gráfico resultante es el siguiente:

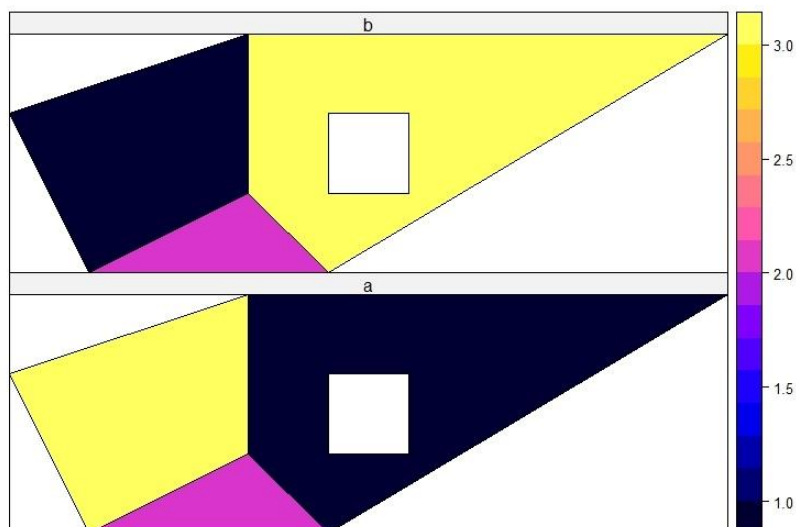


Fig. 31 Gráfico de polígonos espaciales solidos con atributos con agujero

- Generación básica de líneas con datos espaciales.

Ejemplo 11

En este ejemplo se crearán objetos de tipo línea al igual que los polígonos y puntos. A través de la instrucción “`cbind()`” se creará la matriz vectorial para las coordenadas. Se van a crear tres grupos de líneas. Seguidamente con la instrucción “`Line()`” se van a crear los objetos de línea que posteriormente por medio de listas van a pasar a ser de clase línea con su identificador mediante la función “`Lines()`”, y que a su vez, serán reunidas a través de una lista y convertidos en datos espaciales de tipo línea por medio de la función “`SpatialLines()`”. Lo último es simplemente graficar este resultado.

```
# Creación de objetos espacial de líneas

# Creación del primer vector de las líneas: primeras azules
> lineas1 = cbind(c(1,2,3),c(3,2,2))

# Creación del segundo vector de las líneas: segundas azules
> llineas2 = cbind(l1[,1]+.05,l1[,2]+.05)

# Creación del tercer vector de las líneas: líneas rojas
> lineas3 = cbind(c(1,2,3),c(1,1.5,1))

# Creación del objeto línea para el primer vector
> ObjetoS11 = Line(lineas1)

# Creación del objeto línea para el segundo vector
> objetos12 = Line(llineas2)

# Creación del objeto línea para el tercer vector
> objetos13 = Line(lineas3)

# Creación del objeto de clase línea con su identificador
> ClaseObjetoS1 = Lines(list(ObjetoS11, objetos12), ID="a1")
> ClaseObjetoS2 = Lines(list(objetos13), ID="b1")

# Creación de la lista de objetos espaciales líneas
> ObjetoEspacial_1 = SpatialLines(list(ClaseObjetoS1,ClaseObjetoS2))

# Dibujo de la gráfica (S1 en azul y S2 en rojo)
> plot(ObjetoEspacial_1, col = c("red", "blue"))
```

La gráfica resultante es la siguiente

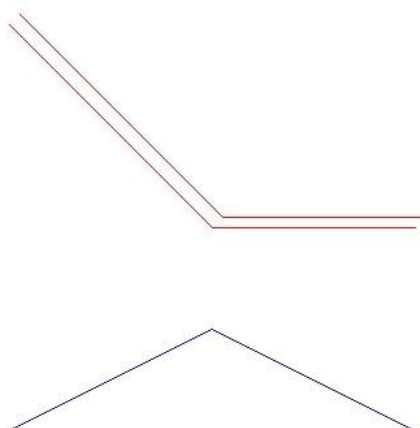


Fig. 32 Gráfico de líneas de datos espaciales mediante “SpatialLines”

- Generación de datos espaciales mediante archivos de datos

Ejemplo 12

En este ejemplo, vamos a utilizar un archivo de datos llamado “meuse10.txt” que corresponde a localizaciones y concentraciones (en un área aproximada de 15 x 15 metros), de metales pesados en la capa superior del suelo, recogidos en una llanura de inundación del río Mosa, cerca de la ciudad holandesa de Stein, los datos son tomados de Rikken y van Rijn⁴ (1993).

Se procede a cargar el archivo “meuse10.txt” por medio de RStudio a través de la pantalla de “Import Data”, seleccionando la ruta y dando clic a “Import”. El archivo se visualiza en la pantalla de Archivo de RStudio.

	id	x	y	cadmium	copper	lead	zinc	elev	dist	om	ffreq	soil	lime	landuse	dist.m
1	1	181072	333611	11.7	85	299	1022	7.909	0.00135803	13.6	1	1	1	Ah	50
2	2	181025	333558	8.6	81	277	1141	6.983	0.01222430	14.0	1	1	1	Ah	30
3	3	181165	333537	6.5	68	199	640	7.800	0.10302900	13.0	1	1	1	Ah	150

Fig. 33 Archivo “meuse10.txt” en RStudio

En la ventana de medio ambiente podemos observar que el archivo contiene 155 observaciones en 15 variables. La descripción de las variables son: Las dos primeras columnas (x, y) representan al sistema de coordenadas RDM (sistema de coordenadas

⁴ Rikken, M.G.J. y van Rijn, R.P.G. (1993). Soil pollution with heavy metals – an inquiry into spatial variation, cost of mapping and the risk evaluation of copper, cadmium lead and zinc in the floodplains of the Meuse west of Stein, Netherlands. Tesis Doctoral, Dept. de Geografía Física, Universidad de Utrecht

topográfico de holandés), las cuatro siguientes concentraciones en partes por millón de metales pesados, la siguiente es “elev” que es la elevación relativa sobre la llanura, la variable “dist” es la distancia GIS al río Mosa, la variable “om” es la materia orgánica del suelo; y las cuatro siguientes son variables de tipo cualitativo y por última la variable “dist.m” es la distancia en metros al Mosa.

La librería a utilizar hay que recordar en la “sp” y la manera de incorporar los datos sería a través de “listas”, “matrices” o “data.frame”, aquí vamos a utilizar las funciones “library()” y “data()” para acceder a ellos.

```
> library(sp)    # Carga la librería “sp” que contiene a “meuse”

> data(meuse)    # Carga un conjunto específico de datos; en este caso “(meuse)” desde una lista o librería
```

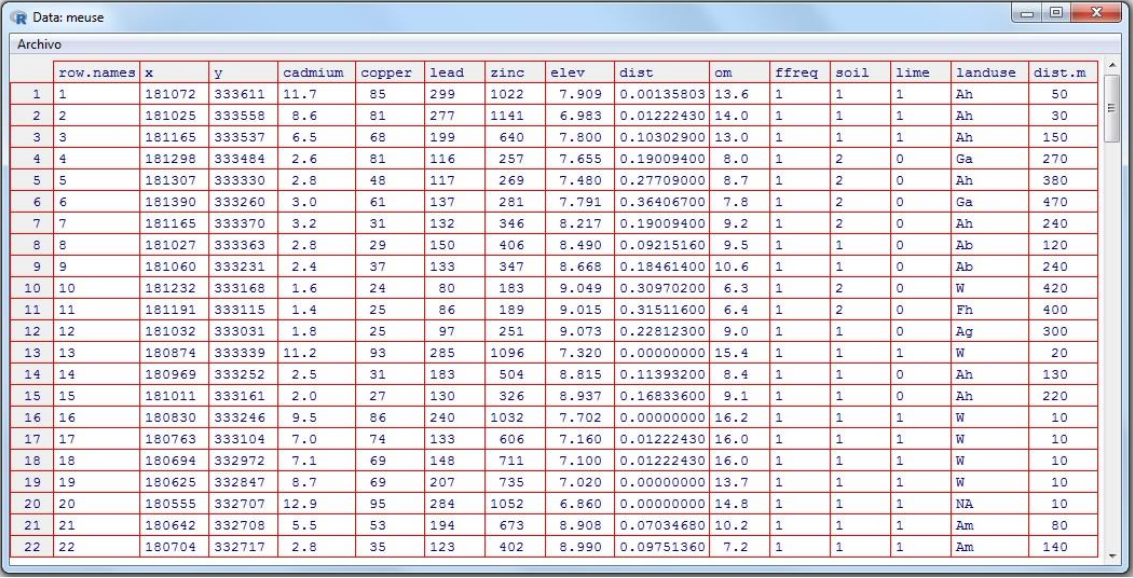
Si pedimos ver el tipo de clase al que pertenece el conjunto de datos “meuse”

```
> class(meuse)
[1] "SpatialPointsDataFrame"
attr(,"package")
[1] "sp"
```

Si queremos ver el contenido del archivo ejecutamos la función “View()” con el nombre del archivo.

```
> view(meuse)
```

En R se nos presentará la pantalla, mostrando el contenido de los datos similar a la siguiente:



	row.names	x	y	cadmium	copper	lead	zinc	elev	dist	om	ffreq	soil	lime	landuse	dist.m
1	1	181072	333611	11.7	85	299	1022	7.909	0.00135803	13.6	1	1	1	Ah	50
2	2	181025	333558	8.6	81	277	1141	6.983	0.01222430	14.0	1	1	1	Ah	30
3	3	181165	333537	6.5	68	199	640	7.800	0.10302900	13.0	1	1	1	Ah	150
4	4	181298	333484	2.6	81	116	257	7.655	0.19009400	8.0	1	2	0	Ga	270
5	5	181307	333330	2.8	48	117	269	7.480	0.27709000	8.7	1	2	0	Ah	380
6	6	181390	333260	3.0	61	137	281	7.791	0.36406700	7.8	1	2	0	Ga	470
7	7	181165	333370	3.2	31	132	346	8.217	0.19009400	9.2	1	2	0	Ah	240
8	8	181027	333363	2.8	29	150	406	8.490	0.09215160	9.5	1	1	0	Ab	120
9	9	181060	333231	2.4	37	133	347	8.668	0.18461400	10.6	1	1	0	Ab	240
10	10	181232	333168	1.6	24	80	183	9.049	0.30970200	6.3	1	2	0	W	420
11	11	181191	333115	1.4	25	86	189	9.015	0.31511600	6.4	1	2	0	Fh	400
12	12	181032	333031	1.8	25	97	251	9.073	0.22812300	9.0	1	1	0	Ag	300
13	13	180874	333339	11.2	93	285	1096	7.320	0.00000000	15.4	1	1	1	W	20
14	14	180969	333252	2.5	31	183	504	8.815	0.11393200	8.4	1	1	0	Ah	130
15	15	181011	333161	2.0	27	130	326	8.937	0.16833600	9.1	1	1	0	Ah	220
16	16	180830	333246	9.5	86	240	1032	7.702	0.00000000	16.2	1	1	1	W	10
17	17	180763	333104	7.0	74	133	606	7.160	0.01222430	16.0	1	1	1	W	10
18	18	180694	332972	7.1	69	148	711	7.100	0.01222430	16.0	1	1	1	W	10
19	19	180625	332847	8.7	69	207	735	7.020	0.00000000	13.7	1	1	1	W	10
20	20	180555	332707	12.9	95	284	1052	6.860	0.00000000	14.8	1	1	1	NA	10
21	21	180642	332708	5.5	53	194	673	8.908	0.07034680	10.2	1	1	1	Am	80
22	22	180704	332717	2.8	35	123	402	8.990	0.09751360	7.2	1	1	1	Am	140

Fig. 34 Contenido en R del Archivo “meuse”

Análogamente, podemos con el mismo comando, ver el contenido en RStudio, la ventana se presentará en el área de archivos de la forma siguiente.

	meuse	y	cadmium	copper	lead	zinc	elev	dist	om	ffreq	soil	lime	landuse	dist.m	optional
1	181072	333611	11.7	85	299	1022	7.909	0.001358003	13.6	1	1	1	Ah	50	TRUE
2	181025	333558	8.6	81	277	1141	6.983	0.01222430	14.0	1	1	1	Ah	50	TRUE
3	181145	333537	6.5	68	199	140	7.800	0.10302900	13.0	1	1	1	Ah	150	TRUE
4	181290	333604	2.6	81	116	257	7.655	0.19009400	8.0	1	2	0	Ga	270	TRUE
5	181307	333330	2.8	48	117	269	7.480	0.27709000	8.7	1	2	0	Ah	380	TRUE
6	181390	333280	3.0	61	137	281	7.791	0.36406700	7.8	1	2	0	Ga	470	TRUE
7	181165	333370	3.2	31	132	346	8.217	0.19009400	9.2	1	2	0	Ah	240	TRUE
8	181027	333363	2.8	29	150	406	8.490	0.09215160	9.5	1	1	0	Ab	120	TRUE
9	181060	333231	2.4	37	133	347	8.668	0.18461400	10.6	1	1	0	Ab	240	TRUE
10	181232	333168	1.6	24	80	183	9.049	0.30970200	6.3	1	2	0	W	420	TRUE
11	181191	333115	1.4	25	86	189	9.015	0.31511600	6.4	1	2	0	Ph	400	TRUE
12	181032	333031	1.8	25	97	251	9.073	0.22812300	9.0	1	1	0	Ag	300	TRUE
13	180874	333339	11.2	93	265	1096	7.320	0.00000000	15.4	1	1	1	W	20	TRUE
14	180989	333252	2.5	31	183	506	8.815	0.11393200	8.4	1	1	0	Ah	130	TRUE
15	181011	333161	2.0	27	136	326	8.937	0.16835600	9.1	1	1	0	Ah	220	TRUE
16	180830	333246	9.5	86	240	1032	7.702	0.00000000	16.2	1	1	1	W	10	TRUE
17	180763	333104	7.0	74	133	606	7.160	0.01222430	16.0	1	1	1	W	10	TRUE
18	180694	332972	7.1	69	148	711	7.100	0.01222430	16.0	1	1	1	W	10	TRUE
19	180625	332847	8.7	69	207	735	7.020	0.00000000	13.7	1	1	1	W	10	TRUE
20	180555	332707	12.9	95	284	1052	6.860	0.00000000	14.8	1	1	1	W	10	TRUE
21	180642	332708	5.5	53	194	673	8.908	0.07034680	10.2	1	1	1	Am	80	TRUE
22	180704	332717	2.8	35	123	402	8.990	0.09751360	7.2	1	1	1	Am	140	TRUE
23	180704	332664	2.9	35	110	343	8.830	0.11393200	7.2	1	1	1	Ag	160	TRUE

Fig. 35 Contenido en RStudio del Archivo “meuse”

- Representación en puntos y resumen estadístico de los datos espaciales mediante un archivo de datos

Una vez cargados en el sistema, se procede a extraer las localizaciones, mediante la función “coordinates()”. La función “coordinates()”, crea un objeto de datos espacial ya sea una matriz, una lista o un data.frame por medio de una asignación. La expresión para realizar esto con el archivo “meuse” es la siguiente:

```
> coordinates(meuse)<-c("x","y") # Crea un objeto de datos espacial:
coordinates(object) <- value
```

Ahora solo debemos de hacer uso de una representación gráfica mediante “plot()”, para representar los puntos de las coordnadas (x,y) del vector-objeto espacial que han sido creadas.

```
> plot(meuse,pch=16,col=2) # Presenta los puntos espaciales con el t
ipo de puntos "pich" de 16 en color rojo
```

La imagen de la localización de los datos espaciales que conforman las coordenadas son los siguientes:



Fig. 36 Imagen de las localizaciones de los datos del fichero “meuse”

El resumen estadístico de estos datos espaciales, se obtiene a través de la función “summary()”. Por lo tanto para los datos espaciales contenidos en “meuse”, se presenta la siguiente información.

```
> summary(meuse)
Object of class SpatialPointsDataFrame
Coordinates:
  min      max
x 178605 181390
y 329714 333611
Is projected: NA
proj4string : [NA]
Number of points: 155
Data attributes:
  cadmium      copper      lead      zinc      elev      dist      om
Min.   : 0.200  Min.   : 14.00  Min.   : 37.0  Min.   : 113.0  Min.   : 5.180  Min.   : 0.00000  Min.   : 1.000
1st Qu.: 0.800  1st Qu.: 23.00  1st Qu.: 72.5  1st Qu.: 198.0  1st Qu.: 7.546  1st Qu.: 0.07569  1st Qu.: 5.300
Median : 2.100  Median : 31.00  Median :123.0  Median : 326.0  Median : 8.180  Median : 0.21184  Median : 6.900
Mean   : 3.246  Mean   : 40.32  Mean   :153.4  Mean   : 469.7  Mean   : 8.165  Mean   : 0.24002  Mean   : 7.478
3rd Qu.: 3.850  3rd Qu.: 49.50  3rd Qu.:207.0  3rd Qu.: 674.5  3rd Qu.: 8.955  3rd Qu.: 0.36407  3rd Qu.: 9.000
Max.   :18.100  Max.   :128.00  Max.   :654.0  Max.   :1839.0  Max.   :10.520  Max.   : 0.88039  Max.   :17.000
NA's   :2

ffreq soil  lime      landuse      dist.m
1:84  1:97  0:111  w      :50  Min.   : 10.0
2:48  2:46  1: 44  Ah     :39  1st Qu.: 80.0
3:23  3:12      Am  :22  Median : 270.0
      Fw   :10  Mean   : 290.3
      Ab   : 8  3rd Qu.: 450.0
      (other):25 Max.   :1000.0
      NA's  : 1
```

Se observa un resumen del análisis estadístico de estos datos espaciales, en cada una de las variables observadas. Así queda conformada la representación mediante puntos.

Por otra parte, además de los datos de las localizaciones de los puntos (x,y), en donde se produjeron las observaciones, también se dispone en la librería “sp”, de las coordenadas del propio río Mosa en el archivo “meuse.riv”, el cual es una matriz de datos, de (176 x 2). Así que se procede a extraer las localizaciones de este archivo que viene incorporado en el CRAN.

```
> data(meuse.riv)      # Cargar el archivo meuse.riv
```

Si presentamos la clase de este archivo tendremos lo siguiente

```
> class(meuse.riv)
[1] "matrix"
```

Para tener un vistazo del contenido de este archivo se utiliza la función “view()”

```
> View(meuse.riv)      # Presenta el archivo en la ventana de archivo
> meuse.riv            # Presenta el archivo en la ventana de consola
```

Ahora se procede a presentar la representación gráfica de los datos espaciales, mediante la función “plot()”

```
> plot(meuse.riv,type="l",col=3,xlab=" ",ylab=" ") # presenta los pun
tos espaciales de tipo "l = líneas" en color "3", sin etiquetas en las
coordenadas, se deja un espacio vacío.
```

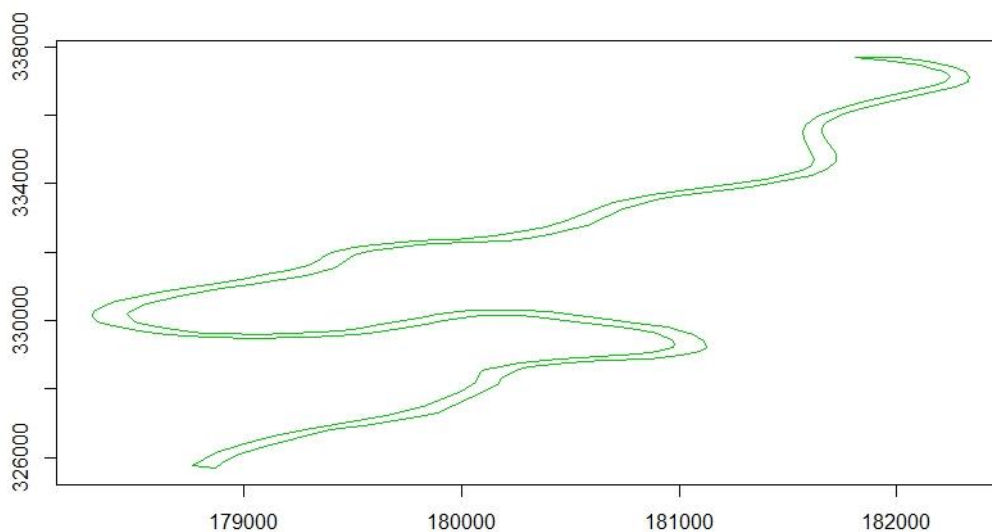


Fig. 37 Imagen de la representación en polígonos del río Mosa

El parámetro “type”, ha sido establecido a “l” para dibujar líneas. Este tipo particular de presentación más similar a un mapa es al que se le denomina “representación en polígonos”. Como se puede observar, teniendo almacenados las coordenadas de localización es bastante sencillo trazar mediante los datos espaciales las imágenes.

- Representación por medio de líneas

La representación de datos espaciales se puede realizar a través de líneas, los objetos creados o manipulados de esta clase pertenecen al tipo de clase “SpatialLines”, estos tipos en R se manejan a través de la librería “sp”, que es el paquete para manejar “datos espaciales” en R.

```
> lineas <- SpatialLines(list(Lines(list(Line(coordinates(meuse))), ID =
"s1")))
> plot(lineas, col = 4)
```

Como se puede observar, se han incluido ya dentro de un solo comando las funciones “Lines()”, y “Line()”, y se han designado como segmentos de la función. Hay que recordar que la función “Lines()”, debe de llevar un identificador único, para construir el objeto, que posteriormente va a utilizar la función que crear el objeto de datos espaciales, para líneas “SpatialLines”. Lo único que hay que hacer es construir la gráfica de las líneas mediante la función “plot()”. La gráfica resultante se observa en la siguiente imagen.

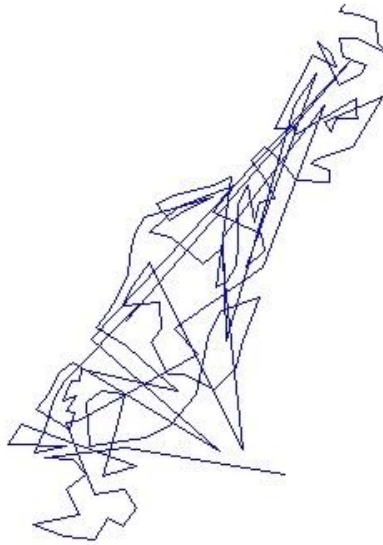


Fig. 38 Imagen de la representación en líneas de las localizaciones del archivo “meuse”

- Representación por medio de Redes (Grids)

Si lo que se quiere representar es un área, basta con tener localizaciones de ella, de manera que la representación de esa gran cantidad de puntos dará la sensación de una representación de toda la zona. Este tipo de gráfica se denomina “Representación en Redes”.

Ejemplo 9

El conjunto de ficheros “meuse”, tiene muchas coordenadas de la zona en donde se hicieron las observaciones. Estas coordenadas se encuentran en el archivo “meuse.grid”.

El primer paso es extraer las coordenadas utilizando la función “data()”

```
> data(meuse.grid)      # Cargar el archivo meuse.grid
```

Ahora se procede a extraer la coordenadas espaciales utilizando “coordenates()”

```
> coordinates(meuse.grid) <-c("x","y")  # Extrae las coordenadas espaciales
```

Podemos ver las características de este archivo.

```
> mode(meuse.grid)
[1] "list"

> class(meuse.grid)
[1] "data.frame"

> length(meuse.grid)
[1] 7
```

Es un “data.frame”, con 7 variables que contienen 3,103 observaciones.

Se puede representar en estos momentos un área mediante la función “plot()”, aplicándolo a estas coordenadas, sin embargo su representación sería muy pobre. R tiene la posibilidad de realizar mejores representaciones mediante la función “image()”, sin embargo esta función solo admite objetos, es decir datos, del tipo “SpatialPixels”, por lo que a las coordenadas espaciales extraídas mediante “coordinates()”, las convertimos ahora en objetos de tipo imagen por medio de la función “as()”.

```
> zona <- as(meuse.grid, "SpatialPixels")
```

Ahora mediante la función “image()” se puede representar los objetos obtenidos.

```
> image(zona,col ="blue")
```

La imagen resultante es la siguiente:

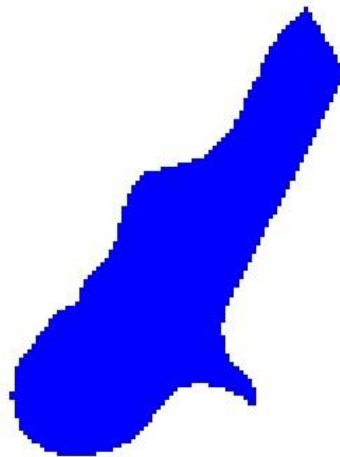


Fig. 39 Imagen de la zona de localización de los datos

Ahora, se puede representar a la vez el río, la zona donde se produjeron las localizaciones y las mismas localizaciones, comenzando los tres gráficos con la zona y utilizando el argumento “add=TRUE” en la función “plot()”.

Para representar la zona y las localizaciones basta con ejecutar la función “image()”. Aquí la secuencia de las expresiones a utilizar.

```
> image(zona,col ="blue")    # Crea la imagen de la zona de las observaciones  
  
> plot(meuse,pch=16,col=2,add=TRUE) # Dibuja las localizaciones y con  
add = TRUE añade el gráfico plot al anterior  
  
> rio<-SpatialPolygons(list(Polygons(list(Polygon(meuse.riv)),"meuse.riv")))  
# Crea la línea del río
```



```
> plot(rio, col=3, add = TRUE)           # Dibuja la línea creca del río  
o en el gráfico con add
```

La imagen que representa todos los datos espaciales es la siguiente.

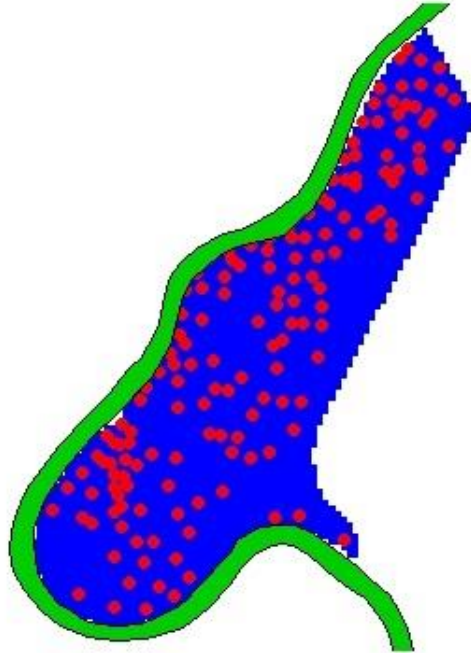


Fig. 40 Imagen de las localizaciones con la zona de localizaciones y el río

Por medio de este ejemplo vemos como R tiene la capacidad de incorporar lo que se denominan “capas”. En la siguiente unidad, veremos como QGIS realiza este mismo proceso. El software QGIS es mucho más intuitivo y sencillo en la representación de datos espaciales.

Finalizamos este apartado indicando que los puntos espaciales de las localizaciones, que se describieron del archivo “meuse”, también se pueden representar manualmente a través de una estructura de datos o “data.frame”, veamos a continuación un ejemplo sencillo de esto.

Ejemplo 13

Se tienen las coordenadas de latitud, longitud y el identificador de los datos de escuelas ubicadas en un área de estudio.

ID	Latitud	Longitud
1	13	-32
2	14	-24
3	17	-35
4	12	-33

5	15	-36
6	18	-34
7	10	-31
8	18	-36
9	17	-37
10	13	-31
11	14	-34
12	16	-30

Se pide, crear una estructura de datos que contenga los datos espaciales, extraer las localizaciones y trazar un gráfico que represente dichas localizaciones.

Lo primero es crear los vectores que contendrán las variables de observación.

Creación del vector de latitud

```
> lat=c(13,14,17,12,15,18,10,18,17,13,14,16)
```

Creación del vector de longitud

```
> lon=c(-32,-24,-35,-33,-36,-34,-31,-36,-37,-31,-34,-30)
```

Creación del vector de Identificación (id)

```
> id=c(1,2,3,4,5,6,7,8,9,10,11,12)
```

Creación del “data.frame” a partir de los vectores creados

```
> Local <- data.frame(lat=c(13,14,17,12,15,18,10,18,17,13,14,16), lon=
c(-32,-24,-35,-33,-36,-34,-31,-36,-37,-31,-34,-30),id=c(1,2,3,4,5,6,7,
8,9,10,11,12))
```

Extracción de las localizaciones

```
> coordinates(Local)=c("lon","lat") Extracción de localizaciones
```

Creación de la imagen espacial a partir de los datos de observaciones

```
> plot(Local, pch = 16, col = "red")
```

La imagen resultante de los puntos espaciales será la siguiente

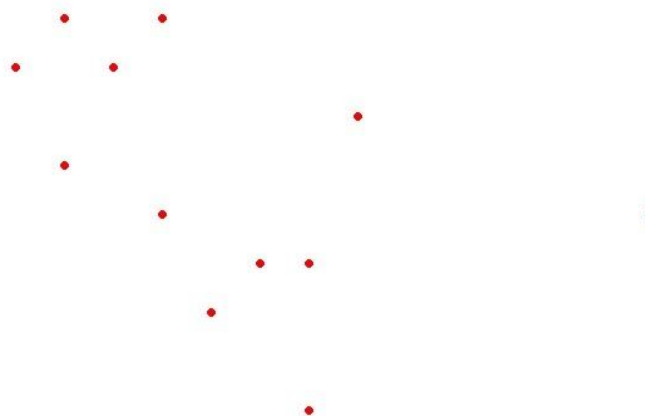


Fig. 41 Imagen de las localizaciones del área de datos

El resumen del análisis estadístico de estos datos espaciales, se obtiene a través de la función “summary()”, que vemos a continuación.

```
> summary(Local)
Object of class SpatialPointsDataFrame
Coordinates:
      min max
lon -37 -24
lat  10  18
Is projected: NA
proj4string : [NA]
Number of points: 12
Data attributes:
      id
Min.   : 1.00
1st Qu.: 3.75
Median : 6.50
Mean   : 6.50
3rd Qu.: 9.25
Max.   :12.00
```

Se observan las coordenadas en latitudes y longitudes mínima y máxima (10, -37: 18 - 24). Se presenta además el valor del 1re y 3re Quantil, y los valores de la media y la mediana de estas coordenadas espaciales.

6) Introducción a los Procesos puntuales Espaciales

De acuerdo con lo expresado por de Cabrero Ortega y García Pérez, sobre los modelos puntuales espaciales: “*Los modelos puntuales espaciales (Spatial Point Patterns) en un inicio fueron utilizados en la década de los años 30 del siglo pasado por botánicos y ecólogos para determinar, por ejemplo, la distribución espacial de los datos y sus causas en unas determinadas especies de estudio, o para comparar la hipótesis de que si era admisible que dos especies se encontraban igualmente distribuidas. Sin embargo, en la actualidad, estos modelos son utilizados en una gran diversidad de campos, como la arqueología, la salud, la educación, astronomía, entre otros. Así por ejemplo es posible analizar si los casos de una determinada enfermedad se encuentran distribuidos geográficamente según un determinado modelo. Prácticamente, todos los casos observables van a ser de tipo pares de datos (x_i, y_i) , y en caso de que se requiera comparar poblaciones, se dispondrá de marcas asociadas que identifiquen las poblaciones a comparar*”⁵.

De esta forma, los tres propósitos para los que se utilizan los “*Procesos Puntuales Espaciales*” indicados por los mismos autores son:

- a) Analizar la distribución Espaciales: Este punto se refiere a analizar la distribución que presentan los datos espaciales, para comprobar si se encuentran distribuidos aleatoriamente, es decir, al azar y sin ningún modelo que rijas las localizaciones observadas; o si están distribuidas regularmente, es decir, si se encuentran distribuidos igualmente espaciadas (uniformemente), y por último, si las localizaciones de datos espaciales se encuentran distribuidas formando grupos o clusters.

⁵ Cabrero Yolanda, García Alfonso. (2015) *Análisis estadístico de datos espaciales con QGIS y R*. Madrid, Librería UNED (Pág. 86)

- b) Analizar la densidad espacial: Este término se refiere a analizar el número de elementos espaciales por unidad de área.
 - c) Análisis de marcas: Este término se refiere a analizar las marcas que presentan los datos espaciales, por ejemplo, el comparar dos variables, o especies de una muestra de observaciones.
- Análisis de la Distribución Espacial.

Los datos que de los ejemplos que se van a utilizar para este apartado se encuentran contenidos dentro de la librería “spatstat”, que es un paquete para el análisis estadístico de datos espaciales. Vamos a introducir los aspectos básicos de dicho paquete, y en ello se va a trabajar con los archivos de datos que se llaman: “cells”, “japanesepines” y “redwood”.

Para acceder a esta librería, hay que recordar que se debe de invocar mediante la función “library()”

```
> library (spatstat)
Loading required package: spatstat.data

Attaching package: 'spatstat.data'

The following object is masked _by_ '.GlobalEnv':

    redwoodfull.extra

Loading required package: nlme
Loading required package: rpart

spatstat 1.63-0      (nickname: 'space camouflage')
For an introduction to spatstat, type 'beginner'
```

Ejemplo 14:

En RStudio, se va a proceder a trabajar con los datos del archivo “cells”, para ello se deben de cargar los datos mediante la función “data()”

```
> data(cells)      # Carga los datos de cells
```

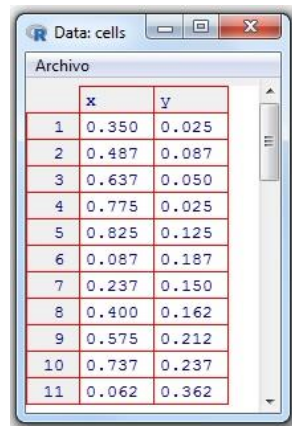
Una descripción de este archivo lo obtenemos con:

```
> mode(cells)
[1] "list"
> class(cells)
[1] "ppp"
> length(cells)
[1] 5
> cells
Planar point pattern: 42 points
window: rectangle = [0, 1] x [0, 1] units
```

Se observa que es una lista y que la clase es de tipo “ppp” (planar point pattern), Es un patrón de puntos con 42 observaciones en una matriz plana de uno por uno, (x,y) Para observar el contenido utilizamos la instrucción “View()”

```
> view(cells)
```

Se presenta la tabla con el contenido del archivo



	x	y
1	0.350	0.025
2	0.487	0.087
3	0.637	0.050
4	0.775	0.025
5	0.825	0.125
6	0.087	0.187
7	0.237	0.150
8	0.400	0.162
9	0.575	0.212
10	0.737	0.237
11	0.062	0.362

Fig. 42 Imagen de la tabla de “cells”

Este conjunto de datos representan la localización de los centros de 42 células observadas bajo el microscópico, pero al decir células, podemos decir: hospitales, escuelas, o centros en donde se han realizado campañas de vacunación, etc. El caso es que representan las localizaciones de 42 elementos. Estos datos han sido recopilados F.H.C. Crick, uno de los descubridores del ADN y Ripley, véase el trabajo de Ripley⁶ en 1977.

Ahora bien, vamos a representar de forma gráfica los datos de este archivo

```
> plot(cells, pch=16, main="Distribución espacial de las células")
```

El gráfico generado es:

Distribución espacial de las células

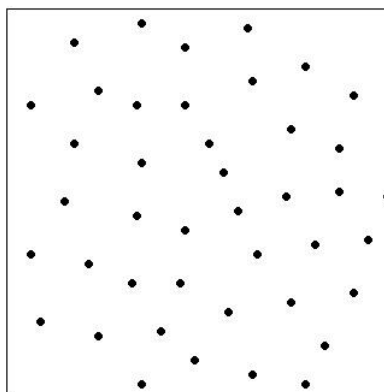


Fig. 43 Gráfico de la distribución espacial de las células

El gráfico sugiere que los datos se encuentran distribuidos regularmente sobre el cuadrado unidad. Es decir, los datos siguen el modelo de estar igualmente espaciados.

⁶ Ripley, B.D. (1977). Modelling spatial patterns (with discussion). Journal of the Royal Statistical Society, Serie B, 39, 172-2012

Ahora vamos a cargar los datos del archivo “japanesepines”, que representa las localizaciones de 65 pinos negros japoneses realizados por Numata⁷ (1961) re-escalados a un cuadrado de lado original.

```
> data(japanesepines)
```

```
> plot(japanesepines, pch= 17, main ="Distribución espacial de los pinos Japoneses")
```

Distribución espacial de los pinos Japoneses

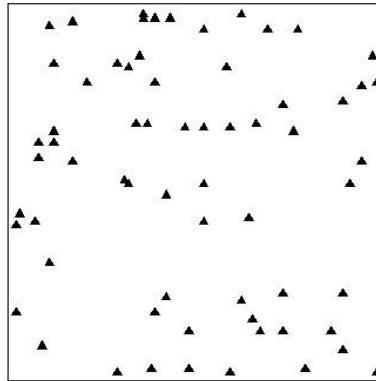


Fig. 44 Gráfico de la distribución espacial de los pinos japoneses

Ahora vamos a cargar los datos de las ubicaciones de 62 secuoyas de California en una región muestral cuadrado. Los datos originales eran 195, procedentes de Strauss (1975), pero se suelen estudiar los 62 que aquí se presentan y que son recopilados en el trabajo de Ripley(1977) en una subregión que se ha re-escalado a un cuadrado unidad.

```
> data(redwood)
```

```
> plot(redwood, pch= 18, main ="Distribución espacial de las secuoyas californianas")
```

Distribución espacial de las secuoyas californianas

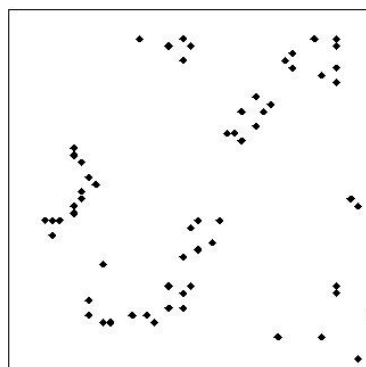


Fig. 45 Gráfico de la distribución espacial de las secuoyas de California

⁷ Numata, M. (1961). Forest vegetation in the vicinity of Choshi. Coastal flora and vegetation at Choshi, Chiba Prefecture. IV. Bulletin of Choshi Marine Laboratory, Chiba University, **3**, 28–48 (en Japonés).

- Procesos Puntuales y Aleatoriedad Espacial Completa (CSR)

Ahora se van a analizar los procesos puntuales espaciales de estas observaciones. Mencionaremos que un proceso Puntual Espacial es un proceso estocástico que genera localizaciones de algunos sucesos de interés dentro de una región concreta en estudio. A su vez, un Modelo Espacial Puntual se define como las localizaciones de los sucesos generados por un proceso puntual en el área de estudio. Si las localizaciones tienen marcas para distinguir varios grupos de datos, se está refiriendo entonces a Procesos y Modelo Espacial Puntual con Marcas.

Hemos mencionado que el primer objetivo dentro del análisis de la distribución espacial de las localizaciones es averiguar si estas localizaciones se encuentran distribuidas al azar en la región de estudio, es decir, que se presente una aleatoriedad en su distribución y que implica que no existe ningún patrón que regule su ubicación. Esto es lo que se denomina “Aleatoriedad Espacial Completa (Complete Spatial Randomness)”, o conocida por sus siglas CSR y se puede decir que es el “ruido blanco” de un proceso puntual espacial, pues caracteriza la ausencia de estructura en los datos. Generalmente es la hipótesis nula de un test estadístico para determinar cuándo hay estructura espacial en un diseño puntual dado.

Para analizar si los datos siguen o no una CSR se formaliza mediante un proceso de Poisson homogéneo de parámetro λ . Esto se puede realizar de dos formas, mediante cuadrados (cuadrats) o por medio de las distancias. Se sabe que los test basados en recuentos de observaciones son menos precisos que los basados en las propias observaciones. De esta forma, para analizar la CSR vamos a considerar métodos basados en las distancias.

Aunque hay varias posibilidades de calcular las distancias, suele utilizarse la distancia (Euclídea), entre una localización y la localización vecina más cercana (nearest-neighboring). De esta forma, se puede demostrar que si las localizaciones se encuentran generadas por un proceso de Poisson homogéneo de parámetro λ , es decir, al azar, la distribución de esas distancias viene dada por la siguiente función lineal.

$$g(w) = 2 \pi \lambda w e^{-2\pi w^2} \quad w > 0$$

O por equivalencia, por la función de distribución:

$$G(w) = 1 - e^{-2\pi w^2} \quad w > 0$$

Por lo tanto, las localizaciones observadas se encontrarán generadas al azar, es decir, sin seguir ningún patrón, si las diferencias entre su función de distribución empírica y este modelo teórico G no son significativas.

Si se representan por d_{ij} la distancia Euclídea entre dos localizaciones (i y j), la distancia entre una localización i y la localización vecina más cercana va a ser:

$$d_i = \min_j \{d_{ij}, \text{ con } j \neq i\}, \text{ para } i = 1, \dots, n$$

Por lo tanto, fijada una distancia w , el estimador de $G(w)$ será la función de distribución empírica

$$\hat{G}(w) = \frac{\text{número de } d_i \leq w}{n}$$

En donde generalmente i y j serán vectores de dos o tres dimensiones. Ahora se calcularán y presentarán los pares $(G(w), \hat{G}(w))$ para las observaciones de los tres grupos de datos espaciales presentados.

Para realizar esto se necesitan cargar las librerías: "lattice" y "spatstat". El paquete "lattice", es un paquete gráfico que contiene la función "xyplot()", que permitirá construir la gráfica de la SCR.

NOTA: Para ver la sintaxis de "xyplot()", así como de muchas otras, hay que recordar que se teclea (> ? Xyplot), lo que presentará la ayuda con la referencia y documentación de la función.

```
> library(lattice)
```

```
Attaching package: 'lattice'
```

```
The following object is masked from 'package:spatstat':
```

```
    panel.histogram
```

```
> library(spatstat)
```

Ahora cargamos la otra librería

```
> library(spatstat)
```

Construimos el vector de secuencias a utilizar mediante la función "seq()", que genera secuencias regulares, en este caso desde cero, hasta un valor de (la raíz cuadrada de 2 / 6 = 0.2357023) con intervalos de crecimiento de 0.005.

```
> r<-seq(0,sqrt(2)/6,by=0.005)
```

Vemos las secuencias generadas.

```
> r
[1] 0.000 0.005 0.010 0.015 0.020 0.025 0.030 0.035 0.040 0.045 0.050
0.055 0.060 0.065 0.070 0.075 0.080 0.085 0.090
[20] 0.095 0.100 0.105 0.110 0.115 0.120 0.125 0.130 0.135 0.140 0.145
0.150 0.155 0.160 0.165 0.170 0.175 0.180 0.185
[39] 0.190 0.195 0.200 0.205 0.210 0.215 0.220 0.225 0.230 0.235
```

Se construyen grupos de simulaciones de CSR, mediante la función "envelope". Mediante la función "as" se transforman los datos "japanesepines" en formato "ppp", ya que la función "envelope()" solo admite datos del tipo "ppp", se hará lo mismo con los otros dos conjuntos de datos.

Para los pinos japoneses sería:

```
> Pinojapo<-envelope(as(japanesepines,"ppp"),fun=Gest,r=r,nrank=2,nsim=99)
Generating 99 simulations of CSR ...
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38
, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55,
56, 57, 58, 59, 60, 61,
62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79
, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96,
97, 98, 99.
Done.
```

Se presentan los patrones de simulación de la CSR y finaliza con “done (hecho)”, Como se mencionó mediante la función “envelope()” se fuerza a que los datos espaciales de entrada sean de formato “ppp”, por medio de la función “as()” y donde los argumentos:

fun: Representa a la función que calcula la estadística de resumen deseada para un patrón de puntos, en este caso Geoestadística.

nrank: Es un entero. Determina la clasificación del valor del envolvente entre los valores simulados nsim. O lo que es lo mismo, fija el coeficiente de confianza entre los entornos de confianza, que establecemos a 2 (es decir, quitamos 2 a cada lado) sobre 100 simulaciones nsim = 99. Eso indica que tendremos entornos de confianza del 96%. Un rango de 1 significa que se utilizarán los valores simulados mínimo y máximo.

nsim: Representa el número de patrones de puntos simulados que se generarán al calcular las envolturas, dispuesto a 99.

Como se observa, al ejecutar la expresión, se generan 99 patrones de simulación de CSR. Ahora construimos los patrones para las secuoyas y las células

```
> Cecuo<-envelope(as(redwood,"ppp"),fun=Gest,r=r,nrank=2,nsim=99)
Generating 99 simulations of CSR ...
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38
, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55,
56, 57, 58, 59, 60, 61,
62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79
, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96,
97, 98, 99.
Done.
```

```
> celulas<-envelope(as(cells,"ppp"),fun=Gest,r=r,nrank=2,nsim=99)
Generating 99 simulations of CSR ...
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38
, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55,
56, 57, 58, 59, 60, 61,
62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79
, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96,
97, 98, 99.
Done.
```

Una vez generada las secuencias, mediante las funciones “rbind()” y “cbind()” se combinan los objetos en filas y columnas. Estos se incluyen en el valor del resultado

```
> resultado1<-rbind(Pinojapo,Cecuo,celulas) # combina los datos
```

La función “rbind()” permite la combinación de objetos por filas y columnas

Al presentar el contenido del cálculo de resultado1 se visualiza lo siguiente

```
> resultado1
Pointwise critical envelopes for G(r)
and observed value for 'as(japanesepines, "ppp")'
Edge correction: "km"
Obtained from 99 simulations of CSR
Alternative: two.sided
Significance level of pointwise Monte Carlo test: 4/100 = 0.04
.....
      Math.label      Description
r      r              distance argument r
obs  hat(G)[obs](r)  observed value of G(r) for data pattern
theo G[theo](r)      theoretical value of G(r) for CSR
lo   hat(G)[lo](r)   lower pointwise envelope of G(r) from simulations
hi   hat(G)[hi](r)   upper pointwise envelope of G(r) from simulations
.....
Default plot formula: .~r
where "." stands for 'obs', 'theo', 'hi', 'lo'
Columns 'lo' and 'hi' will be plotted as shading (by default)
Recommended range of argument r: [0, 0.1]
Available range of argument r: [0, 0.235]
Unit of length: 5.7 metres
```

Se indica que se han obtenido 99 patrones de simulación de la CSR y se presenta el nivel de significancia mediante el test de Montecarlo = 0.04. Se nos informa, que r es el tamaño del vector de secuencias, de los objetos existentes y a trabajar como son “obs” que representan los valores observados y “theo” los valores teóricos, cuyo par vamos a graficar y que deben de estar separado por el circunflejo (~), de la forma (obs~theo). Se nos indica la información puntual alta y baja de G(r), desde las simulaciones mediante los indicadores “lo” y “hi”, se dan pautas por defecto para establecer la gráfica de estos cálculos. El rango recomendado y disponible de argumentos de r, la unidad de longitud.

Ahora, creamos un nuevo cálculo que llamamos resultado2 y que va a combinar el contenido de resultado1 con los valores replicados de las etiquetas en cada valor hasta el tamaño del vector de secuencias calculado.

```
> resultaddo2<-cbind(resultado1,DATASET=rep(c("PINOS JAPONESES","SECUO
YAS - CALIFORNIA","CELULAS"),each=length(r)))
```

En la función “cbind()” se combinan a su vez, los datos obtenidos en “resultado1” con un vector que contiene los nombres, y mediante la función “rep()”, se replican estos nombres por cada uno.

Si presentamos el valor generado de “resultado2” obtendremos lo siguiente.

```
> resultaddo2
```



```

Function value object (class 'fv')
for the function r -> G(r)
.....
      Math.label      Description
r      r              distance argument r
obs    hat(G)[obs](r) observed value of G(r) for data pattern
theo   G[theo](r)     theoretical value of G(r) for CSR
lo     hat(G)[lo](r)  lower pointwise envelope of G(r) from simulations
hi     hat(G)[hi](r)  upper pointwise envelope of G(r) from simulations
y      G[y](r)
.....
Default plot formula: .~r
where "." stands for 'y', 'hi', 'lo', 'theo', 'obs'
Recommended range of argument r: [0, 0.1]
Available range of argument r: [0, 0.235]
Unit of length: 5.7 metres

```

El valor de la función objeto de la clase “fv” para la función $r - G(r)$. Aquí también se nos informa, que r es el tamaño del vector de secuencias, de los objetos existentes y a trabajar como son “obs” que representan los valores observados y “theo” los valores teóricos, cuyo par vamos a graficar y que deben de estar separado por el circunflejo (~), de la forma (obs~theo). Se nos indica la información puntual alta y baja de $G(r)$, desde las simulaciones mediante los indicadores “lo” y “hi”, se dan pautas por defecto para establecer la gráfica de estos cálculos. El rango recomendado y disponible de argumentos de r , la unidad de longitud.

Ahora repetimos el valor creado del conjunto las Etiquetas y los replicamos por cada dato de la secuencia, a fin de que se presenten en el dibujo del gráfico a generar.

```
> DATASET=rep(c("PINOS JAPONESES","SECUOYAS - CALIFORNIA","CELULAS"),each=length(r))
```

Bien, ya solo falta presentar la gráfica por medio de la función “xyplot()”, tomando en cuenta los parámetros que se presentaron en “resultado2”, recordemos que “xyplot()”, pertenece a la librería “lattice”

```
> print(xyplot(obs~theo|DATASET,data=resultado1,type="l",panel=function(x, y,subscripts)
+ {lpolygon(c(x, rev(x)),c(resultado2$lo[subscripts],rev(resultado2$hi[subscripts])),
+         border="red", lwd=2, col = "gold", fill = T)
+   llines(x, y, col="blue", lwd=2)}))
```

Hay que recordar que los símbolos “+” que aparecen es porque la línea que contiene la función es demasiado larga y R continua la instrucción en la siguiente línea.

Se contruyen polígonos mediante la función “lpolygon()” y se utiliza la función “rev()”

La función “rev()”, proporciona una versión inversa de su argumento. Es una función genérica con un método predeterminado para vectores y uno para dendrogramas. Se construye un bode mediante líneas en color rojo y el relleno mediante el argumento “fill” se designa a TRUE es decir que el polígono estará con un relleno sólido de color “gold”.

El gráfico resultante es el siguiente:

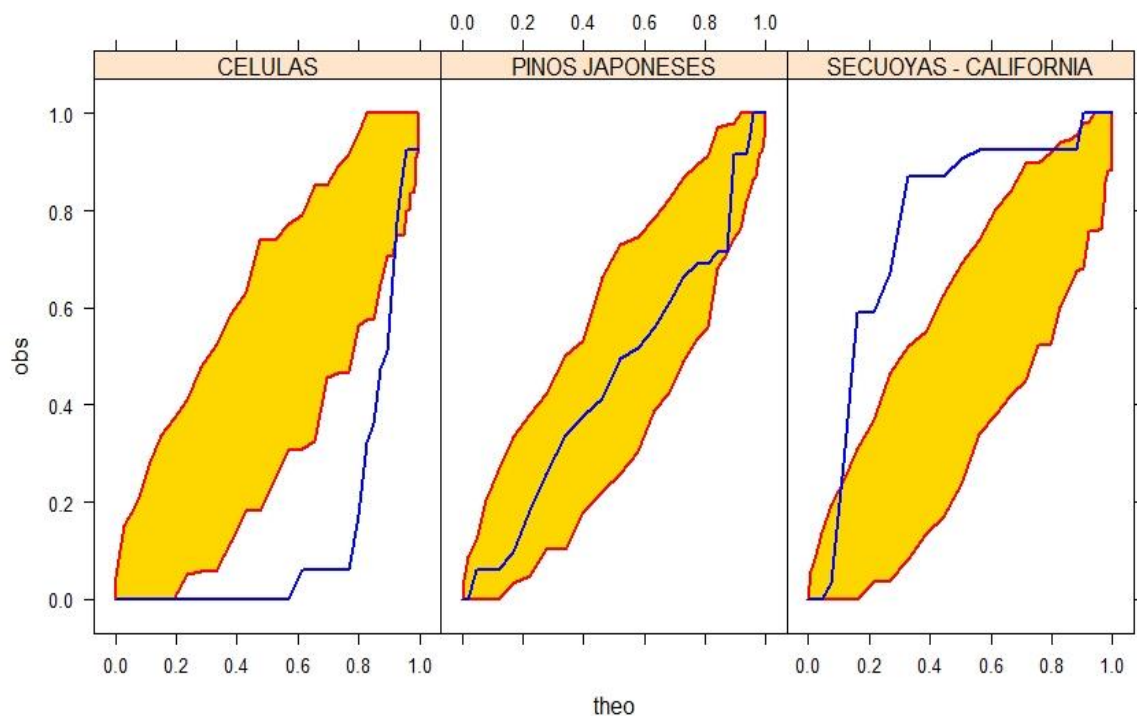


Fig. 46 Gráfico del análisis visual de la CSR

En los gráficos se observa que de los tres casos, el de los pinos japoneses es el que posee la Aleatoriedad Espacial Completa CSR

- Análisis de la CSR, a partir de datos creados

Ejemplo 15.

Ahora consideremos que no vamos a trabajar con datos pre-establecidos, sino con datos propios, de observaciones realizadas por ejemplo a un conjunto de datos de escuelas, centros de salud, etc. Detallaremos un hipotético ejemplo con pares de datos (1) y (2) que serán los pares reales (latitud y longitud) a trabajar.

Utilizamos las mismas librerías

```
> library(lattice)
> library(spatstat)
```

Creamos los vectores.

```
> Dato.Observacion1 <-c(21,22,21.2,22.4,22.8,21.7,22.3,21.5,22.4,21.9
21.2,22.2,21.4,22.6,23.0,21.9,22.5,21.7,22.6,22.1,21.5,22.5,21.7,22.9
23.3,22.2,22.8,22.0,22.9,22.4)
```

```
> Dato.Observacion2 <-c(34.1,35,33.9,34.9,35.1,33.7,33.1,33.4,33.5,33.7,33.7,34.6,33.5,34.5,34.7,33.3,32.7,33.0,33.1,33.3,34.8,35.7,34.6,35.6,35.8,34.4,33.8,34.1,34.2,34.4)
```

Para trabajar con estos grupos de datos es obvio que hay que formar una matriz de datos con dos columnas.

```
> Conjunto.Matriz <-matrix(c(Dato.Observacion1,Dato.Observacion2),ncol=2)
```

```
> Conjunto.Matriz
      [,1] [,2]
[1,] 21.0 34.1
[2,] 22.0 35.0
. . .
[29,] 22.9 34.2
[30,] 22.4 34.4
```

El análisis de la CSR se hace con datos re-escalados en el cuadrado unidad, es decir, que se debe de modificar la escala de éstos para que todos ellos tomen valores en [0,1]. Esto se consigue restando a cada dato “x”, el menor de los valores, “min(x)” y dividiendo el resultado de esta diferencia por la diferencia entre el máximo y el mínimo de los valores, es decir haciendo el cálculo siguiente:

$$\frac{x - \min(x)}{\max(x) - \min(x)}$$

El re-escalamiento se realiza en tres pasos, que son los siguientes.

```
> Reescalado_Dato1<-(Conjunto.Matriz[,1]-min(Conjunto.Matriz[,1]))/(max(Conjunto.Matriz[,1])-min(Conjunto.Matriz[,1]))
```

```
> Reescalado_Dato2<-(Conjunto.Matriz[,2]-min(Conjunto.Matriz[,2]))/(max(Conjunto.Matriz[,2])-min(Conjunto.Matriz[,2]))
```

```
> Conjunto.Matriz2<-matrix(c(Reescalado_Dato1,Reescalado_Dato2),ncol=2)
```

```
> Conjunto.Matriz2
      [,1] [,2]
[1,] 0.00000000 0.45161290
[2,] 0.43478261 0.74193548
[3,] 0.08695652 0.38709677
[4,] 0.60869565 0.70967742
. . .
[28,] 0.43478261 0.45161290
[29,] 0.82608696 0.48387097
[30,] 0.60869565 0.54838710
```

Si ahora presentamos la gráfica del conjunto de datos: “Conjunto.Matriz2” sería la siguiente:

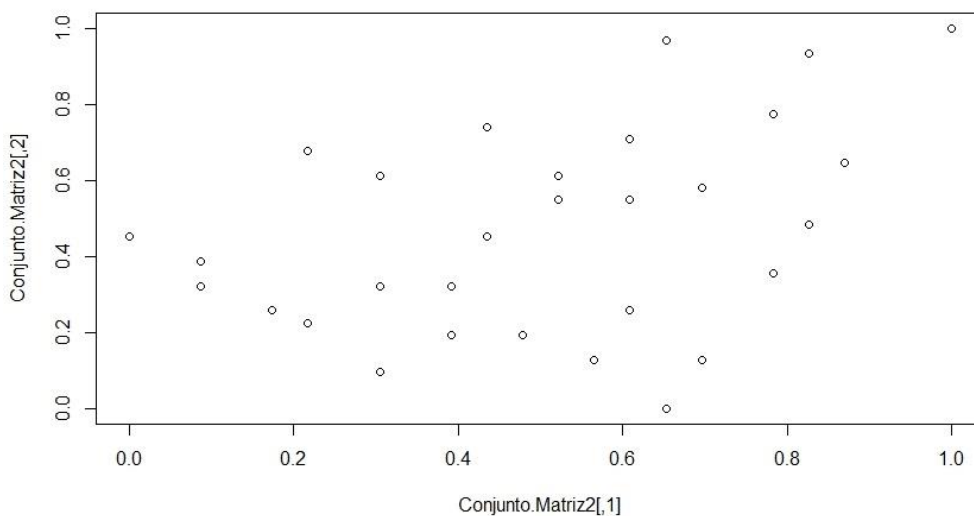


Fig. 47 Gráfico de los puntos de “Conjunto.Matriz2”

La aleatoriedad CSR de los datos se va a verificar si las diferencias entre el modelo teórico $G(w)$ y la distribución empírica $\hat{G}(w)$, no son grandes, en este caso para un conjunto de distancias razonables “w”, cuyo número fijamos a 50 entre 0 y 0.25 definida en la siguiente expresión.

```
> Distancias_W <- seq(0,0.25,len=50)
```

Debido a que el modelo teórico es difícil de manejar, lo que se hace es simular, con la función “envelope()” de la librería “spatstat”, muchas simulaciones propias (aquí podemos indicar las que queramos, indicándolo en el argumento “nsim” de “envelope” y que como el ejemplo anterior lo designamos en 99), del proceso puntual, en este caso “G”, para lo que se utiliza el argumento “fun=Gest” de “envelope”. La función “envelope”, solo admite datos del tipo “ppp”, por eso se deben de transformar los datos de la matriz a datos “ppp” primero. Fue lo mismo que se realizó con los datos del ejemplo anterior.

Sin embargo, los datos del ejemplo anterior, ya eran datos de tipo objeto espacial, estos aun no lo son, por lo que hay que transformar los datos al tipo datos espacial, mediante la función “SpatialPoint”, de la librería “sp”, y utilizar la librería “maptools”, para utilizar la transformación de esos datos que ya son espaciales en datos de tipo “ppp”.

```
> library(sp)
```

```
> Conjunto.Espacial<-SpatialPoints(Conjunto.Matriz2)
```

```
> Conjunto.Espacial
```

```
SpatialPoints:
```

```
      coords.x1  coords.x2
[1,] 0.00000000 0.45161290
[2,] 0.43478261 0.74193548
...
[29,] 0.82608696 0.48387097
[30,] 0.60869565 0.54838710
```

```
> library(maptools)

> Conjunto.Espacial2<-as(Conjunto.Espacial,"ppp")

> Conjunto.Espacial2
Planar point pattern: 30 points
window: rectangle = [0, 1] x [0, 1] units
```

Ahora bien, las distancias a considerar “w”, recordemos se van a incluir en la función “envelope()”, con el argumento “r”, de tal, forma, que con “envelope()” se obtendrán unos “entornos de confianza” entre los que debería de estar la función empírica $\hat{G}(w)$. En estos entornos se puede fijar el coeficiente de confianza mediante el argumento “nrank” que establecemos a 2 (es decir, quitamos 2 a cada lado) sobre 100 simulaciones “nsim = 99”. Eso indica que tendremos entornos de confianza del 96%.

```
> EConfianza <-envelope(Conjunto.Espacial2,fun=Gest,r=Distancias_w,nra
nk=2,nsim=99)
Generating 99 simulations of CSR ...
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38
, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55,
56, 57, 58, 59, 60, 61,
62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79
, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96,
97, 98, 99.

Done.
```

Ahora solo hay que representar la gráfica y sobre-impresionar el dibujo del entorno creado. Por lo que se debe de crear el entorno y proceder a colocar la distribución empírica $\hat{G}(w)$. La gráfica se puede realizar de diversas formas, pero una de las más prácticas es la ya formulada en el ejemplo pasado, ejecutando las siguientes instrucciones.

```
> EConfianza2<-cbind(EConfianza,DATASET=rep(c("PRUEBA"),each=length(Di
stancias_w)))

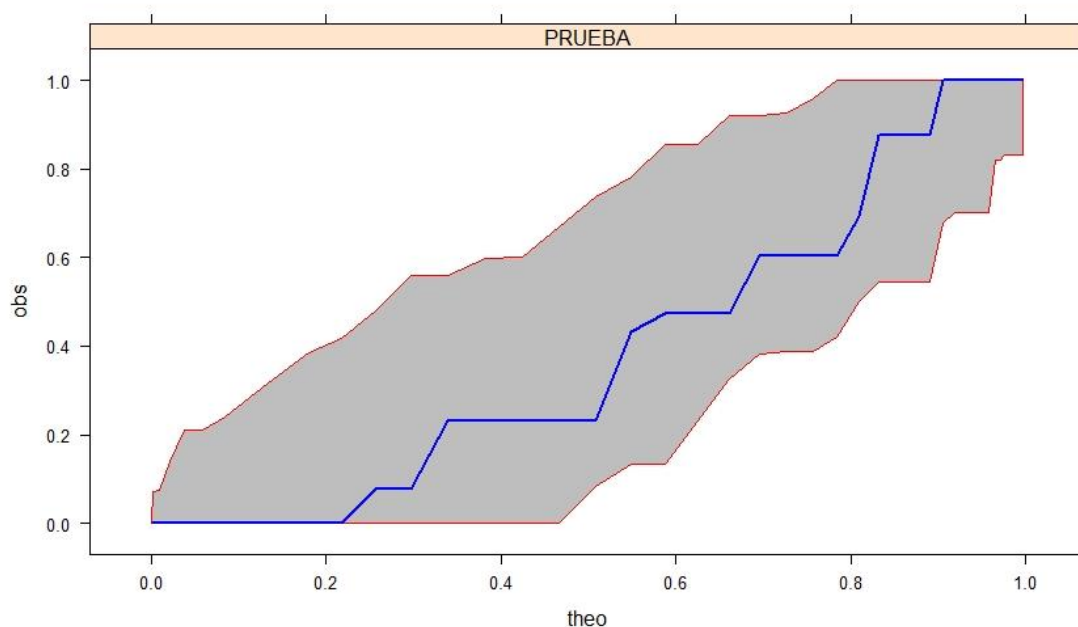
> EConfianza2
Function value object (class 'fv')
for the function r -> G(r)
.....
Math.label      Description
r               distance argument r
obs             hat(G)[obs](r) observed value of G(r) for data pattern
theo           G[theo](r)    theoretical value of G(r) for CSR
lo             hat(G)[lo](r) lower pointwise envelope of G(r) from simulations
hi             hat(G)[hi](r) upper pointwise envelope of G(r) from simulations
y              G[y](r)
.....
Default plot formula: .~r
where "." stands for 'y', 'hi', 'lo', 'theo', 'obs'
Recommended range of argument r: [0, 0.15306]
Available range of argument r: [0, 0.25]
```

```
> DATASET=rep(c("PRUEBA"),each=length(Distancias_W))
```

Y el gráfico resultante se construye mediante la expresión:

```
> print(xyplot(obs~theo|DATASET , data=EConfianza2, type="l",  
+           panel=function(x, y, subscripts)  
+           {  
+             polygon(c(x, rev(x)),  
+                     c(EConfianza2$lo[subscripts], rev(EConfianza  
2$hi[subscripts])),  
+                     border="red", col="gray", fill=T)  
+             lines(x, y, col="blue", lwd=2)}))
```

Cuya salida gráfica es la siguiente:



La conclusión al observar la figura es de que los datos fueron generados al azar.

En la siguiente unidad, se analizarán y se procederá a trabajar con datos espaciales vectoriales y raster, contando también el uso del software QGIS, para el desarrollo y análisis de la representación de datos espaciales más elaborado.

- **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)