Departamento de Estadística e Investigación Operativa Aplicadas y Calidad

Herramientas estadísticas para Big Data

Introducción a la inferencia Estadística, Muestreo y Preproceso de datos.
Scripts y ejemplos prácticos con R

Elena Vázquez Barrachina

Universitat Politècnica de València

Tabla de contenido

1 Obtención de descriptivos con R

1. Obtención de los datos y revisión de variables y tipos

Antes de analizar los datos tenemos que adquirirlos y dar un primer vistazo. Esto también formaría parte de las fases de *obtención* y *preparación de datos*.
En este apartado vamos a trabajar con el **fichero de datos** de R *JaenIndicadores.rda*[1].
Este fichero contiene datos sobre indicadores importantes de los municipios de la provincia de Jaén en el año 2001, e incluye las siguientes variables:

- Código INE del municipio.
- Nombre del municipio.
- Consumo de energía eléctrica en megavatios por hora.
- Consumo medio de agua en invierno, en metros cúbicos por día.
- Consumo medio de agua en verano, en metros cúbicos por día.
- Destino de los residuos sólidos urbanos: las posibilidades son vertedero controlado, vertedero incontrolado y compostaje.
- Cantidad de residuos sólidos urbanos, en toneladas.
- Tipo de municipio (Grande, Mediano o Pequeño)
- Otras variables creadas a partir de las anteriores.

Vamos a cargar, en primer lugar, los datos con la función `load` y ver los objetos que se han cargado en el *work space*:

```
load("JaenIndicadores.rda")

objects()
> [1] "Datos"
```

El archivo contiene un *data frame* denominado `Datos`.
¿Qué variables contiene el *data frame*?. Para un resumen de las variables, podemos usar la función `names()`:

```
names(Datos)
>  [1] "CodigoINE"
>  [2] "Municipio"
>  [3] "Consumo.de.energía.eléctrica"
>  [4] "Consumo.de.agua..Invierno"
>  [5] "Consumo.de.agua..Verano"
>  [6] "Residuos.sólidos.urbanos..Destino"
>  [7] "Residuos.sólidos.urbanos..Cantidad"
>  [8] "Población"
>  [9] "agua.hab"
> [10] "elec.hab"
> [11] "res.hab"
> [12] "Tipo"
```

Si queremos más información acerca de las variables: contenido y tipo, podemos usar la función `str()`

```
str(Datos)
> 'data.frame': 97 obs. of  12 variables:
>  $ CodigoINE                          : num  23001 23002 23003 23004 23005 ...
>  $ Municipio                          : Factor w/ 97 levels "Albanchez de
Mágina",..: 1 2 3 4 5 6 7 8 10 11 ...
>  $ Consumo.de.energía.eléctrica       : num  2165 93991 34985 853 139971 ...
>  $ Consumo.de.agua..Invierno          : num  298 4882 1537 123 8896 ...
>  $ Consumo.de.agua..Verano            : num  400 6342 2633 500 10326 ...
>  $ Residuos.sólidos.urbanos..Destino : Factor w/ 4 levels "..","Compostaje",..:
3 2 2 3 3 3 3 3 3 3 ...
>  $ Residuos.sólidos.urbanos..Cantidad: num  370 6774 3681 114 11776 ...
>  $ Población                          : num  1474 21523 11261 573 37903 ...
>  $ agua.hab                           : num  0.474 0.521 0.37 1.087 0.507 ...
```

```
>  $ elec.hab                          : num  1.47 4.37 3.11 1.49 3.69 ...
>  $ res.hab                           : num  0.251 0.315 0.327 0.198 0.311 ...
>  $ Tipo                              : Factor w/ 3 levels "Grande","Mediano",..:
3 1 1 3 1 1 2 3 1 1 ...
```

Deberíamos cambiar el nombre de las variables o usar otro fichero
Veamos ahora los valores de las diferentes variables para los 5 primeros municipios:

```
head(Datos, 5)
>   CodigoINE           Municipio Consumo.de.energía.eléctrica
> 1     23001 Albanchez de Mágina                         2165
> 2     23002      Alcalá la Real                        93991
> 3     23003           Alcaudete                        34985
> 4     23004        Aldeaquemada                          853
> 5     23005             Andújar                       139971
>   Consumo.de.agua..Invierno Consumo.de.agua..Verano
> 1                       298                    400
> 2                      4882                   6342
> 3                      1537                   2633
> 4                       123                    500
> 5                      8896                  10326
>   Residuos.sólidos.urbanos..Destino Residuos.sólidos.urbanos..Cantidad
> 1              Vertedero controlado                              370.49
> 2                         Compostaje                             6774.11
> 3                         Compostaje                             3680.95
> 4              Vertedero controlado                              113.53
> 5              Vertedero controlado                             11775.50
>   Población  agua.hab elec.hab   res.hab    Tipo
> 1      1474 0.4735414 1.468792 0.2513501 Pequeño
> 2     21523 0.5214886 4.367003 0.3147382  Grande
> 3     11261 0.3703046 3.106740 0.3268759  Grande
> 4       573 1.0872600 1.488656 0.1981326 Pequeño
> 5     37903 0.5071366 3.692874 0.3106746  Grande
```

2. Herramientas estadística unidimensional

2.1. Obtención de parámetros muestrales

Los parámetros muestrales o estadísticos muestrales son fundamentalmente de tres tipos:

- Posición o centralización
- Dispersión
- Forma

La mayoría de estos parámetros se pueden calcular con funciones del paquete *base*:

Parámetro	Función	Paquete
Promedio	`mean()`	base
Desviación típica	`sd()`	base
Varianza	`var()`	base
Cuantiles	`quantile()`	base
Mediana	`median()`	base
Coeficiente de asimetría	`skewness()`	e1071
Coeficiente de curtosis	`kurtosis()`	e1071
Covarianza	`cov(),var()`	base
Coeficiente de correlación	`cor()`	base
Resumen	`summary()`	base
Resumen (Tukey)	`fivenum()`	base

Otros parámetros podemos calcularlos con funciones estándar del paquete *base* o a partir de funciones disponibles en otras librerías.

2.1.1. Media o promedio

Comenzamos con la **media** y la función `mean()`:

```
mean(Datos$agua.hab)
> [1] NA
```

Observamos que el resultado es `NA` porque hay datos que faltan[2]. Hay que decirle a la función que los elimine del cálculo con el parámetro `na.rm`:

```
mean(Datos$agua.hab, na.rm=TRUE)
> [1] 0.5256292
```

Obtengamos la media de *elec.hab* y *res.hab*:

```
mean(Datos$elec.hab,na.rm=TRUE)
> [1] 2.655137
mean(Datos$res.hab,na.rm=TRUE)
> [1] 0.2331974
```

2.1.2. Desviación típica

```
sd(Datos$agua.hab,na.rm=TRUE)
> [1] 0.1371726
sd(Datos$elec.hab,na.rm=TRUE)
> [1] 1.402689
sd(Datos$res.hab,na.rm=TRUE)
> [1] 0.03660991
```

2.1.3. Coeficiente de variación.

```
sd(Datos$agua.hab,na.rm=TRUE)/mean(Datos$agua.hab,na.rm=TRUE)
> [1] 0.2609684
```

2.1.4. Cuantiles.

Por ejemplo, percentil 5 y percentil 95:

```
quantile(Datos$agua.hab,probs=c(0.05,0.95),na.rm=TRUE)
>         5%        95%
> 0.3934972 0.7649731
quantile(Datos$elec.hab,probs=c(0.05,0.95),na.rm=TRUE)
>        5%       95%
> 1.466623 5.185409
quantile(Datos$res.hab,probs=c(0.05,0.95),na.rm=TRUE)
>         5%        95%
> 0.1936857 0.3159832
```

Por ejemplo, 1er y 3er cuartil (Q1 y Q3)

```
sapply(Datos[,9:11], quantile, probs=c(0.25,0.75), na.rm=T)
>      agua.hab elec.hab   res.hab
> 25% 0.4598800 1.754325 0.2114574
> 75% 0.5613026 3.096219 0.2408332
```

2.1.5. Mediana

```
median(Datos$agua.hab, na.rm=T)
> [1] 0.5063754
```

2.1.6. Moda

Moda de la variable destino de los residuos sólidos urbanos

```
# Obtencion de frecuencias absolutas (ver más adelante en el script)
frec.resid<-table(Datos$Residuos.sólidos.urbanos..Destino)
frec.resid
>
>                      ..              Compostaje   Vertedero controlado
>                       1                      24                     61
> Vertedero incontrolado
>                      11
# Obtención del valor (y posición) de máxima frecuencia absoluta
moda<-frec.resid[which.max(frec.resid)]

# O bien
# frec.resid[which(frec.resid == max(frec.resid))]

# Obtención de la moda y el valor de su frecuencia
moda
> Vertedero controlado
>                   61
# Si solo queremos el valor de la moda
names(moda)
> [1] "Vertedero controlado"
```

2.1.7. Coeficiente de asimetría y curtosis.

Para estos parámetros necesitamos la `library(e1071)`

```
# Instalar el paquete Hmisc si es preciso
if(!is.element('e1071', installed.packages())) install.packages('e1071', repos =
'https://cran.rediris.es/', dependencies = T)

# Cargar paquete
library(e1071)
```

Calculemos la **Asimetría**

```
# Asimetria
skewness(Datos$agua.hab,na.rm=TRUE)
> [1] 1.219825
skewness(Datos$elec.hab,na.rm=TRUE)
> [1] 2.052142
skewness(Datos$res.hab,na.rm=TRUE)
> [1] 1.391985
```

Calculemos la **Curtosis**

```
#  Curtosis
kurtosis(Datos$agua.hab,na.rm=TRUE)
> [1] 4.315458
kurtosis(Datos$elec.hab,na.rm=TRUE)
> [1] 5.292531
kurtosis(Datos$res.hab,na.rm=TRUE)
```

```
> [1] 1.40106
```

1.1

2.2. Resúmenes de estadísticos

Hay dos funciones que resultan especialmente útiles: la función `summary()` y la función `fivenum()`. Ambas funciones ofrecen un resumen de los parámetros de posición más importantes.
Si la variable es cuantitativa, la función `summary()` nos proporciona el mínimo, el máximo, el primer y tercer cuartil, la mediana, la media y el número de datos faltantes, si los hay:

```
summary(Datos$agua.hab)
>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
>  0.1363  0.4599  0.5064  0.5256  0.5613  1.0873       3
```

Si la variable es cualitativa se muestran los distintos atributos y sus frecuencias.

```
summary(Datos$Tipo)
>  Grande Mediano Pequeño    NA's
>      33      30      33       1
```

La función `fivenum()` muestra los *Tukey Five-Number*, o sea, nos proporciona también el mínimo, el máximo, el primer y tercer cuartil y la mediana, sólo si la variable es cuantitativa:

```
fivenum(Datos$agua.hab)
> [1] 0.1362745 0.4586279 0.5063754 0.5616029 1.0872600
```

A diferencia de `summary()`, `fivenum()` no calcula la media, ni da información sobre faltantes. Tampoco puede evaluar un *data frame* entero. Además, los cuartiles se calculan como la mediana de cada mitad de datos por debajo y por arriba de la mediana.
Otra función muy práctica es `boxplot.stats()`, genera un resumen de estadísticos de posición como las funciones anteriores, pero además muestra el intervalo de valores entre los cuales consideramos un valor no anómalo (en función del parámetro `coef` que, por defecto, es 1,5) y los valores anómalos. Esto es, esta función calcula los valores necesarios para construir un gráfico *Box & Whisker*:

```
boxplot.stats(Datos$elec.hab)
> $stats
> [1] 0.9391481 1.7423166 2.1954678 3.0997261 5.1169844
>
> $n
> [1] 96
>
> $conf
> [1] 1.976575 2.414361
>
> $out
> [1] 5.471826 5.390684 5.598538 8.011603 9.191959
```

Si queremos ampliar el rango de valores de los bigotes:

```
boxplot.stats(Datos$elec.hab, coef = 2)
> $stats
> [1] 0.9391481 1.7423166 2.1954678 3.0997261 5.5985376
>
> $n
> [1] 96
>
> $conf
```

```
> [1] 1.976575 2.414361
>
> $out
> [1] 8.011603 9.191959
```

1.2

3. Obtención de frecuencias absolutas y relativas

Para calcular las frecuencias absolutas o relativas, conjuntas o marginales,… usaremos básicamente las funciones: `table()`, `prop.table()`.
Cálculo de las **frecuencias absolutas**:

```
Tabla <- table(Datos$Tipo)
Tabla
>
>  Grande Mediano Pequeño
>      33      30      33
```

Cálculo de las **frecuencias relativas**:

```
Tabla.rel<-prop.table(Tabla)
Tabla.rel
>
>  Grande Mediano Pequeño
> 0.34375 0.31250 0.34375
```

Cálculo de las **frecuencias relativas** en porcentaje:

```
Tabla.rel <- round(Tabla.rel*100, 2)
Tabla.rel
>
>  Grande Mediano Pequeño
>   34.38   31.25   34.38
```

Observar que las variables utilizadas son todas cualitativas. Para obtener tablas para variables cuantitativas, primero habría que recodificar la variable y transformarla en categórica (**discretizar**).
Cálculo de las **frecuencias absolutas acumuladas**:

```
Tabla.acum <- cumsum(Tabla)
Tabla.acum
>  Grande Mediano Pequeño
>      33      63      96
```

Cálculo de las **frecuencias relativas acumuladas**:

```
Tabla.rel.acum<-cumsum(Tabla.rel)
Tabla.rel.acum
>  Grande Mediano Pequeño
>   34.38   65.63  100.01
```

1.3

4. Algunos gráficos descriptivos

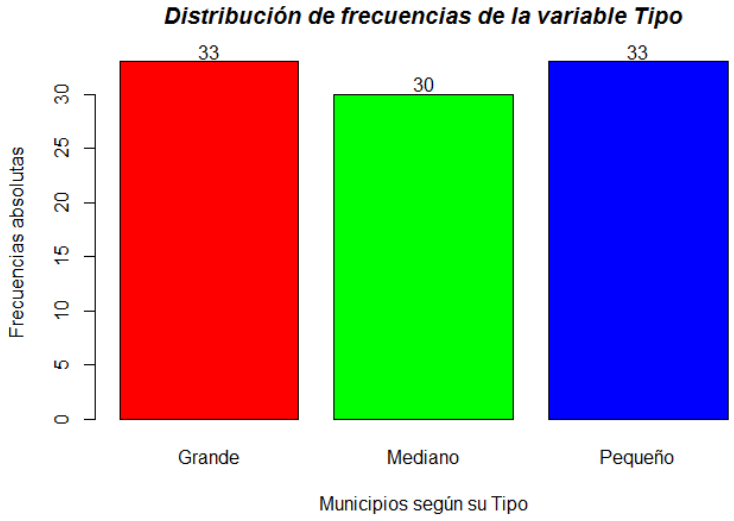
Vamos a ver algunos gráficos asociados a una distribución de frecuencias. Las representaciones gráficas que se muestran son las básicas.
Las funciones gráficas tienen muchísimos parámetros que permiten configurar y mostrar el gráfico con gran nivel de detalle, pero en este caso sólo veremos algunos de ellos.

Con el paquete *base*, las representaciones gráficas básicas adecuadas a las variables cualitativas son los **diagramas de barras** y los **diagramas de tarta o sectores**, mientras que en el caso de las cuantitativas disponemos de los **histogramas** y los **diagramas de caja**.

4.1. Diagrama de barras

Obtención de un diagrama de barras con las frecuencias absolutas de la variable *Tipo* de municipio con la función `barplot()`:

```
barras.tipo<-barplot(Tabla,col=rainbow(3),xlab="Municipios según su Tipo",
ylab="Frecuencias absolutas")

# Si queremos añadir las frecuencias como etiquetas al gráfico:
#1. En el eje X me sitúo en el mismo sitio que están las columnas de diagrama
#2. En el eje Y me sitúo un poco por encima de las frecuencias que me da Tabla
#3. Justo ahí quiero que escriba las frecuencias que me da Tabla

text(barras.tipo,Tabla + 1,labels=Tabla, xpd = TRUE) # xpd = TRUE es para que
"estire" un poco el gráfico y quepa todo

title(main = "Distribución de frecuencias de la variable Tipo", font.main = 4) #
Añadimos el título
```



4.2. Diagrama de sectores

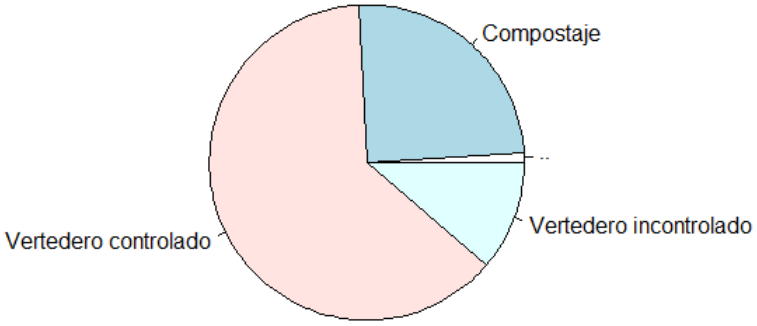
Obtención de un diagrama de barras con las frecuencias relativas de la variable *Residuos.sólidos.urbanos..Destino* con la función `pie()`:
Hagamos un diagrama con las etiquetas solo.
Primero obtenemos las frecuencias:

```
tabla.destino<-prop.table(table(Datos$Residuos.sólidos.urbanos..Destino))
tabla.destino<-round(100*tabla.destino,2)#En porcentaje y redondeando
```

Ahora el diagrama de tarta:

```
sectores.destino<-pie(tabla.destino,labels=names(tabla.destino),main="Distribución
de porcentajes de la variable Destino de los residuos sólidos urbanos")
```

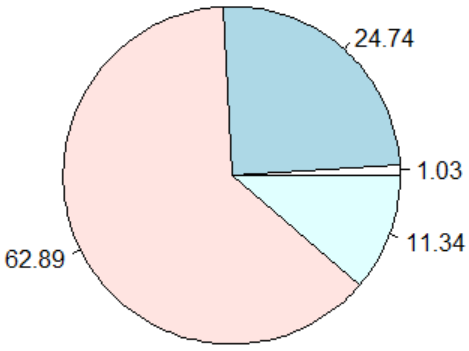
tribución de porcentajes de la variable Destino de los residuos sólidos u



Y ahora un diagrama con las frecuencias relativas en porcentaje:

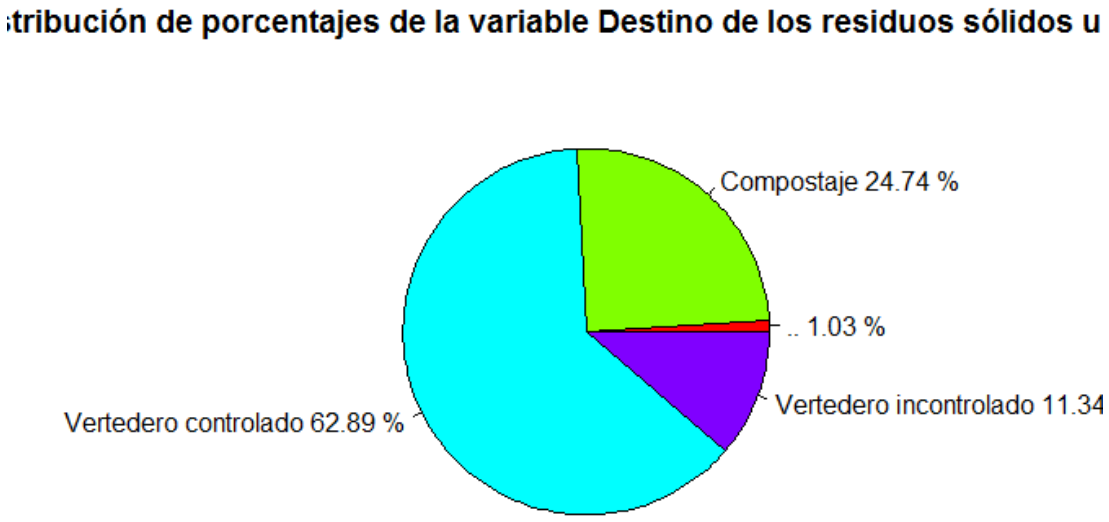
```
sectores.destino<-pie(tabla.destino,labels=tabla.destino,main="Distribución de
porcentajes de la variable Destino de los residuos sólidos urbanos")
```

tribución de porcentajes de la variable Destino de los residuos sólidos u



Ahora un gráfico con etiquetas y sus valores:

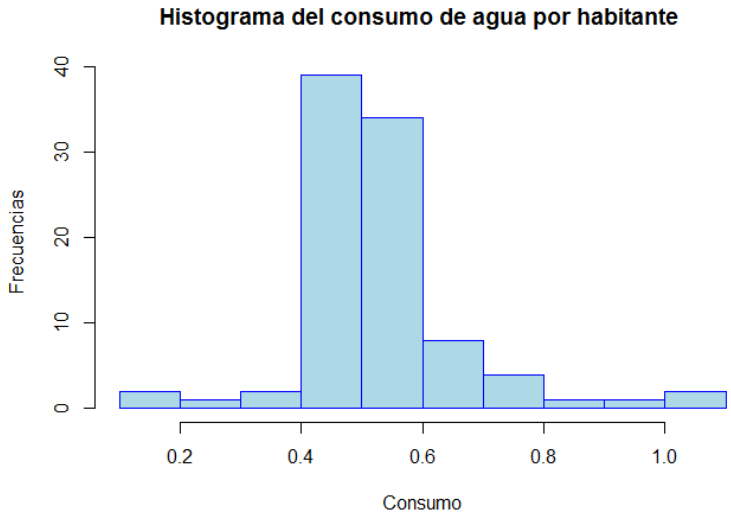
```
sectores.destino<-
pie(tabla.destino,labels=paste(names(tabla.destino),tabla.destino,"%"),main="Distr
ibución de porcentajes de la variable Destino de los residuos sólidos urbanos",
col=rainbow(4))
```



4.3. Histogramas

Los histogramas se utilizan para representar **variables cuantitativas continuas**.
Vamos a construir un histograma para el consumo de agua, con la función `hist()`:

```
hist(Datos$agua.hab, breaks = 10, freq = TRUE, main = "Histograma del consumo de
agua por habitante ",xlab="Consumo",ylab="Frecuencias", col="lightblue",
border="blue")
```

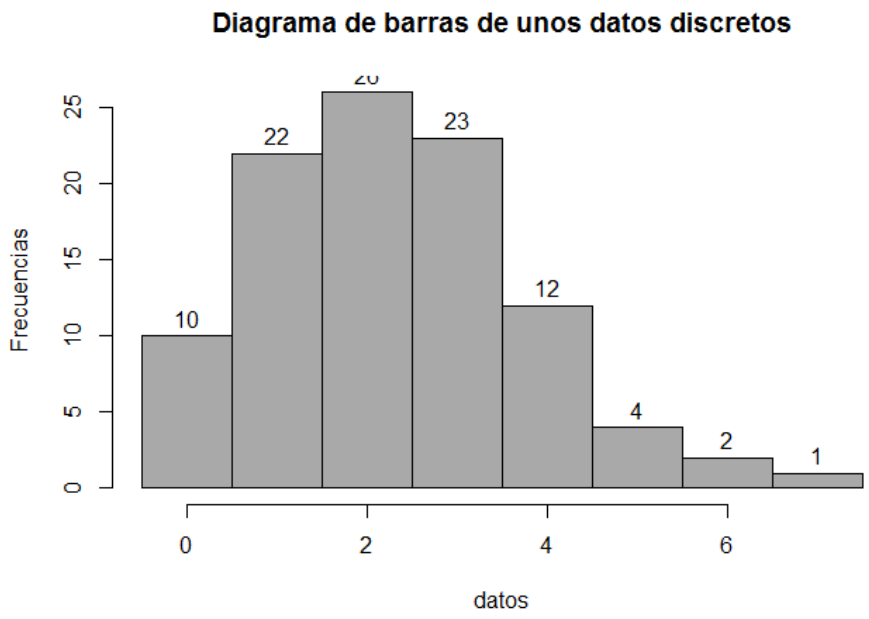


Con un poco de código también puede usarse el histograma para representar las frecuencias de una **variable cuantitaiva discreta**.
En primer lugar generamos unos datos discretos a partir de la distribución de *Poisson*, por ejemplo:

```
datos<-rpois(100,2.5)
head(datos, 10)
>  [1] 1 0 2 2 2 1 1 3 3 4
```

Y ahora los representamos:

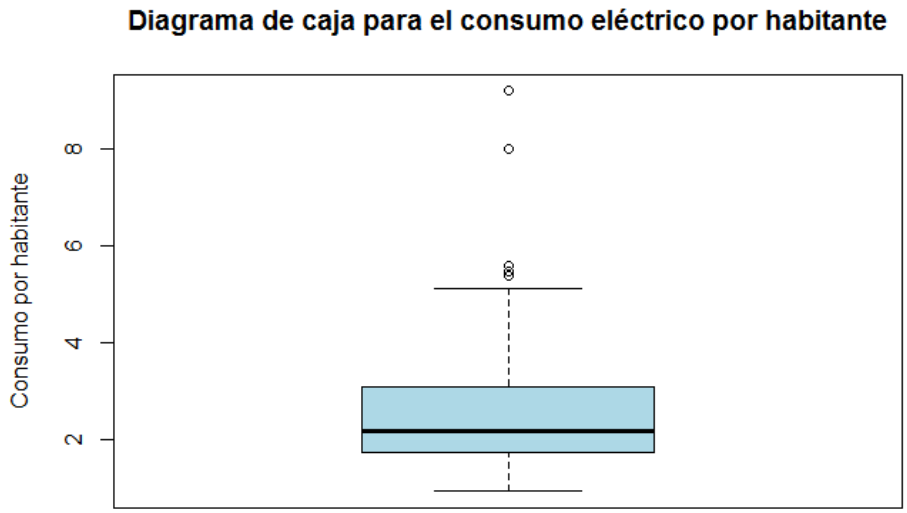
```
# Los puntos de corte del histograma fuerzan a que contemos los 0, los 1, ...
cortes<-(min(datos)-0.5):(max(datos)+0.5)

# El histograma:
hist(datos,breaks=cortes,freq=TRUE,labels=TRUE,ylab="Frecuencias",main="Diagrama
de barras de unos datos discretos", col="darkgray", border="black")
```



4.4. Diagrama de caja o *Box&Whisker*

Es uno de los gráficos más utilizados en el análisis descriptivo de variables cuantitativas.
Obtención de un diagrama de caja para la variable *elec.hab* con la función `boxplot()`:

```
boxplot(Datos$elec.hab,main="Diagrama de caja para el consumo eléctrico por
habitante",ylab="Consumo por habitante", col="lightblue")
```



5. Herramientas estadística k-dimensional

5.1. Parámetros muestrales de múltiples variables

Para obtener la media, desviación típica, etc de varias variables simultáneamente usamos las funciones `lapply()` y `sapply()`.
A estas funciones se les pasa como argumentos, como mínimo, una lista de variables o un subconjunto de un *data set* y la función a aplicar.

5.1.1. Media o promedio

Para obtener la media de los consumos de electricidad, agua y la cantidad de residuos:

```
lapply(Datos[,c("agua.hab", "elec.hab", "res.hab")], mean, na.rm=T) #
referenciando las variables por su nombre
> $agua.hab
> [1] 0.5256292
>
> $elec.hab
> [1] 2.655137
>
> $res.hab
> [1] 0.2331974
# lapply(Datos[,9:11], mean, na.rm=T) para referenciar a las variables por el
número de columna del data frame
```

Para que los resultados queden más claros:

```
lista.medias <- lapply(Datos[,9:11], mean, na.rm=T) # Para tener los resultados en
una lista

lista.medias
> $agua.hab
> [1] 0.5256292
>
> $elec.hab
> [1] 2.655137
>
```

```
> $res.hab
> [1] 0.2331974
```

Si usamos la función `sapply()`,

```
vector.medias <- sapply(Datos[,9:11], mean, na.rm=T) # Para tener los resultados
en un vector

vector.medias
>  agua.hab  elec.hab   res.hab
> 0.5256292 2.6551375 0.2331974
```

Las funciones `lapply()` y `sapply()` pueden usarse para cualquier otra función, estadística o no.

5.1.2. Desviación típica

Para obtener la desviación típica de los consumos de electricidad, agua y la cantidad de residuos y disponer de los resultados en un vector:

```
vector.sd<-sapply(Datos[,9:11], sd, na.rm=T)
vector.sd
>   agua.hab   elec.hab    res.hab
> 0.13717260 1.40268853 0.03660991
```

5.1.3. Coeficiente de variación

```
vector.cv<-vector.sd/vector.medias
vector.cv
>  agua.hab  elec.hab   res.hab
> 0.2609684 0.5282922 0.1569910
```

5.1.4. Cuantiles.

Para obtener el 1er y 3er cuartil (Q1 y Q3) de los consumos de electricidad, agua y la cantidad de residuos:

```
sapply(Datos[,9:11], quantile, probs=c(0.25,0.75), na.rm=T)
>      agua.hab elec.hab   res.hab
> 25% 0.4598800 1.754325 0.2114574
> 75% 0.5613026 3.096219 0.2408332
```

5.1.5. Mediana

Para obtener la mediana de los consumos de electricidad, agua y la cantidad de residuos:

```
sapply(Datos[,9:11], median, na.rm=T)
>  agua.hab  elec.hab   res.hab
> 0.5063754 2.1954678 0.2257808
```

5.1.6. Coeficiente de asimetría y curtosis.

Para obtener los coeficientes de asimetría y curtosis de los consumos de electricidad, agua y la cantidad de residuos:
Calculemos la **Asimetría**

```
# Asimetria
sapply(Datos[,9:11], skewness, na.rm=T)
> agua.hab elec.hab  res.hab
> 1.219825 2.052142 1.391985
```

Calculemos la **Curtosis**

```
#  Curtosis
sapply(Datos[,9:11], kurtosis, na.rm=T)
> agua.hab elec.hab  res.hab
> 4.315458 5.292531 1.401060
```

5.2. Estadísticos por grupos

Cuando se analiza más de una variable, además de calcular los estadísticos de cada una de las variables simultáneamente, nos puede interesar analizar los estadísticos de una para las distintas categorías o rangos de valores de otra.
Por ejemplo, nos puede interesar obtener los parámetros de determinados consumos según el tipo de municipio.
Veamos qué tipos de municipios hay:

```
levels(Datos$Tipo)
> [1] "Grande"  "Mediano" "Pequeño"
```

Como vemos hay tres tipos de municipio, en cuanto a su tamaño.
Para obtener estadísticos muestrales del consumo de agua por habitante para los tres tamaños de municipio, usamos la función `tapply()`.
A esta función hay que pasarle como argumentos:

1. Qué datos manejamos
2. Qué factor es el que determina los grupos
3. Qué función queremos aplicar
4. Información adicional necesaria para la función

Veamos como ejemplo el cálculo de la media, desviación típica y los percentiles 5 y 95 del consumo de agua por habitante para los tres tamaños de municipio:

```
tapply(Datos$agua.hab,Datos$Tipo,mean, na.rm=TRUE)
>    Grande   Mediano   Pequeño
> 0.5135674 0.5118742 0.5494647
tapply(Datos$agua.hab,Datos$Tipo,sd, na.rm=TRUE)
>     Grande    Mediano    Pequeño
> 0.12409556 0.07046884 0.18666335
tapply(Datos$agua.hab,Datos$Tipo,quantile,probs=c(0.05,0.95),na.rm=TRUE)
> $Grande
>        5%       95%
> 0.4048812 0.6520250
>
> $Mediano
>        5%       95%
> 0.4137313 0.6328537
>
> $Pequeño
>        5%       95%
> 0.3049989 0.8514134
```

Si queremos obtener una tabla con la media y la desviación típica del consumo de agua por habitante para cada tipo de municipio:

```
media.tipo<-tapply(Datos$agua.hab,Datos$Tipo,mean, na.rm=TRUE)
desv.tipo<-tapply(Datos$agua.hab,Datos$Tipo,sd, na.rm=TRUE)
resumen<-data.frame(media.tipo, desv.tipo)
resumen
>         media.tipo  desv.tipo
> Grande   0.5135674 0.12409556
```

```
> Mediano  0.5118742 0.07046884
> Pequeño  0.5494647 0.18666335
```

También podría usarse la función `by()`, que en realidad llama a la función `tapply()`

```
by(Datos$agua.hab,Datos$Tipo,mean, na.rm=TRUE)
> Datos$Tipo: Grande
> [1] 0.5135674
> ------------------------------------------------------------
> Datos$Tipo: Mediano
> [1] 0.5118742
> ------------------------------------------------------------
> Datos$Tipo: Pequeño
> [1] 0.5494647
```

5.3. Covarianza y coeficiente de correlación

En el caso de analizar varias variables cuantitativas también puede interesar conocer el grado de relación lineal que hay entre cada para de éstas.
Para tener una medida de este grado de relación disponemos de la **covarianza** y el **coeficiente de correlación (r)**.
Para obtener la **covarianza** entre dos variable podemos usar también la función `var()`, pesándola dos variables como argumentos, en vez de una sola:

```
var(Datos$Consumo.de.agua..Invierno, Datos$Consumo.de.energía.eléctrica, na.rm =T)
> [1] 39420611
```

También podemos usar la función `cov()`[3]:

```
cov(Datos$Consumo.de.agua..Invierno, Datos$Consumo.de.energía.eléctrica,
use="pairwise.complete.obs")
> [1] 39420611
```

Si queremos obtener el **coeficiente de correlación r** (por defecto el de *Pearson*):

```
cor(Datos$Consumo.de.agua..Invierno, Datos$Consumo.de.energía.eléctrica,
use="pairwise.complete.obs")
> [1] 0.9398407
```

A cualquiera de las tres funciones anteriores podemos pasarle como argumento un *data frame* o subconjunto de éste con variables cuantitativas para que obtenga la **matriz de varianzas-covarianzas**.

```
var(Datos[,9:11], na.rm =T)
>               agua.hab     elec.hab       res.hab
> agua.hab  1.881632e-02 -0.009210185 -3.712554e-05
> elec.hab -9.210185e-03  1.997470277  7.725236e-03
> res.hab  -3.712554e-05  0.007725236  1.113212e-03
cov(Datos[,9:11], use="pairwise.complete.obs")
>               agua.hab     elec.hab       res.hab
> agua.hab  1.881632e-02 -0.009210185 -3.712554e-05
> elec.hab -9.210185e-03  1.967535118  9.157241e-03
> res.hab  -3.712554e-05  0.009157241  1.340285e-03
```

Análogamente procederíamos para obtener la **matriz de correlación**:

```
cor(Datos[,9:11], use="pairwise.complete.obs")
>              agua.hab    elec.hab      res.hab
> agua.hab  1.000000000 -0.04750735 -0.008111786
```

```
> elec.hab -0.047507349  1.00000000  0.178321942
> res.hab  -0.008111786  0.17832194  1.000000000
```

1.4 5.4 Resúmenes de estadísticos para múltiples variables

La función `summary()` también pueden usarse para todo un *data frame* y resulta muy útil para un primer paso de un análisis exploratorio:

```
summary(Datos)
>    CodigoINE                   Municipio  Consumo.de.energía.eléctrica
>  Min.   :23001   Albanchez de Mágina: 1   Min.   :   463
>  1st Qu.:23028   Alcalá la Real     : 1   1st Qu.:  3316
>  Median :23053   Alcaudete          : 1   Median :  6978
>  Mean   :23094   Aldeaquemada       : 1   Mean   : 22115
>  3rd Qu.:23079   Andújar            : 1   3rd Qu.: 14978
>  Max.   :23905   Arjona             : 1   Max.   :349561
>                  (Other)            :91   NA's   :1
>  Consumo.de.agua..Invierno Consumo.de.agua..Verano
>  Min.   :  50.0            Min.   :   89
>  1st Qu.: 312.8            1st Qu.:  480
>  Median : 572.0            Median :  820
>  Mean   :1102.1            Mean   : 1488
>  3rd Qu.:1129.2            3rd Qu.: 1656
>  Max.   :8896.0            Max.   :10326
>  NA's   :3                 NA's   :3
>       Residuos.sólidos.urbanos..Destino Residuos.sólidos.urbanos..Cantidad
>  ..                     : 1             Min.   :  113.5
>  Compostaje             :24             1st Qu.:  377.3
>  Vertedero controlado   :61             Median :  602.3
>  Vertedero incontrolado:11              Mean   : 1872.7
>                                         3rd Qu.: 1329.9
>                                         Max.   :39197.5
>                                         NA's   :1
>    Población         agua.hab         elec.hab         res.hab
>  Min.   :   446   Min.   :0.1363   Min.   :0.9391   Min.   :0.1768
>  1st Qu.:  1716   1st Qu.:0.4599   1st Qu.:1.7543   1st Qu.:0.2115
>  Median :  3028   Median :0.5064   Median :2.1955   Median :0.2258
>  Mean   :  6727   Mean   :0.5256   Mean   :2.6551   Mean   :0.2332
>  3rd Qu.:  5780   3rd Qu.:0.5613   3rd Qu.:3.0962   3rd Qu.:0.2408
>  Max.   :111406   Max.   :1.0873   Max.   :9.1920   Max.   :0.3518
>  NA's   :1        NA's   :3        NA's   :1        NA's   :1
>       Tipo
>  Grande :33
>  Mediano:30
>  Pequeño:33
>  NA's   : 1
>
>
>
```

Observar que si las variables son cualitativas no procede calcular ningún estadístico, pero en su lugar muestra los diferentes valores de la variable y sus frecuencias. También ofrece información acerca de los valores faltantes.
La función `fivenum()` es menos completa para resumir todo un *data frame*. No ofrece información si la variable es cualitativa y no podemos resumir varias variables simultáneamente:

```
fivenum(Datos$Consumo.de.energía.eléctrica, na.rm = T)
> [1]    463.0   3244.0   6978.5  15100.0 349561.0
```

Esto daría error:

```
fivenum(Datos[ ,9:12], na.rm = T)
> Error in x[floor(d)] + x[ceiling(d)]: argumento no-numérico para operador
binario
```

Para calcular los *Tukey Five-Number* de varias variables al mismo tiempo tenemos que recurrir a la función `sapply()`:

```
sapply(Datos[ ,9:12], fivenum, na.rm = T)
> Warning in Ops.factor(x[floor(d)], x[ceiling(d)]): '+' not meaningful for
> factors
>       agua.hab  elec.hab   res.hab Tipo
> [1,] 0.1362745 0.9391481 0.1767804   NA
> [2,] 0.4586279 1.7423166 0.2112057   NA
> [3,] 0.5063754 2.1954678 0.2257808   NA
> [4,] 0.5616029 3.0997261 0.2413041   NA
> [5,] 1.0872600 9.1919589 0.3518437   NA
```

1.5 5.5 Tablas de frecuencias cruzadas

Para crear tablas de frecuencias cruzadas también podemos usar la función `table()`
En este caso hemos de indicar como primer argumento la variable cuyos valores aparecerán en las filas y como segundo la variable cuyos valores aparecerán en las columnas.
Por ejemplo, construyamos una tabla de frecuencias cruzadas (**frecuencias conjuntas absolutas**) para las variables destino residuos urbanos (filas) y tipo de municipio (columnas):

```
Tabla2.conj.abs <- table(Datos$Residuos.sólidos.urbanos..Destino, Datos$Tipo)
Tabla2.conj.abs
>
>                          Grande Mediano Pequeño
>   ..                          0       0       0
>   Compostaje                 11       7       6
>   Vertedero controlado       18      20      23
>   Vertedero incontrolado      4       3       4
```

Para obtener las **frecuencias conjuntas relativas** para las variables destino residuos urbanos y tipo de municipio:

```
prop.table(Tabla2.conj.abs)
>
>                              Grande    Mediano    Pequeño
>   ..                      0.00000000 0.00000000 0.00000000
>   Compostaje              0.11458333 0.07291667 0.06250000
>   Vertedero controlado    0.18750000 0.20833333 0.23958333
>   Vertedero incontrolado 0.04166667 0.03125000 0.04166667
```

También podemos obtener las **frecuencias marginales absolutas**:

```
margin.table(Tabla2.conj.abs, 1) # Frec marginales por columnas
>
>                     ..             Compostaje   Vertedero controlado
>                      0                     24                     61
> Vertedero incontrolado
>                     11
margin.table(Tabla2.conj.abs, 2) # Frec marginales por filas
>
>  Grande Mediano Pequeño
```

```
>      33       30       33
```

Y por último las **frecuencias condicionales relativas**:

```
prop.table(Tabla2.conj.abs, 1) # en función de las filas
>
>                            Grande   Mediano   Pequeño
>   ..
>   Compostaje             0.4583333 0.2916667 0.2500000
>   Vertedero controlado   0.2950820 0.3278689 0.3770492
>   Vertedero incontrolado 0.3636364 0.2727273 0.3636364
prop.table(Tabla2.conj.abs, 2) # en función de las columnas
>
>                            Grande   Mediano   Pequeño
>   ..                     0.0000000 0.0000000 0.0000000
>   Compostaje             0.3333333 0.2333333 0.1818182
>   Vertedero controlado   0.5454545 0.6666667 0.6969697
>   Vertedero incontrolado 0.1212121 0.1000000 0.1212121
```

Para un mayor control de lo que se quiere mostrar en las tablas de frecuencias cruzadas pueden usarse las funciones `ftable()` y `xtabs()`.

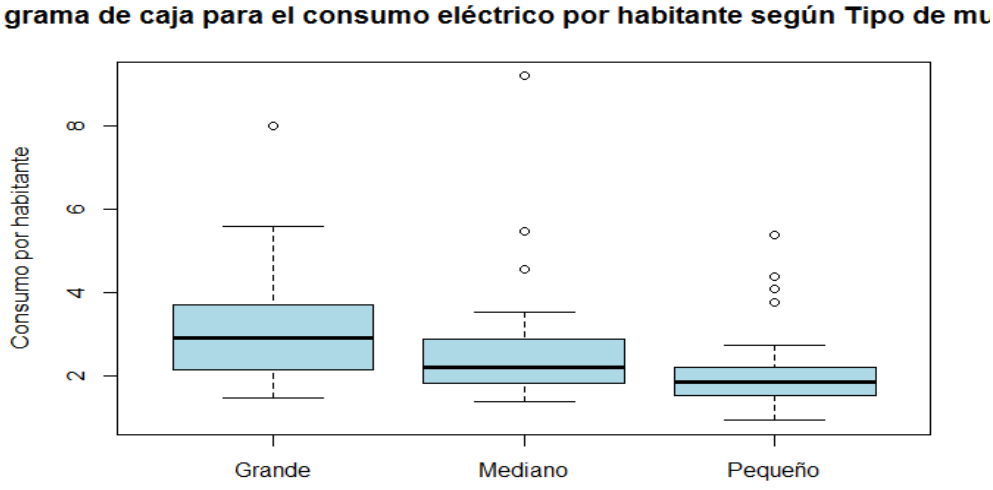
1.6 5.6 Algunos gráficos para múltiples variables

Al igual que resulta útil disponer de los parámetros muestrales de una variable para los distintos valores de otra, o los estadísticos de un subconjunto de variables simultáneamente, también es interesante disponer de representaciones gráficas que muestren varias variables a la vez, o los valores de una en función de otra o representaciones que nos permitan determinar si dos o más variables están relacionadas.
En este sentido, los gráficos elementales son el **diagrama de caja múltiple** y el **diagrama de dispersión**.

1.7 5.6.1 Diagrama *Box&Whisker* múltiple

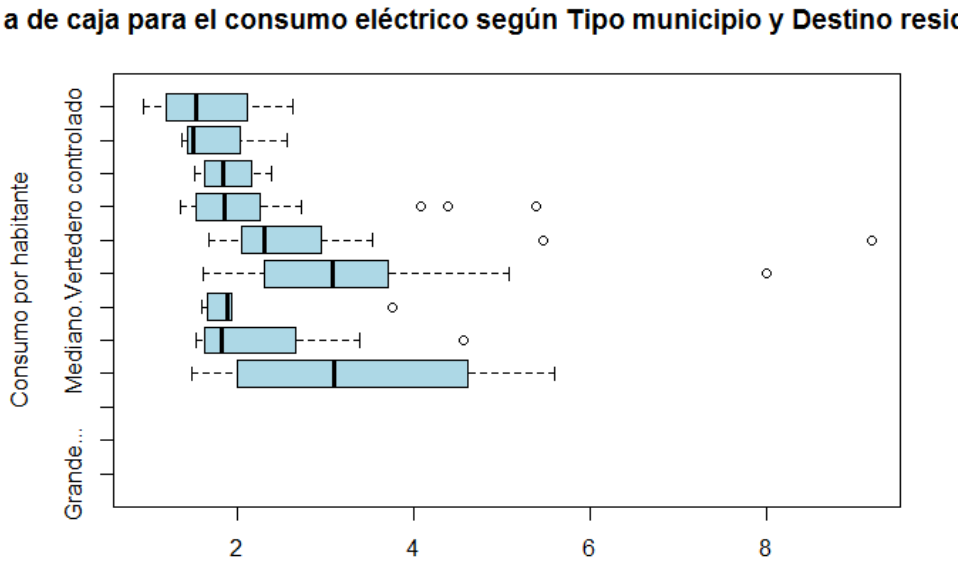
Vamos a representar, por ejemplo, la variable *elec.hab* según tipo de municipio:

```
boxplot(Datos$elec.hab~Datos$Tipo,main="Diagrama de caja para el consumo eléctrico
por habitante según Tipo de municipio",ylab="Consumo por habitante",
col="lightblue")
```



Vamos a ver ahora el diagrama de caja de *elec.hab* según tipo de municipio y destino de los residuos sólidos urbanos:

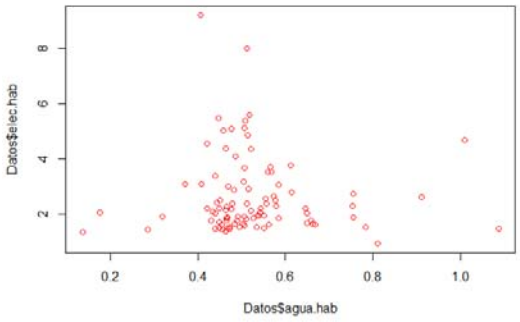
```
boxplot(Datos$elec.hab~Datos$Tipo*Datos$Residuos.sólidos.urbanos..Destino,main="Di
agrama de caja para el consumo eléctrico según Tipo municipio y Destino residuos
solidos",ylab="Consumo por habitante", col="lightblue", horizontal=T)
```



1.8 5.6.2 Diagramas de dispersión

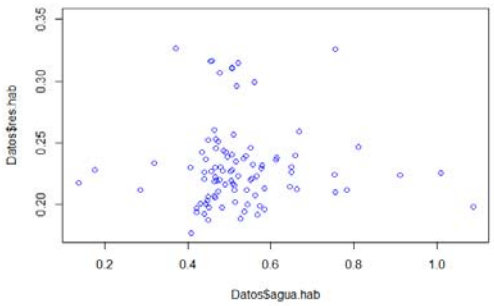
Estas representaciones nos permiten determinar la posible relación entre dos variables cuantitativas. Veamos, por ejemplo, el diagrama de dispersión entre el consumo de agua medio y el de electricidad:

```
plot(Datos$agua.hab, Datos$elec.hab, type="p", col="red")
```



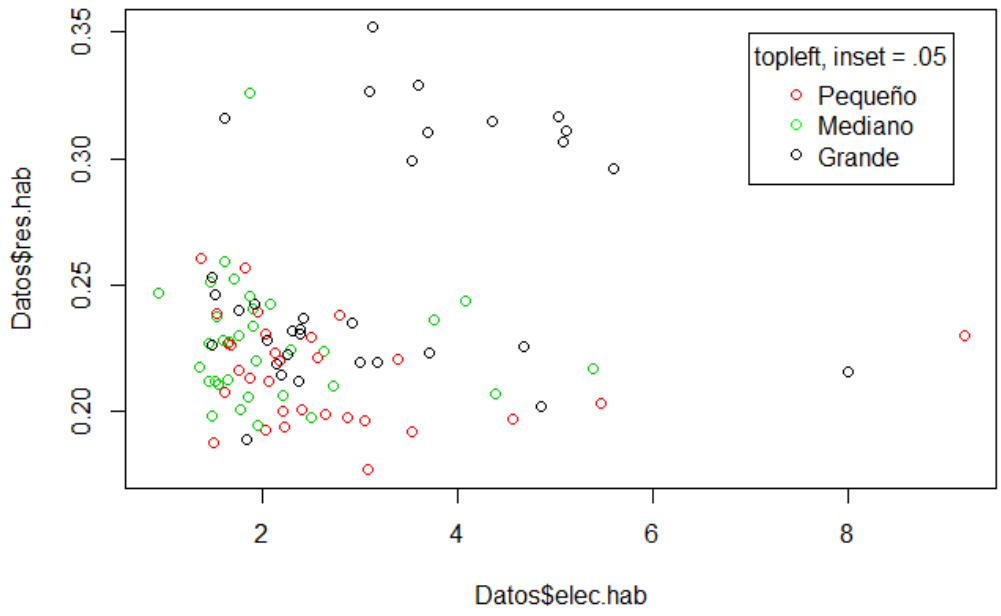
Y ahora el diagrama de dispersión entre el consumo de agua medio y la cantidad de residuos:

```
plot(Datos$agua.hab, Datos$res.hab, type="p", col="blue")
```



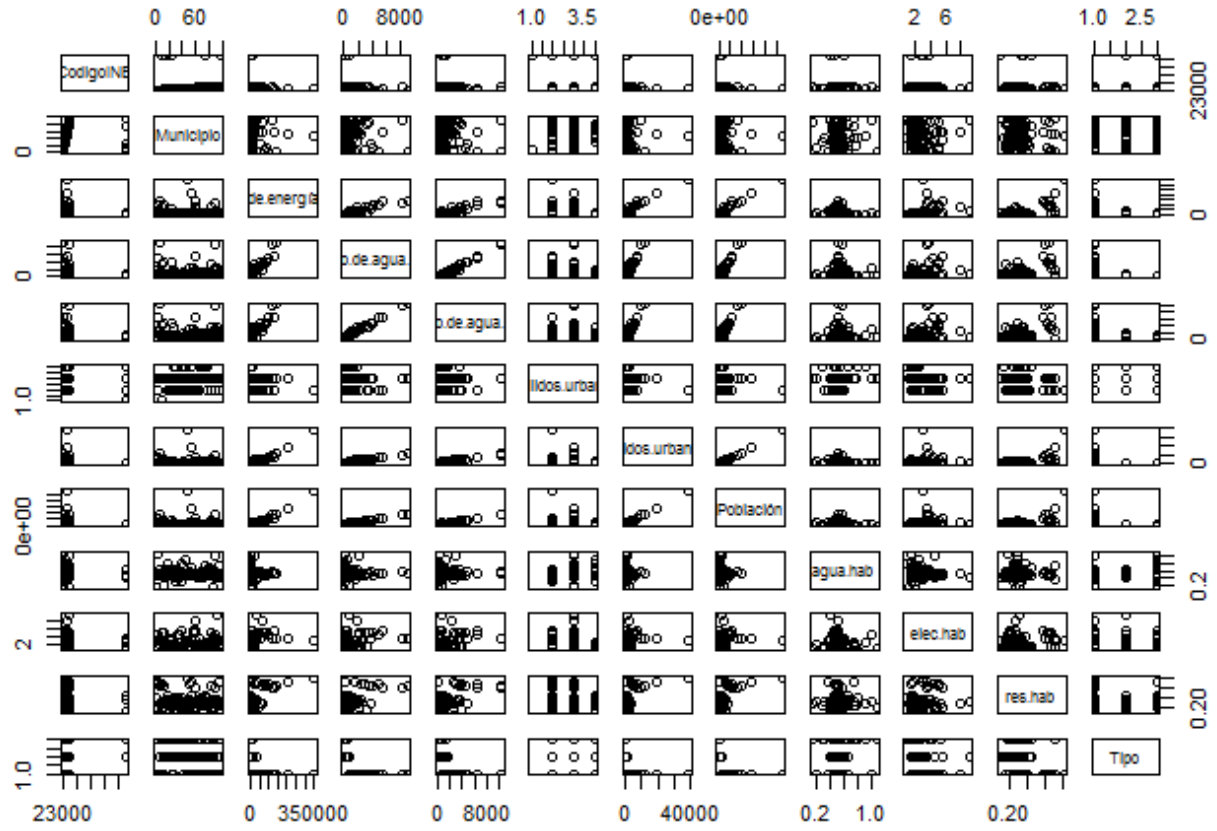
También podemos distinguier los puntos del diagrama, por colores y/o forma, según alguna variable cualitativa o discretizada.
Supongamos que queremos obtener el *diagrama de dispersión* del consumo medio por habitante de electricidad y de la cantidad de residuos generada en función del tamaño del municipio:

```
plot(Datos$elec.hab, Datos$res.hab, type = "p", col = Datos$Tipo)
legend("topright", c("Pequeño", "Mediano", "Grande"), col = c("red", "green",
"black"), pch = 1, title = "topleft, inset = .05", inset = .05)
```

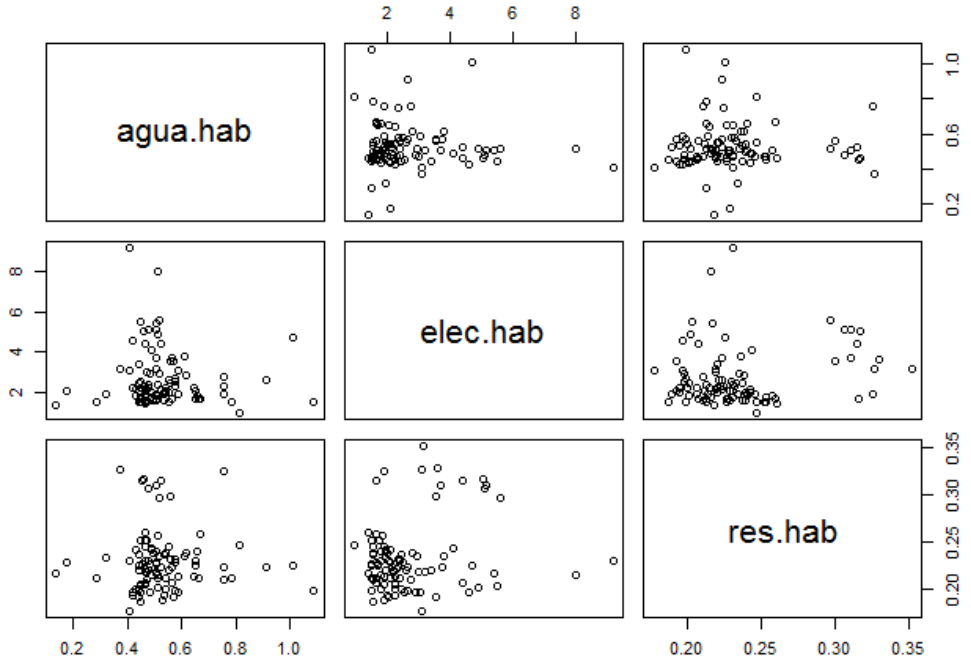


Especialmente útil resulta la función `pairs()` que muestra el diagrama de dispersión para cada par de variables de una hoja de datos o subconjunto de ésta:

```
pairs(Datos)
```




```
pairs(Datos[,9:11])
```



1.9 7 Uso de otros paquetes

Ya hemos visto que para obtener los parámetros de asimetría y curtosis (*forma*) necesitamos el paquete `e1071`.
Por otra parte, hemos visto que las funciones `summary()` y `fivenum()` permiten hacer un resumen de estadísticos de posición, fundamentalmente. Sin embargo, cuando se hace un primer estudio descriptivo, además de los parámetros de posición, es necesario calcular también los de dispersión y forma.
En R podemos programar un *script* que calcule exactamente los parámetros que necesitamos, pero tenemos alguna función que lo hace por nosotros.
Por ejemplo, el **paquete `psych`**, dispone de las funciones `describe()` y `describeBy()` para obtener un resumen de los estadísticos descriptivos de una variable o de un conjunto de ellas o de todo un *data set*, además el resumen se puede visualizar por grupos de acuerdo a los valores de otra variable.
Los parámetros que calculan estas funciones por defecto son:

- La media (*mean*)
- La desviación típica (*sd*)
- La mediana (*median*)
- La media truncada (*trimmed* mean)
- La desviación absoluta (*mad*)
- El mínimo (*min*)
- El máximo (*max*)
- El rango (*range*)
- El coeficiente de asimetría estándar (*skew*)
- El coeficiente de curtosis estándar (*kurtosis*)
- El error estándar (*se*)

Veamos un ejemplo.
En primer lugar cargamos (e instalamos, si no lo hemos hecho previamente) el paquete `psych`:

```
# Instalar el paquete psych si es preciso
if(!is.element('psych', installed.packages())) install.packages('psych', repos =
'https://cran.rediris.es/', dependencies = T)

# Cargar paquete
library(psych)
```

Ahora vamos a obtener el resumen de estadísticos del consumo medio por habitante de agua y electricidad:

```
describe(Datos[,c("agua.hab", "elec.hab")])
>           vars  n mean   sd median trimmed  mad  min  max range skew
> agua.hab    1 94 0.53 0.14   0.51    0.51 0.08 0.14 1.09  0.95 1.22
> elec.hab    2 96 2.66 1.40   2.20    2.42 0.84 0.94 9.19  8.25 2.05
>           kurtosis   se
> agua.hab     4.32 0.01
> elec.hab     5.29 0.14
```

Adicionalmente podemos pedirle que calcule el Rango Intercuartílico (*IQR*) o los cuantiles (1er y 3er cuartil, por ejemplo) para las mismas variables:

```
describe(Datos[,c("agua.hab", "elec.hab")], quant = c(0.25, 0.75), IQR = T)
>           vars  n mean   sd median trimmed  mad  min  max range skew
> agua.hab    1 94 0.53 0.14   0.51    0.51 0.08 0.14 1.09  0.95 1.22
> elec.hab    2 96 2.66 1.40   2.20    2.42 0.84 0.94 9.19  8.25 2.05
>           kurtosis   se  IQR Q0.25 Q0.75
> agua.hab     4.32 0.01 0.10  0.46  0.56
> elec.hab     5.29 0.14 1.34  1.75  3.10
```

Y a continuación obtendremos el resumen de estadísticos del consumo medio por habitante de agua y electricidad, según el tamaño del municipio:

```
describeBy(Datos[,c("agua.hab", "elec.hab")], group = Datos$Tipo)
>
>  Descriptive statistics by group
> group: Grande
>          vars  n mean   sd median trimmed  mad  min  max range skew
> agua.hab    1 31 0.51 0.12   0.51    0.50 0.06 0.18 1.01  0.84 1.45
> elec.hab    2 33 3.18 1.49   2.92    3.03 1.18 1.49 8.01  6.52 1.16
>          kurtosis   se
> agua.hab     7.26 0.02
> elec.hab     1.19 0.26
> ------------------------------------------------------------
> group: Mediano
>          vars  n mean   sd median trimmed  mad  min  max range skew
> agua.hab    1 30 0.51 0.07   0.51    0.51 0.09 0.41 0.65  0.24 0.23
> elec.hab    2 30 2.66 1.53   2.20    2.34 0.72 1.38 9.19  7.81 2.77
>          kurtosis   se
> agua.hab    -1.07 0.01
> elec.hab     8.60 0.28
> ------------------------------------------------------------
> group: Pequeño
>          vars  n mean   sd median trimmed  mad  min  max range skew
> agua.hab    1 33 0.55 0.19   0.49    0.54 0.08 0.14 1.09  0.95 0.70
> elec.hab    2 33 2.12 0.96   1.86    1.94 0.48 0.94 5.39  4.45 1.85
>          kurtosis   se
> agua.hab     0.80 0.03
> elec.hab     2.89 0.17
```

También resulta muy útil la función `describe()` del paquete `Hmisc`:

```
# Instalar el paquete Hmisc si es preciso
if(!is.element('Hmisc', installed.packages())) install.packages('Hmisc', repos =
'https://cran.rediris.es/', dependencies = T)

# Cargar paquete
library(Hmisc)
> Loading required package: lattice
> Loading required package: survival
> Loading required package: Formula
> Loading required package: ggplot2
>
> Attaching package: 'ggplot2'
> The following objects are masked from 'package:psych':
>
>     %+%, alpha
>
> Attaching package: 'Hmisc'
> The following object is masked from 'package:psych':
>
>     describe
> The following object is masked from 'package:e1071':
>
>     impute
> The following objects are masked from 'package:base':
>
>     format.pval, round.POSIXt, trunc.POSIXt, units
```

Veamos un ejemplo:

```
describe(Datos$agua.hab)
> Datos$agua.hab
>        n  missing distinct     Info     Mean      Gmd      .05      .10
>       94        3       94        1   0.5256   0.1346   0.3935   0.4318
>      .25      .50      .75      .90      .95
>   0.4599   0.5064   0.5613   0.6619   0.7650
>
> lowest : 0.1362745 0.1755365 0.2849285 0.3183792 0.3703046
> highest: 0.7823009 0.8113590 0.9114949 1.0106234 1.0872600
describe(Datos$agua.hab~Datos$Tipo)
> Datos$agua.hab ~ Datos$Tipo
>
>  2  Variables      97  Observations
> ---------------------------------------------------------------------------
> Datos$agua.hab
>        n  missing distinct     Info     Mean      Gmd      .05      .10
>       94        3       94        1   0.5256   0.1346   0.3935   0.4318
>      .25      .50      .75      .90      .95
>   0.4599   0.5064   0.5613   0.6619   0.7650
>
> lowest : 0.1362745 0.1755365 0.2849285 0.3183792 0.3703046
> highest: 0.7823009 0.8113590 0.9114949 1.0106234 1.0872600
> ---------------------------------------------------------------------------
> Datos$Tipo
>        n  missing distinct
>       96        1        3
>
> Value       Grande Mediano Pequeño
> Frequency       33      30      33
> Proportion   0.344   0.312   0.344
> ---------------------------------------------------------------------------
```

Para las representaciones gráficas, otros paquetes como `lattice` y `ggplot2`, este último especialmente, ofrecen al usuario mayor control y mejor interfaz para las representaciones.
Por ejemplo, el paquete `psych`dispone también de la función `cor.plot()`que ayuda a detectar los grupos de correlación por colores representando la matriz de correlación, lo cual es especialmente práctico cuando analizamos un número elevado de variables simultáneamente.
Es importante destacar que la matriz r debe ser un objeto tipo **matrix**.
Veamos, como ejemplo, la correlación entre las variables **agua.hab**, **elec.hab** y **res.hab**:

```
dev.off()
> null device
>           1
cor.plot(cor(Datos[,9:11], use="pairwise.complete.obs"))
```

2 Distribuciones de probabilidad con R

Este es el código de los *scripts* que se muestran en la presentación de la unidad DISTRIBUCIONES DE PROBABILIDAD, dentro de la asignatura **Herramientas estadísticas para Big Data** en el **Máster de Big Data Analytics** impartido en la UPV.

La mayoría de ejemplos están sacados de Sáez Castillo, A.J., 2010. Métodos Estadísticos con R y R Commander, Jaén: Universidad de Jaén

2.1 Función de densidad (f(x)), de probabilidad (P(x)) y de distribución (F(x))

2.1.1 DISCRETAS

B(n=10, p=0.25). ¿P(X = 3)?

```
# FUNCIÓN DE PROBABILIDAD, CUANTÍA O MASA P(x)
dbinom(3, 10, 0.25)
```

B(n=10, p=0.25). ¿P(X = 0), P(X = 1), P(X = 2) y P(X = 3)?

```
# FUNCIÓN DE DISTRIBUCIÓN F(x)
dbinom(0:3, 10, 0.25)
```

B(n=10, p=0.25). ¿P(X < 4)?

```
# Sumando las probabilidades
sum(dbinom(0:3, 10, 0.25))

# Utilizando la F(x). P(X <= 3)
pbinom(3, 10, 0.25)

# Utilizando la F(x) y las propiedades de la probabilidad
# P(X < 4) = 1 - P(X >= 4) = 1 - P(X > 3)
1 - pbinom(3, 10, 0.25, lower.tail = F)
```

B(n=10, p=0.25). ¿P(X > 2)?

```
# Utilizando la F(x). P(X > 2)
pbinom(2, 10, 0.25, lower.tail = F)
[1] 0.4744072
```

```
# Utilizando la F(x) y las propiedades de la probabilidad
# P(X > 2) = 1 - P(X <= 2)
1 - pbinom(2, 10, 0.25)
[1] 0.4744072
```

B(n=10, p=0.25). ¿P(2 < X <= 4)?

```
# P(2 < X <= 4) = P(X <= 4) - P(X <= 2)
F4 <- pbinom(4, 10, 0.25)
F2 <- pbinom(2, 10, 0.25)
F4
[1] 0.9218731
```

```
F2
```

```
[1] 0.5255928
```

```
F4 - F2
[1] 0.3962803
```

B(n=10, p=0.25). ¿P(2 <= X <= 4)?

```
# P(2 <= X <= 4) = P(1 < X <= 4) = P(X <= 4) - P(X <= 1)
F4 <- pbinom(4, 10, 0.25)
F1 <- pbinom(1, 10, 0.25)
F4
[1] 0.9218731
```

```
F1
[1] 0.2440252
```

```
F4 - F1
```

2.1.2 CONTINUAS

N(m=5, S=2). ¿Altura de la función de densidad en x = 7.6)?

```
# FUNCIÓN DE DENSIDAD f(x)
dnorm(7.6, 5, 2)
[1] 0.0856843
```

N(m=5, S=2). ¿P(X < 7.6)?

```
# FUNCIÓN DE DISTRIBUCIÓN F(x)
pnorm(7.6, 5, 2)
[1] 0.9031995
```

N(m=5, S=2). ¿P(X < 2)?

```
pnorm(2, 5, 2)
[1] 0.0668072
```

N(m=5, S=2). ¿P(X > 2)?

```
# Utilizando la F(x) y las propiedades de la probabilidad
# P(X > 2) = 1 - P(X <= 2)
1 - pnorm(2, 5, 2)
[1] 0.9331928
```

```
# Especificando la cola de la distribución
pnorm(2, 5, 2, lower.tail = F)
[1] 0.9331928
```

N(m=5, S=2). P(2< X < 7.6)?

```
## P(2< X < 7.6) = F(7.6)-F(2)
pnorm(7.6, 5, 2)-pnorm(2, 5, 2)
```

2.2 CUANTILES Q(x). VALORES CRÍTICOS

2.2.1 DISCRETAS

B(n=15, p=0.65). P(X <= x)=0.05 ¿x?

```
qbinom(0.05, 15, 0.65)
[1] 7
```

B(n=15, p=0.65). Obtener: - Primer y tercer cuartil (Q1 y Q3) - La mediana

```
# Q1 Primer cuartil x / P(X <= x)=0.25
qbinom(0.25, 15, 0.65)
[1] 9
```

```
# Q3 Tercer cuartil x / P(X <= x)=0.75
qbinom(0.75, 15, 0.65)
[1] 11
```

```
# Q3 Tercer cuartil x / P(X >= x)=0.25
qbinom(0.25, 15, 0.65, lower.tail = F)
[1] 11
```

```
# Me Mediana x / P(X >= x)=0.5 o P(X <= x)=0.5
qbinom(0.5, 15, 0.65)
[1] 10
```

```
qbinom(0.5, 15, 0.65, lower.tail = F)
[1] 10
```

Ps(Lambda=5.8). P(X <= x)=0.95 ¿x?

```
qpois(0.95, 5.8)
```

2.2.2 CONTINUAS

N(m=10, S=2). P(X <= x)=0.95 ¿x?

```
qnorm(0.95, 10, 2)
[1] 13.28971
```

N(m=0, S=1). P(Z <= z)=0.95 ¿z?

```
pnorm(0.95, 0, 1)
[1] 0.8289439
```

```
pnorm(0.95) # m=o y S=1 es por defecto
[1] 0.8289439
```

N(m=0, S=1). P(z1 <= Z <= z2)=0.95 ¿z1 y z2?

```
z1 <- qnorm((1-0.95)/2)
z2 <- qnorm((1-0.95)/2, lower.tail = F)
z1
[1] -1.959964
```

```
z2
[1] 1.959964
```

Observar que como la N(0,1) es simétrica con respecto a su media, z1 y z2 tendrán el mismo valor absoluto.
Obviamente, también se podían haber obtenido z1 y z2 como:

```
z1 <- qnorm((0.95 + 0.025), lower.tail = F)
z2 <- qnorm((0.95 + 0.025))
z1
[1] -1.959964
```

```
z2
[1] 1.959964
```

N(m=100, S=10). P(x1 <= Z <= x2)=0.99 ¿x1 y x2?

```
x1 <- qnorm(0.025, 100, 10)
x2 <- qnorm(0.025, 100, 10, lower.tail = F)
```

2.3 GRÁFICOS f(x), F(x) y Q(x) con v.a. CONTINUAS

Veamosla Función de densidad de una distribución Normal N(100, 10)

1. Obtener 2 valores (límites inferior y superior) que contengan la mayoría de los datos de esa normal, por ejemplo el 99%:

```
li<-round(qnorm(0.005, 100, 10), 0)
ls<-round(qnorm(0.005, 100, 10, lower.tail=F), 0)
# o ls<-round(qnorm(0.995, 100, 10), 0)
```

Otra alternativa es usar las propiedades de la distribución Normal:

```
# Valores de X. 99% valores están en [m-3sigma,m+3sigma]
# Límites. [m-3sigma,m+3sigma]
m<-100
sigma<-10
```

```
li<-round(m-3*sigma, 0)
ls<-round(m+3*sigma, 0)
```

2. Número de puntos a dibujar en el eje x

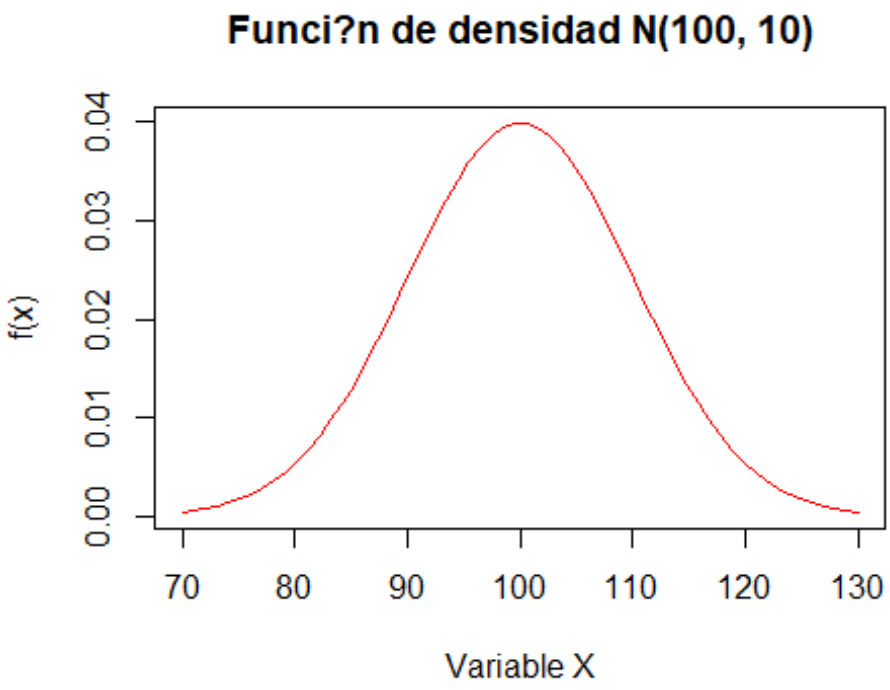
```
npuntos<-ls-li
x<-seq(li, ls, length.out=npuntos)
```

3. Valores de y

```
y<-dnorm(x, m, sigma)
```

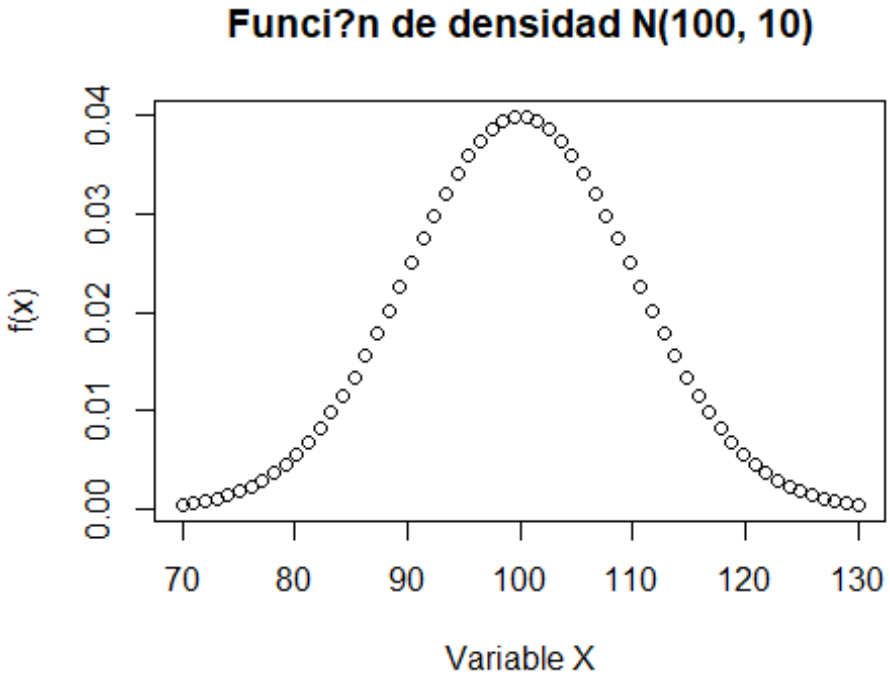
4. Dibujar f(x)

```
fnorm<-plot(x, y, type="l", xlab="Variable X", ylab="f(x)", main="Función de
densidad N(100, 10)", col="red")
```



O bien

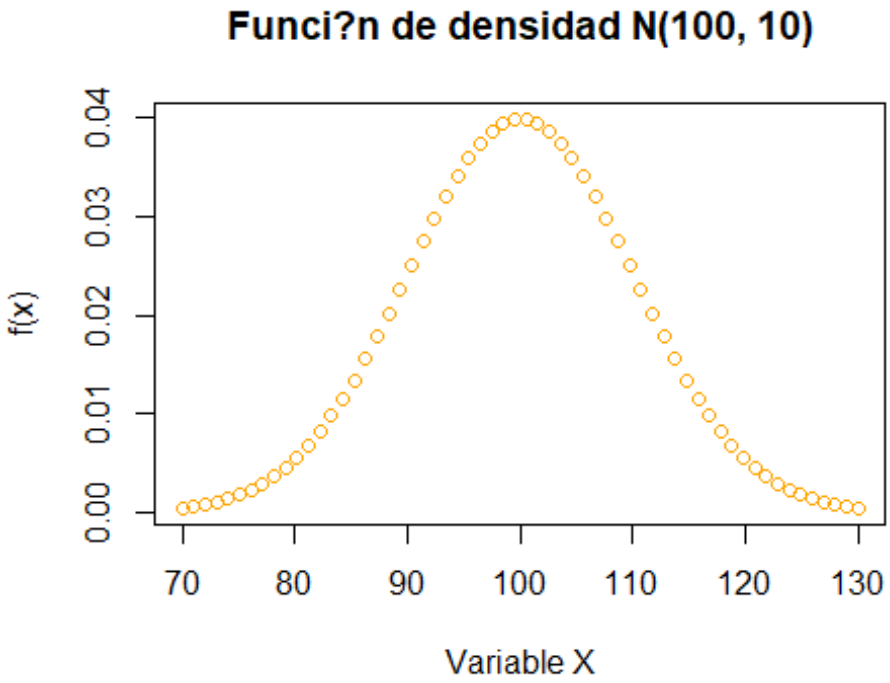
```
fnorm<-plot(x, y, type="p", xlab="Variable X", ylab="f(x)", main="Función de
densidad N(100, 10)", col="black")
```



O también:

O también:

```
fnorm<-plot(x, y, type="b", xlab="Variable X", ylab="f(x)", main="Función de
densidad N(100, 10)", col="orange")
```



2.3.1 Números aleatorios y representación de una variable continua N(m, sigma)

Generar 250 valores aleatorios de una N(100,10)

1. Asignamos los parámetros de la distribución:

```
# Qué media?
m<-100
m

# Qué desviación típica?
sigma<-10
sigma
```

2. Determinamos cuántos valores aleatorios se van a generar:

```
# Cuántos números?
n<-250
n
[1] 250
```

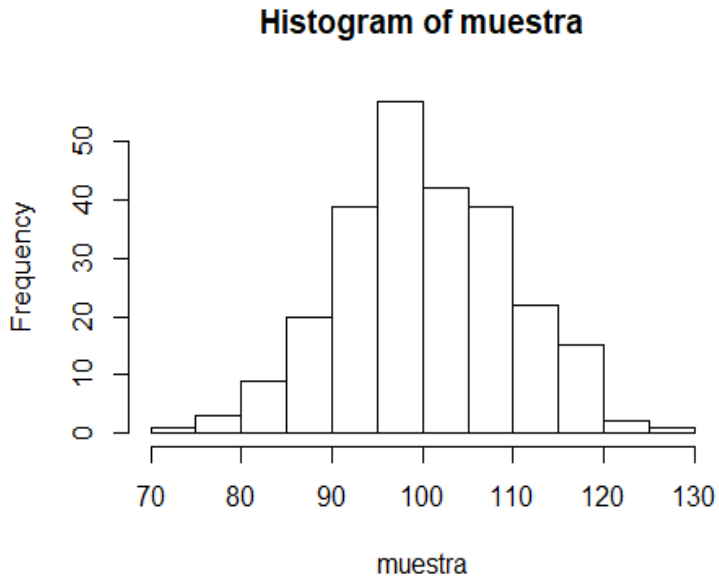
3.Generación de dos conjuntos de n números aleatorios:

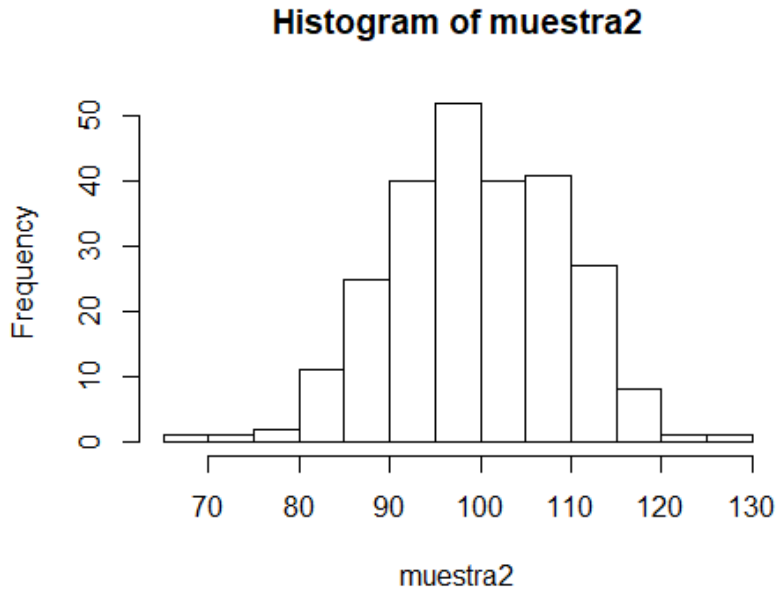
```
muestra <- rnorm(n, m, sigma)
muestra2 <- rnorm(n, m, sigma)

# muestra
hist(muestra)
mean(muestra)
sd(muestra)

# muestra 2
hist(muestra2)
mean(muestra2)
sd(muestra2)
```

Observad que cada muestra aleatoria es diferente y, por tanto, no tienenexactamente la misma media y desviación típica.
Es importante la **semilla** (`set.seed()`). Cuando se usa la misma semilla la secuencia aleatoroia generada es la misma:

```
set.seed(1024)
mean(rnorm(n, m, sigma))

set.seed(1024)
mean(rnorm(n, m, sigma))

set.seed(1024)
mean(rnorm(n, m, sigma))
```

```
# muestra
hist(muestra)
```





2.3.2 HISTOGRAMA de una distribución Normal

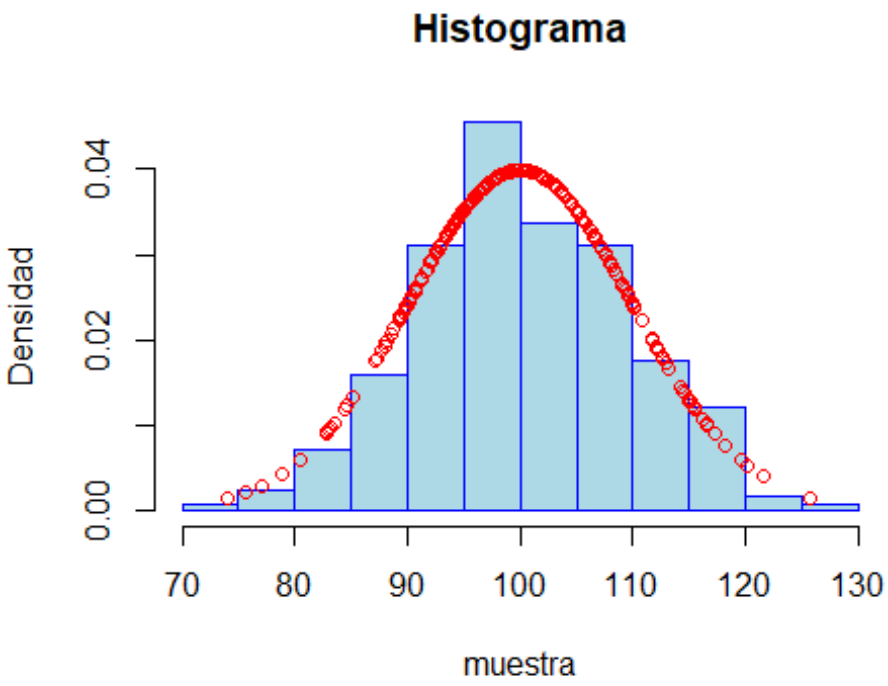
1. Determinar el número de intervalos. Una *regla del pulgar* es usar la raíz cuadrada del número de datos.

```
# Intervalos?
int<-round(sqrt(n), 0)
int
[1] 16
```

2. Dibujar histograma y f(x)

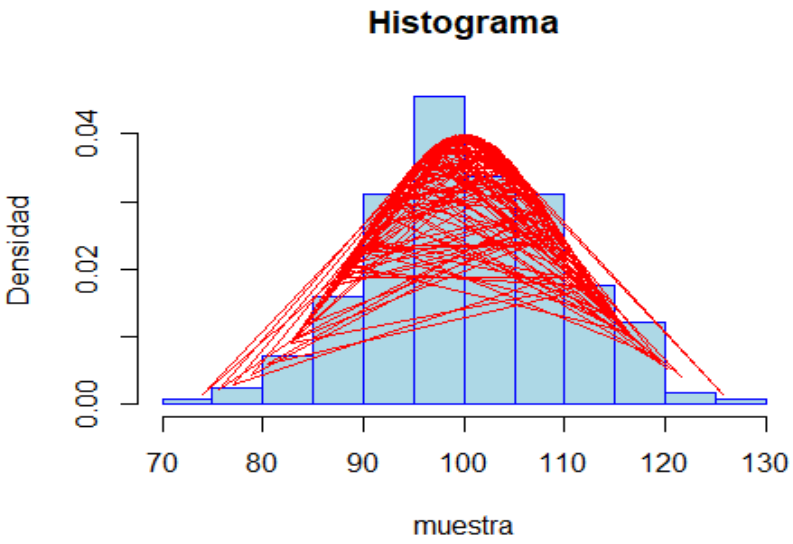
```
# Histograma - MUESTRA
hist(muestra, breaks=int, freq=F, xlab="muestra", ylab="Densidad",
main="Histograma", col="lightblue", border="blue")
# f(x) - POBLACIÓN
```

```
lines(muestra, dnorm(muestra,m,sigma), type="p", col="red")
```



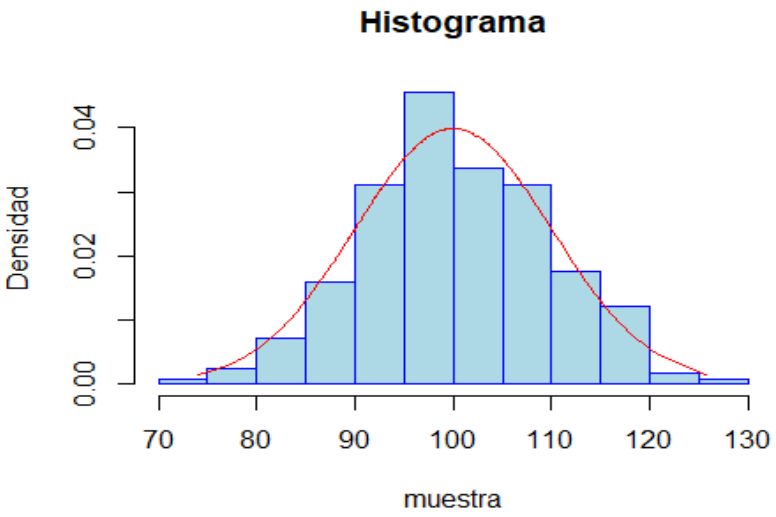
POdríamos hacerlo con líneas:

```
hist(muestra, breaks=int, freq=F, xlab="muestra", ylab="Densidad",
main="Histograma", col="lightblue", border="blue")

lines(muestra, dnorm(muestra,m,sigma), type="l", col="red")
```



Observar que tenemos que ordenar primero los puntos a graficar:

hist(muestra, breaks=int, freq=F, xlab="muestra", ylab="Densidad", main="Histograma", col="lightblue", border="blue")
lines(sort(muestra), dnorm(sort(muestra),m,sigma), type="l", col="red")



2.4 GRÁFICOS P(x), F(x) y Q(x) con v.a. DISCRETAS

Veamos la generación de números aleatorios y representación de una variable de Poisson(lambda)

1. Parámetros de la distribución

```
# Qué media (lambda)?
lambda<-2.5
```

2. Tamaño de la muestra

```
# Cuántos números?
n<-250
```

3. Generación de n números aleatorios

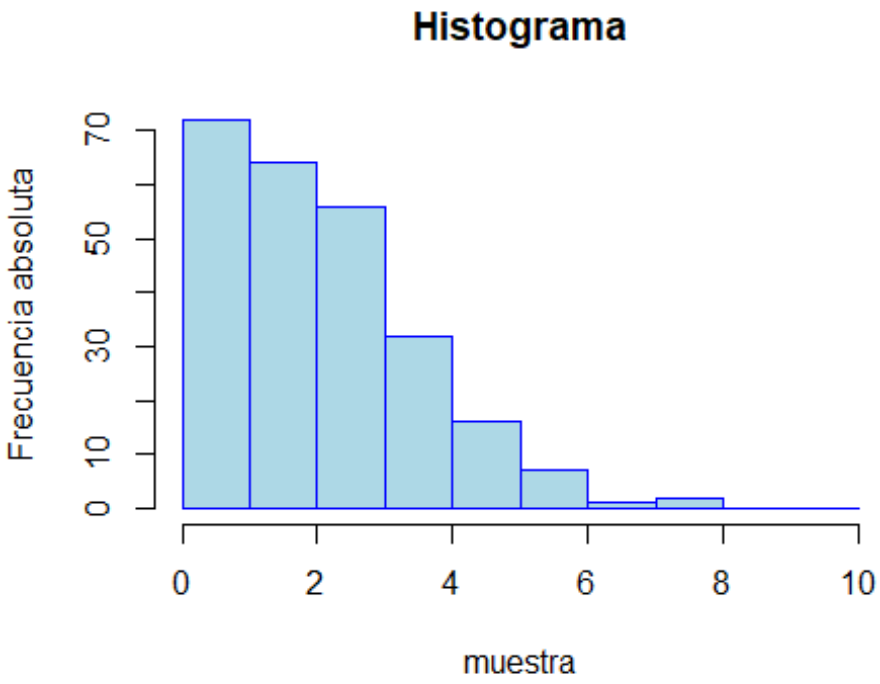
```
muestra<-rpois(n,lambda)
```

4. Límites para el eje X

```
# Intervalos?
# Calculados para que los enteros delimiten las barras (discreta)
c1<-0
c2<-trunc(qpois(0.9999, lambda))
```

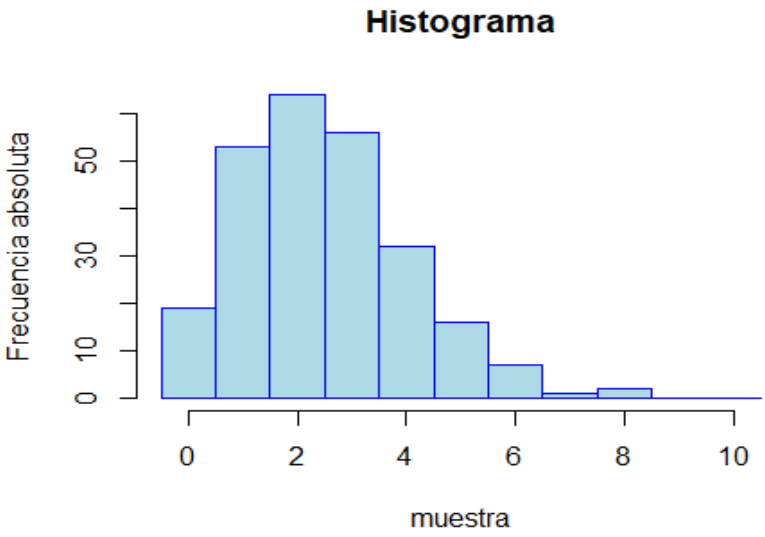
5. Dibujar el histograma

```
# Dibujar histograma con frecuencias absolutas
hist(muestra, breaks=c1:c2, freq=T, xlab="muestra", ylab="Frecuencia absoluta",
main="Histograma", col="lightblue", border="blue")
```



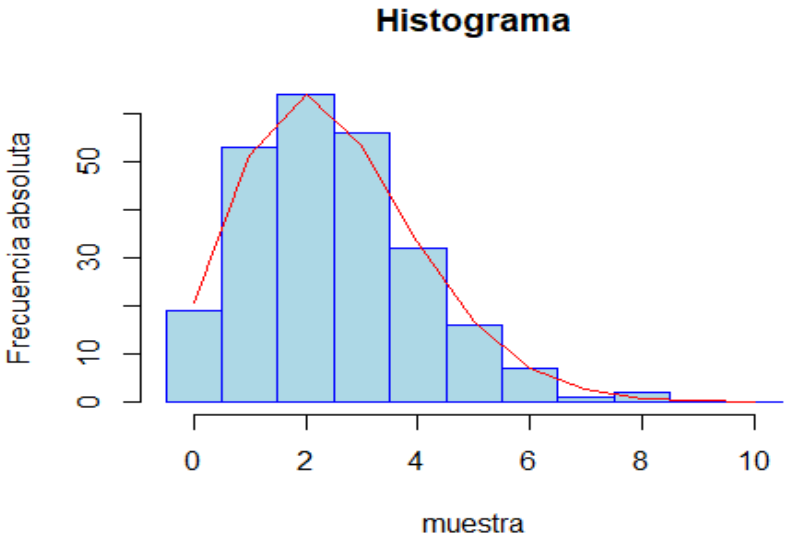
Para que el valor de la variable discreta esté en el centro de la barra:

```
hist(muestra, breaks=(c1-0.5):(c2+0.5), freq=T, xlab="muestra", ylab="Frecuencia
absoluta", main="Histograma", col="lightblue", border="blue")
```



6. Dibujar la f(x)

```
hist(muestra, breaks=(c1-0.5):(c2+0.5), freq=T, xlab="muestra", ylab="Frecuencia
absoluta", main="Histograma", col="lightblue", border="blue")
lines(c1:c2, dpois(c1:c2, lambda)*n, type="l", col="red", xpd=T)
```



2.5 Teorema central del límite

La suma de n (grande) v.a. independientes sigue una distribución normal.
Vamos a ver un ejemplo con la suma de 10 variables uniformes.
Generar 10 v.a uniformes U(2,3):

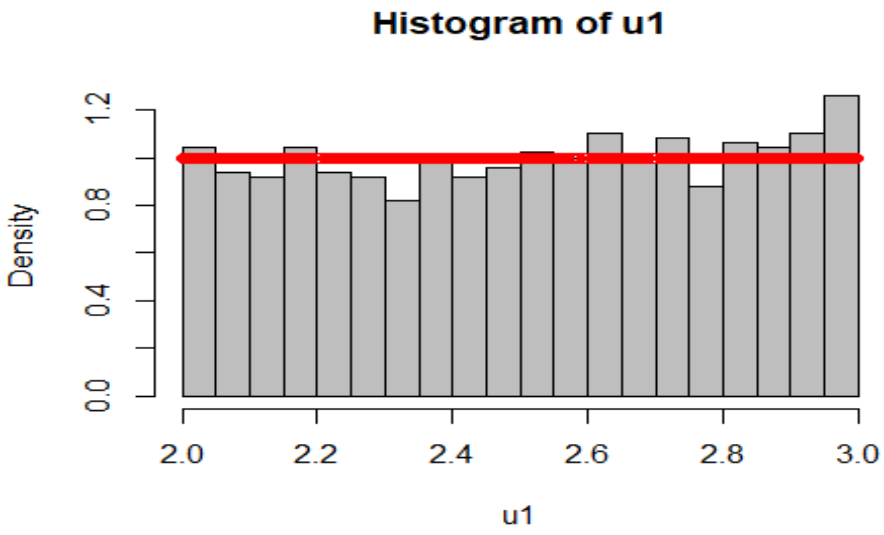
```
a<-2
b<-3
n<-1000
u1<-runif(n, a, b)
u2<-runif(n, a, b)
u3<-runif(n, a, b)
u4<-runif(n, a, b)
u5<-runif(n, a, b)
u6<-runif(n, a, b)
u7<-runif(n, a, b)
u8<-runif(n, a, b)
u9<-runif(n, a, b)
u10<-runif(n, a, b)
```

Otro modo de hacerlo sería con un bucle o:

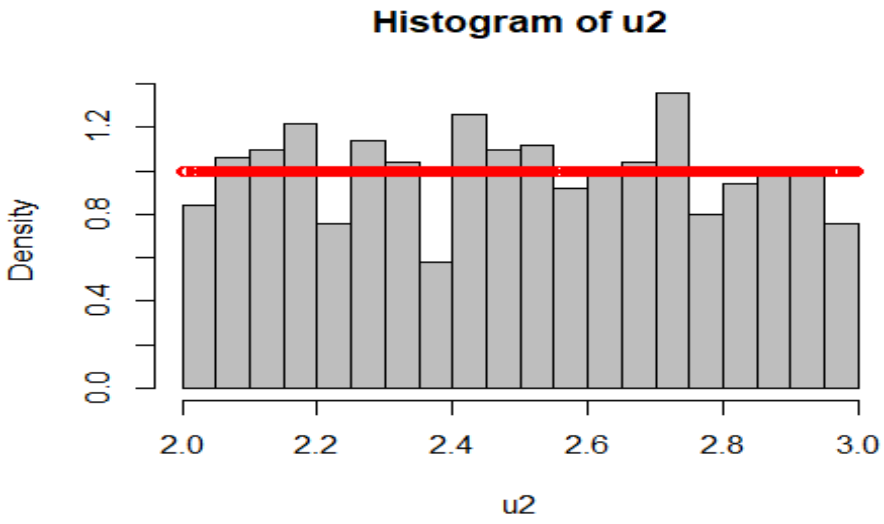
```
# Para generar una matriz con las 10 variables simuladas
u <- matrix(rep(runif(n, a, b), 10), n, 10, byrow = T)
```

Representar el histograma correspondiente y la función de densidad teórica de cada uniforma:

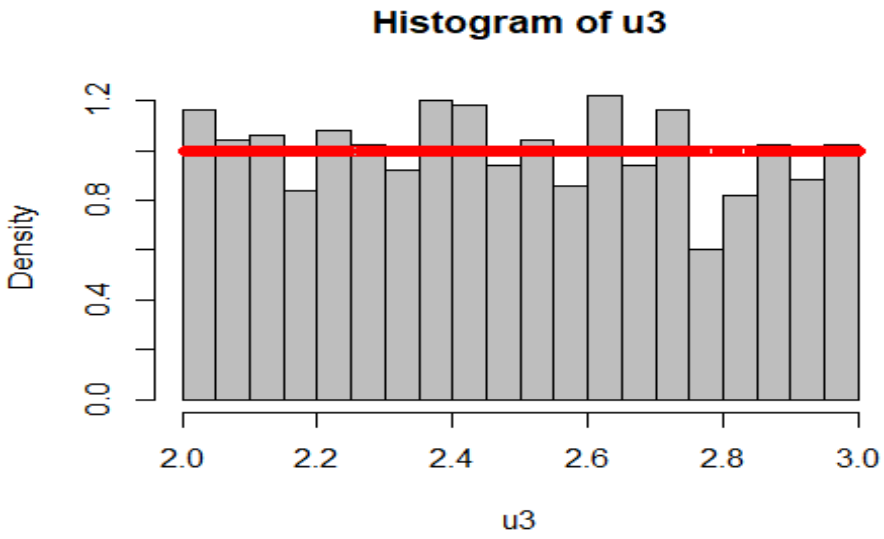
```
hist(u1, breaks=trunc(sqrt(n)), freq=F, col="gray")
lines(u1, dunif(u1, a, b), type="p", col="red")
```




```
hist(u2, breaks=trunc(sqrt(n)), freq=F, col="gray")
lines(u2, dunif(u2, a, b), type="p", col="red")
```




```
hist(u3, breaks=trunc(sqrt(n)), freq=F, col="gray")
lines(u3, dunif(u3, a, b), type="p", col="red")
```



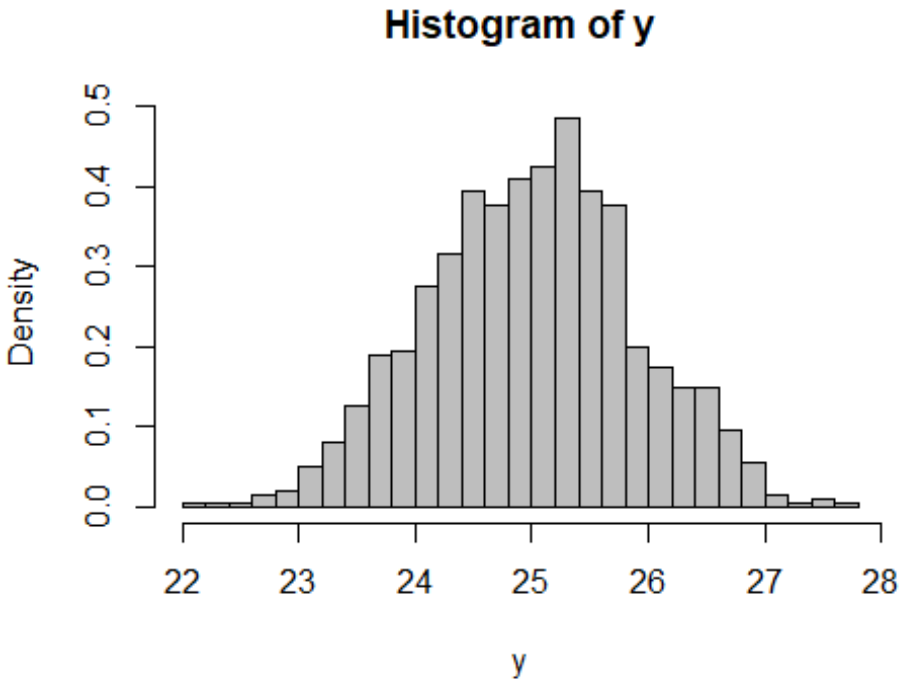

```
# ...
```

Crear una nueva v.a como suma de las anteriores:

```
## y=u1 + u2 + ... + u10
y<-u1+u2+u3+u4+u5+u6+u7+u8+u9+u10
```

Representar el histograma correspondiente

```
hist(y, breaks=trunc(sqrt(n)), freq=F, col="gray")
```



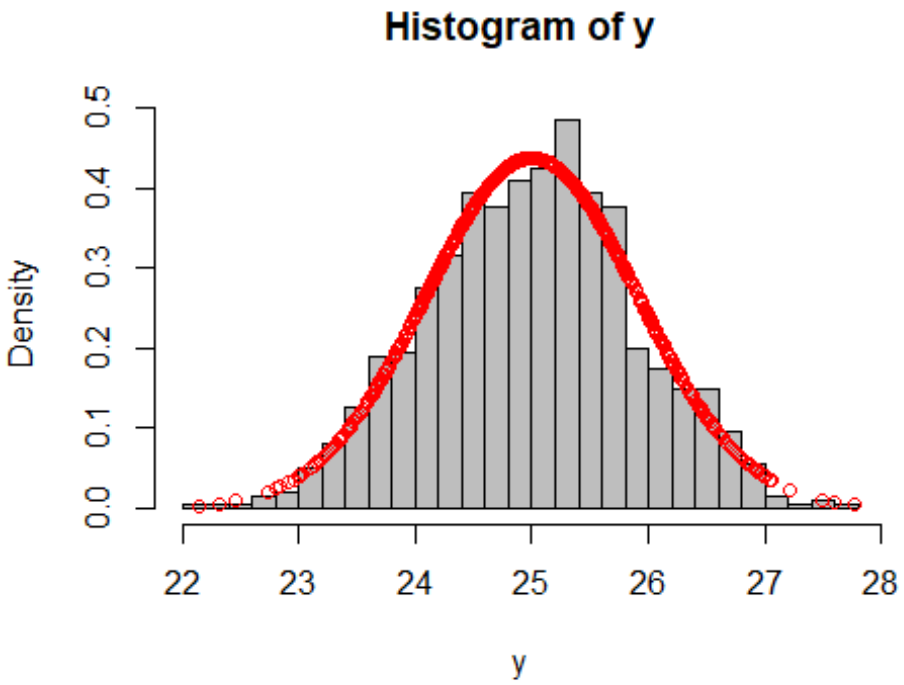
¿Se ajusta a la distribución Normal? La variable y tendrá una media y desviación típica teóricas que se pueden obtener como:

```
## Media y desviación típica de y
m<-10*((a+b)/2)
sigma<-sqrt(10*((b-a)^2)/12)
m
[1] 25
```

```
sigma
[1] 0.9128709
```

Veamos la forma de la función de densidad esperada para y

```
## función de densidad normal
hist(y, breaks=trunc(sqrt(n)), freq=F, col="gray")
lines(y, dnorm(y, m, sigma), type="p", col="red")
```



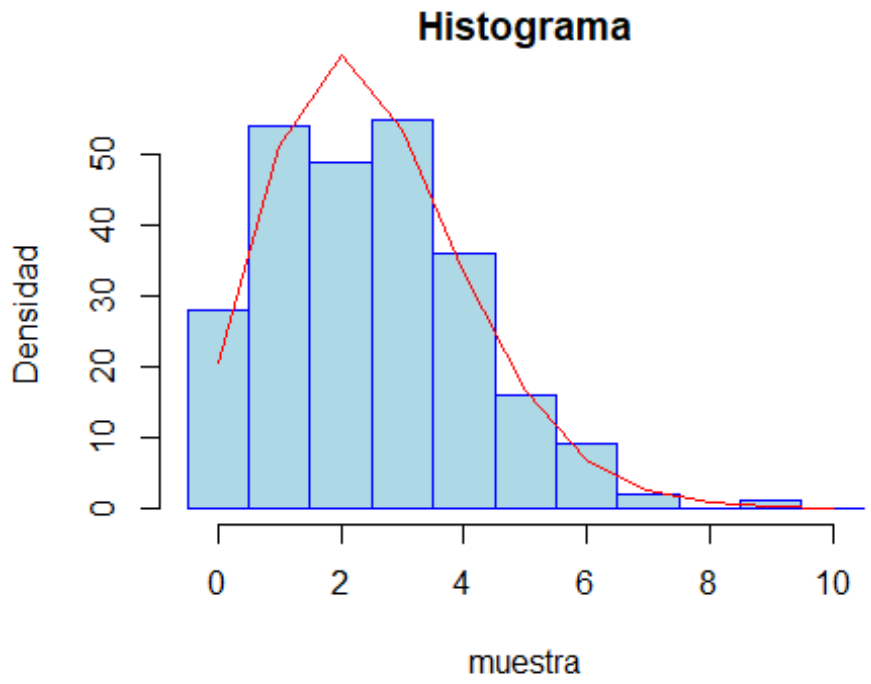
2.6 Aproximación normal

Sea una v.a. Ps(lambda = 2.5) y extraemos una muestra aleatoria de tamaño 250.

```
# Qué media (lambda)?
lambda<-2.5
# Cuántos números?
n<-250
## Generación de n números aleatorios
set.seed(500)
muestra<-rpois(n,lambda)
```

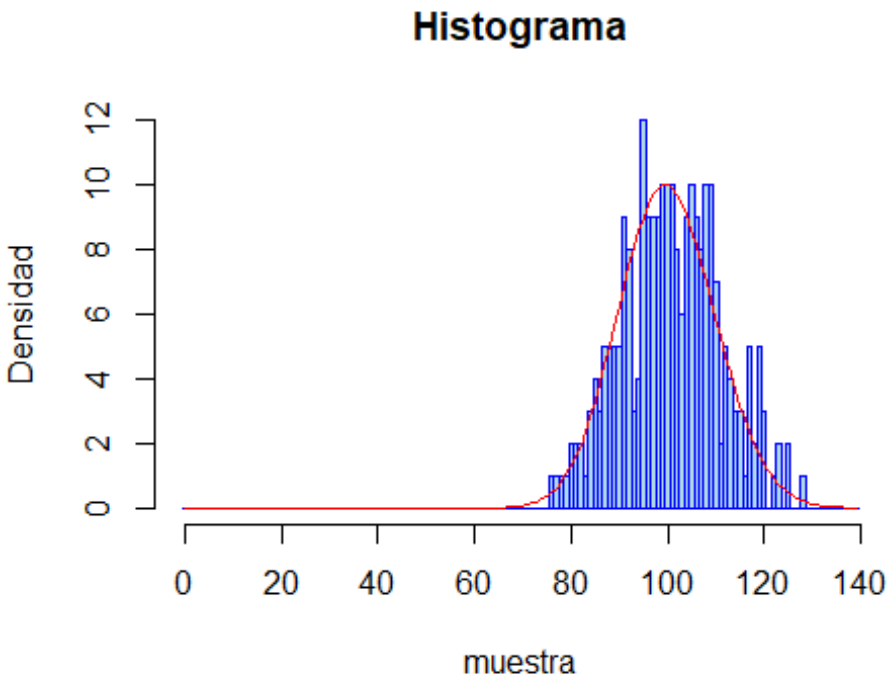
Veamos qué aspecto tiene la distribución de frecuencias mediante un HISTOGRAMA:

```r
# Intervalos?
# Calculados para que los enteros delimiten las barras (discreta)
c1<-0
c2<-trunc(qpois(0.9999, lambda))
# Dibujar histograma con frecuencias absolutas y que el valor de la variable
discreta esté en el centro de la barra
hist(muestra, breaks=(c1-0.5):(c2+0.5), freq=T, xlab="muestra", ylab="Densidad",
main="Histograma", col="lightblue", border="blue")
# Dibujar la f(x)
lines(c1:c2, dpois(c1:c2, lambda)*n, type="l", col="red", xpd=T)
```



Veamos qué aspecto tiene la distribución de frecuencias si cambiamos el lamda por 100, por ejemplo:

```r
# Qué media (lambda)?
lambda<-100
# Cuántos números?
n<-250
## Generación de n números aleatorios
set.seed(500)
muestra<-rpois(n,lambda)
c2<-trunc(qpois(0.9999, lambda))
# Dibujar histograma con frecuencias absolutas y que el valor de la variable
discreta esté en el centro de la barra
hist(muestra, breaks=(c1-0.5):(c2+0.5), freq=T, xlab="muestra", ylab="Densidad",
main="Histograma", col="lightblue", border="blue")
# Dibujar la f(x)
lines(c1:c2, dpois(c1:c2, lambda)*n, type="l", col="red", xpd=T)
```



Conforme aumentamos lambda, la f(x) se asemeja a la distribución normal.
Lo mismo ocurriría con una Binomial si aumentamos n y disminuimos p.

2.7 *ggplot2*

Parámetros de la variable

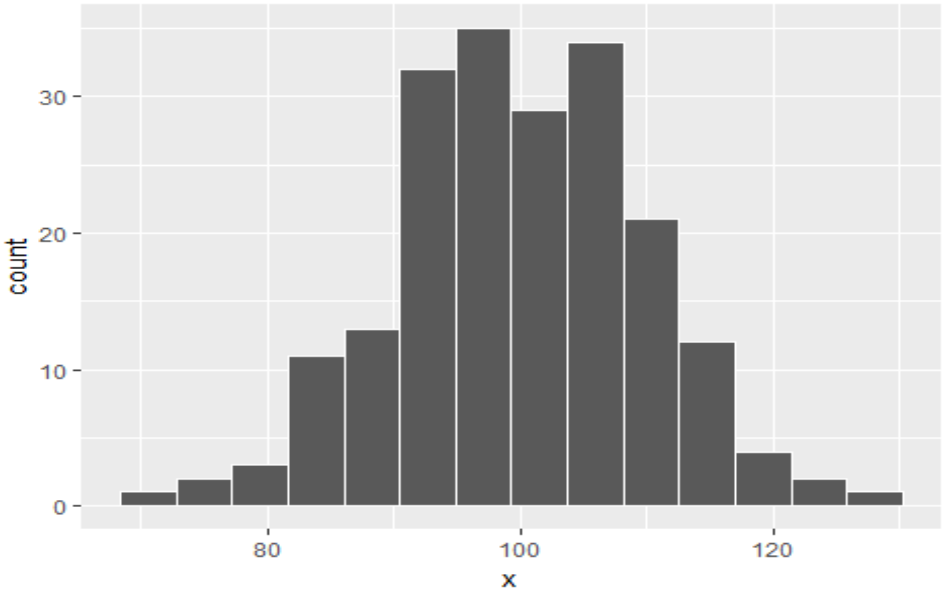
```
# Parámetros
m <- 100
sigma <- 10
n <- 200
# Generar los valores del eje X
x<-rnorm(n, m, sigma)
# Generar los valores del eje Y
y<-dnorm(x, m, sigma)
# Hacer data frame
d <- data.frame(x,y)
```

Cargar el paquete `ggplot2`:

```
if (!is.element("package::ggplot2", search())) library(ggplot2)
```

Dibujar el histograma

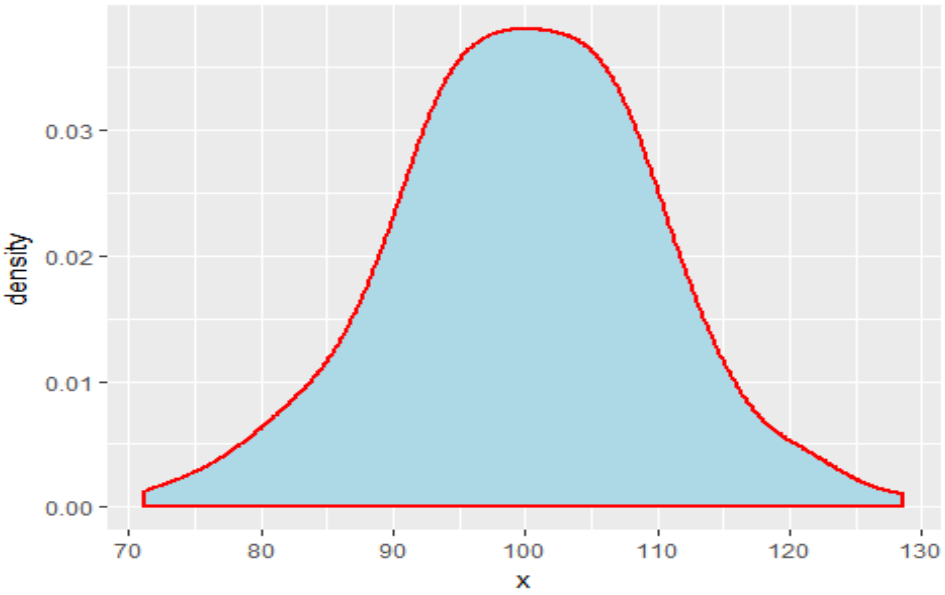
```
## Número de intervalos
int <- trunc(sqrt(n))
# Histograma
g <- ggplot(data = d, mapping = aes(x = x))
histograma <- g + geom_histogram(bins = int, color = "white")
histograma
```



Dibujar f(x)

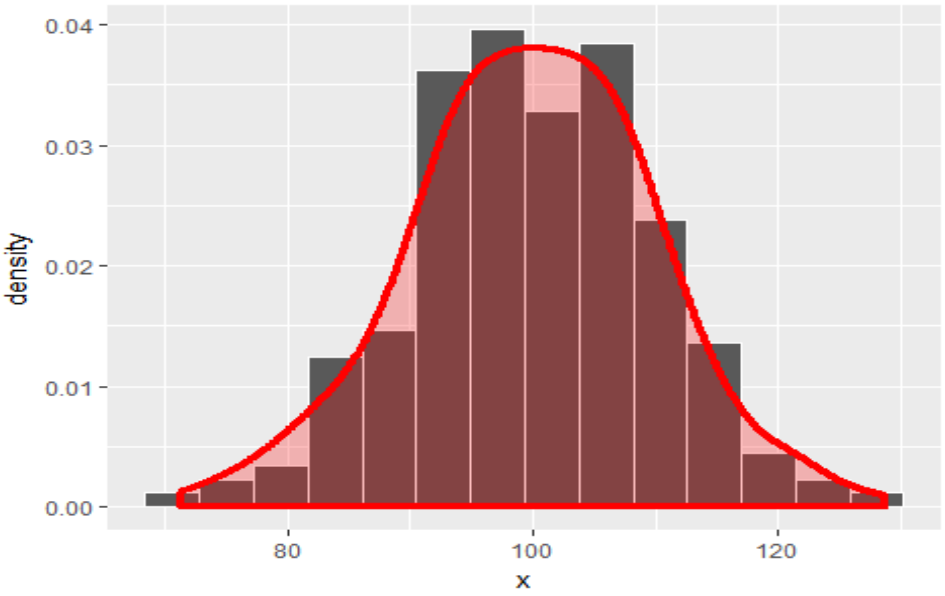
Dibujar f(x)

```
# f(x)
f <- g + geom_density(color = "red", size = 1, fill = "lightblue")
f
```



Dibujar ambos gráficos:

```
ggplot(data = d) +
  geom_histogram(mapping = aes(x = x, y = ..density..), color = "white", bins =
int) +
  geom_density(mapping = aes(x = x), color = "red", fill = "red", alpha = 0.25,
size = 1.5)
```


0.04
0.03
0.02
0.01
0.00
density
80
100
120
x

3 Inferencia con R

Este es el código de los *scripts* que se muestran en la presentación de la unidad INFERENCIA, dentro de la asignatura **Herramientas estadísticas para Big Data** en el **Máster dde Big Data Analytics** impartido en la UPV.

Algunos de los ejemplos están sacados de Sáez Castillo, A.J., 2010. Métodos Estadísticos con R y R Commander, Jaén: Universidad de Jaén

3.1 Estimación puntual de la media y desviación típica de las distribuciones y su error de estimación.

El paquete ***MASS*** contiene una función muy útil llamada ***fitdistr()*** que proporciona las estimaciones máximo verosímiles y los errores estándares asociados a esas estimaciones de las distribuciones más usuales dados unos datos.

3.1.1 Cargar paquete *MASS*

Para cargar y/o instalar paquete ***MASS***

```
if (!is.element("MASS", installed.packages())) {
  install.packages("MASS")
}
if (!is.element("package:MASS", search())) {
  library(MASS)
}
```

3.1.2 Función *fitdistr()*

La llamada a la función se realiza como *fitdistr(muestra, "modelo")*, donde:

- *muestra*: contiene los valores de la variable aleatoria a ajustar (sin valores perdidos)
- *modelo*: la distribución de la variable aleatoria que puede ser: "Poisson", "geometric", "negative binomial", "exponential", "gamma" y "normal".

El resultado de la función es:

- Una **estimación** del parámetro o parámetros (***estimate***). Este valor también lo proporciona el resultado de las funciones t.test y prop.test al efectuar contrastes de hipótesis.
- El **error estándar** (ET) de cada estimación o estimaciones (***sd***)

Es importante comentar que la función *fitdistr()* devuelve el valor exacto de los estimadores para las distribuciones de Poisson, geométrica, exponencial y normal, mientras que para las distribuciones binomial negativa y Gamma utiliza métodos numéricos para proporcionar valores aproximados de las estimaciones, ya que no existen fórmulas explícitas de los estimadores para esas distribuciones.

3.1.3 Ejemplo de distribución de Poisson

Un ejemplo clásico de la distribución de Poisson es el del estudio de muertes por coces de caballos de soldados del ejército prusiano entre 1875 y 1894.

Un ejemplo de estos datos se encuentra en el archivo *DatosMuertesCoces.rda*:

```
# MUESTRA.
load("DatosMuertesCoces.rda")

# Los mismos datos del data set se pueden obtener como
# Datos<-c(rep(0,109), rep(1,65), rep(2,22), rep(3,3), rep(4,1))
```

Veamos las estimaciones de los parámetros de la población a partir de los datos de esta muestra:

```
# AJUSTE
ajuste.poisson<-fitdistr(Datos.muertes$Muertes, "Poisson")
ajuste.poisson

##      lambda
##   0.61000000
##  (0.05522681)
```

Los resultados de la función se almacenan en una lista y son los siguientes:

```
# Ver nombres de los resultados del ajuste
names(ajuste.poisson)

## [1] "estimate" "sd"       "vcov"     "n"        "loglik"
```

Los valores estimados de la distribución de Poisson son la media (*estimate*) y el error de estimación de la media (*sd*)

```
# Estimación de la media (lambda)
lambda<-ajuste.poisson$estimate
lambda

## lambda
##   0.61

# Error de estimación de la media SE
SE<-ajuste.poisson$sd
SE

##     lambda
## 0.05522681
```

Se puede comprobar que el estimador máximo verosímil es la media de la muestra y que el error de estimación es la desviación típica de la media muestral:

```
# Media de la muestra
m<-mean(Datos.muertes$Muertes, na.rm=T)
m

## [1] 0.61
```

```
# Desviación típica de la media muestral
desvm<-sd(Datos.muertes$Muertes, na.rm=T)/(sqrt(length(Datos.muertes$Muertes)))
desvm

## [1] 0.05527001
```

3.1.4 Ejemplo de distribución Normal

Veamos un ejemplo con la distribución *Normal* y el conjunto de datos *datos1* contenido en el archivo *datos.RData*. Vamos a analizar la variable *GASTO* que recoge el gasto total efectuado por un cliente en una tienda *on line* en la primera visita a la web.

```
# MUESTRA.
load("datos.RData")
```

Eliminamos los valores perdidos de la variable, ya que *fitdistr()* no funciona con ellos:

```
gasto<-na.omit(datos1$GASTO)
```

Para obtener la estimación de los parámetros m y sigma de la distribución normal por máxima verosimilitud:

```
# AJUSTE
ajuste.Normal<-fitdistr(gasto, "normal")
ajuste.Normal

##       mean          sd
##   10.0545242    1.9004529
##  ( 0.1706657) ( 0.1206789)
```

Los valores estimados de la distribución de Poisson son la media (*m*) y la desviación típica (*sigma*) y sus errores de estimación:

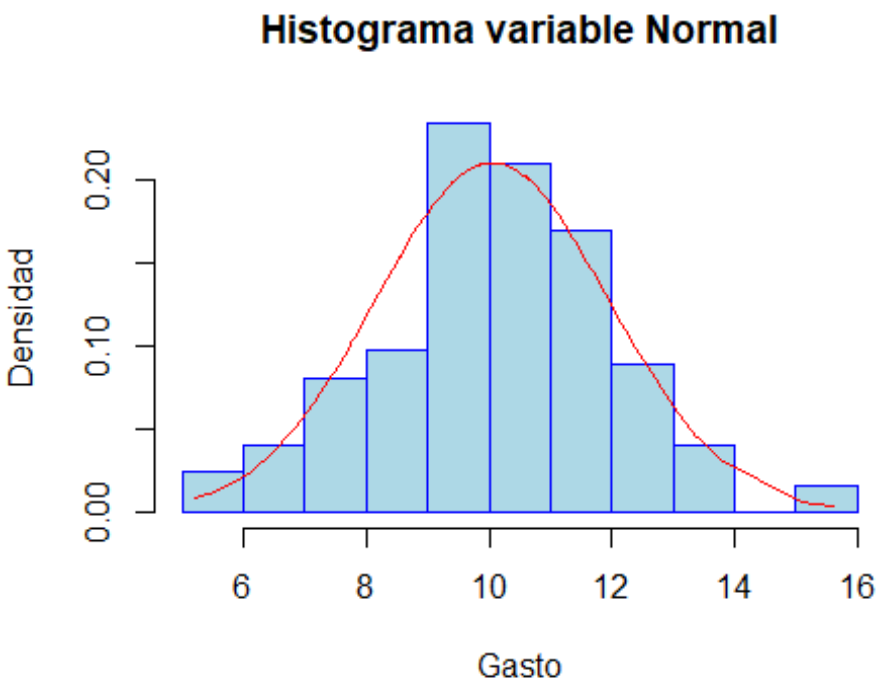
```
# Estimación de m y sigma
ajuste.Normal$estimate

##      mean        sd
## 10.054524  1.900453

mu<-ajuste.Normal$estimate[1]
sigma<-ajuste.Normal$estimate[2]
mu

##     mean
## 10.05452

sigma

##       sd
## 1.900453

# Error de estimación SE
SE<-ajuste.Normal$sd
SE

##      mean        sd
## 0.1706657 0.1206789
```

Se puede comprobar que los estimadores máximo verosímiles coinciden con los parámetros muestrales (m y sigma) y el error de estimación de la media con la desviación típica de la media muestral:

```
# Media y Desviación típica de la muestra
mean(gasto, na.rm=T)

## [1] 10.05452

sd(gasto, na.rm=T)

## [1] 1.908163

# Desviación típica de la media muestral
sd(gasto, na.rm=T)/sqrt(length(gasto))

## [1] 0.1713581
```

Veamos, gráficamente, cómo es el ajuste, comparando los datos muestrales (histograma) con el modelo teórico (función de densidad):

```
# Número de intevalos
int=trunc(sqrt(length(gasto)))

# Histograma
hist.gasto<-hist(gasto, breaks=int, freq=F, xlab="Gasto", ylab="Densidad", main="H
istograma variable Normal", col="lightblue", border="blue")

## Dibujamos la función de densidad teórica esperada para la variable Normal estud
iada, ordenando los valores previamente
lines(sort(gasto), dnorm(sort(gasto), mu, sigma), type="l", col="red")
```



Parece que, efectivamente, los datos muestrales se ajustan al modelo Normal.

3.2 Distribuciones en el muestreo

Sigamos con el caso de la variable *gasto total por cliente en una tienda on line* (X). Supongamos que esta variable sigue una distribución normal, como parece que así es, con una media poblacional (teórica) de 10 € y una desviación típica 2 €

X ~ N(m = 10, σ = 2)

Vamos a generar `J`muestras de `n` valores aleatorios de una normal con esa media y desviación típica teóricas (población) en un *data set* llamado `muestras`:

```
mu <- 10
sigma <- 2
n <- 100
J <- 10

muestras <- data.frame(V1 = 1:n)
for(j in 1:J) {
  set.seed(j*100)
  muestras[,j] <- rnorm(n, mu, sigma)
}

head(muestras, 2)
```

```
##          V1       V2       V3        V4       V5        V6       V7
## 1  8.995615 10.16951 12.74758  7.926902 11.93698  7.759717 9.824359
## 2 10.263062 10.45292 11.72421 11.230567 13.93074 10.396548 9.542067
##         V8        V9      V10
## 1 11.86913 10.244743 9.108443
## 2  9.89877  7.803526 7.588287
```

Veamos qué forma tienen las **distribuciones de frecuencias de cada muestra** y si se ajusta (como cabe esperar) a una normal con esos parámetros:

Calculemos los parámetros muestrales de cada muestra:

```
# Vector de medias muestrales
medias <- sapply(muestras, mean)
head(medias)
```

```
##        V1        V2        V3        V4        V5        V6
## 10.005825 10.057132 10.392881  9.911547  9.858436 10.114941
```

```
# Vector de desviaciones típicas muestrales
desv.tip <- sapply(muestras, sd)
head(desv.tip)
```

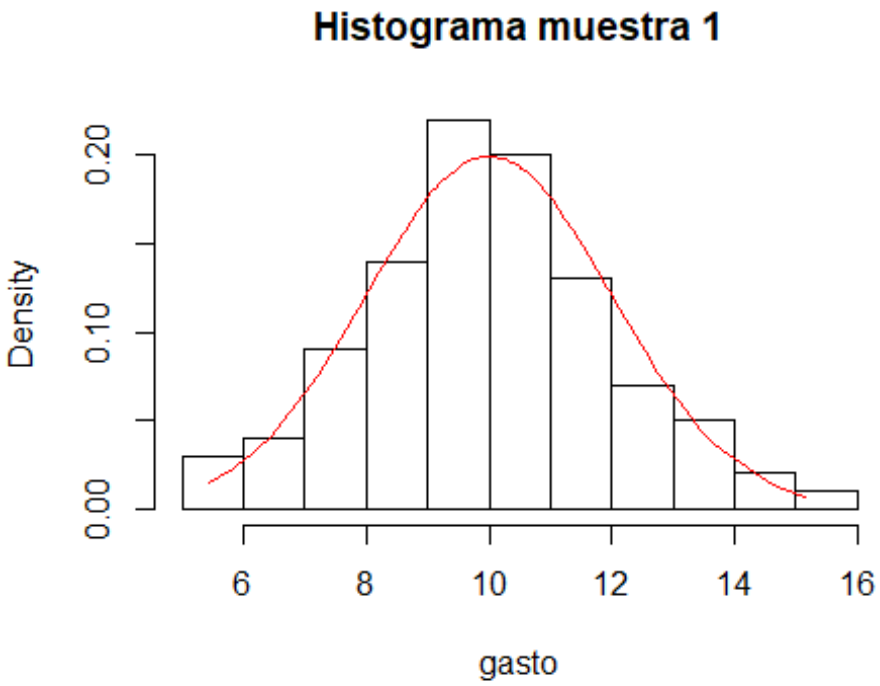
```
##       V1       V2       V3       V4       V5       V6
## 2.041421 1.839633 1.854621 1.863145 2.124520 2.028423
```

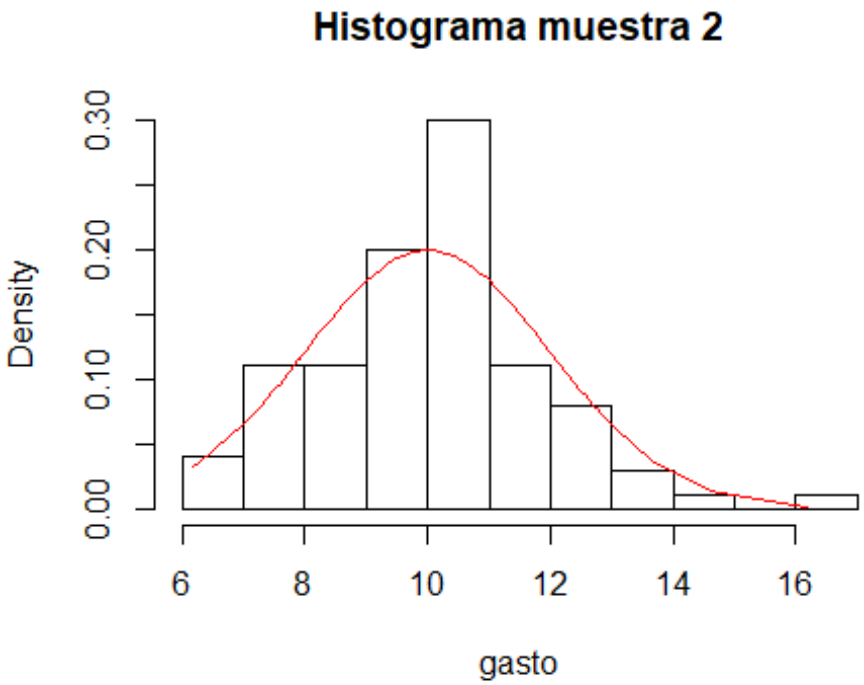
```
# Vector de varianzas muestrales
varianzas <- sapply(muestras, var)
head(varianzas)
```

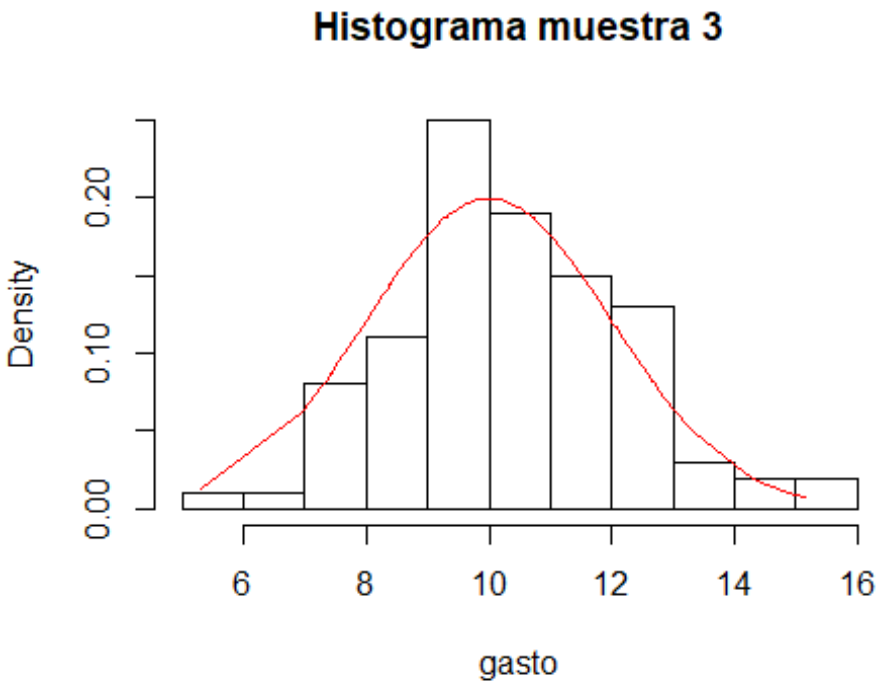
```
##       V1       V2       V3       V4       V5       V6
## 4.167399 3.384248 3.439618 3.471308 4.513583 4.114501
```

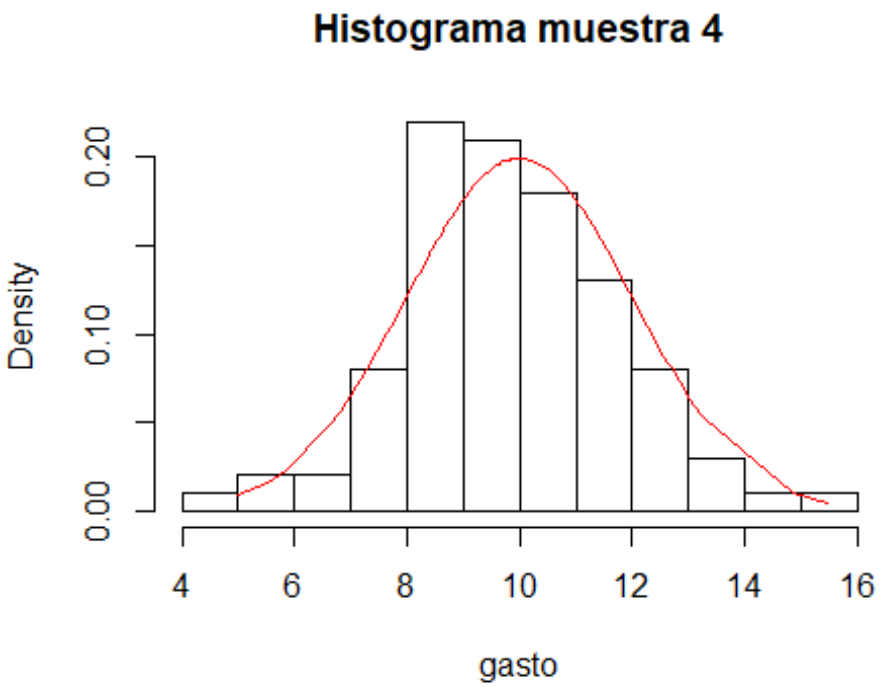
Comprobemos el ajsute a la distribución normal gráficamente:

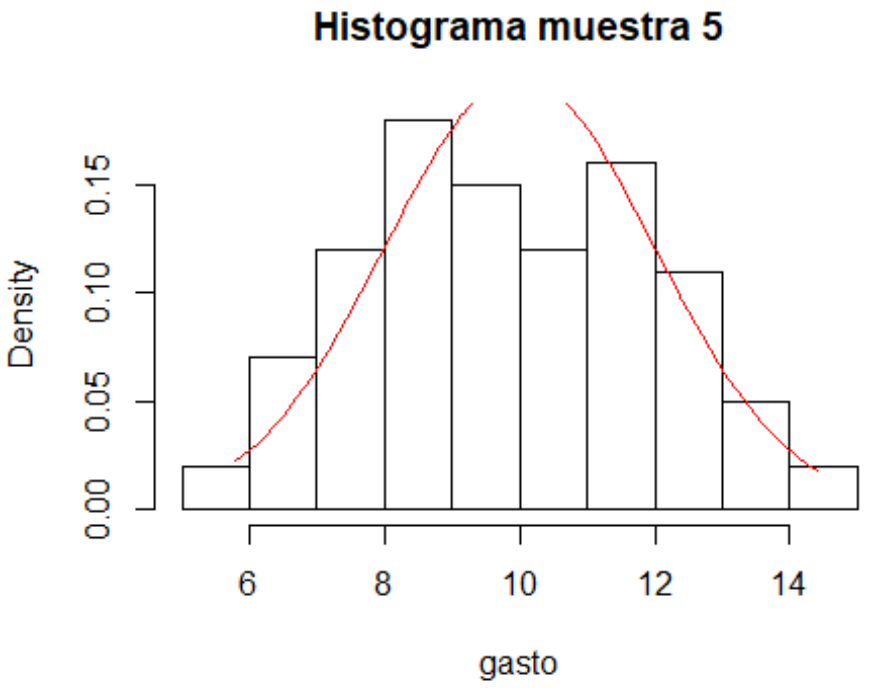
```
for (j in 1:J) {
  # Muestra (observado)
  hist(muestras[,j],breaks = trunc(sqrt(length(muestras[,j]))) ,freq = F, xlab = "
gasto", main = paste("Histograma muestra", j))

  # Población (teórico)
  lines(sort(muestras[,j]), dnorm(sort(muestras[,j]), mu, sigma), type="l", col="r
ed")
}
```

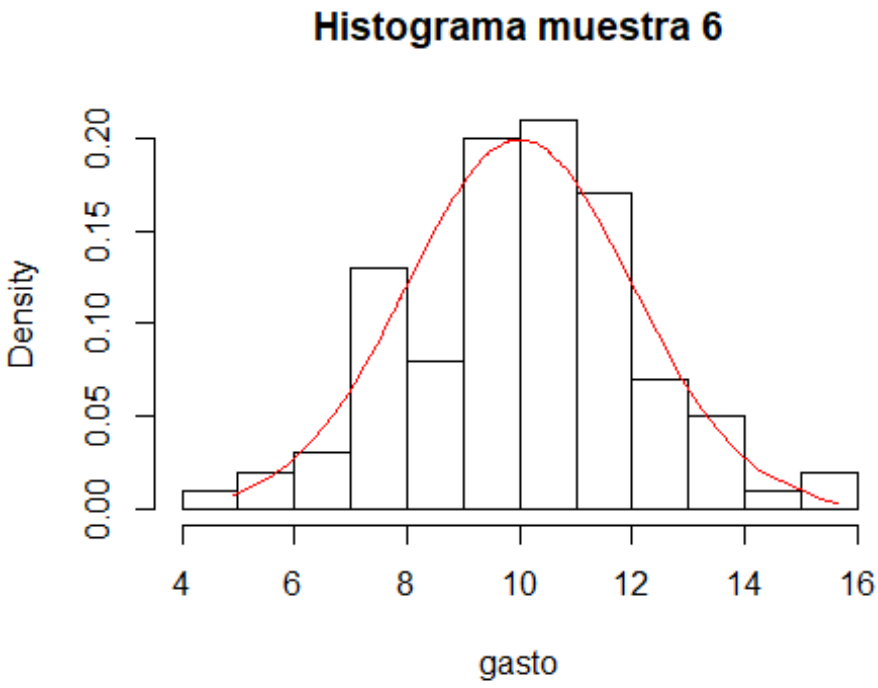

Histograma muestra 1
Density
0.00
0.10
0.20
6
8
10
12
14
16
gasto

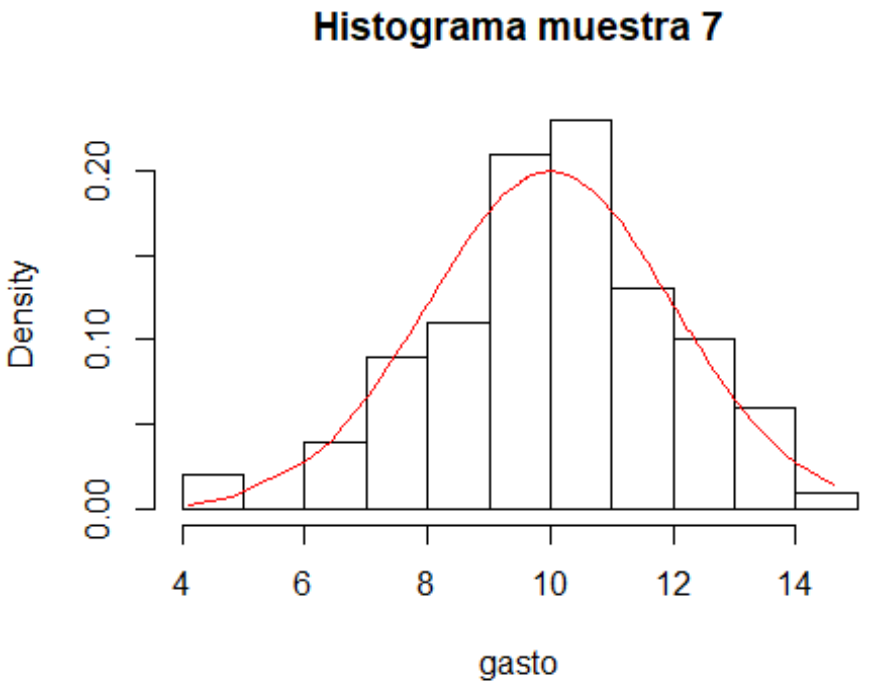

Histograma muestra 2
Density
0.00
0.10
0.20
0.30
6
8
10
12
14
16
gasto

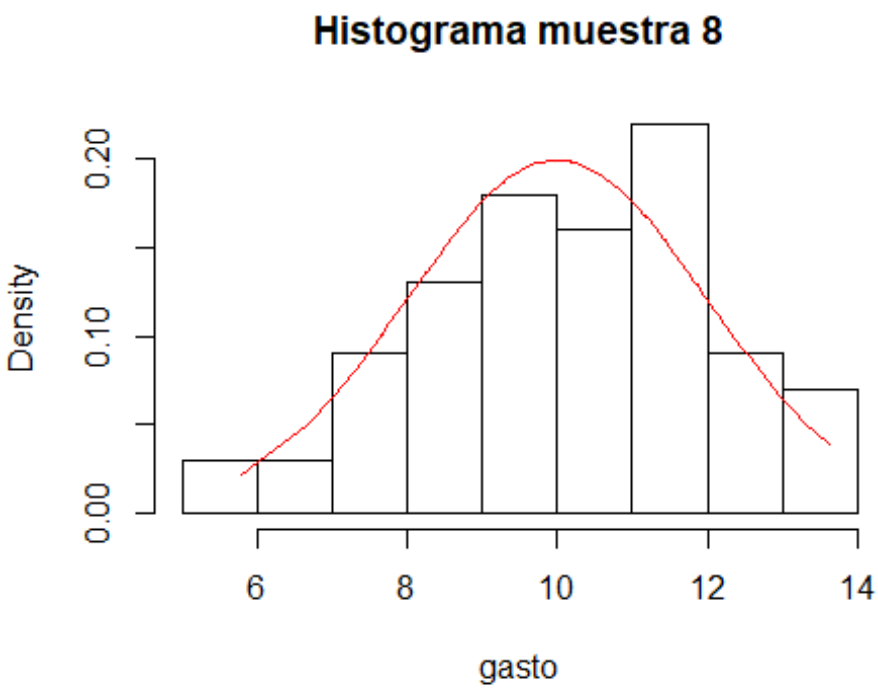

Histograma muestra 3
Density
0.00
0.10
0.20
6
8
10
12
14
16
gasto

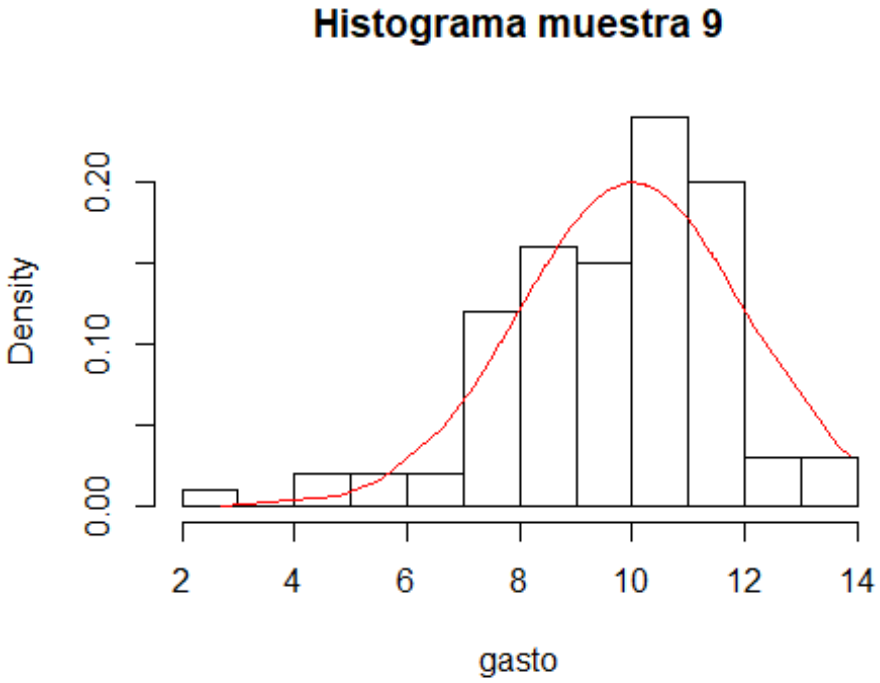

Histograma muestra 4
Density
0.00
0.10
0.20
4
6
8
10
12
14
16
gasto

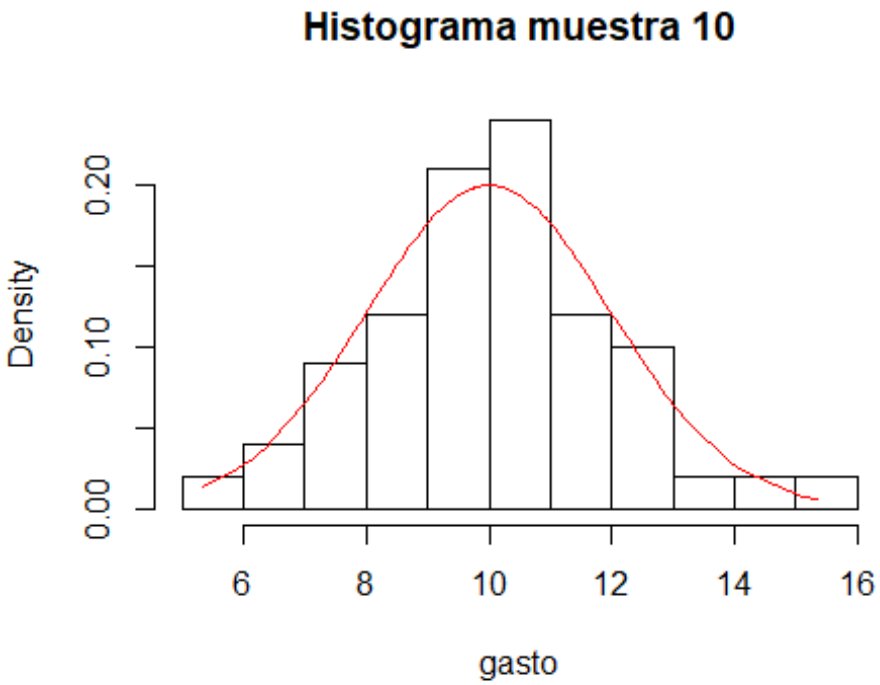

Histograma muestra 5
Density
0.00
0.05
0.10
0.15
6
8
10
12
14
gasto


Histograma muestra 6
Density
0.00
0.05
0.10
0.15
0.20
4
6
8
10
12
14
16
gasto


Histograma muestra 7
Density
0.00
0.10
0.20
4
6
8
10
12
14
gasto


Histograma muestra 8
Density
0.00
0.10
0.20
6
8
10
12
14
gasto



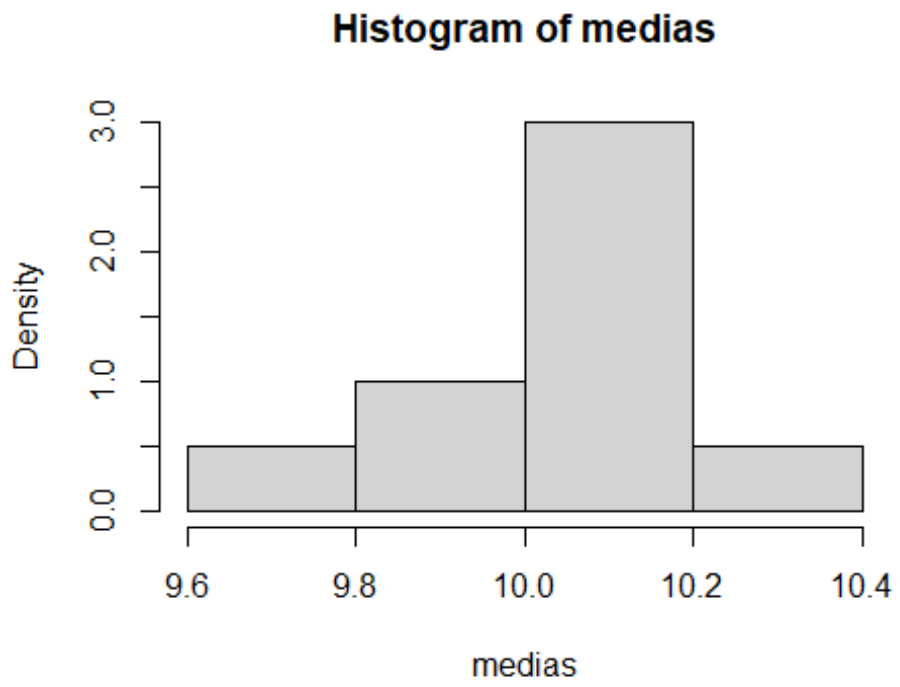


Las muestras parece que siguen un modelo Normal, como cabía esperar, pues como tales se han generado. Se representamos los parámetros muestrales, como las medias o las varianzas de las muestras, cada uno puede seguir un modelo de distribución de frecuencias distinto.

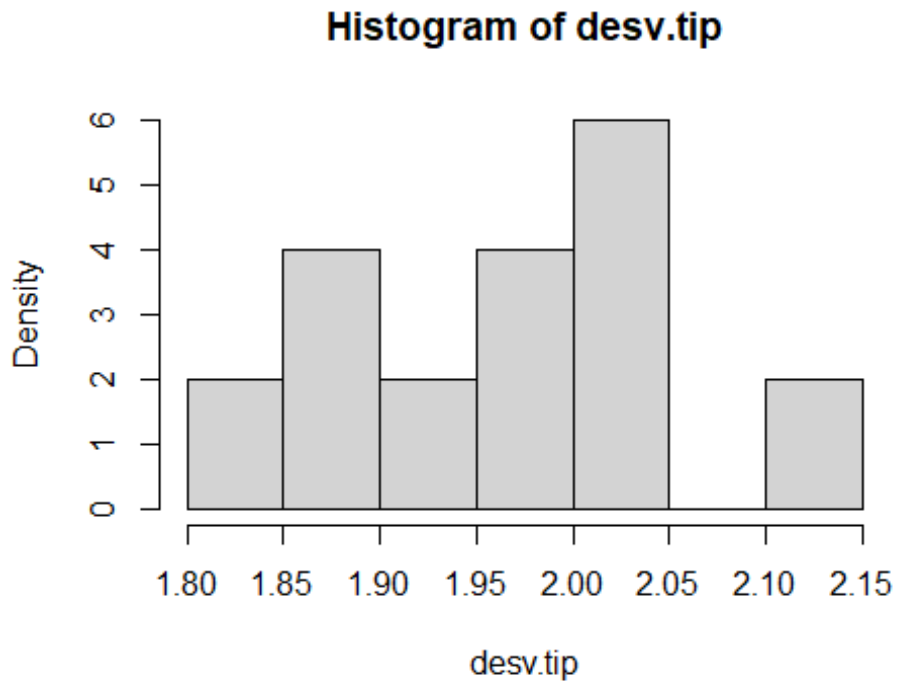
3.2.1 ¿Qué distribución de frecuencias (modelo) tienen los parámetros muestrales?

Vamos a representar las medias, varianzas y desviaciones típicas muestrales:

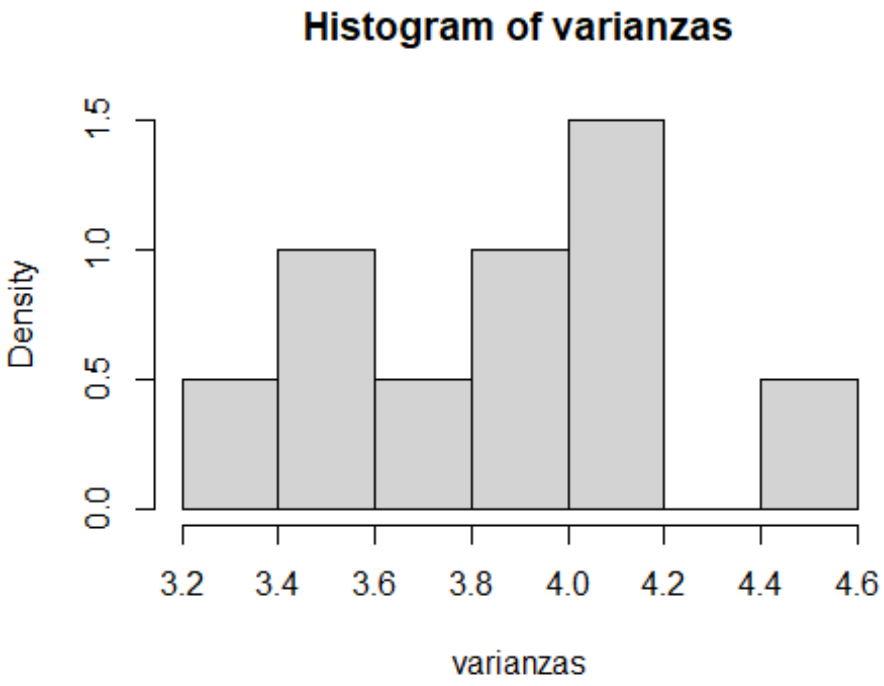
```r
hist(medias, freq = F, col = "lightgrey")
```




```r
hist(desv.tip, freq = F, col = "lightgrey")
```




```r
hist(varianzas, freq = F, col = "lightgrey")
```



Las distribuciones (modelos de probabilidad) de los parámetros muestrales y las de las relaciones entre éstos y los parámetros poblacioneles (**estadísticos**) se llaman **distribuciones en el muestreo**.

Se sabe que si X ~ N(m, sigma):

$$\bar{X} \underset{\substack{N\to\infty \\ (TCL)}}{\approx} N(\mathrm{m_X} = m, \sigma_X = \frac{\sigma}{\sqrt{N}})$$

$$S^2 \underset{\substack{N\to\infty \\ (TCL)}}{\approx} N(m_{S^2} = \sigma^2, \sigma_{S^2} \approx 0)$$

3.2.2 N(0,1)

Si X ~ N(mu = 10, sigma = 2):

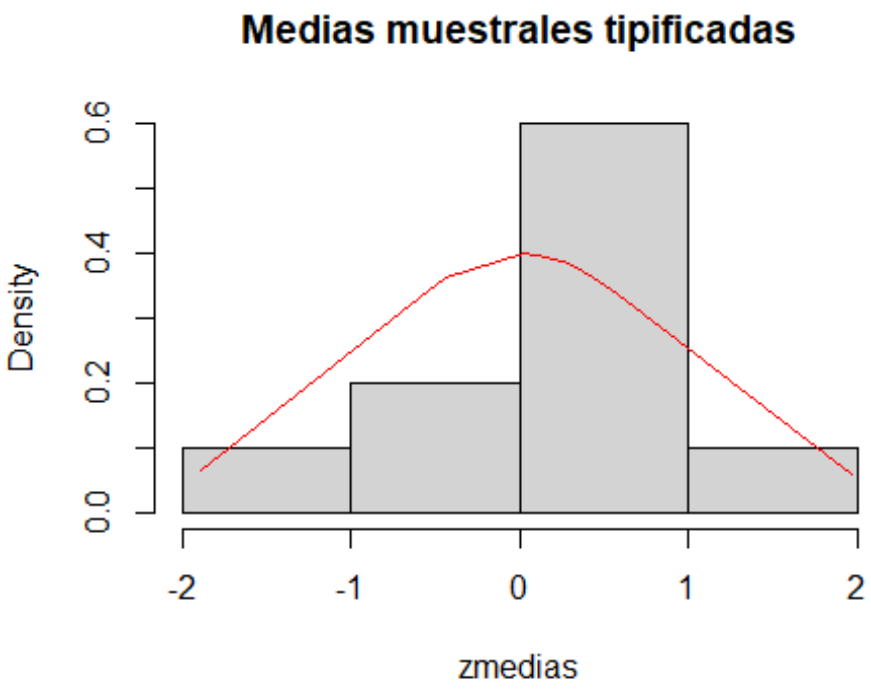
$$\frac{\bar{X} - m}{\sigma/\sqrt{n}} \approx Normal(0,1)$$

Esta expresión es el **estadístico z**

Vamos a comprobarlo:

```
# Muestra
zmedias <- (medias - mu)/(sigma/sqrt(n)) # Medias muestrales tipificadas

hist(zmedias, freq = F, main = "Medias muestrales tipificadas", col = "lightgrey")
```

```r
# Población
lines(sort(zmedias), dnorm(sort(zmedias), 0, 1), type = "l", col = "red")
```



A la vista del gráfico anterior: ¿Es probable que si tomo una muestra aleatoria de clientes, el *gasto* medio (tipificado) sea 2?

¿Es probable que si tomo una muestra aleatoria de clientes, el *gasto* medio (tipificado) sea 0?

¿Entre que par de valores debería estar el gasto medio de una muestra (una media muestral del *gasto*) en el 95% de los casos?

```r
# gasto medio ~ N(mu = 10, sigma = 2)
# media muestral ~ N(mu = 10, sigma = 2/raíz(n))
v1 <- qnorm(0.025, mu, sigma/sqrt(n), lower.tail = F)
v2 <- qnorm(0.025, mu, sigma/sqrt(n))
v1

## [1] 10.39199

v2

## [1] 9.608007
```

3.2.3 t de Student

Si X ~ N(mu = 10, sigma = **?**), esto es, **NO CONOCEMOS LA DESVIACIÓN TIPICA POBLACIONAL** (teórica):

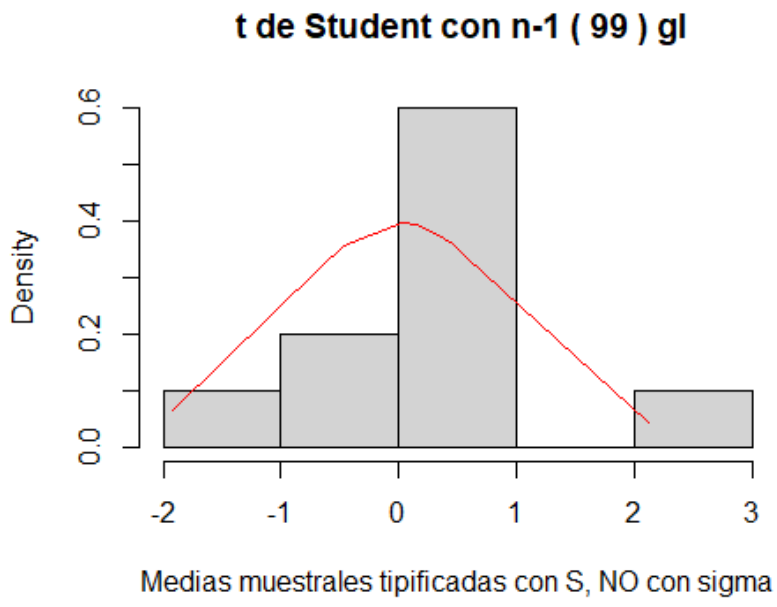
$$t = \frac{\bar{x} - m}{s/\sqrt{N}} \sim t_{N-1}$$

Esta expresión es el **estadístico t**

La diferencia con la N(0,1) es que, en este caso, al no conocer la desviación típica poblacional (sigma), la sustituimos por una estimación, la desviación típica muestral (S). Esto hace que la nueva variable aleatoria no siga un modelo N(0,1), sino uno muy parecido de media 0 y deviación típica teórica mayor que uno (acercándose a 1 cuando aumenta n) denominada **t de Student con *n-1* grados de libertad (gl)** (Gosset)

Vamos a comprobarlo:

```
# Muestra
tmedias <- (medias - mu)/(desv.tip/sqrt(n)) # Medias muestrales tipificadas con S

hist(tmedias, freq = F, xlab = "Medias muestrales tipificadas con S, NO con sigma"
, main = paste(" t de Student con n-1 (", n-1, ") gl"), col = "lightgrey")

# Población
lines(sort(tmedias), dt(sort(tmedias), n-1), type = "l", col = "red")
```



A la vista del gráfico anterior: ¿Es probable que si tomo una muestra aleatoria de clientes, el *gasto* medio (tipificado con S) sea 1?

¿Es probable que si tomo una muestra aleatoria de clientes, el *gasto* medio (tipificado S) sea 3?

3.2.4 Chi cuadrado

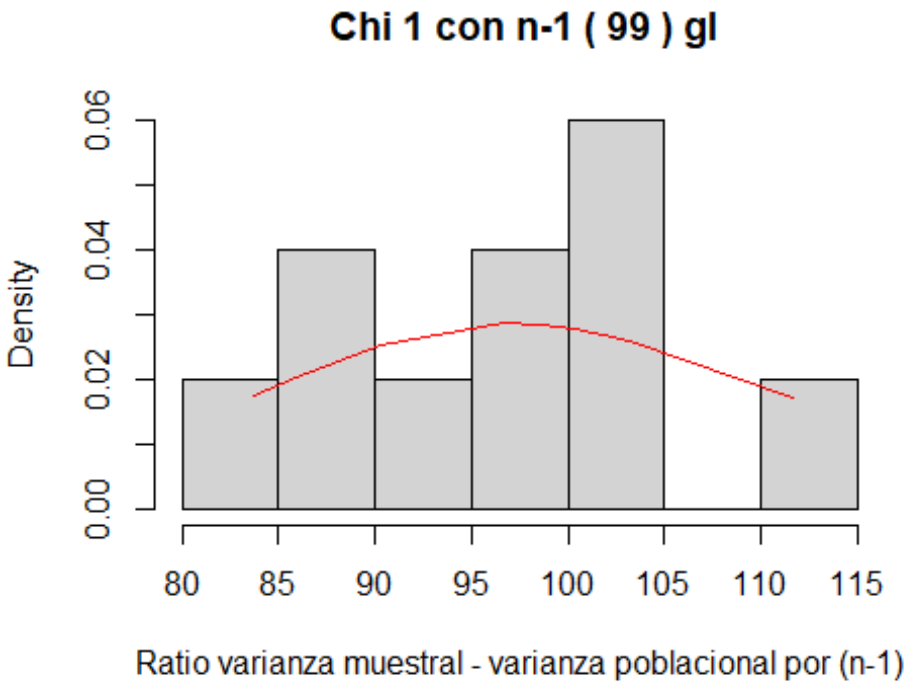
Si X ~ N(mu = 10, sigma = 2):

$$(N-1)\frac{s^2}{\sigma^2} \sim \chi^2_{N-1}$$

Esta expresión es el **estadístico Chi cuadrado**

Vamos a comprobarlo:

```
# Muestra
cocientes <- (n - 1)*(varianzas/sigma^2) # Cociente entre varianza muestral y pobl
acional

hist(cocientes, freq = F, xlab = "Ratio varianza muestral - varianza poblacional p
or (n-1)", main = paste(" Chi 1 con n-1 (", n-1, ") gl"), col = "lightgrey")

# Población
lines(sort(cocientes), dchisq(sort(cocientes), n-1), type = "l", col = "red")
```



3.2.5 F de Snedecor

Si X_1 ~ N(mu_1, $sigma_1$) y X_2 ~ N(mu_2, $sigma_2$):

$$\frac{s_1^2/\sigma_1^2}{s_2^2/\sigma_2^2} \sim F_{N_1-1,N_2-1}$$

$$\text{Si} \quad \sigma_1^2 = \sigma_2^2 \quad \Rightarrow \quad \frac{s_1^2}{s_2^2} \sim F_{N_1-1, N_2-1}$$

Los dos cocientes anteriores son **estadísticos F**

Vamos a comprobarlo:

Supongamos que tenemos un conjunto de muestras extraídas de una población N(mu_1 = 10, $sigma_1$ = 2) y otro conjunto de muestras de otra población N(mu_2 = 15, $sigma_2$ = 2). Ambas poblaciones representan el gasto total de un cliente en dos tiendas on line respectivamente.

Por simplificar hemos supuesto que ambas poblaciones tienen la misma varianza ($sigma_1$ = $sigma_2$ = 2)

```
mu1 <- 10
sigma1 <- 2

mu2 <- 15
sigma2 <- 2

n2 <- 100

J <- 10

muestras2 <- data.frame(V1 = 1:n2)
for(j in 1:J) {
  set.seed(2*j)
  muestras2[,j] <- rnorm(n2, mu2, sigma2)
}

head(muestras2, 2)
```

```
##         V1       V2       V3       V4       V5       V6       V7       V8
## 1 13.20617 15.43351 15.53921 14.83083 15.03749 12.03886 13.67630 15.95283
## 2 15.36970 13.91501 13.74003 16.68080 14.63149 18.15434 18.43791 14.74924
##         V9      V10
## 1 16.85292 17.32537
## 2 18.64564 13.82815
```

Las varianzas de las muestras de la segunda población serán:

```
varianzas2 <- sapply(muestras2, var)
```

Anque la varianza poblacional (teórica) del gasto en la segunda población sea la misma, las varianzas muestrales no lo serán, aunque deberían ser muy parecidas.

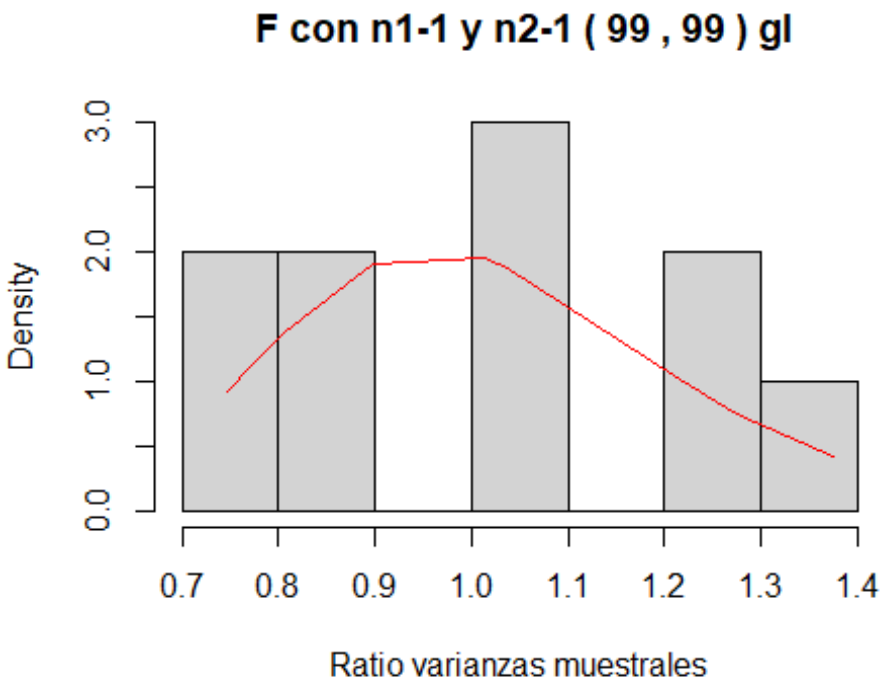
```
head(varianzas2)
```

```
##       V1       V2       V3       V4       V5       V6
## 5.384158 3.340656 4.272303 4.653575 3.543700 2.993648
```

Veamos la distribución de frecuencias de este ratio de varianzas:

```
# Muestra
ratio.varianzas <- varianzas/varianzas2 # Cociente entre varianzas muestrales
```

```
hist(ratio.varianzas, freq = F, xlab = "Ratio varianzas muestrales", main = paste(
" F con n1-1 y n2-1 (", n-1, ",", n2-1, ") gl"), col = "lightgrey")

# Población
lines(sort(ratio.varianzas), df(sort(ratio.varianzas), n-1, n2-1), type = "l", col
= "red")
```



Ahora no podríamos plantear cuestiones como la siguiente.

Si las varianzas poblacionales son iguales, como en este caso, ($sigma_1$ = $sigma_2$ = 2), si tomo dos muestras cualquiera de cada población y obtengo sus varianzas y calculo su ratio:

```
# Varianza de una muestra de la población 1 (la 3, por ejemplo)
S1 <- varianzas[3]

# Varianza de una muestra de la población 2 (la 10, por ejemplo)
S2 <- varianzas[10]

# Ratio de varianzas muestrales
S1/S2

##        V3
## 0.8486754
```

A la vista del histograma anterior, ¿Cómo de probable es que ese valor del ratio sea un valor de una F con 99 y 99 grados de libertad?

3.3 Intervalos de confianza y contraste de hipótesis con funciones del paquete *stats*.

En este apartado se explica, mediante varios ejemplos, la obtención, resolución e interpretación de los **intervalos de confianza** y **contrastes de hipótesis** sobre la **media**, la **varianza** (y **desviación típica**), sobre una **proporción**, la **comparación de medias**, la **comparación de varianzas** y la **comparación de proporciones**, usando algunas de las funciones básicas del paquete *stats*:

- ***t.test()***
- ***cor.test()***
- ***var.test()***
- ***prop.test()***

El *data set* que vamos a utilizar, *datos1*, está contenido en el archivo *datos.RData*, como se ha mencionado antes.

Este *data frame* recoge información sobre 13 características de 131 individuos tales como:

- SEXO
- EDAD
- MES
- ESTATURA
- PESO
- PROVINCIA
- X
- ACCESOS
- TIPO
- GASTO
- COMPRA1
- GASTO2
- COMPRA2

En caso de que no esté cargado aún el data set:

```
load("Datos.RData")
str(datos1)
```

```
## 'data.frame':    131 obs. of  13 variables:
##  $ SEXO     : Factor w/ 2 levels "mujer","varon": 1 1 2 2 2 1 1 1 1 1 ...
##  $ EDAD     : int  20 20 19 19 20 23 21 21 20 22 ...
##  $ MES      : int  NA NA NA 12 6 10 5 4 8 4 ...
##  $ ESTATURA : int  NA 164 185 NA NA 159 160 155 163 172 ...
##  $ PESO     : int  54 NA 70 63 63 54 NA 48 57 59 ...
##  $ PROVINCIA: Factor w/ 5 levels "","Alicante",..: 1 1 1 1 1 2 2 2 2 2 ...
##  $ X        : int  3 4 5 8 5 5 7 3 7 7 ...
##  $ ACCESOS  : int  30 10 30 20 12 65 45 30 25 45 ...
##  $ TIPO     : Factor w/ 6 levels "","aparatos",..: 4 4 4 3 4 6 6 3 2 6 ...
##  $ GASTO    : num  6.12 NA 8.54 NA NA ...
##  $ COMPRA1  : Factor w/ 2 levels "no","si": 2 2 1 1 1 1 1 1 1 1 ...
##  $ GASTO2   : num  7.38 10.98 10.43 12.32 9.39 ...
##  $ COMPRA2  : chr  "no" "no" "no" "no" ...
```

Vamos a calcular los parámetros muestrales de alguna de las variables aleatorias que vamos a usar:

- ***GASTO*** Variable que recoge el gasto total efectuado por un cliente en una tienda *on line* en la primera visita a la web.

```
### Variable "GASTO". Data set "datos1". Archivo "datos.RData"

# Eliminamos los valores perdidos de la variable.
gasto<-na.omit(datos1$GASTO)

## Obtención parámetros muestrales media y S
media.gasto<-mean(gasto, na.rm=T)
S.gasto<-sd(gasto, na.rm=T)
N.gasto<-length(gasto)

media.gasto

## [1] 10.05452

S.gasto

## [1] 1.908163

N.gasto

## [1] 124

# Desviación típica de la media muestral (standar error)
SE.gasto<-S.gasto/sqrt(N.gasto)
```

- ***GASTO2*** Variable que recoge el gasto total efectuado por un cliente en una tienda *on line* en la segunda visita a la web.

```
### Variable "GASTO2". Data set "datos1". Archivo "datos.RData"

# Eliminamos los valores perdidos de la variable.
gasto2<-na.omit(datos1$GASTO2)

## Obtención parámetros muestrales media y S
media.gasto2<-mean(gasto2, na.rm=T)
S.gasto2<-sd(gasto2, na.rm=T)
N.gasto2<-length(gasto2)

media.gasto2

## [1] 10.03871

S.gasto2

## [1] 2.087197

N.gasto2

## [1] 131

# Desviación típica de la media muestral (standar error)
SE.gasto2<-S.gasto2/sqrt(N.gasto2)
```

- ***COMPRA1*** Variable que recoge si un cliente compra en una tienda *on line*, diferente de la tienda asociada los *gasto*y *gasto2*, en la primera visita a la web.

```
### Variable "COMPRA1". Data set "datos1". Archivo "datos.RData"

# Eliminamos los valores perdidos de la variable.
compra1<-na.omit(datos1$COMPRA1)

## Obtención parámetros muestrales media y S
summary(compra1)

##  no  si
## 118  13

N.compra1<-length(compra1)
N.compra1

## [1] 131
```

3.3.1 Intervalos de confianza para la media

Vamos a construir un IC para la media del gasto de los clientes en la web con un nivel de confianza del 95%.

Para ello usaremos la función ***t.test***. Esta función permite realizar el contraste sobre una media o la comparación de las medias de dos poblaciones.

- **Sintaxis de *t.test()***

El test da como resultado la resolución del contraste y un intervalo de confianza para la media o para la comparación de medias, según sea el caso.

Así pues, aunque no nos pidan explícitamente la realización del contraste de hipótesis, usaremos la función ***t.test*** para la obtención del intervalo de confianza.

Recordemos en primer lugar la sintaxis de la función ***t.test***:

```
## Default S3 method:
t.test(x, y = NULL,
       alternative = c("two.sided", "less", "greater"),
       mu = 0, paired = FALSE, var.equal = FALSE,
       conf.level = 0.95, ...)

## S3 method for class 'formula'
t.test(formula, data, subset, na.action, ...)
```

Donde:

- ***x*** es un vector de datos correspondiente a una de las muestras o a la única muestra del problema. Si estamos haciendo un test sobre la media de una población, x contendrá la única muestra; si estamos realizando un test de comparación de medias, x será la primera de las dos muestras.

- ***y*** correspondería a la segunda muestra en un test de comparación de medias. Si no es el caso y estamos en un test sobre una sola población, simplemente no se incluye.

- ***alternative*** especifica la dirección de la hipótesis alternativa. Como puede verse, tiene 3 posibles valores, ***"two.sided"*** (bilateral), ***"less"*** (unilateral a la izquierda) y ***"greater"*** (unilateral a la derecha).
- ***mu*** es el valor hipotético con el que se compara la media o la diferencia de medias en el contraste (hipótesis nula)
- ***paired*** especifca si las dos muestras x e y, en caso de que aparezcan, son apareadas o no. Si se comparan dos muestras, ***var.equal*** especifica si podemos suponer varianzas iguales o no.
- ***conf.level*** es el nivel de confianza de los intervalos que se mostrarán asociados al test (en tanto por 1).

En este primer caso no vamos a comparar medias, vamos a hacer inferencia sobre una sola población, sólo es necesario indicar el argumento *x* que, en nuestro caso, es la variable ***gasto***, el **nivel de confianza** (*conf.level*) en tanto por uno y si se trata de un intervalo de confianza acorde con el contraste bilateral (*alternative = "two.sided"*) o unilateral (*alternative = "less"* o *alternative = "greater"*). Como la opción por defecto para la segunda variable (*y*) es *NULL* (no comparamos nada), no hace falta escribir el segundo argumento.

Los argumentos *paired* y *var.equal*, como la segunda variable (*y*), sólo tienen sentido si comparamos medias, por lo que tampoco los usaremos.

El único argumento que nos puede confundir es el valor del argumento *mu*, que representa el valor de la media poblacional a contrastar. En principio, como no se solicita la realización del contraste propiamente dicho, el valor de la media de la población (parámetro *mu*) nos es indiferente, por defecto R contrastará *mu = 0*, pero no importa, sólo nos interesa el intervalo de confianza.

- **Obtención del IC al 95% de confianza**

Bien, realicemos el test. Los resultados los almacenaremos en un objeto (lista) llamado **test.gasto1**, así, los diferentes resultados generados estarán accesibles cuando queramos:

```
test.gasto1 <- t.test(gasto, alternative = "two.sided", conf.level = 0.95)

test.gasto1

>
>   One Sample t-test
>
> data:  gasto
> t = 58.676, df = 123, p-value < 2.2e-16
> alternative hypothesis: true mean is not equal to 0
> 95 percent confidence interval:
>   9.715331 10.393717
> sample estimates:
> mean of x
>  10.05452
```

El **intervalo de confianza al 95%** es **[9.7153314, 10.393717]**.

No necesitamos ningún resultado adicional del test.

- **Interpretación del IC al 95% de confianza**

El IC es otra forma, más intuitiva, de dar una estimación del gasto medio *teórico* (media poblacional del gasto), usando un intervalo en vez de un sólo valor del parámetro.

El IC está calculado de forma que se tenga una probabilidad "alta", 95% en este caso, de que el verdadero valor de la media del gasto (media poblacional) que tratamos de estimar esté dentro de dicho intervalo.

3.3.2 Contraste o test de hipótesis para la media del gasto

Vamos a plantear un test de hipótesis que permita determinar si el gasto medio *teórico* podría ser de 10€ y a resolverlo sin usar el intervalo de confianza (riesgo de 1ª especie 5%), con un test o contraste.

El contraste que se plantea es:

H_0: $m_A = 10$

H_1: $m_A <> 10$

Ahora sí usaremos la función ***t.test*** y todos sus resultados.

Como tampoco vamos a comparar medias en este caso, sólo es necesario indicar los argumentos *x*, *alternative* y *conf.level* que, en este ejercicio, siguen siendo la variable **gasto**, *"two.sided"* y *0.95* respectivamente.

Los argumentos *paired* y *var.equal* siguen sin tener sentido y tampoco los usaremos.

Como se pide explícitamente contrastar si la media puede ser 10€, ahora sí hay que especificar el valor del argumento *mu = 10*.

Suponemos que el contraste es bilateral, así que incorporamos el argumento *alternative = "two.sided"*.

Por último, ejecutamos el test. Como antes, los resultados los almacenaremos en un objeto (lista) llamado **test.gasto2**:

```
test.gasto2 <- t.test(gasto, alternative = "two.sided",  mu = 10, conf.level = 0.9
5)

test.gasto2
```

```
>
>   One Sample t-test
>
> data:  gasto
> t = 0.31819, df = 123, p-value = 0.7509
> alternative hypothesis: true mean is not equal to 10
> 95 percent confidence interval:
>   9.715331 10.393717
> sample estimates:
> mean of x
>  10.05452
```

El ***p-valor*** es **0.7509**, y es mucho mayor que 0,05, por lo que podemos afirmar que el gasto medio en la web sí podría ser de 10 € con un riesgo de 1ª especie del 5%.

Observar que lo único que cambia entre *test.gasto1* y *test.gasto2* es el valor del estadístico *t* y el *p-valor*.

3.3.3 Contraste de hipótesis para la media del gasto mediante el Intervalos de Confianza.

A partir de los resultados del *t.test()* anteriores vamos a determinar si se puede admitir, y por qué, que el gasto medio de un cliente (teórico) podría ser de 10€, con un nivel de confianza del 95%.

Como la media poblacional que se contraste es 10 y dicho valor sí se encuentra dentro del IC para la media que se ha obtenido en el apartado anterior, podemos afirmar, con un 95% de confianza, que el gasto medio en la web SÍ es de 10€.

3.3.4 Contraste o test de hipótesis para la comparación de varianzas

En este caso vamos a comparar el gasto en la primera visita a la web (gasto) y el gasto en la segunda a visita (gasto2).

Se trata pues de una comparación de medias de poblaciones aparadas: se comparan dos características (gasto y gasto2) para un mismo individuo de la población.

Mediante el test vamos a tratar de determinar si puede admitirse, con un riesgo de primera especie de 1% (**alfa**), que el gasto medio es el mismo en la primera y la segunda visita a la web.

El contrsaste o test de hipótesis que se plantea es:

H_0: $m_A = m_B$

H_1: $m_A <> m_B$

Para realizar el contraste también usaremos la función ***t.test***. Como ahora sí vamos a comparar medias, tendremos que especificar las muestras cuyas medias queremos comparar: *gasto* y *gasto2*.

Los argumentos *paired* y *var.equal* ahora sí son importantes.

En primer lugar vamos a **determinar si las varianzas de el gasto en ambas visitas son iguales** con el ***var.test()*** **(Test F)**.

```
## Default S3 method:
var.test(x, y, ratio = 1,
         alternative = c("two.sided", "less", "greater"),
         conf.level = 0.95, ...)

## S3 method for class 'formula'
var.test(formula, data, subset, na.action, ...)
```

Los parámetros necesarios son similares a los del *test.t* pero lo que se comparan son las varianzas, no las medias. Además, en este caso hay que tener en cuenta que el *nivel de confianza* es ahora 99%.

Para la comparación se usa el *cociente o ratio de varianzas* (***ratio***), si se asume que las varianzas son iguales este ratio será 1 (el cociente de las dos varianzas), que es la hipótesis nula por defecto.

Veamos:

```
var.test1 <- var.test(gasto, gasto2, ratio = 1, alternative = "two.sided", conf.le
vel = 0.99)
var.test1
```

```
>
>   F test to compare two variances
>
> data:  gasto and gasto2
> F = 0.8358, num df = 123, denom df = 130, p-value = 0.3158
> alternative hypothesis: true ratio of variances is not equal to 1
> 99 percent confidence interval:
>  0.5277004 1.3271683
> sample estimates:
> ratio of variances
>          0.8358029
```

El ***p-valor*** es **0.3158**, y es mayor que 0,1, por lo que podemos afirmar que la varianza del gasto en la primera visita no difiere del de la segunda, con un riesgo de 1ª especie del 1%.

3.3.5 Contraste o test de hipótesis para la comparación de medias

Siguiendo el ejemplo del apartado anterior, ahora podemos volver sobre la comparación de medias (***`t.test`***) asumiendo que las varianzas de los gastos en la primera y segunda visita a la web son iguales. Por lo tanto, al llamar al contraste, *`t.test`*, indicaremos *`var.equal = TRUE`*.

Por otra parte, en cuanto al parámetro *`paired`*, éste se usa para indicar si las **poblaciones** comparadas son ***independientes*** o ***apareadas***. Veamos qué quiere decir ésto.

En este *data frame*, los datos analizados corresponden a los **clientes**. Esto quiere decir que de un cliente cualquiera se han medido dos características, por una parte se ha medido el gasto total en la primera visita y por otro el gasto total en la segunda visita, esto es, para cada cliente se comparan dos valores. Esto es lo que se llama comparación de **poblaciones apareadas**, a diferencia de lo que ocurriría si tratáramos de comparar el gasto en la primera visita entre los clienes de diferente sexo. En este último caso tendríamos dos subpoblaciones de clientes, que es lo que se llama comparación de **poblaciones independientes**, no tendríamos el mismo cliente en los dos grupos comparados (hombres y mujeres). En el caso de comparación d emuestras independientes, incluso las muestras que utilizamos en la comparación, pueden tener tamaños diferentes, como sería el caso:

```
cat("Gasto medio en la primera visita a la web según sexo")
```

```
## Gasto medio en la primera visita a la web según sexo
```

```
table(datos1$SEXO)
```

```
##
## mujer varon
##    42    89
```

```
tapply(datos1$GASTO, datos1$SEXO, mean, na.rm = T)
```

```
##     mujer     varon
##  9.920359 10.116082
```

Según se trate de muestras independientes o apareadas, el procedimiento estadístico empleado en la resolución del contraste cambia ligeramente con el fin de sacar partido en el caso de que las muestras sean apareadas, teniendo en cuenta que la variabilidad entre el gasto para un ismo cliente (en la primera o en la segunda visita) es menor que entre clientes diferentes (hombres y mujeres.

Por otra parte, tal y como se ha planteado el contraste inicialmente, consideramos que es bilateral, por lo que incorporamos el argumento *alternative = "two.sided"*.

Por último, la hipótesis a contrastar ($m_A = m_B$) representa que las medias del gasto en la primera y segunda visita pueden considerarse iguales, esto es, la hipótesis nula implícita es que la diferencia de las medias del gasto es cero ($m_A - m_B = 0$). Por tanto, el valor del argumento *mu*, que representa la diferencia de medias que se contrasta en la H_0, debe ser *mu = 0*.

Ya tenemos toda la información necesaria para realizar el contraste ejecutando el *test.t* y los resultados los almacenaremos en un objeto (lista) llamado **test.gasto12**:

```
test.gasto12 <- t.test(gasto, gasto2, alternative = "two.sided",  mu = 0, paired =
F, var.equal = T, conf.level = 0.99)

test.gasto12

>
>   Two Sample t-test
>
> data:  gasto and gasto2
> t = 0.063042, df = 253, p-value = 0.9498
> alternative hypothesis: true difference in means is not equal to 0
> 99 percent confidence interval:
>  -0.6352536  0.6668821
> sample estimates:
> mean of x mean of y
>  10.05452  10.03871
```

El ***p-valor*** es **0.9498**, y es mayor que 0,1 (casi 1, de hecho), por lo que podemos afirmar que gasto medio en la primera visita no difiere significativamente de la del gasto en la segunda, con un riesgo de 1ª especie del 1%.

3.3.6 Intervalos de confianza para la comparación de medias

Vamos a obtener un intervalo de confianza (99%) para la diferencia de medias del gasto según la visita.

En realidad este intervalo de confianza es el que se ha obtenido en el apartado anterior.

Sin embargo, es importante resaltar la interpretación de este intervalo de confianza. El **intervalo de confianza al 99% [-0.6352536, 0.6668821]** mostrado no representa un intervalo para los gastos medios, sino para **la diferencia de las medias entre el gasto en la primera visita y el gasto en la segunda**.

Puesto que hemos concluido que no hay diferencias en cuanto al gasto medio entre la primera y segunda visita, implícitamente estamos aceptando que la diferencia entre las medias es 0, como ya hemos comentado, y como consecuencia el intervalo de confianza deberá contener el valor 0, tal y como sucede en este caso.

3.3.7 ¿Contraste de hipótesis o intervalo de confianza?

¿Aporta alguna información adicional al resultado obtenido en el apartado anterior?

Si lo que queremos es simplemente determinar si podemos admitir que hay o no diferencias en cuanto a la media del gasto en la primera y segunda visita a la web, el cálculo del intervalo de confianza no aportaría nada, sería suficiente realizar el test propiamente dicho y evaluar el *p valor*.

Ahora bien, dar un intervalo de confianza es más informativo que aceptar o rechazar una hipótesis concreta para un nivel de confianza determinado.

En este caso, el intervalo **[-0.6352536, 0.6668821]** nos indica que cualquier hipótesis nula que plantáramos de modo que las medias del gasto en la primera y segunda visita difirieran realmente en -0.6352536 (si $m_A < m_B$) o 0.6668821 (si $m_A > m_B$) sería aceptada también.

Por tanto, si $m_A < m_B$ no sería admisible una diferencia entre las medias menor que ****, en caso contrario tampoco sería admisible una diferencia de medias superior a **0.6668821**.

En este sentido sí aportaría información adicional.

3.3.8 Comparación de proporciones

Tomemos ahora la variable *COMPRA1* del *data set* *datos1* que recoge si un cliente compra o no en la primera visita a la web de otra tienda y veamos si hay diferencias entre hombres y mujeres (*SEXO*):

```
tabla.compra1 <- table(datos1$COMPRA1, datos1$SEXO)
tabla.compra1

##
##      mujer varon
##   no    33    85
##   si     9     4

margin.table(tabla.compra1)

## [1] 131
```

Analizando esta variable se sabe que de los 131 clientes, 9 mujeres compran y 4 hombres compran.

Esto quier decir que la proporción de mujeres que compran en la primera visita, a nivel muestral, no es la misma, pero, ¿esta diferencia ha salido por casualidad o se mantendría en la población si pudiéramos disponer de la información de infinitos clientes?. Esto es lo que vamos a comprobar mediante un contraste de proporciones, siempre con una probabilidad de equivocarnos.

Por tanto, la primera pregunta que nos hacemos es: **¿Puede afirmarse con un nivel de confianza del 90% que la proporción de mujeres que compran en la primera visita es la misma que la de los hombres?**

Para responder a esta pregunta podemos construir un **IC para la diferencia de proporciones de clientes que compran o no en la primera visita a la web de otra tienda** o realizar un **contraste para la comparación de proporciones**, que en R podemos llevar a cabo con la función ***prop.test()***.

Recordemos la sintaxis de esta función:

```
prop.test(x, n, p = NULL,
          alternative = c("two.sided", "less", "greater"),
          conf.level = 0.95, correct = TRUE)
```

Donde:

- ***x*** puede especificar dos cosas. O bien simplemente el número de éxitos, o bien, mediante una matriz de dos columnas, el número de éxitos y de fracasos en cada muestra.
- ***n*** especifica el número de datos de la muestra en el caso en que x sea el número de éxitos, y es ignorado en el caso en que
- ***x*** proporcione también el número de fracasos.
- ***p*** es el vector de probabilidades de éxito bajo la hipótesis nula. Debe ser un vector de la misma dimensión que el número de elementos especificado en x.
- ***alternative*** especifica la dirección de la hipótesis alternativa, tomando los valores "two.sided", "greater" o "less".
- ***conf.level*** es el nivel de confianza de los intervalos que se muestran entre los resultados.
- ***correct*** especifica si se usa la corrección por continuidad de Yates. Obsérvese que la opción por defecto es que sí se use esta corrección.

A la función *prop.test()* hay que pasarle como argumentos la información para que estime las proporciones y las compare. Esto se puede hacer de dos formas.

En cualquier de estas dos formas nos va a hacer falta las siguiente información:

```
# Casos analizados
n <- 131

# Conteo de éxitos: compra en la primera visita
# Mujeres
compra.M<-9

# Hombres
compra.H<-4
```

- La primera forma de pasarle información a la función es mediante la construcción de una tabla de contingencia (tabla de frecuencias cruzadas) con el número de *éxitos* y de *fracasos* en cada muestra. Es importante resaltar que en este caso el "éxito" representa que el cliente compra en la primera visita, en algunas situaciones puede llevar a confusión, por ejemplo si nuestro "éxito" consistiera en no comprar.

Construyamos la matriz:

```
# Alternativa 1: creando previamente tabla de contingencia
# Conteo de éxitos: número de hombres y mujeres que compran en la primera visita
compra.M

> [1] 9

compra.H

> [1] 4
```

```
# Conteo de fracasos: número de hombres y mujeres que NO compran en la primera vis
ita
No.compra.M<-n-compra.M
No.compra.H<-n-compra.H

# Construcción de la tabla de contingencia
tabla<-matrix(c(compra.M,compra.H, No.compra.M, No.compra.H), 2, 2)
tabla

>      [,1] [,2]
> [1,]    9  122
> [2,]    4  127
```

El nivel de confianza es del 90%, esto implica que el nivel de significación *alfa* es 0,1. Suponemos también que el contraste es bilateral.

Realizamos el contraste:

```
prop.test(tabla, alternative = "two.sided", conf.level = 0.9)

>
>   2-sample test for equality of proportions with continuity
>   correction
>
> data:  tabla
> X-squared = 1.295, df = 1, p-value = 0.2551
> alternative hypothesis: two.sided
> 90 percent confidence interval:
>  -0.01342923  0.08976511
> sample estimates:
>     prop 1     prop 2
> 0.06870229 0.03053435
```

- La segunda forma de pasarle información a la función consiste en construir dos vectores, uno con el número de *éxitos* (fallos de estación) y otro con los *casos analizados* en cada muestra.

Construyamos los vectores:

```
# Alternativa 2: creando 2 vectores
# nº de hombres o mujeres que compran en la primera visita
compran <- c(compra.M, compra.H)

# nº de clientes analizados
analizados <- c(c(n, n))

# Vectores
compran

> [1] 9 4

analizados

> [1] 131 131
```

Realizamos el contraste:

```
test.HM <- prop.test(compran, analizados, alternative = "two.sided", conf.level =
0.9)
test.HM

>
>   2-sample test for equality of proportions with continuity
>   correction
>
> data:  compran out of analizados
> X-squared = 1.295, df = 1, p-value = 0.2551
> alternative hypothesis: two.sided
> 90 percent confidence interval:
>  -0.01342923  0.08976511
> sample estimates:
>     prop 1     prop 2
> 0.06870229 0.03053435
```

En ambos casos el resultado de *prop.test()* ha sido el mismo.

Como ***p-valor*** es **0.2551237**, y es mayor que 0,1, podemos afirmar que la proporción de mujeres que compran en la primera visita no difiere significativamente de la de los hombres, con un riesgo de 1ª especie del 10%.

Se llega a la misma conclusión observando que el 0 (una diferencia de 0 entre ambas proporciones) está en el intervalo de confianza para la diferencia de proporciones obtenido.

Este intervalo ha sido **[-0.0134292, 0.0897651]**

Como se puede observar, en este contraste no hemos usado la información muestral, la información sobre el éxito o fracaso se nos daba a priori.

Sin embargo, se nos podía haber planteado la pregunta de **si la proporción de los que compran en la primera visita es la misma en ambos sexos según los datos disponibles en la muestra**.

Lo que cambiaría en este caso es el modo de obtener los *casos analizados* (o los *éxitos*). Estos no se calcularían sobre 131, sino sobre el número total de clientes de cada sexo.

Veamos:

```
# El nº de clientes que compran es el mismo
compran

> [1] 9 4

# nº de mujeres que compran
nM <- sum(datos1$SEXO == "mujer")

# nº de hombres que compran
nH <- sum(datos1$SEXO == "varon")

# nº de estaciones analizadas de cada tipo
analizados <- c(c(nM, nH))

# Comparación de proporciones
prop.test(compran, analizados, alternative = "two.sided", conf.level = 0.9)
```

```
> Warning in prop.test(compran, analizados, alternative = "two.sided",
> conf.level = 0.9): Chi-squared approximation may be incorrect

>
>   2-sample test for equality of proportions with continuity
>   correction
>
> data:  compran out of analizados
> X-squared = 7.3576, df = 1, p-value = 0.006678
> alternative hypothesis: two.sided
> 90 percent confidence interval:
>  0.04158896 0.29709483
> sample estimates:
>     prop 1     prop 2
> 0.21428571 0.04494382
```

3.3.9 Contraste de hipótesis e Intervalo de confianza para una proporción

Si seguimos con el ejemplo de los clientes que compran o no en la primera visita a la web:

```
frec.compran <- summary(datos1$COMPRA1)
frec.compran

##  no  si
## 118  13

N.compran <- length(na.omit(datos1$COMPRA1))
N.compran

## [1] 131
```

Analizando esta variable se sabe que de los 131 clientes, 13 compran y 118 no. Esto quier decir que la proporción de clientes que compran en la primera visita ha sido de 13/131, mientras la proporción de los que no lo han hecho es de 118/131.

Es decir, a nivel muestral hay una proporción (**P**) de clientes que compran en la primera visita de **0,099**.

Pero, ¿esa proporción es fruto del azar o se mantendría en la población? ¿la proporción de clientes que compran en la primera visita podría ser del 5% en la población?.

Lo que nos planteamos es un contraste de hipótesis:

H_0: P = 0.05

H_1: P <> 0.05

Para ello también podemos usar la función *prop.test()*:

```
test.P <- prop.test(frec.compran["si"], N.compran, p = 0.05, alternative = "two.si
ded", conf.level = 0.9, correct = F)

test.P

##
##  1-sample proportions test without continuity correction
```

```
##
## data:  frec.compran["si"] out of N.compran, null probability 0.05
## X-squared = 6.6858, df = 1, p-value = 0.009718
## alternative hypothesis: true p is not equal to 0.05
## 90 percent confidence interval:
##  0.06405002 0.15064223
## sample estimates:
##          p
## 0.09923664
```

Como ***p-valor*** es **0.0097183**, y es menor que 0,1, podemos afirmar que la proporción de los que compran no puede ser 0,05, con un riesgo de 1ª especie del 10%.

Se llega a la misma conclusión observando que el valor de P planteado en la hipótesis nula (0.05) está en el intervalo de confianza para P obtenido **[0.06405, 0.1506422]**.

4 Muestreo

```
> ## Establecer directorio de trabajo (no haría falta si se carga el proyecto)
>
> ## Cargar datos
> load("datos.Rdata")
```

4.1 Obtención de muestras

4.1.1 Muestreo aleatorio simple

Función **sample**:

sample(x, size, replace = FALSE, prob = NULL)

- **Muestrear índices o código de individuo** (números de fila):

```
# Ejemplo 1
x <- 1:100
set.seed(111)
sample(x, size = 10, replace = F)
##  [1] 60 72 37 50 98 40  2 97 95  9
# Ejemplo 2
set.seed(222)
sample(100, size = 10, replace = T)
##  [1] 94  7 50  1 92 96 36 42 58 15
```

- **Muestrear el propio valor de la variable**:

```
# Ejemplo 3
# Muestra de 10 elementos de una población normal (con media 4 y desviación típica
1) de tamaño 500
x<-rnorm(500, 4, 1)
sample(x, 10, replace=T)
##  [1] 4.334473 3.748581 4.810628 2.436822 4.183274 4.675893 4.430474
##  [8] 4.924990 3.932792 5.569572
# Ejemplo 4
load("datos.RData")
sample(datos1$PROVINCIA, size = 4)
## [1] Valencia Valencia Valencia Teruel
## Levels:  Alicante Castellon Teruel Valencia
```

- **Permutaciones**:

```
for(i in 1:4) {
  cat("Permutación", i, "=", sample(5, size = 5), "\n")
  }
## Permutación 1 = 5 1 4 2 3
## Permutación 2 = 1 3 4 5 2
## Permutación 3 = 1 2 5 3 4
## Permutación 4 = 1 2 3 5 4
```

- **Ensayos de Bernoulli**:

```
sample(c(0,1), 100, replace = TRUE)
##   [1] 0 0 1 1 0 0 0 1 0 1 1 0 0 1 0 0 1 1 1 1 0 0 0 0 1 1 0 1 1 0 0 0 1 1 0
##  [36] 0 1 1 1 0 1 1 1 0 0 1 0 1 1 1 1 1 0 1 0 1 1 1 0 0 0 1 0 1 0 1 0 1 1 1
##  [71] 1 0 1 0 0 0 1 0 1 1 0 0 1 1 1 1 0 0 0 1 0 0 1 0 1 0 1 0 0 0
```

Observar que si no hay reemplazamiento, cuando la población se “acaba” ya no se puede muestrear:

```
sample(c(0,1), 100, replace = FALSE)

Error in sample.int(length(x), size, replace, prob) : cannot take a sample larger than the
population when 'replace = FALSE'
```

4.1.2

4.1.3 Muestreo estratificado

Datos sobre el tamaño de la población, de la muestra y los estratos:

```
N <- 600 #Población
n <- 20 # Tamaño muestra
k <- 4 # Estratos
N1 <- 200 # Tamaño deñ estrato 1
N2 <- 150 # Tamaño deñ estrato 2
N3 <- 150 # Tamaño deñ estrato 3
N4 <- 100 # Tamaño deñ estrato 4
```

Cálculo de la proporción de inmdividuos en cada estrato:

```
p <- n/N  # Proporción de la muestra con respecto a la población
p
## [1] 0.03333333
n1 <- round(p*N1)   #Número de individuos del estrato 1
n2 <- round(p*N2) #Número de individuos del estrato 2
n3 <- round(p*N3) #Número de individuos del estrato 3
n4 <- round(p*N4) #Número de individuos del estrato 4

n1
## [1] 7
n2
## [1] 5
n3
## [1] 5
n4
## [1] 3
```

Ahora podemos usar los ni en la función `sample()` y obtener los individuos de cada estrato.

4.2 Cálculo del tamaño de la muestra para la estimación de un parámetro

```
> #### PARA LA ESTIMACIÓN DE LA MEDIA EN UNA POBLACIÓN INFINITA ######
> ##################################################################
> ### Estimación del del gasto mediol por cliente en una tienda on line (X)
>
> ## Estimación de la varianza muestral a partir de la
> ## variable "GASTO". Data set "datos1". Archivo "datos.RData"
> S2.gasto<-var(gasto, na.rm=T)
> S2.gasto
[1] 3.570017
>
> ## Elección del Nivel de significación y de la precisión
> # Alfa
> alfa<-0.05
```

```
> z<-qnorm(alfa/2, lower.tail=F)
> alfa
[1] 0.05
> z
[1] 1.959964
>
> # Error
> e<-0.05 # Error de e*100%
>
> ## Tamaño de la muestra redondeando al entero inmediatamente superior
> n<-ceiling(z^2*S2.gasto/e^2)
>
> alfa
[1] 0.05
> e
[1] 0.05
> n
[1] 5486
```

```
> #### PARA LA ESTIMACIÓN DE UNA PROPORCIÓN EN UNA POBLACIÓN INFINITA ######
> ##########################################################################
> ### Sea X la variable definida como "nº de clientes que compran en la primera vi
sita a la web" de una determinada tienda on line.
> ### Vamos a estimar n para la proporción (P) de clientes que compran en la prime
ra visita a la web de dicha tienda
>
> ## Estimación de la proporción y varianza muestral a partir de la
> ### Variable "COMPRA1". Data set "datos1". Archivo "datos.RData"
> compra1<-datos1$COMPRA1
> frec.compra1<-table(compra1)
> prop.compra1<-prop.table(frec.compra1)
> frec.compra1
compra1
 no  si
118  13
> p.compra1<-prop.compra1["si"]  # o bien p.compra1<-prop.compra1[2]
> p.compra1
        si
0.09923664
>
> S2.compra1<-p.compra1*(1-p.compra1)
> S2.compra1
        si
0.08938873
>
> ## Elección del Nivel de significación y de la precisión
> # Alfa
> alfa<-0.05
> z<-qnorm(alfa/2, lower.tail=F)
> alfa
[1] 0.05
> z
[1] 1.959964
>
> # Error
> e<-0.05 # Error de e*100%
>
> ## Tamaño de la muestra redondeando al entero inmediatamente superior
> n<-ceiling(z^2*S2.gasto/e^2)
>
> alfa
[1] 0.05
> e
[1] 0.05
> n
[1] 5486
```

4.3 Influencia del tamaño de la muestra

```
> ### INREVALOS DE CONFIANZA PARA LA COMPARACIÓN DE MEDIAS
> ### INFLUENCIA DEL TAMAÑO DE LA MUESTRA
> ######################################################################
```

1. **Obtener 2 muestras aleatorias de tamaño 10 de una distribución normal con sigma=2:**
   - media 10
   - media 11

Obtener el intervalo de confianza para la comparación de las 2 muestras aleatorias independientes (usar `t.test`).

```
set.seed(1010)
muestra1.10 <- rnorm(10, 10, 2)
muestra2.10 <- rnorm(10, 11, 2)
test.10 <- t.test(muestra1.10, muestra2.10, alternative = "two.sided", conf.level
= 0.95)

test.10$conf.int
## [1] -1.826977  2.087928
## attr(,"conf.level")
## [1] 0.95
cat("amplitud IC (n=10)", test.10$conf.int[2]-test.10$conf.int[1])
## amplitud IC (n=10) 3.914906
```

2. **Obtener ahora otras 2 muestras aleatorias de tamaño 1000 de una distribución normal con sigma=2:**
   - media 10
   - media 11 Obtener el intervalo de confianza para la comparación de las 2 muestras aleatorias independientes.

```
set.seed(1010)
muestra1.1000 <- rnorm(1000, 10, 2)
muestra2.1000 <- rnorm(1000, 11, 2)

test.1000 <- t.test(muestra1.1000, muestra2.1000, alternative = "two.sided",
conf.level = 0.95)

test.1000$conf.int
## [1] -1.2059580 -0.8563001
## attr(,"conf.level")
## [1] 0.95
cat("amplitud IC (n=1000)", test.1000$conf.int[2]-test.1000$conf.int[1])
## amplitud IC (n=1000) 0.3496579
```

¿Qué se observa en cuanto a la amplitud de los intervalos de confianza?
Conforme aumenta el tamaño de la muestra la amplitud de los IC disminuye (permaneciendo constantes los demás elementos para su cálculo)

4.4 Tamaño del efecto

```
> ### d de Cohen PARA LA COMPARACIÓN DE MEDIAS
> ##############################################
> ### Ejemplo de comparación del gasto total por cliente en una tienda on line (X) entre hombres y mujeres
> ### Variables "SEXO" y "GASTO". Data set "datos1". Archivo "datos.RData"
> ## Obtención de las muestras
> gasto.mujer<-na.omit(datos1[which(datos1$SEXO=="mujer"), "GASTO"])
> gasto.varon<-na.omit(datos1[which(datos1$SEXO=="varon"), "GASTO"])
>
> ## Obtención de los parámetros muestrales
> media.mujer<-mean(gasto.mujer)
> media.varon<-mean(gasto.varon)
> S2.mujer<-var(gasto.mujer)
> S2.varon<-var(gasto.varon)
> n.mujer<-length(gasto.mujer)
> n.varon<-length(gasto.varon)
>
> media.mujer
[1] 9.928857
> media.varon
[1] 10.09705
> S2.mujer
[1] 3.069534
> S2.varon
[1] 3.840814
> n.mujer
[1] 42
> n.varon
[1] 87
>
> ## ¿Podemos suponer que las varianzas son iguales?
> ## Test de F para comparación de varianzas
> var.test(gasto.mujer, gasto.varon, ratio=1, alternative="two.sided", conf.level=0.95)

        F test to compare two variances

data:  gasto.mujer and gasto.varon
F = 0.7992, num df = 41, denom df = 86, p-value = 0.4305
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.4814966 1.3953079
sample estimates:
ratio of variances
         0.7991883

>
> ## d de Cohen
> Sp<-sqrt((S2.mujer*(n.mujer-1)+S2.mujer*(n.mujer-1))/(n.mujer+n.varon-2))
> d<-(media.mujer-media.varon)/Sp
> d ## d de Cohen
[1] -0.1194692
>
> ## Test t bilateral asumiendo igualdad de varianzas. mujer = hombre
> t.test(gasto.mujer, gasto.varon, alternative="two.sided", mu=0, var.equal=T, conf.level=0.95)

        Two Sample t-test

data:  gasto.mujer and gasto.varon
t = -0.4723, df = 127, p-value = 0.6375
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.8728396  0.5364620
sample estimates:
mean of x mean of y
 9.928857 10.097046
```

4.5 Potencia estadística

```
>
> ### Instalar pwr si es la primera vez que se usa
>
> ### Cargar paquete pwr
> library(pwr)
>
> ### Obtener n para un test sobre la media de una población
> n=NULL
> efecto<-0.6
> alfa<-0.05
> potencia<-0.8
> tipo<-"one.sample"
> res<-pwr.t.test(n, efecto, alfa, potencia, tipo)
> ceiling(res$n)
[1] 24
>
> ### Obtener el tamaño del efecto para un n dado en un test sobre la media de una
población
> n=length(datos1$GASTO)
> efecto<-NULL
> alfa<-0.05
> potencia<-0.8
> tipo<-"one.sample"
> pwr.t.test(n, efecto, alfa, potencia, tipo)

     One-sample t test power calculation

              n = 131
              d = 0.2466023
      sig.level = 0.05
          power = 0.8
    alternative = two.sided

>
> ### Obtener el tamaño del efecto para un n dado en un test sobre la media de una
población
> n=length(datos1$GASTO)
> efecto<-NULL
> alfa<-0.05
> potencia<-0.9
> tipo<-"one.sample"
> pwr.t.test(n, efecto, alfa, potencia, tipo)

     One-sample t test power calculation

              n = 131
              d = 0.285312
      sig.level = 0.05
          power = 0.9
    alternative = two.sided
```

5 Preproceso de datos

Script: Curso Big Data-1-5Preproceso.R

```
## Establecer directorio de trabajo (no haría falta si se carga el proyecto)

#Antes de nada, nos situamos en el directorio de trabajo
setwd("C:/Users/Elena Vázquez/Dropbox/Doc-bigdata EVB/Datos/Curso Big Data")
load("datos.RData")
```

5.1 Limpieza

```
> load("datos.RData")
> ###############################################################################
> ##### DETECCIÓN Y TRATAMIENTO DE VALORES ESPECIALES ############################
> ###############################################################################

> ## is.finite() determina si los valores son “normales”

> valores<-c(1, Inf, NA, NaN, NULL)

> is.finite(valores)
[1]  TRUE FALSE FALSE FALSE


> ###################################################################
> ##### TRATAMIENTO DE VALORES PERDIDOS ##############################
> ###################################################################

> ##### DETECCIÓN DE VALORES PERDIDOS ################################
> ###################################################################

> ## Temperatura recogidas por 3 estaciones meteorológicas
> ## Data set "temper". Archivo "datos.RData"
> attach(temper)

> ## El data frame tiene algundato anomalo?
> anyNA(temper)
[1] TRUE

> ## Donde están los datos anómalos?

> summary(temper)

 Estacion   Temperatura
 A   :52   Min.   :16.11
 B   :52   1st Qu.:18.55
 C   :51   Median :19.79
 NA's: 1   Mean   :19.95
           3rd Qu.:21.21
           Max.   :32.00
           NA's   :2

> detach(temper)
> ##### ELIMINACIÓN U OMISIÓN DE VALORES PERDIDOS ###########################
> ###################################################################
```

```
> attach(datos1)
> ## na.rm()
> ## Un ejemplo
> edad<-c(23, 16, NA)
> edad
[1] 23 16 NA
> mean(edad)
[1] NA
> mean(edad, na.rm=T)
[1] 19.5

> ## Otro ejemplo
> # Insertamos algunos faltantes
> datos1[c(1,4,5,13,15), "ESTATURA"]<-NA
> datos1[c(2,7), "PESO"]<-NA
> datos1[c(2,4,5,7,11), "GASTO"]<-NA
> View(datos1)

> # Calculo de la correlacion con eliminación de perdidos por casos
> cor(PESO, ESTATURA, use="complete.obs")
[1] 0.7333834

> # Cuantos casos competos hay en el frame?
> com<-complete.cases(datos1)
> com
  [1] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE
TRUE  TRUE  TRUE  TRUE
 [21]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
TRUE  TRUE  TRUE  TRUE
 [41]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
TRUE  TRUE  TRUE  TRUE
 [61]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
TRUE  TRUE  TRUE  TRUE
 [81]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
FALSE  TRUE  TRUE  TRUE
[101]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
TRUE  TRUE  TRUE  TRUE
[121]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE

> table(com)
com
FALSE  TRUE
   11   120

> ## Eliminar todos los casos con dato1 perdidos
> datos1<-na.omit(datos1)

> ## Eliminar todos los casos perdidos de la variable MES
> datos1<-na.omit(MES)
> detach(datos1)

> ### Ejemplo de funciones de tratamiento de valores perdidos
> (g <- as.data.frame(matrix(c(1:5, NA), ncol = 2)))
  V1 V2
1  1  4
2  2  5
3  3 NA


> g
  V1 V2
1  1  4
```

```
2  2  5
3  3 NA

> na.omit(g)
  V1 V2
1  1  4
2  2  5
> na.exclude(g)

  V1 V2
1  1  4
2  2  5

> na.fail(g)
Error in na.fail.default(g) : missing values in object

> na.pass(g)
  V1 V2
1  1  4
2  2  5
3  3 NA


> ##### IMPUTACIÓN de VALORES PERDIDOS ########################
> ############################################################

> ## Cargar paquete Hmisc
> # library(Hmisc)

> ## O bien cargar paquete e1071 para data.frames y matrices
> library(e1071)

> #### IMPUTACIÓN NUMÉRICA ##########################

> ### Ejemplo
> ## Crear vector
> x<- matrix(1:10, ncol=2)
> x
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10

> ## Generar datos perdidos
> x[c(1,3,7)] <- NA
> x
     [,1] [,2]
[1,]   NA    6
[2,]    2   NA
[3,]   NA    8
[4,]    4    9
[5,]    5   10

> ## Medias y Medianas

> colMeans(x, na.rm = T)
[1] 3.666667 8.250000
> sapply(list(x[,1], x[,2]), median, na.rm= T)
[1] 4.0 8.5
```

```
> ## Imputar la media
> x<- impute(x, "mean")
> x
         [,1]  [,2]
[1,] 3.666667  6.00
[2,] 2.000000  8.25
[3,] 3.666667  8.00
[4,] 4.000000  9.00
[5,] 5.000000 10.00

> # Nuevas Medias
> colMeans(x, na.rm = T)
[1] 3.666667 8.250000

> ## Imputar la mediana
> ## Volver a generar los datos perdidos
> x[c(1,3,7)] <- NA
> x
     [,1] [,2]
[1,]   NA    6
[2,]    2   NA
[3,]   NA    8
[4,]    4    9
[5,]    5   10
> x<- impute(x, "median")
> x
     [,1] [,2]
[1,]    4  6.0
[2,]    2  8.5
[3,]    4  8.0
[4,]    4  9.0
[5,]    5 10.0

> # Nuevas Medias
> colMeans(x, na.rm = T)
[1] 3.8 8.3


> ###############################################################################
> ##### DETECCIÓN Y TRATAMIENTO DE VALORES ANÓMALOS ##############################
> ###############################################################################
> attach(temper)

> ## Graficamente: Donde están los datos anómalos?

> #Diagrama de caja (deafult)
> caja.temper<-boxplot(temper$Temperatura, xlab="Todas las estaciones", ylab="Temp
eratura ºC", main="¿hay datos anómalos?", col="lightblue")

> # Sin cerar el gráfico, podemos señalar en el boxplot los datos anómalos e ident
ificarlos
> identify(rep(1,length(temper$Temperatura)), temper$Temperatura, temper$Estacion)
[1] 2



> # O bien
> boxplot(temper$Temperatura, ylab="Temperatura ºC")
> identify(rep(1,length(temper$Temperatura)), temper$Temperatura, rownames(temper)
)
[1] 2
```

```
> # O bien
> boxplot(Temperatura~Estacion, ylab="Temperatura ºC", main="¿Hay datos anómalos?"
, col="yellow")
> # Para cambiar el coeficiente que determina el dato anómalo e identificar numéri
camente los datos anómalos y otros valores del boxplot

> boxplot.stats(temper$Temperatura, coef=2)
$stats
[1] 16.11 18.54 19.79 21.22 24.63

$n
[1] 154

$conf
[1] 19.44878 20.13122

$out
[1] 32

> ## Papel Probabilístico normal
> qqnorm(Temperatura)
> qqline(Temperatura)
> detach(temper)
```

5.2 Integración

```
> ##############################################################################
> ##### FILTRADO ###############################################################
> ##############################################################################
>
> attach(datos1)


> #### Filtrado de variables
> ## Creamos un nuevo data frame que contenga solo las variables EDAD, ESTATURA y
PESO
> nuevos1<-subset(datos1, select = c("EDAD", "ESTATURA", "PESO"))
> head(nuevos1)
  EDAD ESTATURA PESO
1   20       NA   54
2   20      164   NA
3   19      185   70
4   19       NA   63
5   20       NA   63
6   23      159   54


> #### Filtrado de casos
> ## Creamos un nuevo data frame que contenga los valores de todas las variables
> ## de las mujeres mayores de 20 años
> nuevos2<-subset(datos1, SEXO=="mujer" & EDAD > 20)
> head(nuevos2, 10)
    SEXO EDAD MES ESTATURA PESO PROVINCIA X ACCESOS     TIPO  GASTO COMPRA1 GASTO2 COMPRA2
6  mujer   23  10      159   54  Alicante 5      65     yoga 10.902      no 12.780      si
7  mujer   21   5      160   NA  Alicante 7      45     yoga     NA      no  9.619      si
8  mujer   21   4      155   48  Alicante 3      30  fitness 10.097      no  6.895      si
10 mujer   22   4      172   59  Alicante 7      45     yoga  8.964      no 10.015      si
12 mujer   21   9      160   49  Alicante 4      30           5.856      no 11.422      si
14 mujer   21   9      163   56  Alicante 3      45  fitness  9.617      no 11.842      si
30 mujer   22   6      162   48 Castellon 3      75 aparatos 11.403      no  9.714      si
```

```
31 mujer   21   6      170   63 Castellon 7      65 aparatos 10.133      si 10.573      si
34 mujer   21  11      163   54 Castellon 7      10 aparatos  9.154      no  5.362      no
37 mujer   21   4      160   55 Castellon 4      30     yoga 11.639      no  7.436      no
```

```
> #### Filtrado de variables y casos
> ## Creamos un nuevo data frame que contenga el SEXO, ESTATURA y PESO
> ## de las mujeres mayores de 20 años
> nuevos3<-subset(datos1, SEXO=="mujer" & EDAD > 20, select = c("EDAD", "ESTATURA"
, "PESO"))

> head(nuevos3, 10)

   EDAD ESTATURA PESO
6    23      159   54
7    21      160   NA
8    21      155   48
10   22      172   59
12   21      160   49
14   21      163   56
30   22      162   48
31   21      170   63
34   21      163   54
37   21      160   55
```

```
> ##############################################################################
> ##### ORDENAR ################################################################
> ##############################################################################

>
> ## Ordenar todos los casos de un data frame en función de una variable  (p.e. ED
AD) orden creciente
> orden1<-datos1[order(MES, decreasing = F) ,]

> head(orden1)
    SEXO EDAD MES ESTATURA PESO PROVINCIA X ACCESOS     TIPO  GASTO COMPRA1 GASTO2 COMPRA2
20 varon   21   1      180   72  Alicante 3      15 aparatos 12.289      no  8.427      si
21 varon   20   1      171   75  Alicante 8      15     yoga 13.391      no 11.128      si
33 mujer   20   1      161   48 Castellon 5      60  pilates 10.464      si 11.458      no
40 mujer   21   1      161   46 Castellon 3      30  fitness  6.184      no  9.756      no
42 varon   20   1      183   76 Castellon 5      15  pilates  8.346      no  5.672      no
45 varon   19   1      179   67 Castellon 7      30  pilates  5.544      no  7.241      no
```

```
> ## Ordenar todos los casos de un data frame en función de una variable (p.e. EDA
D) orden decreciente
> orden2<-datos1[order(MES, decreasing = T) ,]
```

```
> head(orden2)
    SEXO EDAD MES ESTATURA PESO PROVINCIA X ACCESOS     TIPO  GASTO COMPRA1 GASTO2 COMPRA2
12 mujer   21   9      160   49  Alicante 4      30           5.856      no 11.422      si
14 mujer   21   9      163   56  Alicante 3      45  fitness  9.617      no 11.842      si
28 varon   21   9      180   90  Alicante 7      15  fitness 10.654      no  9.279      si
46 varon   20   9      171   71 Castellon 3      13  fitness  9.490      no 11.628      no
59 varon   20   9      178   69 Castellon 7      45          11.755      no 12.728      no
68 mujer   23   9      165   55    Teruel 2      35 aparatos 11.068      no  7.247      no
```

```
> ## Ordenar todos los casos de un data frame en función de dos variable (p.e. MES
y EDAD) orden creciente
> orden3<-datos1[order(MES, EDAD) ,]
> head(orden3)
    SEXO EDAD MES ESTATURA PESO PROVINCIA X ACCESOS     TIPO  GASTO COMPRA1 GASTO2 COMPRA2
45 varon   19   1      179   67 Castellon 7      30  pilates  5.544      no  7.241      no
21 varon   20   1      171   75  Alicante 8      15     yoga 13.391      no 11.128      si
33 mujer   20   1      161   48 Castellon 5      60  pilates 10.464      si 11.458      no
42 varon   20   1      183   76 Castellon 5      15  pilates  8.346      no  5.672      no
61 mujer   20   1      168   61    Teruel 7       7 aparatos  7.942      no  8.970      no
20 varon   21   1      180   72  Alicante 3      15 aparatos 12.289      no  8.427      si


> #### Ordenar todos los casos de un data frame en función de una variable  (p.e.
EDAD) orden creciente,
> ## dejando al final los valores perdidos
> orden4<-datos1[order(MES, EDAD, na.last = T) ,]
> tail(orden4)
     SEXO EDAD  MES ESTATURA PESO PROVINCIA X ACCESOS     TIPO  GASTO COMPRA1 GASTO2 COMPRA2
130 varon   22    9      174   80  Valencia 7      20     yoga 10.104      no  9.474      no
68  mujer   23    9      165   55    Teruel 2      35 aparatos 11.068      no  7.247      no
128 varon   23    9      176   74  Valencia 6      10     yoga 12.057      no  7.354      no
3   varon   19 <NA>      185   70           5      30  pilates  8.541      no 10.426      no
1   mujer   20 <NA>       NA   54           3      30  pilates  6.124      si  7.385      no
2   mujer   20 <NA>      164   NA           4      10  pilates     NA      si 10.980      no
>
> detach(datos1)


> ##############################################################################
> ##### AÑADIR VARIABLES Y CASOS ###############################################
> ##############################################################################

> #### Añadir variables (vertical)
> ## Simulamos una nueva variable que contiene la precipitación media en l/m2
> x<-rnorm(length(Temperatura), 300, 50)
>
> ## Añadimos la nueva variable al data frame
> temper$Precipitacion<-x
>
> head(temper)

Estacion Temperatura Precipitacion
1        A          NA      287.7228
2        A       32.00      332.1161
3        B          NA      277.7155
4        B       22.73      249.7067
5     <NA>       23.42      384.5467
6        C       19.56      207.2325



> ## O bien
> temper1<-cbind(temper, x)
>
> head(temper1)

Estacion Temperatura Precipitacion        x
1        A          NA      287.7228 287.7228
2        A       32.00      332.1161 332.1161
3        B          NA      277.7155 277.7155
```

```
4         B       22.73      249.7067 249.7067
5      <NA>       23.42      384.5467 384.5467
6         C       19.56      207.2325 207.2325
```

```
> #### Añadir casos (horizontal)
> ## Creamos dos grupos de datos a partir de datos1
> muestra1<-datos1[1:5,]
> muestra2<-datos1[6:10,]
>
> ## Combinamos los casos de ambos data frames
> muestra<-rbind(muestra1, muestra2)
>
>
> ################## fin ###################################
```

5.3 Transformaciones

5.4

```
#############################################################################
##### NORMALIZAR o ESTANDARIZAR o TIPIFICAR ##################################
#############################################################################
# Estandarizar las variables ESTATURA y PESO y añadirlas al dataset datos1
datos1$est.norm <- scale(ESTATURA, center = T, scale = T)
datos1$pes.norm <- scale(PESO, center = T, scale = T)
summary(datos1[,c("ESTATURA", "PESO", "est.norm", "pes.norm")])

#############################################################################
##### TRANSFORMACIÓN LOGARÍTMICA #############################################
#############################################################################
# Verificar la asimetría positiva de la variable ACCESOS
boxplot(ACCESOS, horizontal = T, main)

# Transformación logarítmica
y<-log(ACCESOS, exp(1))

# Verificar la nueva forma d ela distribución
boxplot(y, horizontal = T)

#############################################################################
##### DISCRETIZACIÓN VAR CONTINUA ############################################
```

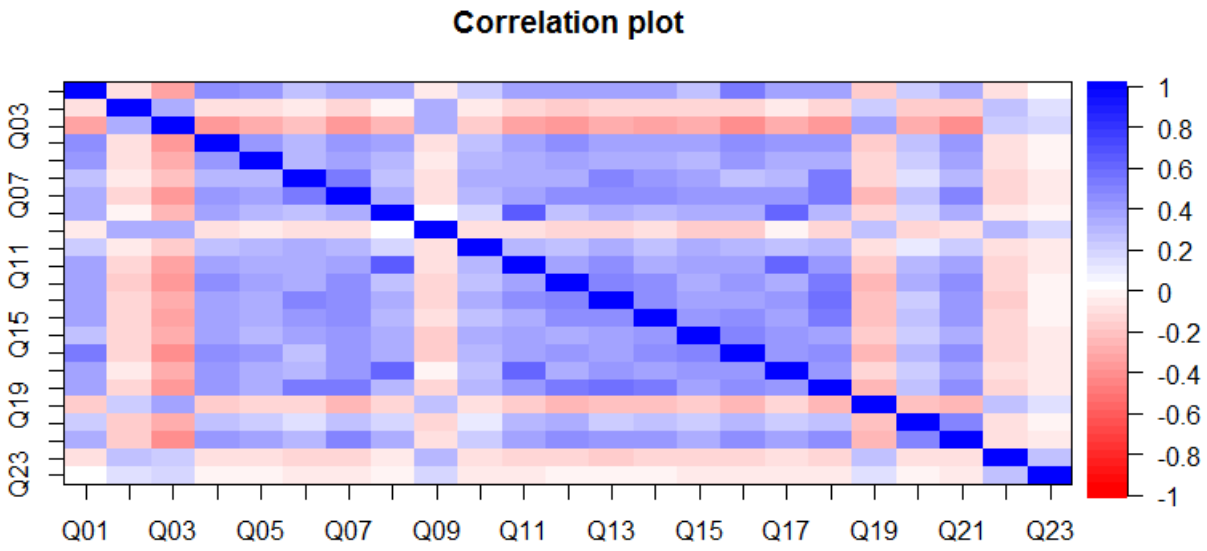
```
################################################################################
## Tabla d efrecuencias de ESTATURA
table(ESTATURA)
pie(table(ESTATURA))


## Discretizar con 5 tramos de estatura
est.disc <- cut(ESTATURA, breaks = 5)

## Tabular la nueva variable
table(est.disc)
pie(table(est.disc))
```

5.5 Reducción: Análisis de Componentes Principales

```
> #### PAQUETES
> ## Instalar paquetes
> # install.packages("corpcor")
> # install.packages("psych")
> # install.packages("GPArotation")
>
> ## Cargar paquetes
> library(corpcor)
> library(GPArotation)
> library(psych)
>
 > #### DATOS
 > ## Leer archivo de datos, visualizar y verificar contenido
 > load("RAQ.RData")
 > objects()
  [1] "alfa"           "bartlett.res"   "kmo"            "mcp1"
  [5] "mcp2"           "mcp2.RVar"      "mcp3"           "mcp3.RVar"
  [9] "ncomp"          "p.mcp2.Rvar"    "prueba"         "raq.kmo"
 [13] "raqDatos"       "raqKMO"         "raqR"           "resid.2"
 [17] "resid.2.Rvar"   "resid.3"        "resid.3.Rvar"   "residual.stats"
 [21] "residuos"
 > str(raqDatos)
 'data.frame':  2571 obs. of  23 variables:
  $ Q01: int  4 5 4 3 4 4 4 4 3 4 ...
  $ Q02: int  5 5 3 5 5 5 3 4 3 2 ...
  $ Q03: int  2 2 4 5 3 3 3 3 5 2 ...
  $ Q04: int  4 3 4 2 4 4 4 4 2 3 ...
  $ Q05: int  4 4 2 3 4 2 4 4 1 4 ...
  $ Q06: int  4 4 5 3 3 2 4 4 3 5 ...
  $ Q07: int  3 4 4 2 3 2 4 4 1 4 ...
  $ Q08: int  5 4 4 4 4 4 4 4 1 4 ...
  $ Q09: int  5 1 4 4 2 2 3 2 3 3 ...
  $ Q10: int  4 4 4 2 4 3 4 4 3 4 ...
  $ Q11: int  5 4 3 4 4 4 4 4 1 4 ...
  $ Q12: int  4 3 3 4 3 2 4 3 1 3 ...
  $ Q13: int  4 5 4 4 3 3 4 4 1 4 ...
  $ Q14: int  4 3 2 3 4 3 4 4 1 5 ...
  $ Q15: int  4 2 4 3 4 1 4 3 1 4 ...
```

```
 $ Q16: int  3 3 3 3 4 4 4 4 1 3 ...
 $ Q17: int  5 4 4 4 4 3 4 4 1 4 ...
 $ Q18: int  4 4 3 2 3 1 4 4 1 4 ...
 $ Q19: int  3 3 5 4 3 5 3 2 4 3 ...
 $ Q20: int  4 2 2 2 2 1 4 3 1 3 ...
 $ Q21: int  4 2 3 2 4 3 4 4 1 4 ...
 $ Q22: int  4 2 4 2 2 5 2 2 3 2 ...
 $ Q23: int  1 4 4 3 2 2 2 2 3 2 ...
> View(raqDatos)
>
> ## Datos por defecto
> attach(raqDatos)
The following objects are masked from raqDatos (pos = 4):

    Q01, Q02, Q03, Q04, Q05, Q06, Q07, Q08, Q09, Q10, Q11, Q12, Q13, Q14,
    Q15, Q16, Q17, Q18, Q19, Q20, Q21, Q22, Q23

>
> ## Preparar datos de entrada
> # Obtener la matriz de correlación R
> raqR<-cor(raqDatos)
>
> #### ADECUACIÓN MUESTRAL
> dim(raqDatos)
[1] 2571   23
>
> ### Comprobarque las variables no correlacionan ni demasiado poco,ni mucho (r>0
.3 y r<0.9)
> ## Examinar la matriz R
> raqR
             Q01         Q02         Q03         Q04         Q05         Q06
Q01  1.000000000 -0.09872403 -0.3366489  0.43586018  0.40243992  0.21673399
Q02 -0.098724032  1.00000000  0.3183902 -0.11185965 -0.11934658 -0.07420968
Q03 -0.336648879  0.31839020  1.0000000 -0.38046016 -0.31030879 -0.22674048
Q04  0.435860179 -0.11185965 -0.3804602  1.00000000  0.40067225  0.27820154
Q05  0.402439917 -0.11934658 -0.3103088  0.40067225  1.00000000  0.25746014
Q06  0.216733985 -0.07420968 -0.2267405  0.27820154  0.25746014  1.00000000
Q07  0.305365139 -0.15917448 -0.3819533  0.40861502  0.33939179  0.51358048
Q08  0.330737608 -0.04962257 -0.2586342  0.34942939  0.26862697  0.22283175
Q09 -0.092339458  0.31464054  0.2998036 -0.12454637 -0.09570151 -0.11264384
Q10  0.213681706 -0.08400316 -0.1933887  0.21581010  0.25820925  0.32223023
Q11  0.356786290 -0.14382984 -0.3506397  0.36865655  0.29782882  0.32807072
Q12  0.345381133 -0.19486946 -0.4099513  0.44164706  0.34674325  0.31250937
Q13  0.354646283 -0.14274026 -0.3179193  0.34429168  0.30182159  0.46640487
Q14  0.337879655 -0.16469991 -0.3707551  0.35080964  0.31533810  0.40224407
Q15  0.245752635 -0.16499581 -0.3123968  0.33423089  0.26137190  0.35989309
Q16  0.498618057 -0.16755228 -0.4186478  0.41586725  0.39491795  0.24433888
Q17  0.370550512 -0.08699527 -0.3273715  0.38273945  0.31041722  0.28226121
Q18  0.347118037 -0.16389415 -0.3752329  0.38200149  0.32209148  0.51332164
Q19 -0.189011027  0.20329748  0.3415737 -0.18597751 -0.16532210 -0.16675017
Q20  0.213897945 -0.20159437 -0.3248338  0.24291796  0.19966945  0.10092489
Q21  0.329153138 -0.20461730 -0.4171878  0.41029317  0.33461494  0.27233273
Q22 -0.104408664  0.23087487  0.2036569 -0.09838349 -0.13253593 -0.16513541
Q23 -0.004480593  0.09967828  0.1502065 -0.03381815 -0.04165684 -0.06868743
             Q07         Q08         Q09         Q10         Q11         Q12
Q01  0.30536514  0.33073761 -0.09233946  0.21368171  0.35678629  0.34538113
Q02 -0.15917448 -0.04962257  0.31464054 -0.08400316 -0.14382984 -0.19486946
Q03 -0.38195325 -0.25863421  0.29980362 -0.19338871 -0.35063969 -0.40995127
Q04  0.40861502  0.34942939 -0.12454637  0.21581010  0.36865655  0.44164706
Q05  0.33939179  0.26862697 -0.09570151  0.25820925  0.29782882  0.34674325
Q06  0.51358048  0.22283175 -0.11264384  0.32223023  0.32807072  0.31250937
Q07  1.00000000  0.29749696 -0.12829828  0.28372299  0.34474770  0.42298591
Q08  0.29749696  1.00000000  0.01573316  0.15860850  0.62929768  0.25198582
Q09 -0.12829828  0.01573316  1.00000000 -0.13418658 -0.11552479 -0.16739436
Q10  0.28372299  0.15860850 -0.13418658  1.00000000  0.27143657  0.24582591
Q11  0.34474770  0.62929768 -0.11552479  0.27143657  1.00000000  0.33529466
Q12  0.42298591  0.25198582 -0.16739436  0.24582591  0.33529466  1.00000000
Q13  0.44211926  0.31424716 -0.16743882  0.30196707  0.42316548  0.48871303
Q14  0.44070276  0.28058958 -0.12150197  0.25468730  0.32532025  0.43270398
```

```
Q15  0.39136675  0.29968600 -0.18657099  0.29523438  0.36482687  0.33179910
Q16  0.38854534  0.32149420 -0.18886556  0.29058576  0.36907763  0.40805908
Q17  0.39074283  0.59014022 -0.03681556  0.21832214  0.58683495  0.33269383
Q18  0.50086685  0.27974433 -0.14957782  0.29250304  0.37341373  0.49296482
Q19 -0.26912031 -0.15947671  0.24931170 -0.12723487 -0.19965203 -0.26665953
Q20  0.22095420  0.17515089 -0.15864747  0.08406520  0.25533736  0.29802585
Q21  0.48300388  0.29571756 -0.13594310  0.19313633  0.34643407  0.44063832
Q22 -0.16820488 -0.07917265  0.25684622 -0.13090831 -0.16198921 -0.16728557
Q23 -0.07029016 -0.05023839  0.17077441 -0.06191796 -0.08637256 -0.04642506
            Q13         Q14         Q15         Q16         Q17         Q18
Q01  0.35464628  0.33787966  0.24575263  0.49861806  0.37055051  0.34711804
Q02 -0.14274026 -0.16469991 -0.16499581 -0.16755228 -0.08699527 -0.16389415
Q03 -0.31791928 -0.37075510 -0.31239678 -0.41864780 -0.32737145 -0.37523290
Q04  0.34429168  0.35080964  0.33423089  0.41586725  0.38273945  0.38200149
Q05  0.30182159  0.31533810  0.26137190  0.39491795  0.31041722  0.32209148
Q06  0.46640487  0.40224407  0.35989309  0.24433888  0.28226121  0.51332164
Q07  0.44211926  0.44070276  0.39136675  0.38854534  0.39074283  0.50086685
Q08  0.31424716  0.28058958  0.29968600  0.32149420  0.59014022  0.27974433
Q09 -0.16743882 -0.12150197 -0.18657099 -0.18886556 -0.03681556 -0.14957782
Q10  0.30196707  0.25468730  0.29523438  0.29058576  0.21832214  0.29250304
Q11  0.42316548  0.32532025  0.36482687  0.36907763  0.58683495  0.37341373
Q12  0.48871303  0.43270398  0.33179910  0.40805908  0.33269383  0.49296482
Q13  1.00000000  0.44978632  0.34219704  0.35837775  0.40837657  0.53293713
Q14  0.44978632  1.00000000  0.38011484  0.41841820  0.35374183  0.49830615
Q15  0.34219704  0.38011484  1.00000000  0.45427861  0.37310235  0.34287045
Q16  0.35837775  0.41841820  0.45427861  1.00000000  0.40976309  0.42197911
Q17  0.40837657  0.35374183  0.37310235  0.40976309  1.00000000  0.37560681
Q18  0.53293713  0.49830615  0.34287045  0.42197911  0.37560681  1.00000000
Q19 -0.22697105 -0.25405813 -0.20980230 -0.26704702 -0.16288096 -0.25663183
Q20  0.20396327  0.22592173  0.20625622  0.26514025  0.20523013  0.23518040
Q21  0.37443078  0.39938896  0.29971557  0.42054273  0.36349147  0.43010427
Q22 -0.19535632 -0.16983754 -0.16790617 -0.15579385 -0.12629066 -0.15982631
Q23 -0.05298304 -0.04847418 -0.06200665 -0.08152195 -0.09167243 -0.08041698
            Q19         Q20         Q21         Q22          Q23
Q01 -0.1890110  0.21389794  0.32915314 -0.10440866 -0.004480593
Q02  0.2032975 -0.20159437 -0.20461730  0.23087487  0.099678285
Q03  0.3415737 -0.32483385 -0.41718781  0.20365686  0.150206522
Q04 -0.1859775  0.24291796  0.41029317 -0.09838349 -0.033818152
Q05 -0.1653221  0.19966945  0.33461494 -0.13253593 -0.041656841
Q06 -0.1667502  0.10092489  0.27233273 -0.16513541 -0.068687430
Q07 -0.2691203  0.22095420  0.48300388 -0.16820488 -0.070290157
Q08 -0.1594767  0.17515089  0.29571756 -0.07917265 -0.050238392
Q09  0.2493117 -0.15864747 -0.13594310  0.25684622  0.170774410
Q10 -0.1272349  0.08406520  0.19313633 -0.13090831 -0.061917956
Q11 -0.1996520  0.25533736  0.34643407 -0.16198921 -0.086372565
Q12 -0.2666595  0.29802585  0.44063832 -0.16728557 -0.046425059
Q13 -0.2269710  0.20396327  0.37443078 -0.19535632 -0.052983042
Q14 -0.2540581  0.22592173  0.39938896 -0.16983754 -0.048474181
Q15 -0.2098023  0.20625622  0.29971557 -0.16790617 -0.062006650
Q16 -0.2670470  0.26514025  0.42054273 -0.15579385 -0.081521950
Q17 -0.1628810  0.20523013  0.36349147 -0.12629066 -0.091672426
Q18 -0.2566318  0.23518040  0.43010427 -0.15982631 -0.080416984
Q19  1.0000000 -0.24859386 -0.27489793  0.23392259  0.122434401
Q20 -0.2485939  1.00000000  0.46770448 -0.09970186 -0.034665293
Q21 -0.2748979  0.46770448  1.00000000 -0.12902148 -0.067664367
Q22  0.2339226 -0.09970186 -0.12902148  1.00000000  0.230369402
Q23  0.1224344 -0.03466529 -0.06766437  0.23036940  1.000000000
>
> # Más útil gráficamente
> cor.plot(raqR)
```




```
>
> ## Realizar Test de Bartlett. Contrastar la H0 de que R=I (r<0.3).
> # Si podemos aceptar esta hipótesis, nuestros datos no son adecuados para el AF
.
> # Debemos obtener p-valores < 0.05 (alfa)
>
> alfa<-0.05
> bartlett.res<-cortest.bartlett(raqR, dim(raqDatos)[1])
> if (bartlett.res[2]<alfa) {print("No hay evidencia para suponer poca relación.
Los datos son adecuados")} else print("Los datos NO son adecuados para este análi
sis")
[1] "No hay evidencia para suponer poca relación. Los datos son adecuados"
>
> # El test se puede hacer a partir de las variables origibales
> # cortest.bartlett(raqDatos)
>
> ## Realizar Kaiser-Meyer-Olkin (KMO). Contrastar que r<0.9 y examinar matriz an
timagen.
>
> # Llamada a kmo
> raqKMO <- KMO(raqDatos)
>
> # Llamada a kmo by G. Jay Kerns, Ph.D., Youngstown State University (http://tol
stoy.newcastle.edu.au/R/e2/help/07/08/22816.html)
> # Cargar a KMO
> source("kmo.R")
>
> raq.kmo <- kmo(raqDatos)
> raq.kmo
$overall
[1] 0.9302245

$report
[1] "The KMO test yields a degree of common variance marvelous."

$individual
      Q01       Q02       Q03       Q04       Q05       Q06       Q07       Q08
0.9297610 0.8747754 0.9510378 0.9553403 0.9600892 0.8913314 0.9416800 0.8713055
      Q09       Q10       Q11       Q12       Q13       Q14       Q15       Q16
0.8337295 0.9486858 0.9059338 0.9548324 0.9482270 0.9671722 0.9404402 0.9336439
      Q17       Q18       Q19       Q20       Q21       Q22       Q23
0.9306205 0.9479508 0.9407021 0.8890514 0.9293369 0.8784508 0.7663994

$AIS
              [,1]         [,2]         [,3]         [,4]          [,5]         [
,6]
 [1,]  0.627153118 -0.014264007  0.032684206 -0.103441650 -1.037583e-01  0.012088
782
```

```
 [2,] -0.014264007  0.811755252 -0.109450940 -0.028631346  7.727771e-03 -0.036374
795
 [3,]  0.032684206 -0.109450940  0.601893661  0.050984914  2.433620e-02 -0.024815
703
 [4,] -0.103441650 -0.028631346  0.050984914  0.614852103 -8.837819e-02 -0.004017
651
 [5,] -0.103758317  0.007727771  0.024336201 -0.088378188  7.090423e-01 -0.022431
442
 [6,]  0.012088782 -0.036374795 -0.024815703 -0.004017651 -2.243144e-02  0.573142
384
 [7,]  0.013059119  0.010812929  0.040454676 -0.049445748 -2.717225e-02 -0.151630
513
 [8,] -0.027742025 -0.021089865 -0.003996693 -0.041960612 -1.634178e-02  0.013146
784
 [9,] -0.011472364 -0.153262421 -0.097431470  0.020499303 -1.470427e-02  0.007177
224
[10,] -0.008658329 -0.009762474 -0.011244544  0.004236945 -7.030633e-02 -0.078722
167
[11,] -0.022064941  0.023201921  0.034186816 -0.011953216 -1.886611e-05 -0.043447
306
[12,] -0.003979867  0.021383623  0.051009738 -0.091752577 -3.723802e-02  0.026094
115
[13,] -0.049926946 -0.005077376 -0.018181988  0.013383379  2.645271e-03 -0.092109
464
[14,] -0.024527325  0.016237776  0.041983788 -0.002600799 -1.725476e-02 -0.058639
638
[15,]  0.056809108  0.026845165  0.005124834 -0.039445344  9.636461e-03 -0.078628
196
[16,] -0.153020286 -0.007522866  0.045802079 -0.020823640 -5.903514e-02  0.056645
166
[17,] -0.026661898 -0.018384278  0.019122891 -0.019771474 -1.076964e-02  0.022254
595
[18,] -0.013045034  0.011714556  0.021451839 -0.013716299  1.493170e-03 -0.131528
207
[19,]  0.008672592 -0.023479847 -0.083483937 -0.023826851 -1.327839e-02 -0.010287
564
[20,] -0.010925121  0.045232762  0.051626828 -0.002724226 -8.176177e-03  0.033270
691
[21,]  0.003767931  0.027566236  0.040049040 -0.050058353 -2.857293e-02  0.021641
617
[22,]  0.001054318 -0.099886582 -0.004649695 -0.023288160  2.654928e-02  0.027530
455
[23,] -0.044822472 -0.002060071 -0.056616249 -0.012816343 -3.825819e-03  0.013362
232
              [,7]          [,8]         [,9]        [,10]         [,11]
 [1,]  0.013059119 -0.0277420250 -0.011472364 -0.008658329 -2.206494e-02
 [2,]  0.010812929 -0.0210898651 -0.153262421 -0.009762474  2.320192e-02
 [3,]  0.040454676 -0.0039966925 -0.097431470 -0.011244544  3.418682e-02
 [4,] -0.049445748 -0.0419606124  0.020499303  0.004236945 -1.195322e-02
 [5,] -0.027172245 -0.0163417807 -0.014704269 -0.070306334 -1.886611e-05
 [6,] -0.151630513  0.0131467843  0.007177224 -0.078722167 -4.344731e-02
 [7,]  0.530473739 -0.0075535688 -0.019180626 -0.021520785  2.250863e-02
 [8,] -0.007553569  0.5098493087 -0.062208913  0.032887772 -2.015734e-01
 [9,] -0.019180626 -0.0622089131  0.780016094  0.033713150  2.240585e-02
[10,] -0.021520785  0.0328877721  0.033713150  0.802646950 -5.653395e-02
[11,]  0.022508629 -0.2015734127  0.022405846 -0.056533948  4.697590e-01
[12,] -0.023976511  0.0180486093 -0.001900561 -0.012680730 -2.719009e-03
[13,] -0.020888782  0.0009523293  0.039655398 -0.040009293 -5.034302e-02
[14,] -0.030671484 -0.0130510906 -0.029714559 -0.008056592  1.858099e-02
[15,] -0.045137483 -0.0193594501  0.048849295 -0.067752682 -2.883092e-02
[16,] -0.010577284 -0.0031974354  0.033607617 -0.053707659  2.466825e-03
[17,] -0.041541853 -0.1503268370 -0.042750313  0.007880771 -1.121229e-01
[18,] -0.045065228  0.0122084002 -0.003463633 -0.016681035 -1.091403e-02
[19,]  0.044062049  0.0298246527 -0.087018152 -0.007303013 -3.541011e-03
[20,]  0.029839248  0.0127751183  0.028689911  0.032694075 -4.786649e-02
[21,] -0.111906752 -0.0108148909 -0.019980361  0.011249010 -2.321623e-03
[22,]  0.008498244 -0.0148291041 -0.101440618  0.015269216  2.142715e-02
[23,] -0.005620156  0.0015036112 -0.078101774  0.012521640  6.291945e-03
             [,12]         [,13]         [,14]        [,15]         [,16]
```

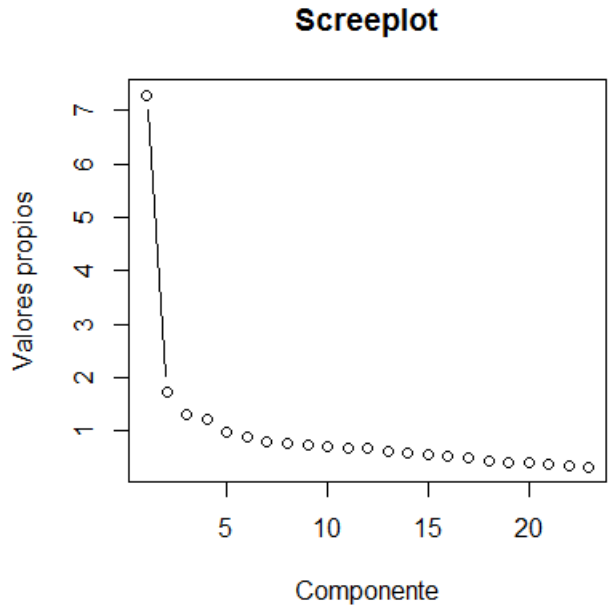
```
 [1,] -0.003979867 -0.0499269456 -0.0245273252  0.056809108 -0.153020286
 [2,]  0.021383623 -0.0050773761  0.0162377764  0.026845165 -0.007522866
 [3,]  0.051009738 -0.0181819885  0.0419837881  0.005124834  0.045802079
 [4,] -0.091752577  0.0133833788 -0.0026007994 -0.039445344 -0.020823640
 [5,] -0.037238024  0.0026452706 -0.0172547624  0.009636461 -0.059035136
 [6,]  0.026094115 -0.0921094642 -0.0586396376 -0.078628196  0.056645166
 [7,] -0.023976511 -0.0208887824 -0.0306714840 -0.045137483 -0.010577284
 [8,]  0.018048609  0.0009523293 -0.0130510906 -0.019359450 -0.003197435
 [9,] -0.001900561  0.0396553976 -0.0297145589  0.048849295  0.033607617
[10,] -0.012680730 -0.0400092935 -0.0080565924 -0.067752682 -0.053707659
[11,] -0.002719009 -0.0503430182  0.0185809852 -0.028830917  0.002466825
[12,]  0.575501008 -0.1114757308 -0.0483506266 -0.016190501 -0.022213223
[13,] -0.111475731  0.5494652433 -0.0569919843 -0.004979672  0.014060536
[14,] -0.048350627 -0.0569919843  0.6070035304 -0.058969619 -0.046150485
[15,] -0.016190501 -0.0049796722 -0.0589696187  0.655546920 -0.137812029
[16,] -0.022213223  0.0140605358 -0.0461504852 -0.137812029  0.536854026
[17,]  0.003157819 -0.0475192420 -0.0157657049 -0.049220560 -0.039461084
[18,] -0.079021863 -0.0896507793 -0.0805449780  0.021891662 -0.046937807
[19,]  0.027194136  0.0059297117  0.0304844455  0.006757464  0.030364853
[20,] -0.041898800  0.0113005163  0.0007480665 -0.025874233 -0.003316873
[21,] -0.044445498 -0.0182731673 -0.0363989731  0.020844352 -0.046249347
[22,]  0.012477538  0.0354047137  0.0205919013  0.018362943 -0.002181511
[23,] -0.020240842 -0.0212023173 -0.0190981192 -0.018244263  0.016098449
              [,17]         [,18]         [,19]         [,20]         [,21]         [,
22]
 [1,] -0.026661898 -0.013045034  0.008672592 -0.0109251211  0.003767931  0.001054
318
 [2,] -0.018384278  0.011714556 -0.023479847  0.0452327615  0.027566236 -0.099886
582
 [3,]  0.019122891  0.021451839 -0.083483937  0.0516268279  0.040049040 -0.004649
695
 [4,] -0.019771474 -0.013716299 -0.023826851 -0.0027242256 -0.050058353 -0.023288
160
 [5,] -0.010769643  0.001493170 -0.013278387 -0.0081761769 -0.028572928  0.026549
277
 [6,]  0.022254595 -0.131528207 -0.010287564  0.0332706908  0.021641617  0.027530
455
 [7,] -0.041541853 -0.045065228  0.044062049  0.0298392484 -0.111906752  0.008498
244
 [8,] -0.150326837  0.012208400  0.029824653  0.0127751183 -0.010814891 -0.014829
104
 [9,] -0.042750313 -0.003463633 -0.087018152  0.0286899108 -0.019980361 -0.101440
618
[10,]  0.007880771 -0.016681035 -0.007303013  0.0326940755  0.011249010  0.015269
216
[11,] -0.112122917 -0.010914029 -0.003541011 -0.0478664911 -0.002321623  0.021427
154
[12,]  0.003157819 -0.079021863  0.027194136 -0.0418987996 -0.044445498  0.012477
538
[13,] -0.047519242 -0.089650779  0.005929712  0.0113005163 -0.018273167  0.035404
714
[14,] -0.015765705 -0.080544978  0.030484446  0.0007480665 -0.036398973  0.020591
901
[15,] -0.049220560  0.021891662  0.006757464 -0.0258742335  0.020844352  0.018362
943
[16,] -0.039461084 -0.046937807  0.030364853 -0.0033168730 -0.046249347 -0.002181
511
[17,]  0.505549313 -0.017196614 -0.029607073  0.0093954885 -0.021343045  0.006514
419
[18,] -0.017196614  0.507773950  0.018885098 -0.0015453311 -0.037757448 -0.015627
644
[19,] -0.029607073  0.018885098  0.791081637  0.0689982743  0.020677430 -0.093130
611
[20,]  0.009395489 -0.001545331  0.068998274  0.7297949321 -0.203976791 -0.008719
382
[21,] -0.021343045 -0.037757448  0.020677430 -0.2039767910  0.546465907 -0.016292
080
[22,]  0.006514419 -0.015627644 -0.093130611 -0.0087193817 -0.016292080  0.832994
213
```

```
[23,]  0.037100122  0.015381707 -0.032641178 -0.0229392353  0.009011910 -0.153800
018
             [,23]
 [1,] -0.044822472
 [2,] -0.002060071
 [3,] -0.056616249
 [4,] -0.012816343
 [5,] -0.003825819
 [6,]  0.013362232
 [7,] -0.005620156
 [8,]  0.001503611
 [9,] -0.078101774
[10,]  0.012521640
[11,]  0.006291945
[12,] -0.020240842
[13,] -0.021202317
[14,] -0.019098119
[15,] -0.018244263
[16,]  0.016098449
[17,]  0.037100122
[18,]  0.015381707
[19,] -0.032641178
[20,] -0.022939235
[21,]  0.009011910
[22,] -0.153800018
[23,]  0.914348653

$AIR
              [,1]         [,2]         [,3]         [,4]          [,5]         [
,6]
 [1,]  0.929761030 -0.019991353  0.053197504 -0.166580176 -1.555966e-01  0.020163
434
 [2,] -0.019991353  0.874775439 -0.156583882 -0.040526946  1.018605e-02 -0.053328
178
 [3,]  0.053197504 -0.156583882  0.951037838  0.083810123  3.725262e-02 -0.042250
849
 [4,] -0.166580176 -0.040526946  0.083810123  0.955340346 -1.338516e-01 -0.006767
926
 [5,] -0.155596577  0.010186046  0.037252619 -0.133851633  9.600892e-01 -0.035187
615
 [6,]  0.020163434 -0.053328178 -0.042250849 -0.006767926 -3.518762e-02  0.891331
392
 [7,]  0.022640977  0.016477785  0.071594044 -0.086578823 -4.430549e-02 -0.274994
113
 [8,] -0.049060328 -0.032782369 -0.007214728 -0.074943776 -2.717956e-02  0.024320
212
 [9,] -0.016402662 -0.192606681 -0.142196068  0.029600700 -1.977223e-02  0.010734
293
[10,] -0.012203517 -0.012094420 -0.016177808  0.006031221 -9.319575e-02 -0.116065
594
[11,] -0.040651671  0.037572822  0.064292632 -0.022241387 -3.268952e-05 -0.083732
492
[12,] -0.006624586  0.031285686  0.086670274 -0.154244574 -5.829449e-02  0.045434
763
[13,] -0.085050739 -0.007602504 -0.031616325  0.023025567  4.238029e-03 -0.164135
635
[14,] -0.039752806  0.023132278  0.069458617 -0.004257218 -2.630130e-02 -0.099417
859
[15,]  0.088599095  0.036800323  0.008158644 -0.062131025  1.413447e-02 -0.128275
927
[16,] -0.263714560 -0.011395743  0.080574423 -0.036244591 -9.568555e-02  0.102118
277
[17,] -0.047350276 -0.028698042  0.034666651 -0.035462712 -1.798802e-02  0.041343
446
[18,] -0.023116560  0.018246436  0.038803369 -0.024548017  2.488502e-03 -0.243810
514
[19,]  0.012312649 -0.029300299 -0.120985239 -0.034164154 -1.772959e-02 -0.015278
145
```

```
[20,] -0.016148755  0.058767894  0.077895996 -0.004066842 -1.136616e-02  0.051443
456
[21,]  0.006436276  0.041388851  0.069831391 -0.086359414 -4.590258e-02  0.038670
243
[22,]  0.001458692 -0.121471280 -0.006566645 -0.032540847  3.454583e-02  0.039843
834
[23,] -0.059190649 -0.002391188 -0.076317703 -0.017093180 -4.751519e-03  0.018458
306
              [,7]         [,8]         [,9]        [,10]         [,11]         [,
12]
 [1,]  0.022640977 -0.049060328 -0.016402662 -0.012203517 -4.065167e-02 -0.006624
586
 [2,]  0.016477785 -0.032782369 -0.192606681 -0.012094420  3.757282e-02  0.031285
686
 [3,]  0.071594044 -0.007214728 -0.142196068 -0.016177808  6.429263e-02  0.086670
274
 [4,] -0.086578823 -0.074943776  0.029600700  0.006031221 -2.224139e-02 -0.154244
574
 [5,] -0.044305485 -0.027179557 -0.019772226 -0.093195746 -3.268952e-05 -0.058294
488
 [6,] -0.274994113  0.024320212  0.010734293 -0.116065594 -8.373249e-02  0.045434
763
 [7,]  0.941679984 -0.014524438 -0.029818037 -0.032981005  4.508993e-02 -0.043394
132
 [8,] -0.014524438  0.871305451 -0.098646099  0.051410433 -4.118838e-01  0.033319
600
 [9,] -0.029818037 -0.098646099  0.833729475  0.042607424  3.701451e-02 -0.002836
658
[10,] -0.032981005  0.051410433  0.042607424  0.948685759 -9.206816e-02 -0.018657
736
[11,]  0.045089926 -0.411883847  0.037014508 -0.092068161  9.059338e-01 -0.005229
377
[12,] -0.043394132  0.033319600 -0.002836658 -0.018657736 -5.229377e-03  0.954832
379
[13,] -0.038691099  0.001799269  0.060573186 -0.060246038 -9.909040e-02 -0.198238
120
[14,] -0.054051436 -0.023460123 -0.043183904 -0.011542330  3.479650e-02 -0.081805
679
[15,] -0.076542651 -0.033486535  0.068313199 -0.093403322 -5.195400e-02 -0.026359
390
[16,] -0.019820478 -0.006111569  0.051934725 -0.081817431  4.912162e-03 -0.039963
215
[17,] -0.080218060 -0.296096886 -0.068077838  0.012371565 -2.300780e-01  0.005854
400
[18,] -0.086830924  0.023993999 -0.005503575 -0.026129164 -2.234664e-02 -0.146180
354
[19,]  0.068017723  0.046961702 -0.110776444 -0.009164929 -5.808701e-03  0.040303
389
[20,]  0.047957387  0.020943227  0.038025680  0.042717552 -8.175110e-02 -0.064651
427
[21,] -0.207846545 -0.020488949 -0.030603423  0.016985193 -4.582184e-03 -0.079254
366
[22,]  0.012784279 -0.022754799 -0.125845912  0.018673832  3.425358e-02  0.018021
248
[23,] -0.008069758  0.002202210 -0.092481145  0.014616488  9.600446e-03 -0.027902
917
             [,13]        [,14]        [,15]        [,16]        [,17]         [,18
]
 [1,] -0.085050739 -0.039752806  0.088599095 -0.263714560 -0.04735028 -0.02311656
0
 [2,] -0.007602504  0.023132278  0.036800323 -0.011395743 -0.02869804  0.01824643
6
 [3,] -0.031616325  0.069458617  0.008158644  0.080574423  0.03466665  0.03880336
9
 [4,]  0.023025567 -0.004257218 -0.062131025 -0.036244591 -0.03546271 -0.02454801
7
 [5,]  0.004238029 -0.026301304  0.014134472 -0.095685554 -0.01798802  0.00248850
2
```

```
 [6,] -0.164135635 -0.099417859 -0.128275927  0.102118277  0.04134345 -0.24381051
4
 [7,] -0.038691099 -0.054051436 -0.076542651 -0.019820478 -0.08021806 -0.08683092
4
 [8,]  0.001799269 -0.023460123 -0.033486535 -0.006111569 -0.29609689  0.02399399
9
 [9,]  0.060573186 -0.043183904  0.068313199  0.051934725 -0.06807784 -0.00550357
5
[10,] -0.060246038 -0.011542330 -0.093403322 -0.081817431  0.01237157 -0.02612916
4
[11,] -0.099090396  0.034796500 -0.051954001  0.004912162 -0.23007798 -0.02234664
5
[12,] -0.198238120 -0.081805679 -0.026359390 -0.039963215  0.00585440 -0.14618035
4
[13,]  0.948226989 -0.098684298 -0.008297146  0.025888307 -0.09016077 -0.16972614
5
[14,] -0.098684298  0.967172178 -0.093482608 -0.080844887 -0.02846007 -0.14508003
5
[15,] -0.008297146 -0.093482608  0.940440197 -0.232304235 -0.08549937  0.03794387
0
[16,]  0.025888307 -0.080844887 -0.232304235  0.933643945 -0.07574590 -0.08989995
7
[17,] -0.090160774 -0.028460074 -0.085499366 -0.075745901  0.93062054 -0.03394110
5
[18,] -0.169726145 -0.145080035  0.037943870 -0.089899957 -0.03394110  0.94795084
1
[19,]  0.008993999  0.043991831  0.009383637  0.046594261 -0.04681694  0.02979705
8
[20,]  0.017845460  0.001123943 -0.037408060 -0.005299078  0.01546811 -0.00253855
1
[21,] -0.033347416 -0.063199259  0.034826111 -0.085387827 -0.04060627 -0.07167800
1
[22,]  0.052332319  0.028958752  0.024849600 -0.003262183  0.01003859 -0.02402908
9
[23,] -0.029912816 -0.025635340 -0.023565058  0.022977354  0.05456795  0.02257425
4
              [,19]        [,20]        [,21]        [,22]        [,23]
 [1,]  0.012312649 -0.016148755  0.006436276  0.001458692 -0.059190649
 [2,] -0.029300299  0.058767894  0.041388851 -0.121471280 -0.002391188
 [3,] -0.120985239  0.077895996  0.069831391 -0.006566645 -0.076317703
 [4,] -0.034164154 -0.004066842 -0.086359414 -0.032540847 -0.017093180
 [5,] -0.017729586 -0.011366160 -0.045902583  0.034545829 -0.004751519
 [6,] -0.015278145  0.051443456  0.038670243  0.039843834  0.018458306
 [7,]  0.068017723  0.047957387 -0.207846545  0.012784279 -0.008069758
 [8,]  0.046961702  0.020943227 -0.020488949 -0.022754799  0.002202210
 [9,] -0.110776444  0.038025680 -0.030603423 -0.125845912 -0.092481145
[10,] -0.009164929  0.042717552  0.016985193  0.018673832  0.014616488
[11,] -0.005808701 -0.081751099 -0.004582184  0.034253576  0.009600446
[12,]  0.040303389 -0.064651427 -0.079254366  0.018021248 -0.027902917
[13,]  0.008993999  0.017845460 -0.033347416  0.052332319 -0.029912816
[14,]  0.043991831  0.001123943 -0.063199259  0.028958752 -0.025635340
[15,]  0.009383637 -0.037408060  0.034826111  0.024849600 -0.023565058
[16,]  0.046594261 -0.005299078 -0.085387827 -0.003262183  0.022977354
[17,] -0.046816943  0.015468108 -0.040606268  0.010038594  0.054567953
[18,]  0.029797058 -0.002538551 -0.071678001 -0.024029089  0.022574254
[19,]  0.940702074  0.090808637  0.031448821 -0.114725725 -0.038379495
[20,]  0.090808637  0.889051429 -0.322997230 -0.011183153 -0.028081649
[21,]  0.031448821 -0.322997230  0.929336944 -0.024147602  0.012749096
[22,] -0.114725725 -0.011183153 -0.024147602  0.878450848 -0.176229853
[23,] -0.038379495 -0.028081649  0.012749096 -0.176229853  0.766399409

>
> ## Evaluar el valor del determinante de R
> det(raqR)
[1] 0.0005271037
>
> #### EXTRACCIÓN DE FACTORES. Funcion principal()
> ## Inicialmente extraemos tantas Componentes como variables
> # Número de componentes
```

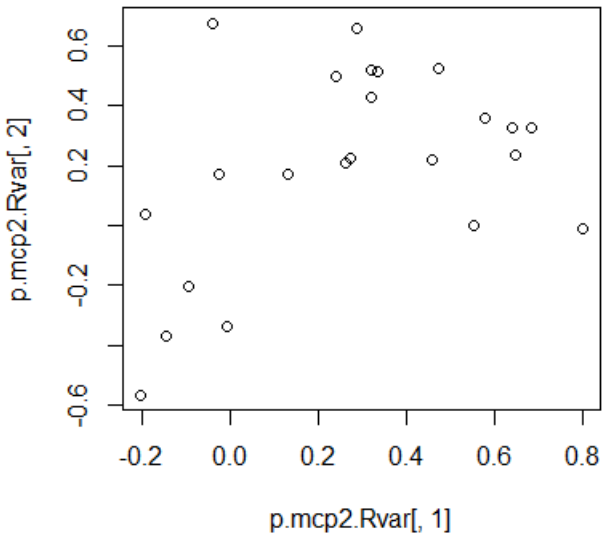
```
> ncomp <- dim(raqDatos)[2]
>
> ## Primer modelo. Modelo inicial sin rotación
> ####################################################
> mcp1<- principal(raqR, nfactors = ncomp, rotate = "none")
> # O bien mcp1<- principal(raqDatos, nfactors = ncomp, rotate = "none")
>
> # Selección de componentes a mantener. Scree plot
> plot(mcp1$values, type = "b", ylab = "Valores propios", xlab = "Componente", ma
in = "Screeplot")
```

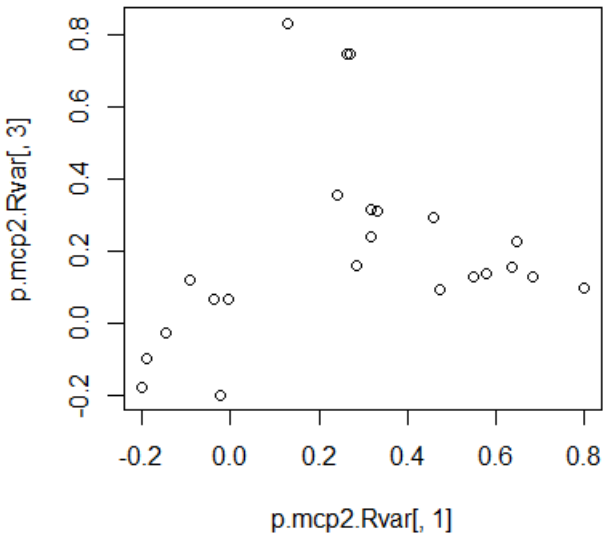



```
>
> ## Segundo modelo. Retenemos 4 componentes sin rotción
> ##########################################################
> ncomp <- 4
> mcp2 <- principal(raqR, ncomp, rotate = "none")
>
> # La funcion devuelve los resultados del PCA en una lista
> # Podemos acceder de manera individual a cada elemento:
>
> # Comunalidades
> mcp2$communality
      Q01       Q02       Q03       Q04       Q05       Q06       Q07       Q08
0.4346477 0.4137525 0.5297160 0.4685890 0.3430498 0.6539317 0.5452943 0.7394635
      Q09       Q10       Q11       Q12       Q13       Q14       Q15       Q16
0.4844805 0.3347726 0.6896049 0.5133281 0.5358284 0.4882649 0.3779918 0.4870822
      Q17       Q18       Q19       Q20       Q21       Q22       Q23
0.6828085 0.5973378 0.3432423 0.4839965 0.5499069 0.4635443 0.4121913
>
> # Valores propios
> mcp2$values
 [1] 7.2900471 1.7388287 1.3167515 1.2271982 0.9878779 0.8953304 0.8055604 0.7828
199
 [9] 0.7509712 0.7169577 0.6835877 0.6695016 0.6119976 0.5777377 0.5491875 0.5231
504
[17] 0.5083962 0.4559399 0.4238036 0.4077909 0.3794799 0.3640223 0.3330618
>
> # Matriz de cargas o pesos
> mcp2$loadings

Loadings:
    PC1    PC2    PC3    PC4
Q01  0.586  0.175 -0.215  0.119
```

```
Q02 -0.303  0.548  0.146
Q03 -0.629  0.290  0.213
Q04  0.634  0.144 -0.149  0.153
Q05  0.556  0.101          0.137
Q06  0.562          0.571
Q07  0.685          0.252  0.103
Q08  0.549  0.401 -0.323 -0.417
Q09 -0.284  0.627          0.103
Q10  0.437          0.363 -0.103
Q11  0.652  0.245 -0.209 -0.400
Q12  0.669                 0.248
Q13  0.673          0.278
Q14  0.656          0.198  0.135
Q15  0.593          0.117 -0.113
Q16  0.679         -0.138
Q17  0.643  0.330 -0.210 -0.342
Q18  0.701          0.298  0.125
Q19 -0.427  0.390
Q20  0.436 -0.205 -0.404  0.297
Q21  0.658         -0.187  0.282
Q22 -0.302  0.465 -0.116  0.378
Q23 -0.144  0.366          0.507

                 PC1   PC2   PC3   PC4
SS loadings    7.290 1.739 1.317 1.227
Proportion Var 0.317 0.076 0.057 0.053
Cumulative Var 0.317 0.393 0.450 0.503
>
> # Rotación
> mcp2$rotation
[1] "none"
>
> # Medida del ajuste
> mcp2$fit.off
[1] 0.9645252
>
> ## Obtencion de los residuos del modelo 2
> residuos<-factor.residuals(raqR, mcp2$loadings)
> resid.2 <- residuos
>
> ## Tercer modelo. Retenemos 2 componentes sin rotación
> ##########################################################
>
> ncomp <- 2
> mcp3 <- principal(raqR, ncomp, rotate = "none")
>
> ## Obtencion de los residuos del modelo 3
> resid.3 <- factor.residuals(raqR, mcp3$loadings)
>
> ## Ajuste del modelo. Residuos. SRMR
> # funcion residual.stats (Andy Field, Jeremy Miles, and Z.F., 2012. Discovering
statistics using R, SAGE Publications)
> source("residual.stats.R")
>
> residual.stats(residuos)
Root means squared residual =  0.05549286
Number of absolute residuals > 0.05 =  91
Proportion of absolute residuals > 0.05 =  0.3596838
>
>
> ## Rotacion
> ##########################################################
> # Rotacion ortogonal varimax del modelo 2 (4 componentes)
> ##########################################################
> ncomp <- 4
> mcp2.RVar <- principal(raqR, ncomp, rotate = "varimax")
>
> p.mcp2.Rvar <- matrix(mcp2.RVar$loadings, 23,4)
> colnames(p.mcp2.Rvar) <- c("CP1", "CP2", "CP3", "CP4")
```

```
> plot(p.mcp2.Rvar[,1], p.mcp2.Rvar[,2])
> plot(p.mcp2.Rvar[,1], p.mcp2.Rvar[,3])
```






```
>
> # Rotacion ortogonal varimax del modelo 3 (2 componentes)
> ########################################################
> ncomp <- 2
> mcp3.RVar <- principal(raqR, ncomp, rotate = "varimax")
>
>
> ### Comparacion del ajuste entre los modelos 2 y 3
> ########################################################
> # Calculamos los residuos de los modelos 2 y 3 sin rotar
> # Modelo 2: 4 componentes sin rotar
> resid.2 <- factor.residuals(raqR, mcp2$loadings)
> stats.resid.2 <- residual.stats(resid.2)
Root means squared residual =  0.05549286
Number of absolute residuals > 0.05 =  91
Proportion of absolute residuals > 0.05 =  0.3596838
> mcp2$fit.off
[1] 0.9645252
>
> # Modelo 3: 2 componentes sin rotar
> resid.3 <- residuos
> stats.resid.3 <- residual.stats(resid.3)
Root means squared residual =  0.05549286
Number of absolute residuals > 0.05 =  91
Proportion of absolute residuals > 0.05 =  0.3596838
> mcp3$fit.off
[1] 0.9533968
>
> # Modelo 4: 4 componentes rotacion Varimax
> resid.2.Rvar <- factor.residuals(raqR, mcp2.RVar$loadings)
> stats.resid.2.Rvar <- residual.stats(resid.2.Rvar)
Root means squared residual =  0.05549286
Number of absolute residuals > 0.05 =  91
Proportion of absolute residuals > 0.05 =  0.3596838
> mcp2.RVar$fit.off
[1] 0.9645252
>
> # Modelo 5: 2 componentes rotacion Varimax
> resid.3.Rvar <- factor.residuals(raqR, mcp3.RVar$loadings)
> stats.resid.3 <- residual.stats(resid.3.Rvar)
```

```
Root means squared residual =  0.06360403
Number of absolute residuals > 0.05 =  114
Proportion of absolute residuals > 0.05 =  0.4505929
> mcp3.RVar$fit.off
[1] 0.9533968
>
> ## Interpretacion del modelo
> # Interpretacion del modelo 2 rotado
> print.psych(mcp2.RVar, cut = 0.3, sort = T)
Principal Components Analysis
Call: principal(r = raqR, nfactors = ncomp, rotate = "varimax")
Standardized loadings (pattern matrix) based upon correlation matrix
    item   RC3   RC1   RC4   RC2   h2   u2 com
Q06    6  0.80                    0.65 0.35 1.0
Q18   18  0.68  0.33              0.60 0.40 1.5
Q13   13  0.65                    0.54 0.46 1.6
Q07    7  0.64  0.33              0.55 0.45 1.7
Q14   14  0.58  0.36              0.49 0.51 1.8
Q10   10  0.55                    0.33 0.67 1.2
Q15   15  0.46                    0.38 0.62 2.6
Q20   20        0.68              0.48 0.52 1.1
Q21   21        0.66              0.55 0.45 1.5
Q03    3       -0.57        0.37 0.53 0.47 2.3
Q12   12  0.47  0.52              0.51 0.49 2.1
Q04    4  0.32  0.52  0.31        0.47 0.53 2.4
Q16   16  0.33  0.51  0.31        0.49 0.51 2.6
Q01    1        0.50  0.36        0.43 0.57 2.4
Q05    5  0.32  0.43              0.34 0.66 2.5
Q08    8              0.83        0.74 0.26 1.1
Q17   17              0.75        0.68 0.32 1.5
Q11   11              0.75        0.69 0.31 1.5
Q09    9                    0.65 0.48 0.52 1.3
Q22   22                    0.65 0.46 0.54 1.2
Q23   23                    0.59 0.41 0.59 1.4
Q02    2       -0.34        0.54 0.41 0.59 1.7
Q19   19       -0.37        0.43 0.34 0.66 2.2

                       RC3  RC1  RC4  RC2
SS loadings           3.73 3.34 2.55 1.95
Proportion Var        0.16 0.15 0.11 0.08
Cumulative Var        0.16 0.31 0.42 0.50
Proportion Explained  0.32 0.29 0.22 0.17
Cumulative Proportion 0.32 0.61 0.83 1.00

Mean item complexity =  1.8
Test of the hypothesis that 4 components are sufficient.

The root mean square of the residuals (RMSR) is  0.06

Fit based upon off diagonal values = 0.96
>
> # Interpretacion del modelo 3
> print.psych(mcp3, cut = 0.3, sort = T)
Principal Components Analysis
Call: principal(r = raqR, nfactors = ncomp, rotate = "none")
Standardized loadings (pattern matrix) based upon correlation matrix
    item   PC1   PC2   h2   u2 com
Q18   18  0.70       0.49 0.51 1.0
Q07    7  0.69       0.47 0.53 1.0
Q16   16  0.68       0.46 0.54 1.0
Q13   13  0.67       0.46 0.54 1.0
Q12   12  0.67       0.45 0.55 1.0
Q21   21  0.66       0.44 0.56 1.0
Q14   14  0.66       0.43 0.57 1.0
Q11   11  0.65       0.49 0.51 1.3
Q17   17  0.64  0.33 0.52 0.48 1.5
Q04    4  0.63       0.42 0.58 1.1
Q03    3 -0.63       0.48 0.52 1.4
Q15   15  0.59       0.35 0.65 1.0
```

```
Q01    1  0.59       0.37 0.63 1.2
Q06    6  0.56       0.33 0.67 1.1
Q05    5  0.56       0.32 0.68 1.1
Q08    8  0.55  0.40 0.46 0.54 1.8
Q10   10  0.44       0.19 0.81 1.0
Q20   20  0.44       0.23 0.77 1.4
Q19   19 -0.43  0.39 0.33 0.67 2.0
Q09    9        0.63 0.47 0.53 1.4
Q02    2 -0.30  0.55 0.39 0.61 1.6
Q22   22 -0.30  0.47 0.31 0.69 1.7
Q23   23        0.37 0.16 0.84 1.3

                        PC1  PC2
SS loadings            7.29 1.74
Proportion Var         0.32 0.08
Cumulative Var         0.32 0.39
Proportion Explained   0.81 0.19
Cumulative Proportion  0.81 1.00

Mean item complexity =  1.3
Test of the hypothesis that 2 components are sufficient.

The root mean square of the residuals (RMSR) is  0.06

Fit based upon off diagonal values = 0.95
>
> ### Obtención de las puntuaciones factoriales para el modelo 2 rotado
> prueba <- principal(raqR, nfactors = 4, rotate = "varimax", scores = T)
>
> detach(raqDatos)
```