Estadística descriptiva y gráficos con R

PID_00293463

Marta Casals Fontanet
Alícia Vila Grifo

Tiempo mínimo de dedicación recomendado: 6 horas

Universitat Oberta de Catalunya

Marta Casals Fontanet

Estadística de formación y profesión. Licenciada en ITM. Máster en Educación y nuevas tecnologías. Profesora de Matemáticas en secundaria y universidad. Profesora colaboradora en el máster de Bioinformática y bioestadística de la UOC y la UB.

Alicia Vila Grifo

Licenciada en Matemáticas por la Universidad de Valencia y profesora consultora de Probabilidad, Estadística y Análisis de Datos en diferentes estudios de la UOC. Actualmente es profesora funcionaria de Informática en el Departamento de Educación de la Generalitat de Cataluña, en los ámbitos de programación y bases de datos. También participa en la coordinación de proyectos de aplicaciones web y de análisis de datos.

El encargo y la creación de este recurso de aprendizaje UOC han sido coordinados por los profesores: Álvaro Leitao Rodríguez y David Cantón Fabà

Cómo citar este recurso de aprendizaje con el estilo Harvard:

Casals, M. y Vila A. (2024). *Estadística descriptiva y gráficos con R* [Recurso de aprendizaje textual]. 1.a ed. Fundació Universitat Oberta de Catalunya (FUOC).

Primera edición: febrero 2024

Av. Tibidabo, 39-43, 08035 Barcelona
Autoría: Marta Casals Fontanet, Alícia Vila Grifo
Producción: FUOC

Índice

1. Introducción al LAB2

El **LAB2** es un recurso didáctico complementario de la asignatura **Software para el análisis de datos (SAD)** del máster interuniversitario de Bioinformática y Bioestadística de la Universitat Oberta de Catalunya (UOC) y la Universitat de Barcelona (UB).

Este **LAB2** forma parte de un conjunto de laboratorios prácticos que conjugan contenidos teóricos con ejemplos, ejercicios y casos prácticos reales del ámbito de conocimiento del máster.

El **LAB2** explica los conceptos básicos de estadística descriptiva y gráficos con el lenguaje **R** y su aplicación al ámbito biosanitario. En realidad, aunque **R** es un lenguaje de programación, no suele utilizarse para desarrollar grandes proyectos sino más bien para realizar paquetes de funciones o bloques de programas que puedan ser reutilizados con un objetivo específico. Por ello, las aplicaciones más habituales suelen estar dirigidas a la extracción, manipulación, tratamiento e interpretación de conjuntos de datos con fines específicos.

En este **LAB2**, se introduce al estudiante a la estadística descriptiva, los gráficos y la regresión en **R**.

2. Instrucciones del LAB2

El **LAB2** contiene una serie de apartados con introducciones teóricas, ejemplos y casos prácticos, además de otros ejercicios que se propondrán para que sean resueltos por el estudiante. La temporización y las pautas para trabajar el **LAB2** serán indicadas en el aula de la asignatura. También, incluye la propuesta de solución de los diversos ejercicios y casos prácticos, que servirá para que el estudiante pueda autoevaluar las soluciones realizadas por él mismo.

En ocasiones, para realizar los ejercicios, se utilizarán conjuntos de datos de paquetes propios de **RStudio**, como es el caso de *biopsy*, *anorexia* y *birthwt* del paquete *MASS*, *Iris* o *airquality* y *esoph* del paquete *datasetsDb*, así como también otros conjuntos de datos abiertos de repositorios externos. Todos los ejercicios se realizarán en el entorno de desarrollo integrado para **R**, **RStudio** (https://cran.rstudio.com/), como lenguaje de programación para la informática estadística y los gráficos.

3. Contenidos del LAB2

En este **LAB2** trabajaremos los siguientes contenidos:

- Resumen de datos y variables en R.
- Etiquetar y recodificar en R.
- Tablas de frecuencias con R.
- Tablas de contingencia y test (ji al cuadrado-Fisher).
- Gráficos básicos en R.

- Paquete ggplot2 para la creación de gráficos.
- Guardar y exportar gráficos en R.
- Regresión lineal con R.
- ANOVA.
- Ejercicios de práctica.

4. Resumen de datos y variables en R

En este apartado aprenderemos a resumir y concretar la información de nuestras variables. De esta manera podremos caracterizar nuestros grupos de individuos, ver sus singularidades y, en algunos casos, compararlos con otros grupos. **R** dispone de muchas funciones y comandos para resumir nuestros objetos en **R** (datasets, variables, vectores, etc.).

A continuación, disponéis de una tabla con los comandos más usados y su significado:

Comando	**Explicación del estadístico**
sort (data)	Ordenar datos
summary (data)	Resúmenes estadísticos
fivenum (data)	Cinco valores (mín., Q1, media, Q3, máx.)
min (data)	Mínimo
max (data)	Máximo
range (data)	Rango (máx.–mín.)
mean (data)	Media
median (data)	Mediana
sd (data)	Desviación típica
var (data)	Varianza
cor (data)	Correlación
Q1, Q3	Cuartil 0,25 - Cuartil 0,75

Ejemplo 1:

Vamos a ver un ejemplo de cómo utilizar estos comandos:

```
edad <- c(20, 21, 22, 23, 24, 25, 26, 27) #vector edad
mean(edad) #media de la variable edad

## [1] 23.5

median(edad) #mediana de la variable edad

## [1] 23.5

var(edad) #varianza de la variable edad
```

```
## [1] 6
```

```
sd(edad) #desviación típica de la variable edad
```

```
## [1] 2.44949
```

```
min(edad) #valor mínimo de la variable edad
```

```
## [1] 20
```

```
max(edad) #valor máximo de la variable edad
```

```
## [1] 27
```

```
range(edad) #valores mín. y máx. de la variable edad
```

```
## [1] 20 27
```

R, como paquete estadístico, proporciona diferentes maneras de resumir nuestros datos. Si queremos escoger un único estadístico para todo el conjunto de datos, se puede usar la función *sapply()*. Es un buen método para obtener estadísticas descriptivas, ya que podéis añadir opciones como excluir o no los datos faltantes.

Para poner un ejemplo, buscaremos la media de las variables del paquete **Iris**.

```
sapply(iris, mean, na.rm=TRUE) #de esta forma no tenemos en cuenta los
missings o valores faltantes
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
##     5.843333     3.057333     3.758000     1.199333           NA
```

En este ejemplo, la función *sapply()* calcula la media de todas las variables del paquete **Iris** y excluye los NA.

Ejemplo 2:

Construimos variables en nuestro script y buscamos resúmenes y diferentes estadísticos descriptivos con los datos:

```
#definición de las variables en vectores

id=c(100,110,120,130,140,150,160,170,180,190)
edad=c(18,19,NA,18,24,27,22,25,22,15)
gen=c(2,1,2,2,1,2,1,1,2,1)
peso=c(65,58,56,61,84,99,50,64,87,87)
altura=c(161,170,174,165,150,171,181,170,184,190)
niv_col=c(1,2,3,1,3,2,3,1,2,3)

#usaremos diferentes comandos estadísticos en diferentes variables
sort(edad)  #ordenamos la variable edad
```

```
## [1] 15 18 18 19 22 22 24 25 27
```

```r
summary(edad) #buscamos un resumen estadístico básico de la variable edad
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   15.00   18.00   22.00   21.11   24.00   27.00       1
```

```r
fivenum(gen) #buscamos los cinco estadísticos descriptivos básicos
```

```
## [1] 1.0 1.0 1.5 2.0 2.0
```

```r
mean(peso) #media de la variable peso
```

```
## [1] 71.1
```

```r
median(altura) #mediana de la variable altura
```

```
## [1] 170.5
```

```r
var(altura) #varianza de la variable altura
```

```
## [1] 134.9333
```

```r
cor(altura, peso) #correlación de las variables altura y peso
```

```
## [1] 0.07103622
```

```r
sd(altura) #desviación típica de altura
```

```
## [1] 11.61608
```

5. Etiquetar y recodificar en R

Antes de entrar en la recodificación de variables como tal, podemos hablar de etiquetar las variables que tenemos en un conjunto de datos. Para etiquetar variables, usaremos la opción *labels()* en nuestra variable. Se recomienda generar variables nuevas cuando se etiqueta, pero no es obligatorio.

Ejemplo 3:

Usando las variables definidas anteriormente, etiquetamos de nuevo las variables **niv_col** y **gen**.

```r
gen2=factor(gen, levels=c(1,2), labels=c("Hombre","Mujer")) #generamos un
factor con sus nuevas etiquetas
niv_col2=factor(niv_col, levels=c(1,2,3), labels=c("Colesterol Bajo",
"Colesterol Normal","Colesterol Alto"))
```

Para recodificar variables en data frames tenemos distintas opciones:

- Usar los corchetes **[]** que nos permiten seleccionar grupos de casos o grupos de variables para realizar determinados procedimientos.
- Usar el comando *ifelse()*.

- Usar la función *recode()* que se incluye en la librería **car** y es extremadamente útil.
- Calcular nuevas variables.

Ejemplo 4:

Para mostrar cómo podemos recodificar variables, vamos a ver un ejemplo nuevo:

```
Ident <- 1:5;
Genero <- c("Masculino", "Femenino", "Masculino", "Masculino", "Femenino")
Salud <- c(0, 1, 1, 0, 1) #donde 0 es sano y 1 enfermo
Opinion <- c("Mucho", "Poco", "Mucho", "Bastante", "Mucho")
Fuma <- c("1", "2", "1", "2", "2")
Opin_Pacientes <- data.frame(Ident, Genero, Salud, Opinion, Fuma)
Opin_Pacientes
```

```
##   Ident    Género Salud  Opinión Fuma
## 1     1 Masculino     0    Mucho    1
## 2     2  Femenino     1     Poco    2
## 3     3 Masculino     1    Mucho    1
## 4     4 Masculino     0 Bastante    2
## 5     5  Femenino     1    Mucho    2
```

En primer lugar, queremos cambiar los valores de la variable Salud, ya que incluir un 0 puede ocasionar problemas de análisis posteriores. Cambiamos los valores a 1 y 2:

```
Opin_Pacientes[Opin_Pacientes$Salud==1] <- 2
Opin_Pacientes[Opin_Pacientes$Salud==0] <- 1
Opin_Pacientes
```

```
##   Ident    Género Salud Opinión Fuma
## 1     2 Masculino     1       Mucho    1
## 2     2  Femenino     2        Poco    2
## 3     3 Masculino     2       Mucho    1
## 4     4 Masculino     1    Bastante    2
## 5     5  Femenino     2       Mucho    2
```

En segundo lugar, nos gustaría recodificar la variable **Salud** del conjunto de datos inicial (con valores 1, 0) a un factor con **Sano** y **Enfermo**:

```
```{r}
reco1 <- Opin_Pacientes
reco1$Salud <- factor(reco1$Salud)
levels(reco1$Salud)[1] <- "Sano"
levels(reco1$Salud)[2] <- "Enfermo"
reco1
```
```

```
##   Ident    Género   Salud Opinión Fuma
## 1     1 Masculino Sano          Mucho    1
## 2     2  Femenino Enfermo        Poco    2
## 3     3 Masculino Enfermo       Mucho    1
```

```
## 4     4 Masculino Sano       Bastante    2
## 5     5  Femenino Enfermo       Mucho    2
```

Por otro lado, queremos cambiar la variable Género y hacerla numérica (valores Masculino = 0 y Femenino = 1).

```
Opin_Pacientes$Género <- ifelse(Opin_Pacientes$Género == "Masculino",0,1)
Opin_Pacientes

##   Ident Género Salud Opinión Fuma
## 1     1      0     1       Mucho    1
## 2     2      1     2        Poco    2
## 3     3      0     2       Mucho    1
## 4     4      0     1    Bastante    2
## 5     5      1     2       Mucho    2
```

También recodificar intervalos. Por ejemplo, podemos hacer que la edad cambie en función de un intervalo.

```
datos_reco <- data.frame(edad=c(22,21,34,40), sexo=c("H","M","H","H"))
datos_reco$edad_cod <- ifelse(datos_reco$edad >= 15 & datos_reco$edad <= 25,
"joven",
      ifelse(datos_reco$edad >= 26 & datos_reco$edad <= 65, "adulto",
"otro"))
datos_reco
##    edad sexo edad_cod
## 1    22   H   joven
## 2    21   M   joven
## 3    34   H  adulto
## 4    40   H  adulto
```

6. Tablas de frecuencias con R

Otra de las herramientas relevantes para resumir los datos son las **tablas de frecuencias** (absolutas y relativas) y también las **tablas de contingencia** (o tablas de frecuencia cruzada). Para generarlas, podemos usar los comandos de la tabla siguiente:

Comando	**Explicación de la tabla**
table(data)	Tabla de frecuencias absolutas
prop.table(data)	Tabla de frecuencias relativas
table(data1, data2)	Tabla de frecuencias absolutas cruzadas
prop.table(data1,data2)	Tabla de frecuencias relativas cruzadas

Ejemplo 5:

Para poder entender el funcionamiento de estos comandos, pondremos unos cuantos ejemplos:

```r
#definición de las variables en vectores
id=c(100,110,120,130,140,150,160,170,180,190)
edad=c(18,19,NA,18,24,27,22,25,22,15)
gen=c(2,1,2,2,1,2,1,1,2,1)
peso=c(65,58,56,61,84,99,50,64,87,87)
altura=c(161,170,174,165,150,171,181,170,184,190)
niv_col=c(1,2,3,1,3,2,3,1,2,3)

#tablas de frecuencias absolutas
table(edad)
```

```
## edad
## 15 18 19 22 24 25 27
##  1  2  1  2  1  1  1
```

```r
table(niv_col)
```

```
## niv_col
## 1 2 3
## 3 3 4
```

El comando *table()* no muestra los valores faltantes por defecto. Así que debemos usar el siguiente código:

```r
table(edad, useNA="ifany") #utilizamos "useNA" para que aparezca NA
```

```
## edad
##   15   18   19   22   24   25   27 <NA>
##    1    2    1    2    1    1    1    1
```

```r
#tabla de frecuencias relativas
prop.table(edad)
```

```
##  [1] NA NA NA NA NA NA NA NA NA NA
```

```r
prop.table(altura)
```

```
##  [1] 0.09382284 0.09906760 0.10139860 0.09615385 0.08741259 0.09965035
##  [7] 0.10547786 0.09906760 0.10722611 0.11072261
```

```r
#tabla de frecuencias absolutas cruzadas
table(niv_col,gen)
```

```
##        gen
## niv_col 1 2
##       1 1 2
##       2 1 2
##       3 3 1
```

```r
#tabla de frecuencias relativas cruzadas
prop.table(table(niv_col,gen))
```

```
##         gen
## niv_col   1   2
##       1 0.1 0.2
##       2 0.1 0.2
##       3 0.3 0.1
```

En el caso de tener tablas de contingencia (es decir, una tabla de frecuencias absolutas cruzadas) con variables categóricas, podemos usar la **prueba de la ji al cuadrado** con el comando *chisq.test()* para poder comprobar la relación o independencia de los datos. Para usar la prueba podemos utilizar este código:

```
data_cont <- table(niv_col,gen) #creamos la tabla de contingencia
chisq.test(data_cont) #aplicamos la prueba de la ji al cuadrado
```

```
## Warning in chisq.test(data_cont): Chi-squared approximation may be
incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  data_cont
## X-squared = 1.6667, df = 2, p-value = 0.4346
```

Si miramos el **p valor (*p-value*)** en el resultado de la prueba podremos determinar si las variables nivel de colesterol y sexo están relacionadas o no. En este caso, el p-valor es mayor que 0.05 (p = 0.43), la prueba no es significativa y por lo tanto las variables son independientes.

En caso de que al realizar la **prueba de la ji al cuadrado** observemos un *warning*, puede ser debido a que los valores esperados sean menores que cinco, o la proporción supera al 20 % de las celdas, con valores observados inferiores a cinco. Cuando esto sucede, es recomendable utilizar un test de independencia denominado Fisher con el comando *fisher.test()*.

```
fisher.test(data_cont)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  data_cont
## p-value = 0.5714
## alternative hypothesis: two.sided
```

7. Gráficos básicos en R

Para generar gráficos básicos con **R** debemos conocer los comandos específicos que se utilizan. A continuación, os mostramos una tabla con los comandos para generar los gráficos básicos más usados en **R** y que ya vimos en el **LAB1** de una manera sencilla:

Comando	Tipo de gráfico
stem(data)	Diagrama de tallo y de hojas
hist(data)	Histograma
boxplot()	Gráfico de caja
plot()	Gráfico de puntos
pairs()	Gráfico de correlación por pares
barplot()	Gráfico de barras
pie()	Gráfico de sector
persp()	Gráfico tridimensional

7.1. Funciones y opciones de los gráficos

R dispone de múltiples funciones diseñadas para la representación gráfica de datos. Las podemos dividir en funciones gráficas de «**alto nivel**» que generan gráficos completos y de «**bajo nivel**» que solo añaden elementos a un gráfico existente.

7.1.1. Funciones para gráficos de «alto nivel»

- **xlim, ylim**: controlan, respectivamente, la extensión de los ejes X e Y. Por ejemplo, si escribimos *xlim=c(0,10)*, indica que el eje X se extiende de 0 a 10 y si escribimos *ylim=c(-5,5)*, indica que el eje Y va de -5 a 5. Si no se incluyen estos valores, **R** los ajusta por defecto, de modo que se incluyan todos los valores disponibles en el data frame.
- **xlab, ylab**: sirven para poner las etiquetas en los ejes X e Y, respectivamente.
- **main**: indica el título del gráfico.
- **sub**: permite especificar un subtítulo.

7.1.2. Funciones para gráficos de «bajo nivel»

Permiten añadir líneas, puntos, etiquetas, etc. a un gráfico ya existente. Son de gran utilidad para completar un gráfico. Entre estas funciones cabe destacar las siguientes:

- **lines()**: permite añadir líneas (uniendo puntos concretos) a una gráfica ya existente.
- **abline()**: se usa para añadir líneas horizontales, verticales u oblicuas, indicando pendiente y ordenada.
- **points()**: se usa para añadir puntos al gráfico.
- **legend()**: se usa para añadir una leyenda.
- **text()**: se usa texto en las posiciones que se indiquen.
- **grid()**: se usa para añadir una malla de fondo.
- **title()**: se usa para añadir un título o subtítulo al gráfico.

Los argumentos más usados en cualquier tipo de función son los siguientes:

pch	Indica la forma en que se dibujan los puntos (círculo, cuadrado, estrella, etc.)
lty	Indica la forma en que se dibujan las líneas (continuas, a trozos, etc.)

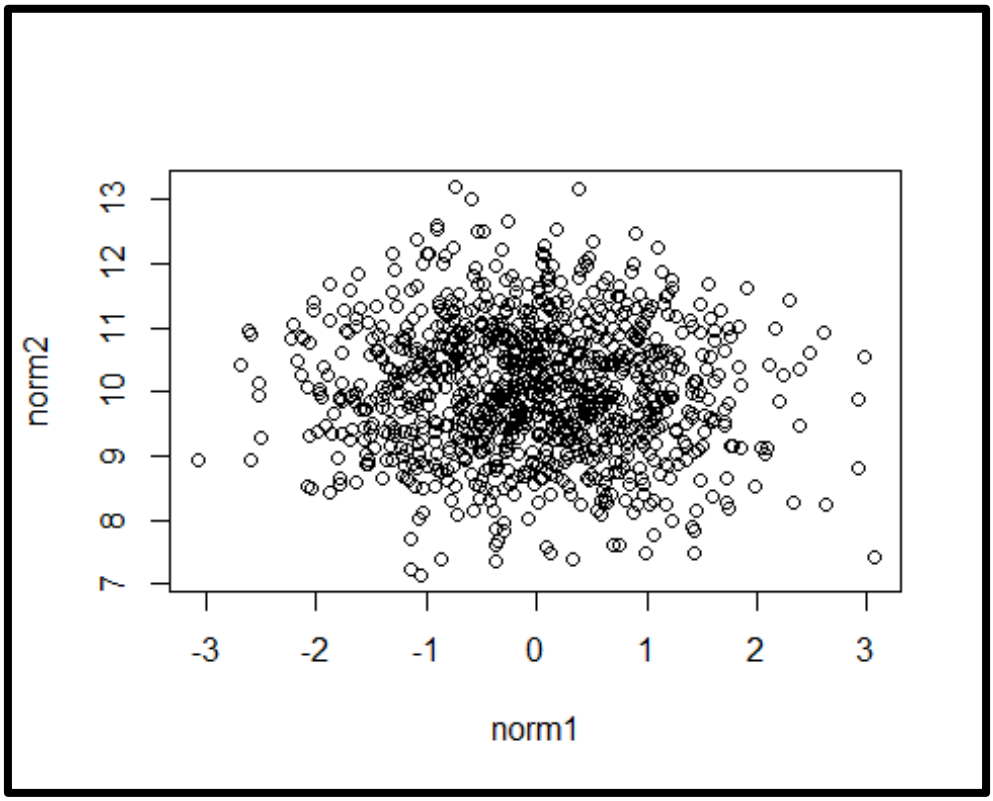
pch	Indica la forma en que se dibujan los puntos (círculo, cuadrado, estrella, etc.)
lwd	Ancho de las líneas
col	Color usado para el gráfico
font	Fuente usada en el texto
las	Cambia el estilo de las etiquetas de los ejes

Si queremos poner colores a nuestros gráficos y que tengan mejor visualización, podemos usar la Lista de colores de R.

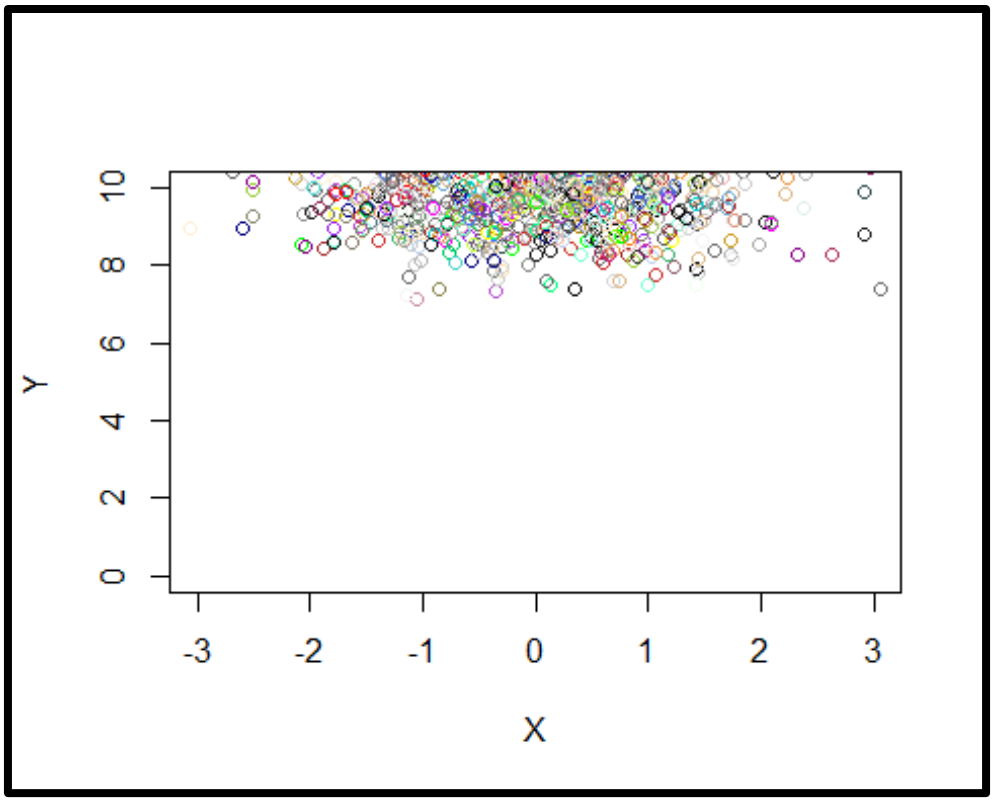
Ejemplo 6:

Vamos a crear un gráfico a partir de dos distribuciones normales y a usar, a modo de ejemplo, algunas de las opciones detalladas en la tabla anterior:

```
#generamos distribuciones con datos normales
set.seed(999)
norm1 <- rnorm(1000, mean=0,sd=1) #normal de media 0 y desviación 1
norm2 <- rnorm(1000, mean=10,sd=1) #normal de media 10 y desviación 1
plot(x=norm1, y=norm2) #creamos un gráfico de puntos solo con los datos
```



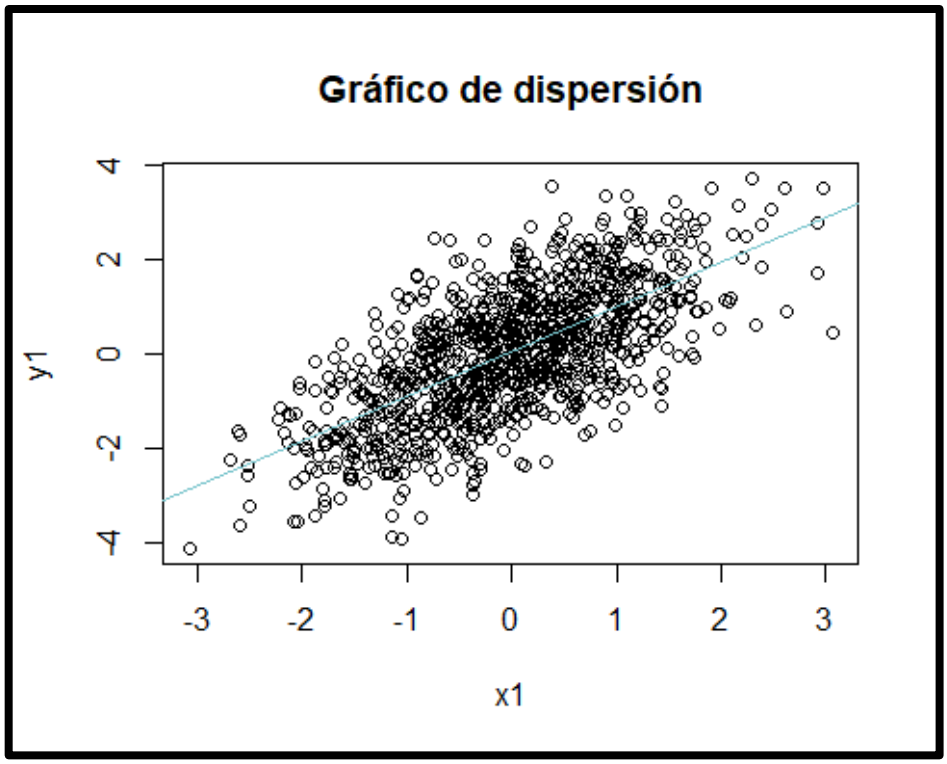

```
plot(x=norm1, y=norm2, xlab="X", ylab="Y", xlim=c(-3,3), ylim=c(0,10),
col=sample(colors(), 100)) #usamos las opciones para las etiquetas de los
ejes, adaptamos sus límites y ponemos colores diferentes para los puntos
```



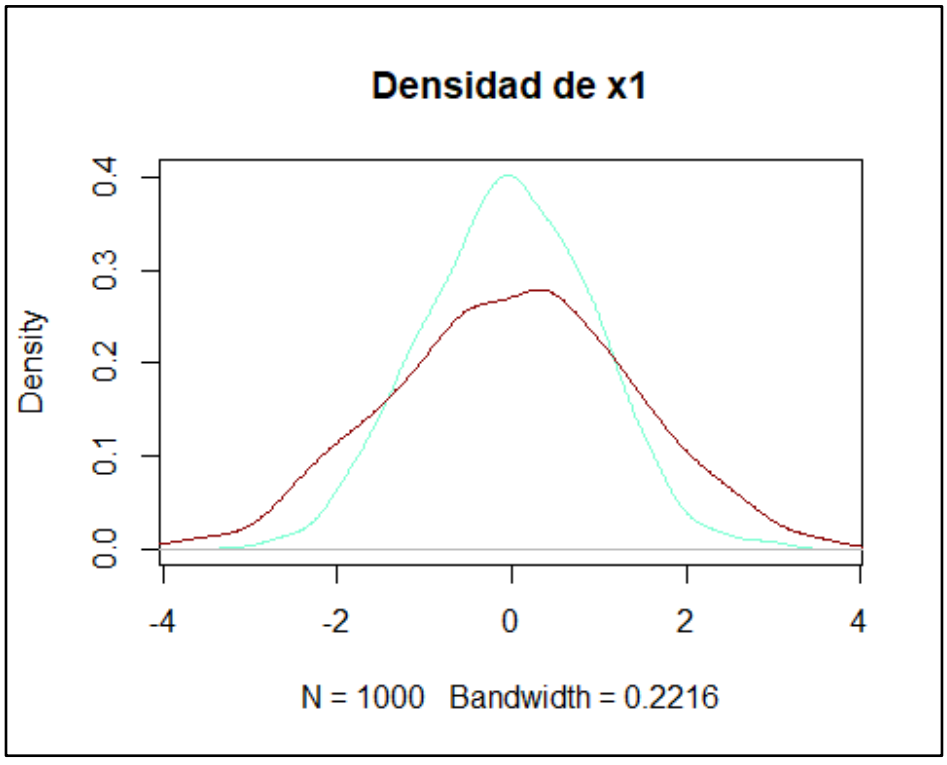
Ejemplo 7:

Otro ejemplo con datos simulados para ver cómo aplicamos la función *plot()* en gráficos estadísticos:

```
set.seed(999)
x1 <- rnorm(1000) #simulamos una variable x1
y1 <- x1 + rnorm(1000) #simulamos una variable y1
plot(x1, y1, main="Gráfico de dispersión") #creamos el gráfico de dispersión
de las variables
abline(lm(y1~x1), col="cadetblue3") #creamos la recta de regresión de color
verde
```



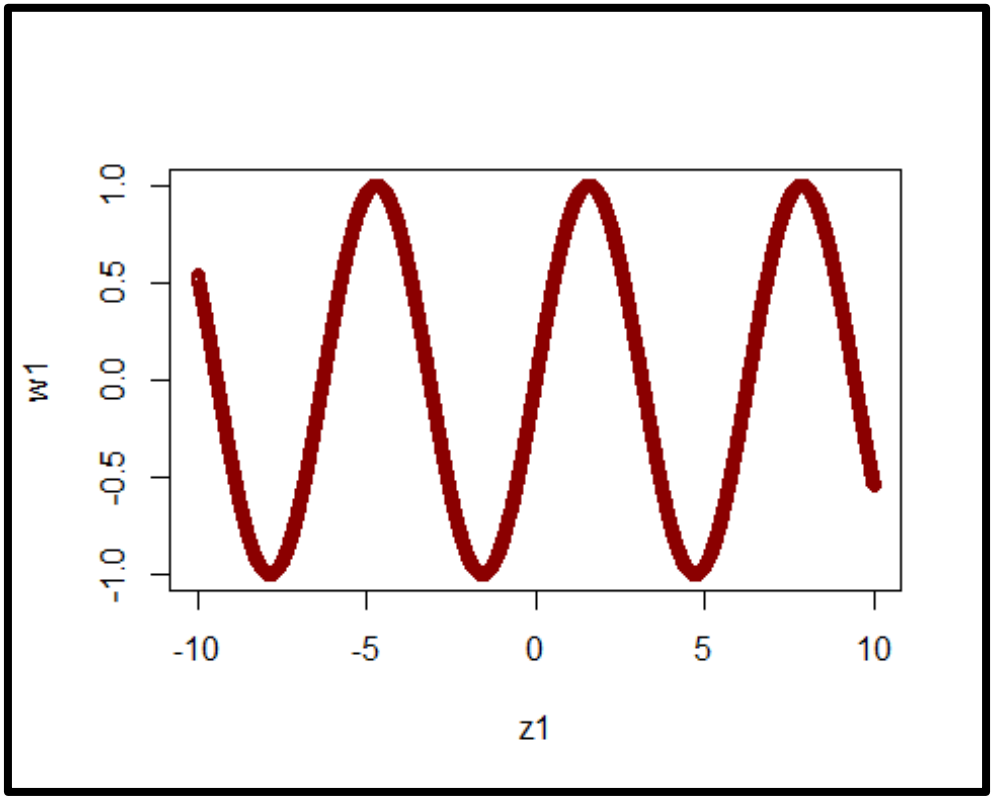

```r
plot(density(x1), main="Densidad de x1", col="aquamarine") #generamos el gráfico de densidad para ver cómo es la distribución
lines(density(y1), col="darkred") #generamos un gráfico con las densidades de las dos variables.
```



Ejemplo 8:

También podemos generar gráficos con el comando *plot()* sobre funciones específicas de **R**. Por ejemplo, de una función trigonométrica como es sinus.

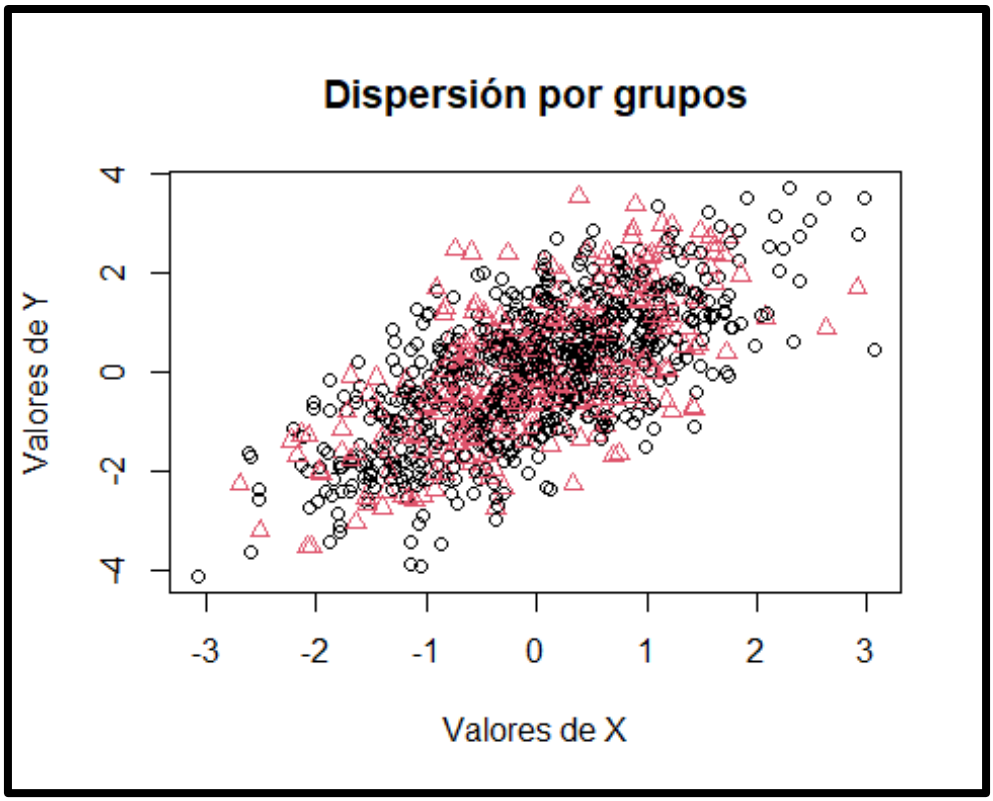
```
z1 <- seq(-10, 10, 0.01) #generamos datos con una secuencia
w1 <- sin(z1) #buscamos el sinus de nuestra variable
plot(z1,w1, col="darkred") #creamos el gráfico
```



Ejemplo 9:

Por último, como ejemplo, podemos usar el comando *plot()* para realizar gráficos de dispersión por grupos.

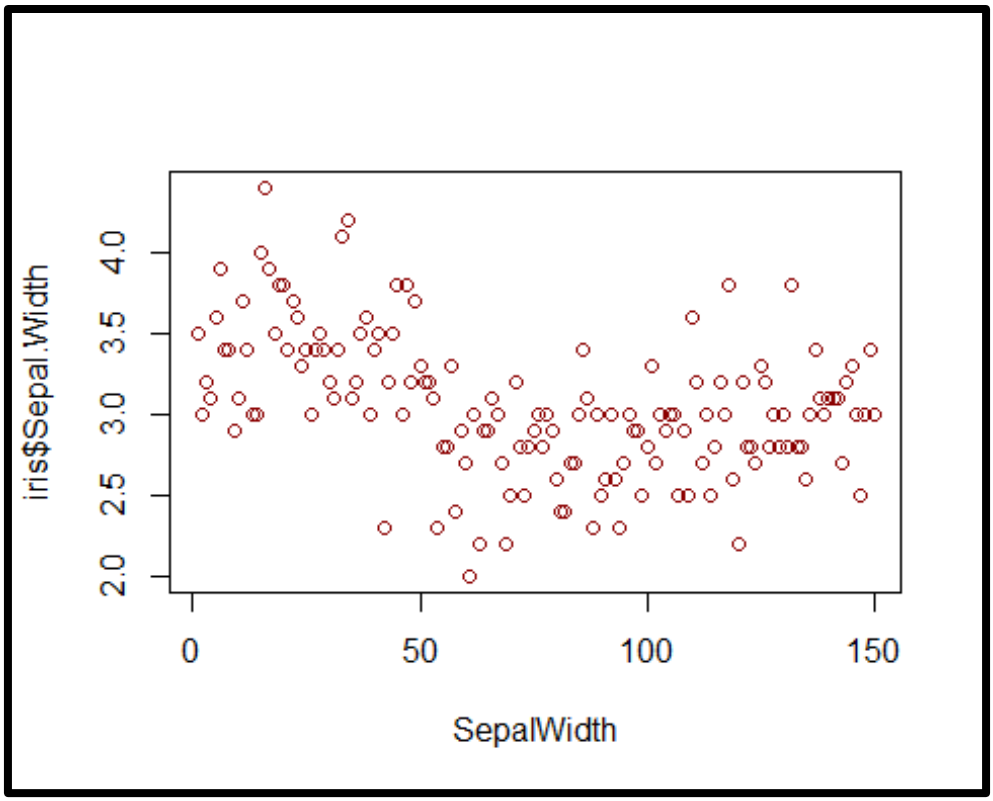
```
grupos <- rbinom(10000, 1, 0.3) + 1  #simular unos datos para dos grupos
plot(x1, y1, main = "Dispersión por grupos", xlab = "Valores de X",ylab =
"Valores de Y", col = grupos, pch = grupos)#gráficos de dispersión; color y
forma en función de los grupos generados
```



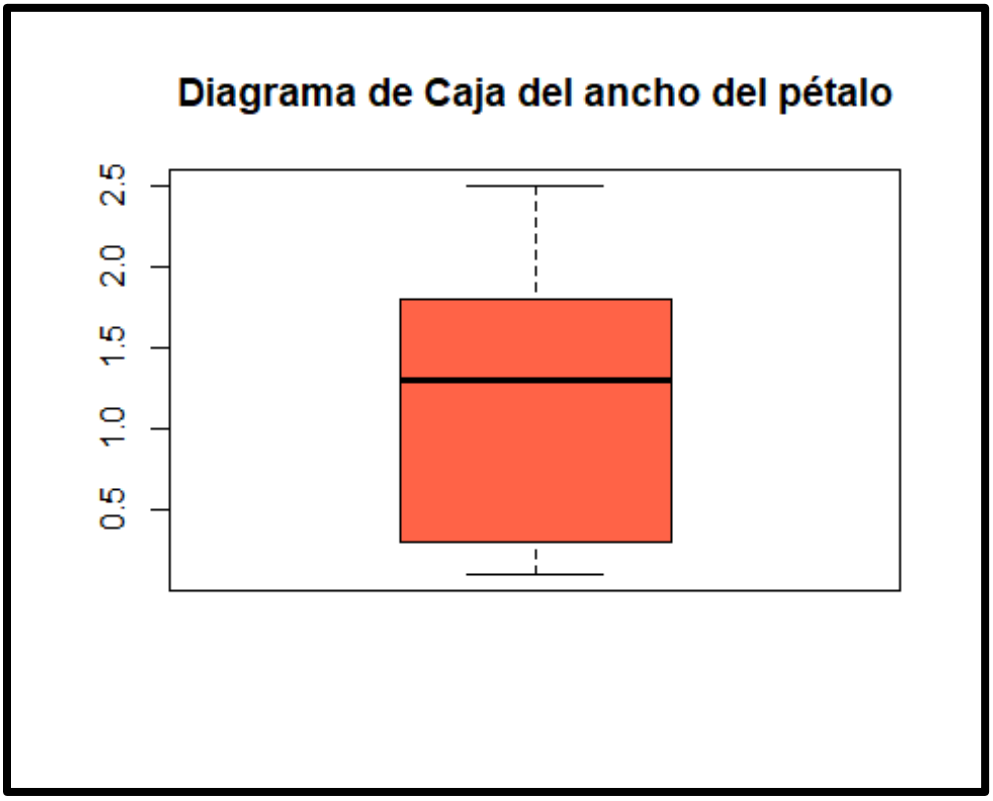
Ejemplo 10:

Vamos a escoger el conjunto de datos **Iris** del paquete **MASS** de **R** para generar más gráficos diferentes con diferentes opciones:

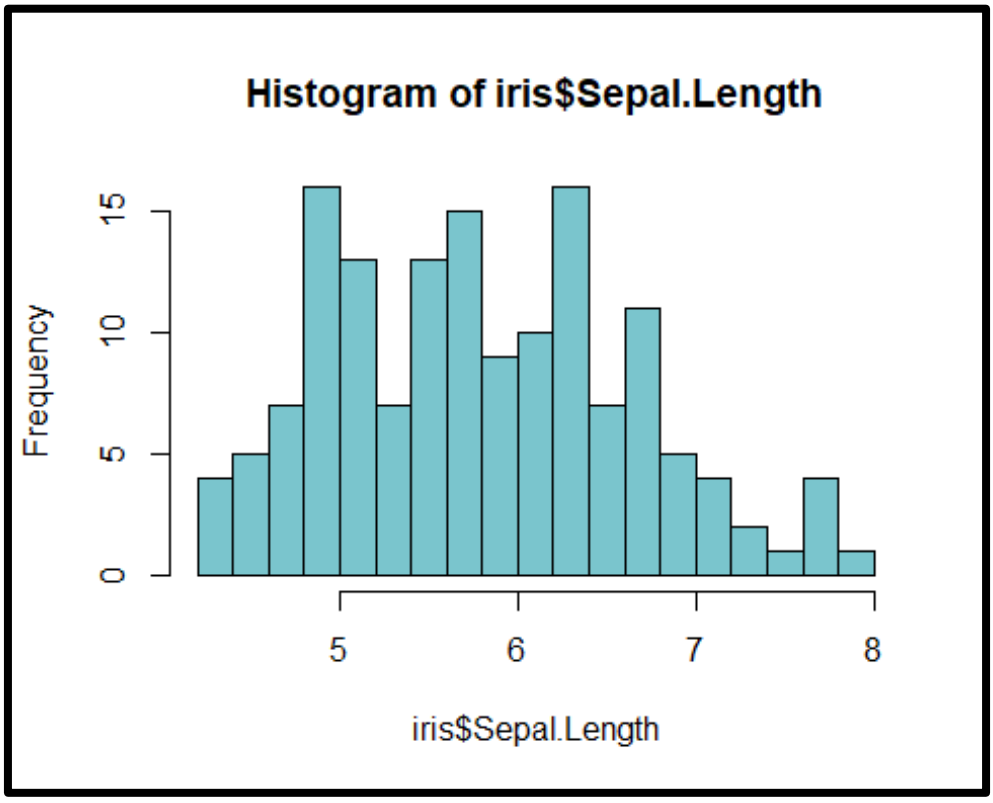
```
library(MASS) #
data("iris") #cargamos los datos
plot (iris$Sepal.Width, col="dark red", xlab="SepalWidth") #gráfico de
dispersión de color rojo intenso con etiqueta del eje X
```




```r
boxplot(iris$Petal.Width, col="tomato", main="Diagrama de Caja del ancho del pétalo") #diagrama de caja de color rojo con un título
```

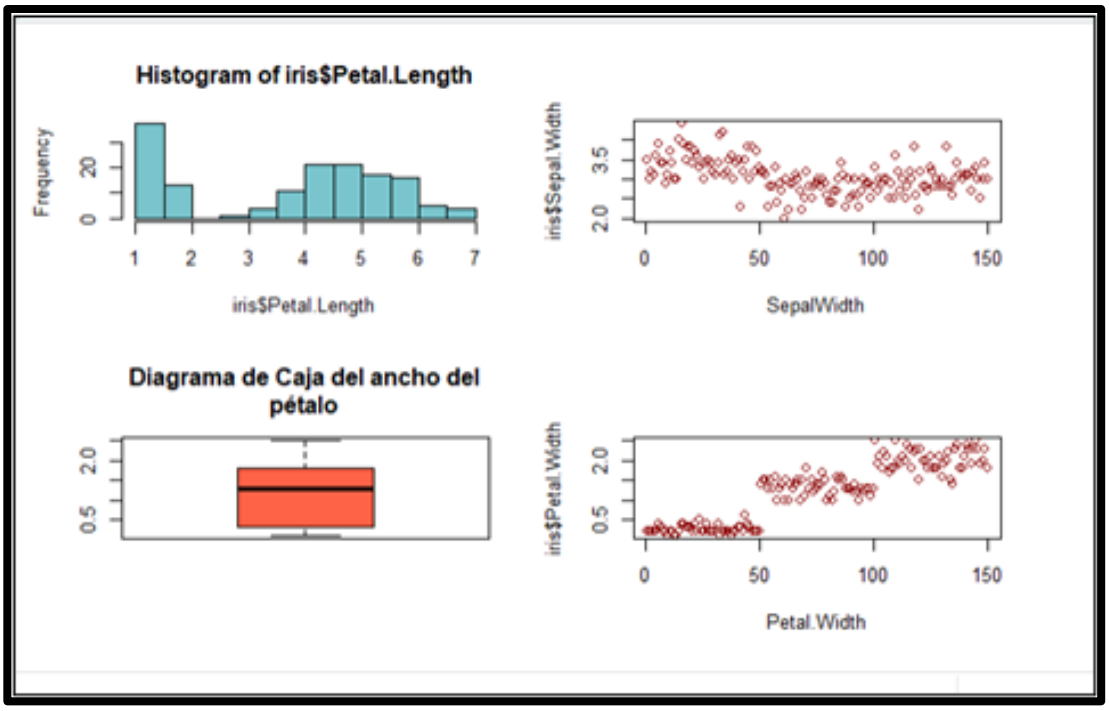


```r
hist(iris$Sepal.Length, breaks=20, col="cadetblue3") #histograma con 20 intervalos de clase y color azul
```



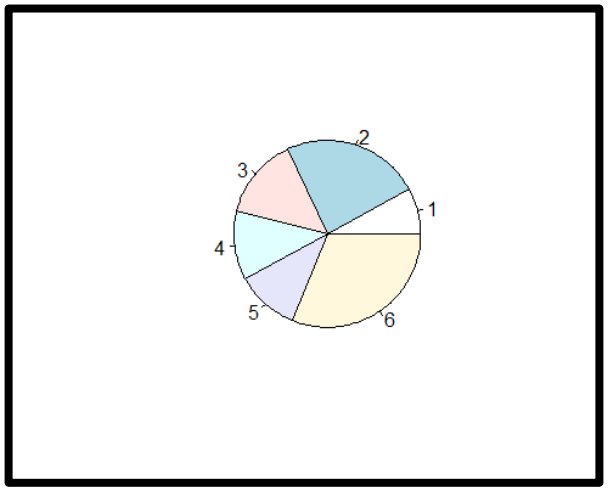
Si, para nuestro trabajo, queremos agrupar gráficos e imprimirlos todos juntos, se debe generar una matriz cuadrada con el número de filas y columnas específica para el número de gráficos que queremos y usar el comando *par()*.

```r
par(mfrow=c(2,2)) #matriz 2x2
hist(iris$Petal.Length, breaks=20, col="cadetblue3") #histograma
plot (iris$Sepal.Width, col="dark red", xlab="SepalWidth") #gráfico de dispersión
boxplot(iris$Petal.Width, col="tomato", main="Diagrama de Caja del ancho del pétalo") #diagrama de caja
plot (iris$Sepal.Width, col="dark red", xlab="SepalWidth") #gráfico de dispersión de color rojo oscuro
```

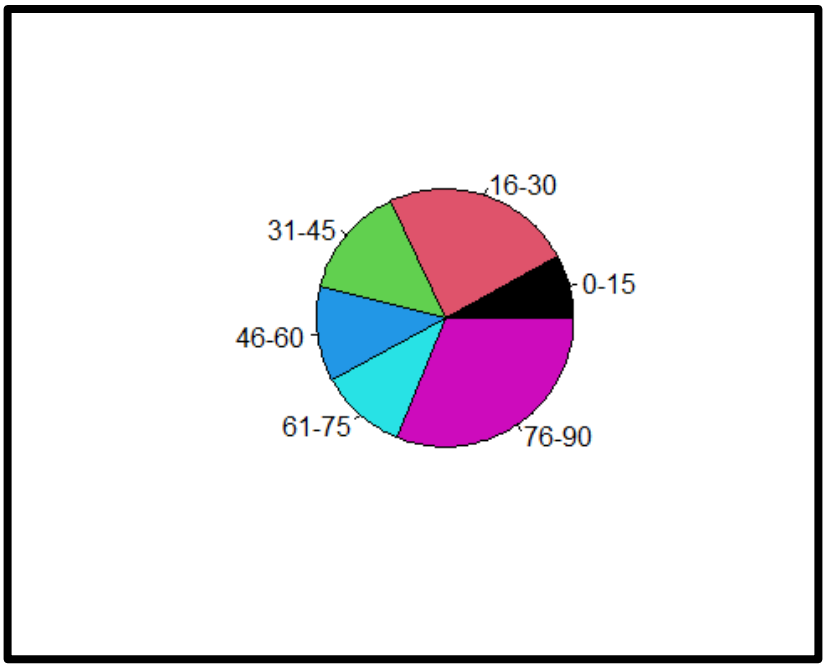


Para hacer un gráfico de sector, un *pie chart*, usaremos un vector de datos inventado:

```
data_pie <- c(8,24,14,12,11,31) #datos inventados
pie(data_pie) #creamos un diagrama de sector
```




```
pie(data_pie, col=1:6, labels = c("0-15", "16-30", "31-45", "46-60", "61-75",
"76-90")) #diagrama de sector y ponemos etiquetas a cada sector
```

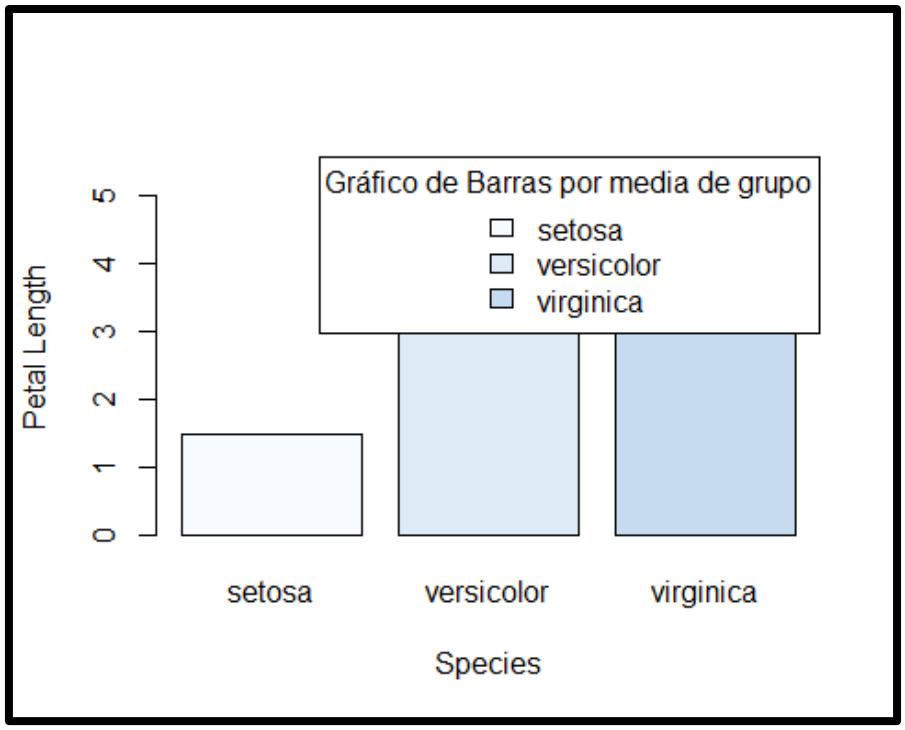


Si queremos poner una leyenda a un gráfico, usaremos la opción *legend()* justo después de crear el gráfico de la siguiente forma:

```
data(iris)
data_agreg = aggregate(iris[,1:4], by = list(iris$Species), FUN = mean)
data_agreg
```

```
##      Group.1 Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     setosa        5.006       3.428        1.462       0.246
## 2 versicolor        5.936       2.770        4.260       1.326
## 3  virginica        6.588       2.974        5.552       2.026
```

```
barplot(Petal.Length~Group.1, data = data_agreg,xlab = c('Species'), ylab =
c('Petal Length'), col=blues9)
legend(x = "topright", legend = c("setosa", "versicolor", "virginica"), fill
= blues9,
       title = "Gráfico de Barras por media de grupo")
```



8. Paquete ggplot2 para la creación de gráficos

8.1. El paquete ggplot2

La librería **ggplot2** permite personalizar los gráficos con temas. Forma parte del conjunto de librerías llamado **tidyverse**.

Los elementos necesarios para representar un gráfico con **ggplot2** son los siguientes:

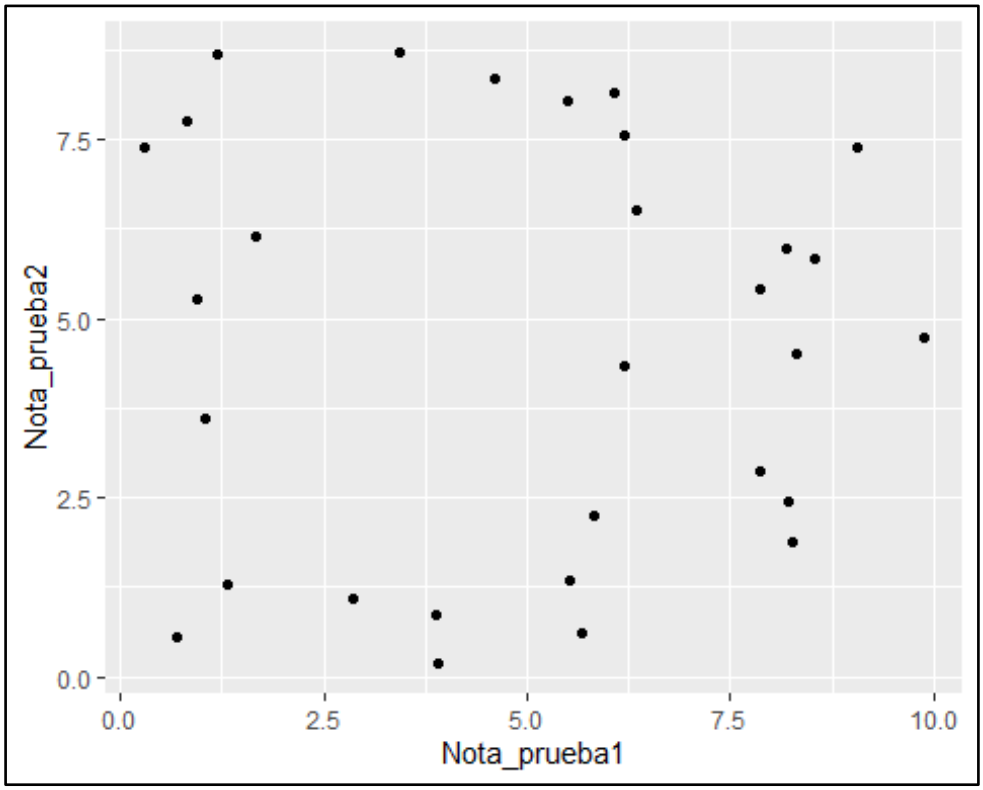
- Un **data frame** que contiene los datos que se quieren visualizar.
- Los **aesthetics**, es decir, una lista de relaciones entre las variables del fichero de datos y determinados aspectos del gráfico como son coordenadas, formas o colores.
- Los **geoms**, que especifican los elementos geométricos del gráfico.

Ejemplo 11:

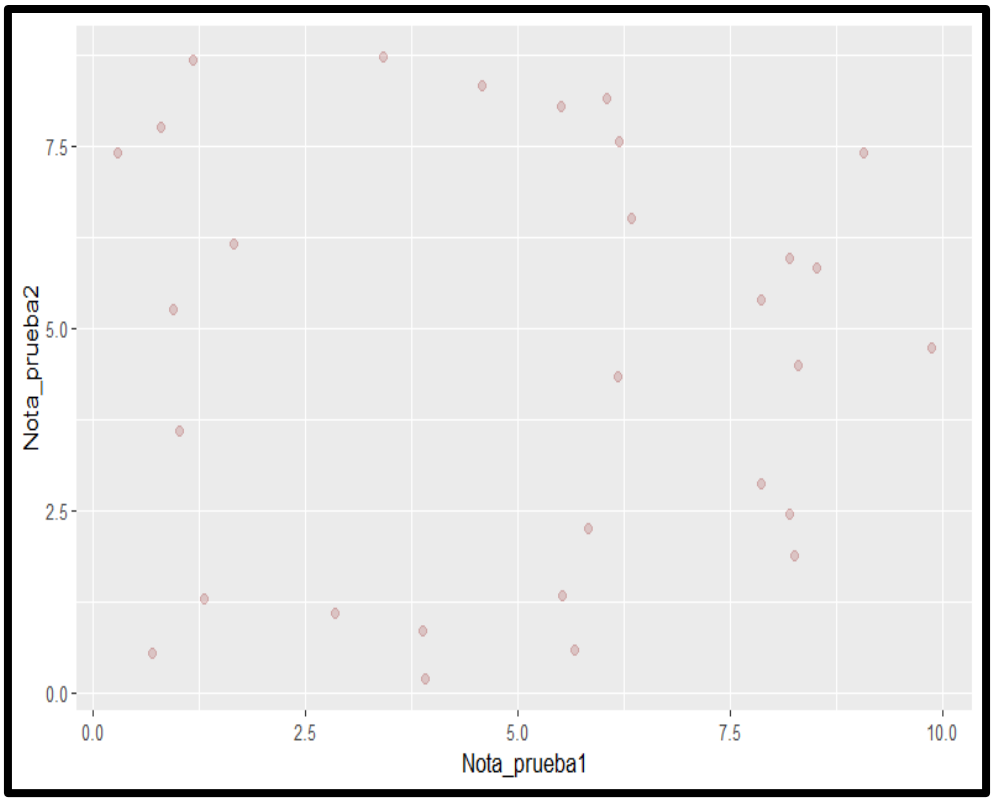
Para poder crear un gráfico con la función *ggplot()* debemos cargar el paquete **tidyverse**, escoger unos datos (deben pertenecer a un data frame) y empezar a añadir las capas.

```
#creamos un dataset con las notas de los internos de diferentes
especialidades de un hospital
set.seed(999)
Id_Medico<- c(seq(1:30))
Nota_prueba1 <- c(round(runif(30, min=0, max=10),2)) #nota 1a prueba
Nota_prueba2 <- c(round(runif(30, min=0, max=10),2)) #nota 2a prueba
Especialidad <-
factor(c(1,1,2,3,2,3,1,2,3,1,1,2,3,1,2,3,1,2,3,3,2,1,3,1,1,1,1,2,3,2))
```

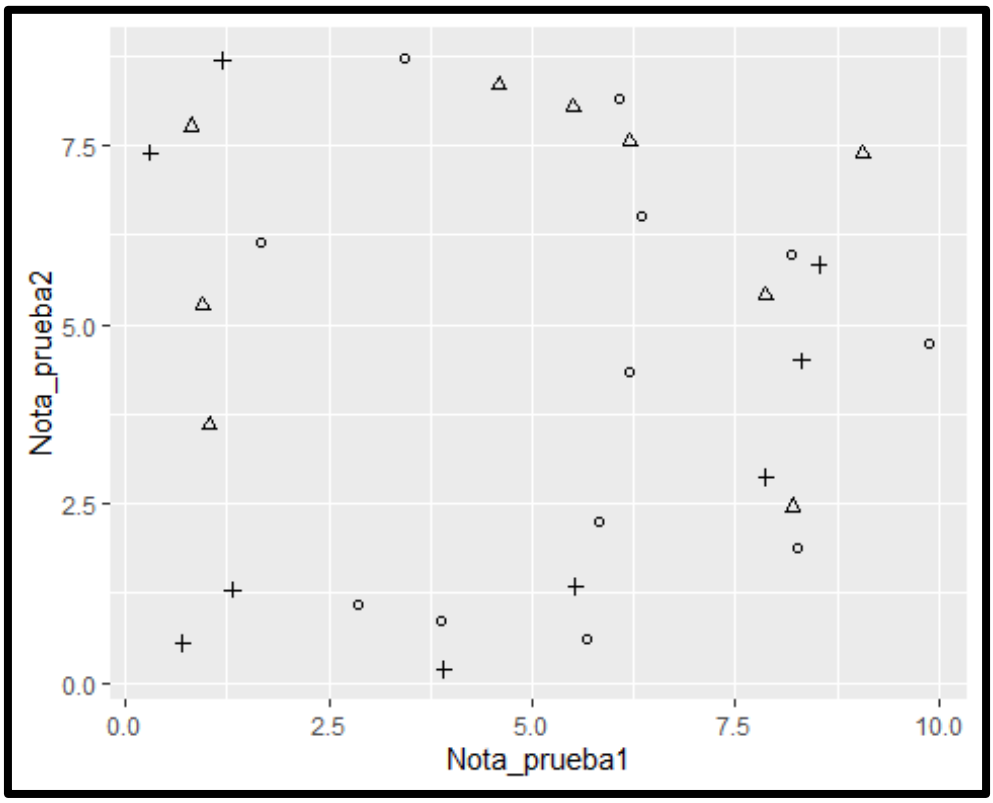
```
Data_Not <-data.frame(Id_Médico, Nota_prueba1, Nota_prueba2)

library(tidyverse)

ggplot(data= Data_Not) + geom_point(aes(x = Nota_prueba1, y = Nota_prueba2))
#creamos un primer gráfico añadiendo los datos
```




```
ggplot(data = Data_Not) + geom_point(aes(x = Nota_prueba1, y =
Nota_prueba2),col = 'yellow', size = 2, alpha = 1/6) #cambio de color, tamaño
y gradiente de cada punto
```




```
ggplot(data = Data_Not) + geom_point(aes(x = Nota_prueba1, y = Nota_prueba2),
shape=Especialidad) #cada punto se detalla con un tipo de las especialidades
```

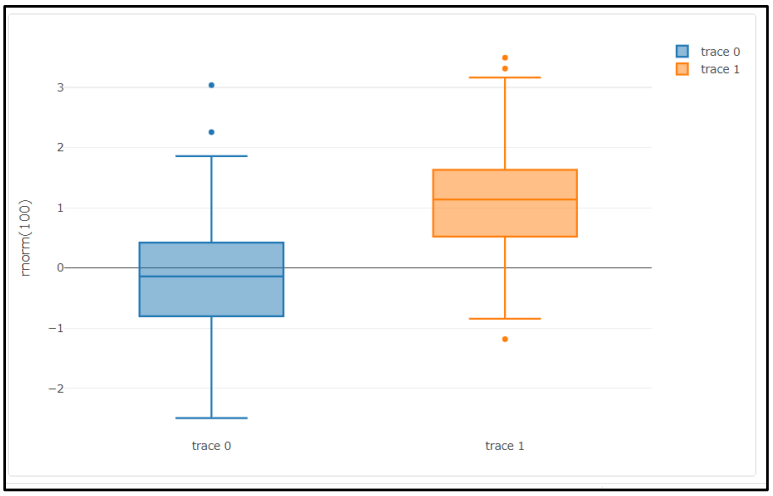


En diferentes ocasiones, se necesita implementar gráficos dinámicos en nuestros informes. En particular, si queremos que nuestros gráficos se publiquen en internet y los usuarios puedan interactuar.

Como introducción a los gráficos interactivos y dinámicos, introduciremos el paquete **Plotly** de **R**. Este paquete es gratuito y de código abierto.

Vamos a crear un ejemplo de cómo usar este paquete con R. Para ello, deberéis tener instalado este paquete:

```
install.packages("plotly")
library(plotly)
graf1 <- plot_ly(y = ~rnorm(100), type = "box")
graf2 <- graf1 %>% add_trace(y = ~rnorm(100, 1))
graf2
```



La imagen anterior reproduce el gráfico, pero no podemos observar su interactividad.

Para más información y ejemplos con código, podéis ir a la página del paquete Plotly con R y podréis practicar con los gráficos dinámicos.

¡Hay mucha variedad de ejemplos con código para empezar a usar esta herramienta!

9. Guardar y exportar gráficos en R

Podemos guardar los gráficos en diferentes formatos. Para llevarlo a cabo, deberemos especificar, antes de empezar a compilar el código, el formato con el que queremos guardar el gráfico:

Comando	Extensión
pdf («migrafico.pdf»)	Se guarda como un **pdf**.
win.metafile(«migrafico.wmf»)	Se guarda como un Windows Metafile **wmf**.
jpeg(«migrafico.jpg»)	Se guarda como imagen en **jpg**.
bmp(«migrafico.bmp»)	Se guarda como imagen en **bmp**.
png(«migrafico.png»)	Se guarda como imagen en **png**.
postscript(«migrafico.ps»)	Se guarda como imagen en **ps**.

Ejemplo 12:

Si queremos guardar un gráfico en un documento PDF (.pdf), por ejemplo, deberemos usar el código que hay a continuación:

```
pdf("migrafico.pdf") #abrimos el dispositivo gráfico
plot(iris$Petal.Width)
dev.off #cerramos el dispositivo gráfico

file.show("migrafico.pdf") #podemos visualizar el archivo que hemos generado
```

Con todos los formatos de exportación (resumidos en la tabla anterior) procederemos de la misma forma. Es muy importante cerrar el dispositivo gráfico con el comando *dev.off()*.

Otra manera sencilla de proceder es usando el menú que encontraremos en la ventana inferior derecha de **RStudio *Plot > Export > Save Images***, y lo podremos guardar como imagen, como documento .pdf o en el portapapeles.

10. Regresión lineal con R

10.1. Regresión lineal simple

Como ejemplo de regresión en este laboratorio de programación, nos centraremos en la regresión lineal simple.

El objetivo del análisis de la regresión lineal simple es determinar una función matemática sencilla que describa el comportamiento de una variable dados los valores de otra u otras variables. Buscamos la función que mejor explique la relación entre la variable dependiente y las independientes.

Para explicar cómo hacer una regresión lineal en **R** usaremos una serie de pasos que numeramos a continuación:

- **Paso 1**: acceder a los datos mediante la importación de un fichero, datos de un paquete de **R** o de un conjunto de datos ya activo en vuestro entorno.
- **Paso 2**: hacer un resumen estadístico de las variables que queremos analizar. Podemos usar el comando *summary()* o *fivenum()*.

- **Paso 3**: representar las variables con un diagrama de dispersión de dos en dos. Podemos usar el comando *pairs()*.
- **Paso 4**: calcular el coeficiente de correlación de Pearson, que es una medida de dependencia lineal entre dos variables aleatorias cuantitativas. A diferencia de la covarianza, la correlación de Pearson es independiente de la escala de medida de las variables. De esta forma, podemos comprobar si hay relación entre las variables que queremos analizar.
- **Paso 5**: generar el modelo de regresión si encontramos relación entre dos variables. Probaremos con el modelo más sencillo, que es ver si se ajusta a una recta definida con la expresión **Y=mx+n**. El comando básico es **lm** (*linear models*). El primer argumento de este comando es una fórmula y ~ x en la que se especifica cuál es la variable respuesta o dependiente (y) y cuál es la variable represora o independiente (x). Mediante el comando *summary* podemos mirar los principales resultados.
- **Paso 6**: crear el gráfico de nube de puntos de nuestro modelo y la **recta de regresión** con mínimos cuadrados ajustada a la nube de puntos.
- **Paso 7**: comprobar los resultados. Nos interesan los **parámetros estimados del modelo** y el valor del ajuste de la recta con el **coeficiente de determinación** o «R-squared». También podemos mirar el **valor p o *p-value*** que nos indica, en inferencia, la probabilidad de obtener un valor igual o más extremo que el observado, suponiendo que se cumple una hipótesis nula. Una serie de datos nos dice que es estadísticamente significativa si su valor p es menor o igual que 0.05
- **Paso 8**: validar el modelo. El **diagnóstico del modelo de regresión** es un conjunto de procedimientos que buscan evaluar la validez de un modelo previamente ajustado. Esta evaluación puede ser una exploración de los supuestos estadísticos del modelo.

Si un modelo de regresión ajustado representa adecuadamente los datos, sus residuos deberán:

- Tener varianza constante (homogeneidad de la varianza).
- Estar aproximadamente distribuidos de forma normal y ser independientes el uno del otro.

Para hacer esta validación y comprobar si los errores siguen una distribución normal, podemos generar, por ejemplo, un histograma, un diagrama de tallo y hojas, o un Q-Q Plot de los residuos.

- **Paso 9**: si se requiere, podemos buscar una predicción puntual de nuestro modelo; usaremos el comando *predict()* con los nuevos datos.
- **Paso 10**: si se requiere, podemos buscar los intervalos de confianza para nuestros parámetros; se buscarán con el comando *confint()*.

Ejemplo 13:

Creamos un dataset **«Mujeres»** con datos inventados para veinticinco mujeres con un identificador, el peso, la cantidad de grasa en sangre y la edad.

- **Paso 1**: incorporamos los datos en nuestro entorno.

```
library(knitr)
id <- seq(1:25)
peso <- c(63, 72, 79, 75, 27, 84, 73,65, 70, 76, 69, 89,65,57,59, 69, 60, 79,
75, 82, 59, 67, 85, 55,63)
grasa <- c(288,385,402, 365, 209, 354, 190, 405, 263, 451,
302,290,346,254,395,434,220,374,308,220,311,181,274,303,244 )
edad <- c(28,36,57, 44, 24,46,20, 52, 30, 57,
25,31,52,23,60,48,34,51,50,34,46,23,37,40,30)
Mujeres <- data.frame(id, peso, grasa, edad)
head(Mujeres)
```

```
##   id peso grasa edad
## 1  1   63   288   28
## 2  2   72   385   36
## 3  3   79   402   57
## 4  4   75   365   44
## 5  5   27   209   24
## 6  6   84   354   46
```

```
names(Mujeres)
```

```
## [1] "id"    "peso"  "grasa" "edad"
```

- **Paso 2**: hacemos un resumen de las variables del conjunto de datos.

```
str(Mujeres)
```

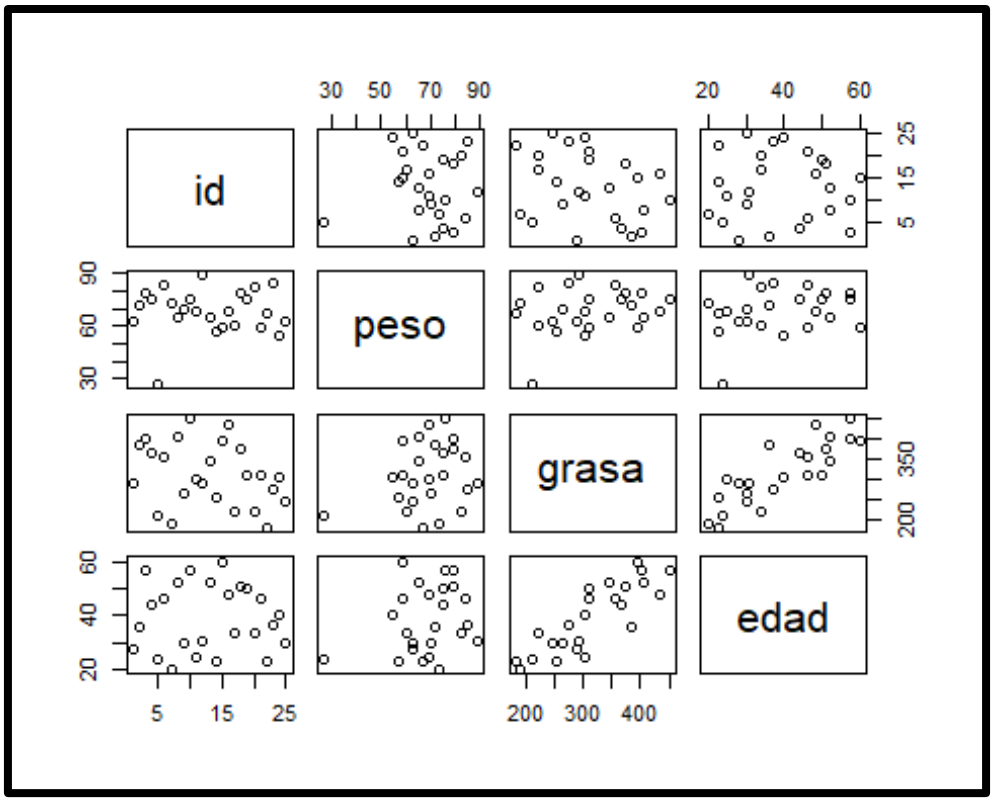
```
## 'data.frame':    25 obs. of  4 variables:
##  $ id: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ peso: núm.  63 72 79 75 27 84 73 65 70 76 ...
##  $ grasa: núm.  288 385 402 365 209 354 190 405 263 451 ...
##  $ edad: núm.  28 36 57 44 24 46 20 52 30 57 ...
```

```
summary(Mujeres)
```

```
##        id          peso           grasa            edad
##  Min.   : 1   Min.   :27.00   Min.   :181.0   Min.   :20.00
##  1st Qu.: 7   1st Qu.:63.00   1st Qu.:254.0   1st Qu.:30.00
##  Median :13   Median :69.00   Median :303.0   Median :37.00
##  Mean   :13   Mean   :68.68   Mean   :310.7   Mean   :39.12
##  3rd Qu.:19   3rd Qu.:76.00   3rd Qu.:374.0   3rd Qu.:50.00
##  Max.   :25   Max.   :89.00   Max.   :451.0   Max.   :60.00
```

- **Paso 3**: miramos las relaciones que pueda haber entre cada par de variables. Para ello podemos representar una matriz de diagramas de dispersión.

```
pairs(Mujeres)
```



Vemos en el gráfico una relación (que parece lineal) entre la edad y la grasa.

- **Paso 4**: miramos el grado de relación lineal de las variables.

```
kable(cor(Mujeres$grasa, Mujeres$edad))
```

x
0.8373534

Observamos que el coeficiente es elevado y se acerca a 1. Parece que sí que hay relación entre estas dos variables de nuestro conjunto de datos.

- **Paso 5**: generamos el modelo con la variable dependiente que será la variable **grasa en sangre** y la variable **edad**.

```
modelreg <- lm(grasa ~ edad, data = Mujeres)
summary(modelreg)

##
## Call:
## lm(formula = grasa ~ edad, data = Mujeres)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -63.478 -26.816  -3.854  28.315  90.881
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 102.5751    29.6376   3.461  0.00212 **
## edad          5.3207     0.7243   7.346 1.79e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 43.46 on 23 degrees of freedom
## Multiple R-squared:  0.7012, Adjusted R-squared:  0.6882
## F-statistic: 53.96 on 1 and 23 DF,  p-value: 1.794e-07
```

En este resultado podemos observar los parámetros de la ecuación de la recta de mínimos cuadrados. Estos vienen dados por la columna «**Estimate**» del apartado «**Coefficients»** y relacionan la cantidad de grasas en la sangre en función del peso. En este ejemplo la ecuación de la recta de mínimos cuadrados es:

```
y=102.575+5.321x
```

Vemos en esta tabla que el **coeficiente de determinación** (mide el ajuste de la recta a los datos) es de 0.7012 en Multiple R-squared.

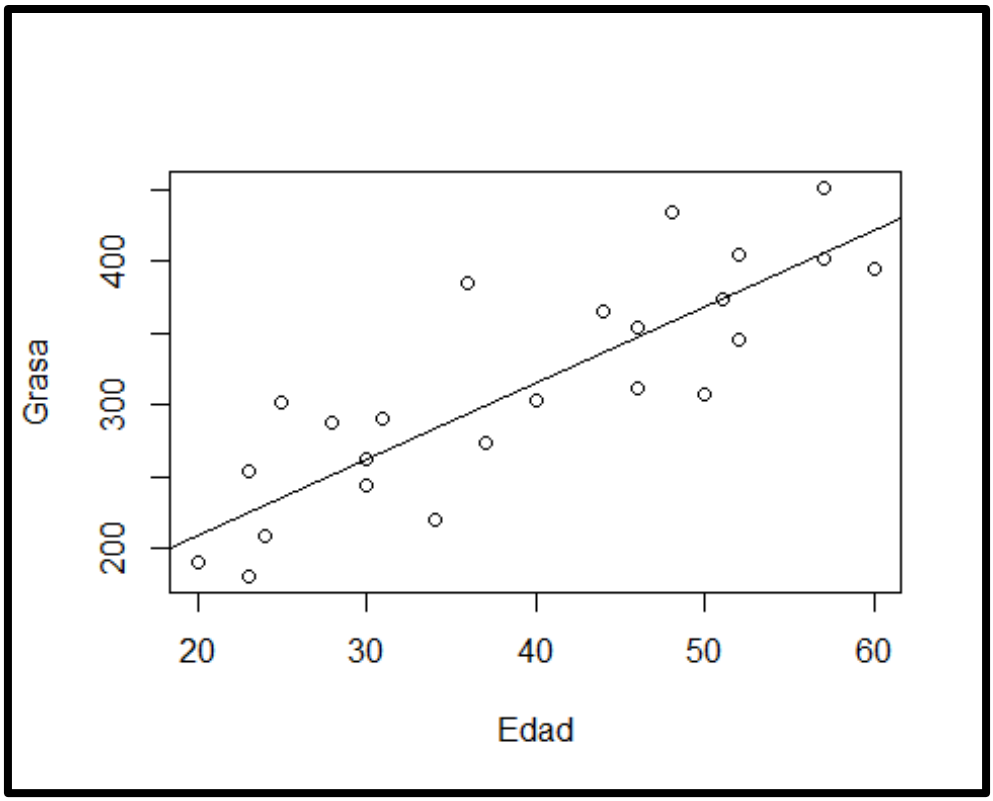
La **desviación típica de los errores** es 43.46 (Residual standard error).

Los **errores típicos de los estimadores de los parámetros β0 y β1** se encuentran en la columna «**Std. Error**».

Sus valores son **29.638** y **0.724**, respectivamente.

- **Paso 6**: generamos el gráfico de dispersión (nube de puntos) con la recta de mínimos cuadrados ajustada.

```
plot(Mujeres$edad, Mujeres$grasa, xlab='Edad', ylab='Grasa')
abline(modelreg)
```

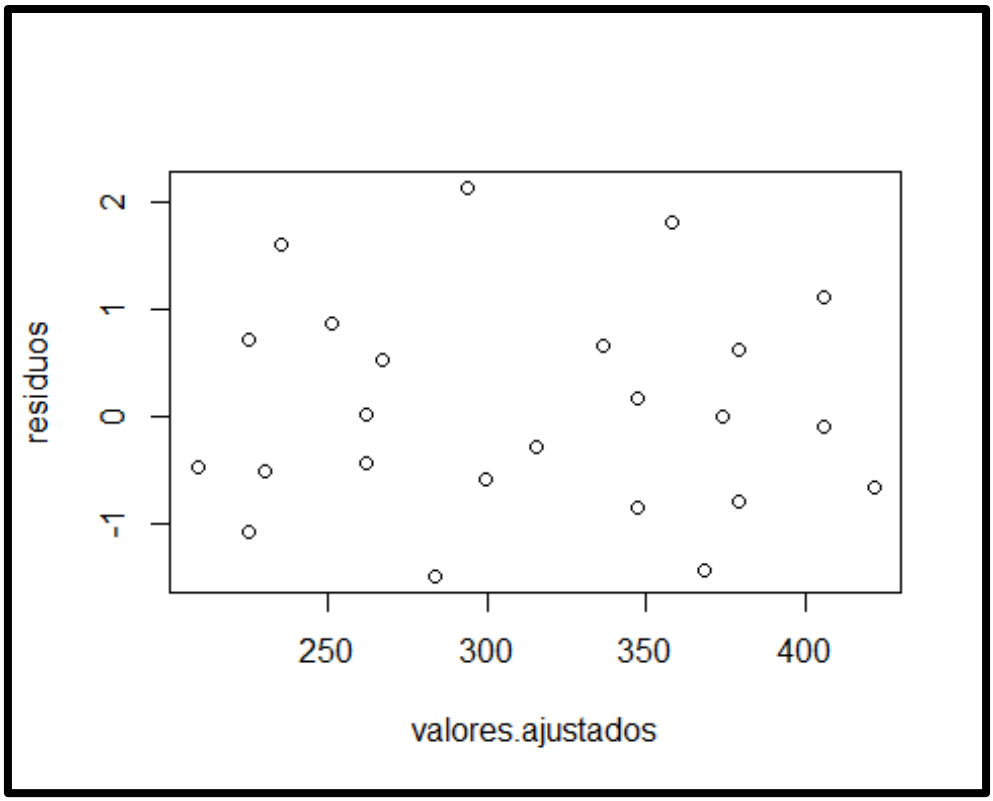


En este gráfico vemos cómo la recta del modelo de mínimos cuadrados se ajusta bastante bien a los datos.

- **Paso 7**: hacemos la validación del modelo para ver que se cumplen las condiciones.

Los valores ajustados y los residuos se pueden obtener con los comandos *fitted() y residuals()*, respectivamente. Los residuos estandarizados se obtienen con *rstandard().* Por ejemplo, el siguiente código obtiene una representación de los residuos estandarizados frente a los valores ajustados, que resulta útil al llevar a cabo el diagnóstico del modelo:

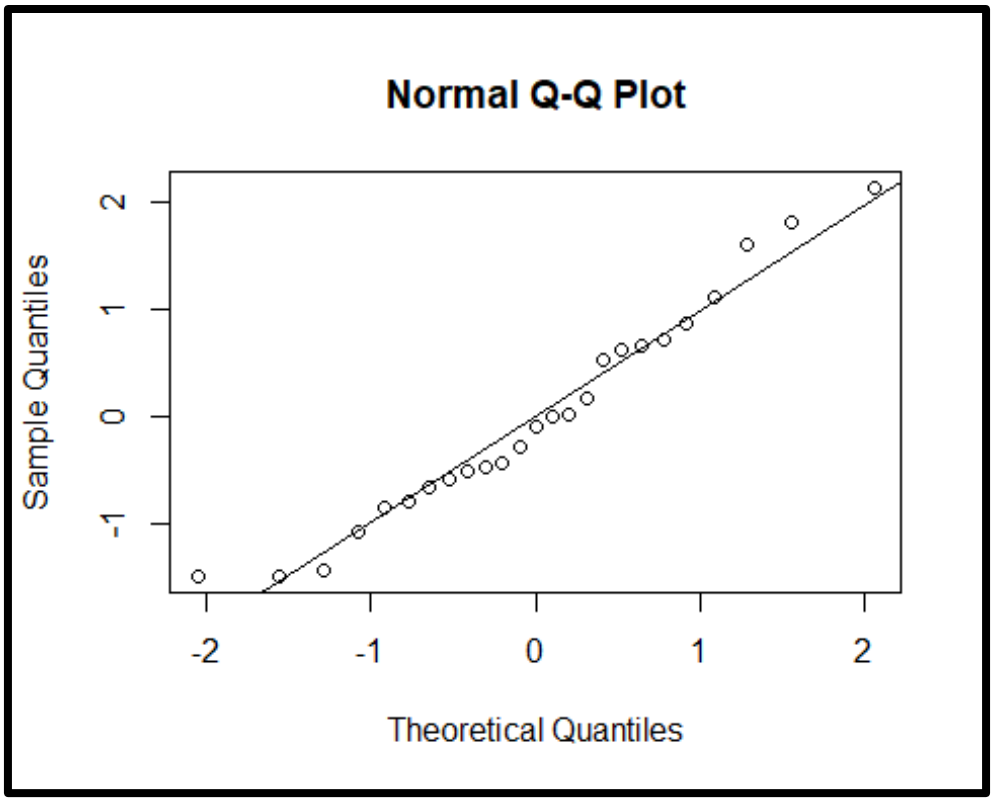
```
residuos <- rstandard(modelreg)
valores.ajustados <- fitted(modelreg)
plot(valores.ajustados, residuos)
```



No se observa ningún patrón especial, por lo que, tanto la homocedasticidad como la linealidad, resultan hipótesis razonables.

- **Paso 8**: la hipótesis de normalidad se suele comprobar mediante un Q-Q plot de los residuos. El siguiente código sirve para obtenerlo:

```
qqnorm(residuos)
qqline(residuos)
```



- **Paso 9**: supongamos que nos piden hacer una predicción puntual para la cantidad de grasas de mujeres con edades comprendidas entre 30 y 40 años.

En primer lugar, deberemos generar un vector con las nuevas edades.

```
pred_edad <- data.frame(edad = seq(30, 40))
predict(modelreg, pred_edad)
```

```
##        1        2        3        4        5        6        7        8
## 262.1954 267.5161 272.8368 278.1575 283.4781 288.7988 294.1195 299.4402
##        9       10       11
## 304.7608 310.0815 315.4022
```

La primera observación será la de la mujer de 30 años y así sucesivamente. Si queremos predecir exactamente el nivel de grasa de una mujer de 35 años será de 288.79.

- **Paso 10**: si se nos pide el intervalo de confianza (al 95 %) de los parámetros estimados podremos usar este código.

```
confint(modelreg) #por defecto el intervalo es del 95 %
```

```
##                 2.5 %     97.5 %
## (Intercept) 41.265155 163.885130
## edad         3.822367   6.818986
```

```
confint(modelreg, level=0.90) #con intervalo de confianza del 90 %
```

```
##                    5 %       95 %
## (Intercept) 51.780153 153.370132
## edad         4.079335   6.562018
```

En definitiva, en un modelo de regresión lineal simple tratamos de explicar la relación que existe entre la variable respuesta Y y una única variable explicativa X, a partir del ajuste de un modelo estimado de recta lineal a los datos usando el método de los mínimos cuadrados.

10.2. Regresión lineal múltiple

Aunque no forma parte de los contenidos de este laboratorio vamos a hacer un pequeño recordatorio sobre el modelo de regresión lineal múltiple y los pasos mínimos que debemos seguir con R. Veréis que coinciden muchos de los pasos con la regresión lineal simple, aunque se diferencia de este método por la necesidad de crear un modelo con los mejores predictores. Hay muchos métodos para encontrar los mejores predictores para un modelo múltiple, pero una opción es empezar con todas las variables del conjunto de datos como predictores y seleccionar aquellos que son mejores con el método AIC.

- **Paso 1**: activar los datos y observar con resúmenes cómo se comportan las variables.
- **Paso 2**: comprobar que no hay valores atípicos.
- **Paso 3**: analizar la relación entre las variables a partir de gráficos de dispersión y/o el cálculo del coeficiente de Pearson.
- **Paso 4**: generar el modelo final. Para ello podemos usar diferentes formas.
- **Paso 5**: seleccionar los mejores predictores y generar el modelo final.
- **Paso 6**: validar los supuestos del modelo de regresión lineal múltiple.
- **Paso 7**: buscar predicciones puntuales.

Ejemplo 14:

Vamos a ver un ejemplo de regresión lineal múltiple; para ello usaremos un conjunto de datos **state.x77** que contiene variables específicas sobre características de las ciudades de EE. UU. Es un dataset muy usado en ejemplos de este tipo de regresión.

```
#activar los datos y observar cómo son los tipos de datos y las variables con resúmenes estadísticos

library(knitr)
datos_usa <- data.frame(state.x77)
str(datos_usa)
```

```
## 'data.frame':    50 obs. of  8 variables:
##  $ Population: núm.  3615 365 2212 2110 21198 ...
##  $ Income    : núm.  3624 6315 4530 3378 5114 ...
##  $ Illiteracy: núm.  2.1 1.5 1.8 1.9 1.1 0.7 1.1 0.9 1.3 2 ...
##  $ Life.Exp  : núm.  69 69.3 70.5 70.7 71.7 ...
##  $ Murder    : núm.  15.1 11.3 7.8 10.1 10.3 6.8 3.1 6.2 10.7 13.9 ...
##  $ HS.Grad   : núm.  41.3 66.7 58.1 39.9 62.6 63.9 56 54.6 52.6 40.6 ...
```

```
##  $ Frost      : núm.  20 152 15 65 20 166 139 103 11 60 ...
##  $ Area       : núm.  50708 566432 113417 51945 156361 ...
```

```
kable(head(datos_usa)) #tabla de los primeros registros
```

	Population	Income	Illiteracy	Life.Exp	Murder	HS.Grad	Frost	Area
Alabama	3615	3624	2.1	69.05	15.1	41.3	20	50708
Alaska	365	6315	1.5	69.31	11.3	66.7	152	566432
Arizona	2212	4530	1.8	70.55	7.8	58.1	15	113417
Arkansas	2110	3378	1.9	70.66	10.1	39.9	65	51945
California	21198	5114	1.1	71.71	10.3	62.6	20	156361
Colorado	2541	4884	0.7	72.06	6.8	63.9	166	103766

```
summary(datos_usa) #resumen de los datos
```

```
##    Population        Income       Illiteracy       Life.Exp
##  Min.   :  365   Min.   :3098   Min.   :0.500   Min.   :67.96
##  1st Qu.: 1080   1st Qu.:3993   1st Qu.:0.625   1st Qu.:70.12
##  Median : 2838   Median :4519   Median :0.950   Median :70.67
##  Mean   : 4246   Mean   :4436   Mean   :1.170   Mean   :70.88
##  3rd Qu.: 4968   3rd Qu.:4814   3rd Qu.:1.575   3rd Qu.:71.89
##  Max.   :21198   Max.   :6315   Max.   :2.800   Max.   :73.60
##      Murder          HS.Grad          Frost             Area
##  Min.   : 1.400   Min.   :37.80   Min.   :  0.00   Min.   :  1049
##  1st Qu.: 4.350   1st Qu.:48.05   1st Qu.: 66.25   1st Qu.: 36985
##  Median : 6.850   Median :53.25   Median :114.50   Median : 54277
##  Mean   : 7.378   Mean   :53.11   Mean   :104.46   Mean   : 70736
##  3rd Qu.:10.675   3rd Qu.:59.15   3rd Qu.:139.75   3rd Qu.: 81163
##  Max.   :15.100   Max.   :67.30   Max.   :188.00   Max.   :566432
```

```
Density_population <- datos_usa$Population * 1000 / datos_usa$Area
datos_usa2 <- cbind(datos_usa,Density_population) #unimos el conjunto de
datos con la nueva variable de densidad de población
cor_usa2<-cor(datos_usa2)
as.data.frame(cor_usa2)
```

```
##                    Population     Income   Illiteracy    Life.Exp
Murder
## Population          1.00000000  0.2082276  0.107622373 -0.06805195
0.3436428
## Income              0.20822756  1.0000000 -0.437075186  0.34025534 -
0.2300776
## Illiteracy          0.10762237 -0.4370752  1.000000000 -0.58847793
0.7029752
## Life.Exp           -0.06805195  0.3402553 -0.588477926  1.00000000 -
0.7808458
## Murder              0.34364275 -0.2300776  0.702975199 -0.78084575
1.0000000
```

```
## HS.Grad            -0.09848975  0.6199323 -0.657188609  0.58221620 -
0.4879710
## Frost              -0.33215245  0.2262822 -0.671946968  0.26206801 -
0.5388834
## Area                0.02254384  0.3633154  0.077261132 -0.10733194
0.2283902
## Density_population  0.24622789  0.3299683  0.009274348  0.09106176 -
0.1850352
##                        HS.Grad        Frost        Area Density_population
## Population         -0.09848975 -0.332152454  0.02254384        0.246227888
## Income              0.61993232  0.226282179  0.36331544        0.329968277
## Illiteracy         -0.65718861 -0.671946968  0.07726113        0.009274348
## Life.Exp            0.58221620  0.262068011 -0.10733194        0.091061763
## Murder             -0.48797102 -0.538883437  0.22839021       -0.185035233
## HS.Grad             1.00000000  0.366779702  0.33354187       -0.088367214
## Frost               0.36677970  1.000000000  0.05922910        0.002276734
## Area                0.33354187  0.059229102  1.00000000       -0.341388515
## Density_population -0.08836721  0.002276734 -0.34138851        1.000000000
```

```
kable(cor_usa2) #miramos la tabla con las correlaciones
```

	Population	Income	Illiteracy	Life.Exp	Murder	HS.Grad	Frost	Area	Density_ population
Population	1.0000000	0.2082276	0.1076224	-0.0680520	0.3436428	-0.0984897	-0.3321525	0.0225438	0.2462279
Income	0.2082276	1.0000000	-0.4370752	0.3402553	-0.2300776	0.6199323	0.2262822	0.3633154	0.3299683
Illiteracy	0.1076224	-0.4370752	1.0000000	-0.5884779	0.7029752	-0.6571886	-0.6719470	0.0772611	0.0092743
Life.Exp	-0.0680520	0.3402553	-0.5884779	1.0000000	-0.7808458	0.5822162	0.2620680	-0.1073319	0.0910618
Murder	0.3436428	-0.2300776	0.7029752	-0.7808458	1.0000000	-0.4879710	-0.5388834	0.2283902	-0.1850352
HS.Grad	-0.0984897	0.6199323	-0.6571886	0.5822162	-0.4879710	1.0000000	0.3667797	0.3335419	-0.0883672
Frost	-0.3321525	0.2262822	-0.6719470	0.2620680	-0.5388834	0.3667797	1.0000000	0.0592291	0.0022767
Area	0.0225438	0.3633154	0.0772611	-0.1073319	0.2283902	0.3335419	0.0592291	1.0000000	-0.3413885
Density_ population	0.2462279	0.3299683	0.0092743	0.0910618	-0.1850352	-0.0883672	0.0022767	-0.3413885	1.0000000

```
model_reg_mult <- lm(Life.Exp ~ Population + Income + Illiteracy + Murder +
                HS.Grad + Frost + Area + Density_population, data =
datos_usa2) #modelo con todas las variables
summary(model_reg_mult) #resumen del modelo con todas las variables
```

```
##
## Call:
## lm(formula = Life.Exp ~ Population + Income + Illiteracy + Murder +
##     HS.Grad + Frost + Area + Density_population, data = datos_usa2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.47514 -0.45887 -0.06352  0.59362  1.21823
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)         6.995e+01  1.843e+00  37.956  < 2e-16 ***
## Population          6.480e-05  3.001e-05   2.159   0.0367 *
## Income              2.701e-04  3.087e-04   0.875   0.3867
## Illiteracy          3.029e-01  4.024e-01   0.753   0.4559
## Murder             -3.286e-01  4.941e-02  -6.652 5.12e-08 ***
## HS.Grad             4.291e-02  2.332e-02   1.840   0.0730 .
## Frost              -4.580e-03  3.189e-03  -1.436   0.1585
## Area               -1.558e-06  1.914e-06  -0.814   0.4205
## Density_population -1.105e-03  7.312e-04  -1.511   0.1385
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7337 on 41 degrees of freedom
## Multiple R-squared:  0.7501, Adjusted R-squared:  0.7013
## F-statistic: 15.38 on 8 and 41 DF,  p-value: 3.787e-10
```

Para buscar las variables que son mejores predictoras para nuestro estudio, usaremos la función *step().*

```
step(object=model_reg_mult, direction="both", trace=1)
```

```
## Start:  AIC=-22.89
## Life.Exp ~ Population + Income + Illiteracy + Murder + HS.Grad +
##     Frost + Area + Density_population
##
##                       Df Sum of Sq    RSS     AIC
## - Illiteracy           1    0.3050 22.373 -24.208
## - Area                 1    0.3564 22.425 -24.093
## - Income               1    0.4120 22.480 -23.969
## <none>                             22.068 -22.894
## - Frost                1    1.1102 23.178 -22.440
## - Density_population   1    1.2288 23.297 -22.185
## - HS.Grad              1    1.8225 23.891 -20.926
## - Population           1    2.5095 24.578 -19.509
## - Murder               1   23.8173 45.886  11.707
##
## Step:  AIC=-24.21
## Life.Exp ~ Population + Income + Murder + HS.Grad + Frost + Area +
##     Density_population
##
##                       Df Sum of Sq    RSS     AIC
## - Area                 1    0.1427 22.516 -25.890
## - Income               1    0.2316 22.605 -25.693
## <none>                             22.373 -24.208
## - Density_population   1    0.9286 23.302 -24.174
## - HS.Grad              1    1.5218 23.895 -22.918
## + Illiteracy           1    0.3050 22.068 -22.894
## - Population           1    2.2047 24.578 -21.509
## - Frost                1    3.1324 25.506 -19.656
## - Murder               1   26.7071 49.080  13.072
```

```
##
## Step:  AIC=-25.89
## Life.Exp ~ Population + Income + Murder + HS.Grad + Frost +
Density_population
##
##                      Df Sum of Sq    RSS     AIC
## - Income              1     0.132 22.648 -27.598
## - Density_population  1     0.786 23.302 -26.174
## <none>                            22.516 -25.890
## - HS.Grad             1     1.424 23.940 -24.824
## + Area                1     0.143 22.373 -24.208
## + Illiteracy          1     0.091 22.425 -24.093
## - Population          1     2.332 24.848 -22.962
## - Frost               1     3.304 25.820 -21.043
## - Murder              1    32.779 55.295  17.033
##
## Step:  AIC=-27.6
## Life.Exp ~ Population + Murder + HS.Grad + Frost + Density_population
##
##                      Df Sum of Sq    RSS     AIC
## - Density_population  1     0.660 23.308 -28.161
## <none>                            22.648 -27.598
## + Income              1     0.132 22.516 -25.890
## + Illiteracy          1     0.061 22.587 -25.732
## + Area                1     0.043 22.605 -25.693
## - Population          1     2.659 25.307 -24.046
## - Frost               1     3.179 25.827 -23.030
## - HS.Grad             1     3.966 26.614 -21.529
## - Murder              1    33.626 56.274  15.910
##
## Step:  AIC=-28.16
## Life.Exp ~ Population + Murder + HS.Grad + Frost
##
##                      Df Sum of Sq    RSS     AIC
## <none>                            23.308 -28.161
## + Density_population  1     0.660 22.648 -27.598
## + Income              1     0.006 23.302 -26.174
## + Illiteracy          1     0.004 23.304 -26.170
## + Area                1     0.001 23.307 -26.163
## - Population          1     2.064 25.372 -25.920
## - Frost               1     3.122 26.430 -23.877
## - HS.Grad             1     5.112 28.420 -20.246
## - Murder              1    34.816 58.124  15.528

##
## Call:
## lm(formula = Life.Exp ~ Population + Murder + HS.Grad + Frost,
##     data = datos_usa2)
##
## Coefficients:
```

```
## (Intercept)   Population       Murder      HS.Grad        Frost
##   7.103e+01    5.014e-05   -3.001e-01    4.658e-02   -5.943e-03
```

Si nos fijamos en el final de los resultados, veremos que nos determina el modelo final con el método AIC y es el que usaremos finalmente. La esperanza de vida dependerá de los asesinatos, las heladas, los universitarios y la población.

```
model_reg_mult2 <- (lm(formula = Life.Exp ~ Population  + Murder + HS.Grad +
Frost, data = datos_usa2))
summary(model_reg_mult2) #resultados con el modelo final
```

```
##
## Call:
## lm(formula = Life.Exp ~ Population + Murder + HS.Grad + Frost,
##     data = datos_usa2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.47095 -0.53464 -0.03701  0.57621  1.50683
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.103e+01  9.529e-01  74.542  < 2e-16 ***
## Population   5.014e-05  2.512e-05   1.996  0.05201 .
## Murder      -3.001e-01  3.661e-02  -8.199 1.77e-10 ***
## HS.Grad      4.658e-02  1.483e-02   3.142  0.00297 **
## Frost       -5.943e-03  2.421e-03  -2.455  0.01802 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7197 on 45 degrees of freedom
## Multiple R-squared:  0.736,  Adjusted R-squared:  0.7126
## F-statistic: 31.37 on 4 and 45 DF,  p-value: 1.696e-12
```

Ahora imaginemos que nos piden una predicción de una ciudad que tiene **7000 habitantes**, un **índice de asesinatos con un valor de 5**, un valor de **HS.Grad igual a 45** y **Frost = 120**.

```
Valores_predic <- data.frame(Population = 7000,
                             Murder = 5, HS.Grad = 45, Frost = 120 )
Valores_predic
```

```
##   Population Murder HS.Grad Frost
## 1       7000      5      45   120
```

```
predicción <- predict(model_reg_mult2, newdata = Valores_predic)
predicción
```

```
##        1
## 71.26037
```

La esperanza de vida que resulta en la predicción es de 71.2 años. Parece bastante fiable, ya que si nos fijamos en los valores que hay en el conjunto de datos y los comparamos con New Jersey, que tiene valores parecidos, el valor es bastante parecido.

11. ANOVA

Aunque el estudio de ANOVA (Análisis de varianza) se verá con más profundidad en otros **LAB**, explicamos aquí los pasos más importantes para realizar un breve estudio:

- **Paso 1**: activar los datos.
- **Paso 2**: hacer un resumen de los datos.
- **Paso 3**: buscar valores atípicos significativos en nuestros datos.
- **Paso 4**: comprobar que la variable tiene una distribución «aproximadamente» normal para cada categoría de la variable independiente.
- **Paso 5**: comprobar la homogeneidad de varianza. Comprobaremos que las varianzas de la respuesta en cada grupo son iguales.
- **Paso 6**: comprobar si existen diferencias entre los grupos realizando la prueba ANOVA.
- **Paso 7**: comprobar si entre grupos existen diferencias. Usamos las pruebas *post hoc* como la prueba de comparaciones múltiples de Tukey.

Ejemplo 15:

Para este ejemplo usaremos unos datos llamados **goggles** del paquete **WRS2** de **R**. Estos datos contienen información sobre 48 individuos (24 hombres y 24 mujeres) y tres variables de estudio. Las variables son **sexo (*gender*)**, **alcohol** y **nivel de atracción (*attractiveness*)**.

El objetivo del estudio era comprobar si, después de consumir alcohol o no durante toda la noche, las percepciones subjetivas del atractivo físico se vuelven menos rigurosas. Tenemos una variable dependiente numérica (***attractiveness***) y la queremos comparar entre tres niveles con una variable categórica (**alcohol**) que corresponden a muestras independientes, ya que tenemos individuos distintos.

Para ello, vamos a usar un **ANOVA** de una vía o un factor entre grupos (paso a paso) para evaluar si existen diferencias entre los grupos:

Paso 1: acceder a los datos que queremos de nuestro paquete.

```
#install.packages("WRS2") #instalar la librería
library (WRS2)
data("goggles") #activar datos
head("goggles") #primeros registros

## [1] "goggles"
```

Paso 2: resumen de las variables del conjunto de datos.

```
summary(goggles)
```

```
##     gender      alcohol   attractiveness
##  Female:24   None   :16   Min.   :20.00
##  Male  :24   2 Pints:16   1st Qu.:53.75
##              4 Pints:16   Median :60.00
##                           Mean   :58.33
##                           3rd Qu.:66.25
##                           Max.   :85.00
```

Para las variables categóricas, nos hace falta un resumen de frecuencias. Observamos que hay dieciséis individuos por cada grupo según el alcohol que han tomado al final de la noche.

Nota: no son demasiados individuos por grupo, pero como ejemplo de la realización de un ANOVA nos valdrá.

Para ver los resúmenes estadísticos por grupo, podemos usar la función *describeBy()* de la librería **psych**. Este comando nos permite ver un resumen específico de cada grupo.

```
# install.packages("psych")
library(psych)
describeBy(goggles$attractiveness, goggles$alcohol)
```

```
##
##  Descriptive statistics by group
## group: None
##    vars  n  mean   sd median trimmed   mad min max range skew kurtosis
se
## X1    1 16 63.75 8.47   62.5   63.57 11.12  50  80    30 0.29    -1.07
2.12
## ------------------------------------------------------------
## group: 2 Pints
##    vars  n  mean   sd median trimmed  mad min max range skew kurtosis   se
## X1    1 16 64.69 9.91     65   64.64 7.41  45  85    40 0.08    -0.23 2.48
## ------------------------------------------------------------
## group: 4 Pints
##    vars  n  mean    sd median trimmed   mad min max range  skew kurtosis
se
## X1    1 16 46.56 14.34     50   46.79 14.83  20  70    50 -0.22    -1.21
3.59
```

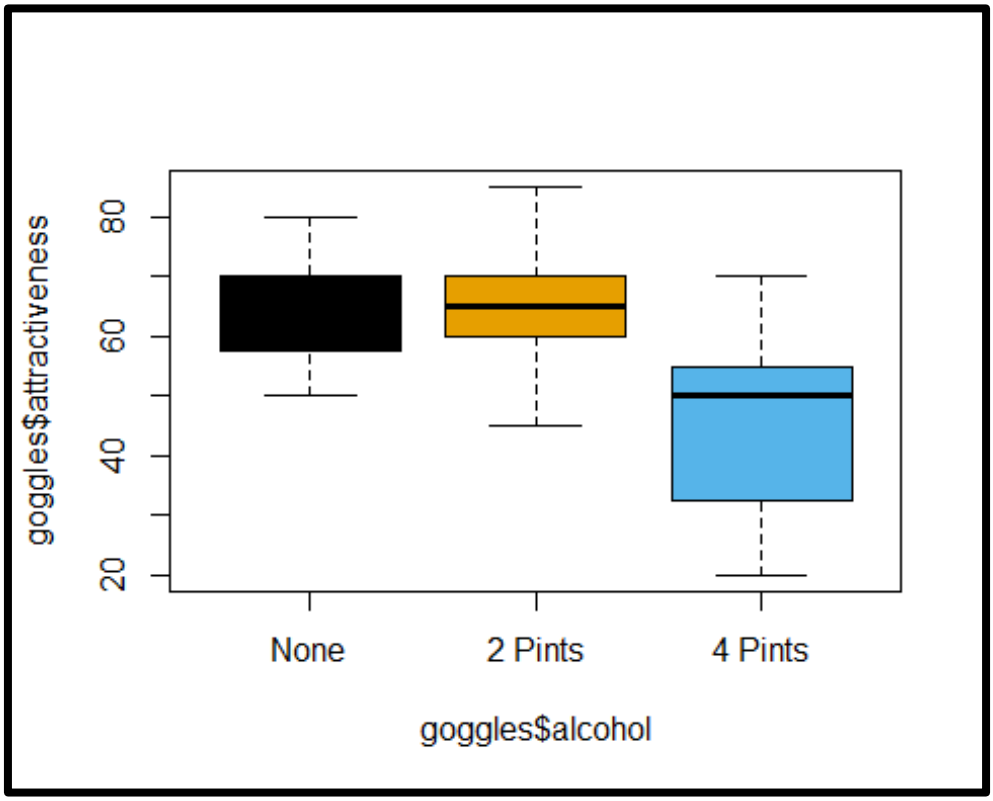
Si nos fijamos en los datos, comprobamos que en los dos primeros grupos se presentan unos valores parecidos en el valor del atractivo de las parejas que encontraron al final de la noche. En cambio, en el tercer grupo (que consumió más alcohol) los valores en el atractivo son menores.

Los valores se nos presentan como la **media ± sd**. De esta manera, el grupo que **no consumió alcohol (*none*)**, sus valores se encuentran en **63.75 ± 8.47**; el que consumió **dos pintas de**

alcohol, sus valores se encuentran en **64.69 ± 9.91** y, por último, aquellos que consumieron **cuatro pintas de alcohol**, sus valores se encuentran en **46.56 ± 14.34**.

Paso 3: para ver más claros estos datos, podemos hacer un gráfico de caja con los tres grupos con colores diferenciados.

```
boxplot(goggles$attractiveness ~ goggles$alcohol, col=palette.colors())
```



Paso 4: es importante verificar que no hay valores atípicos en nuestros datos. Para ello podemos usar la función *rep.outlier()* de la librería **rapportools**. Este comando funcionará si la variable es numérica. Si no, siempre podremos observar los datos como en este caso con tan pocos individuos.

Paso 5: por otro lado, vamos a comprobar la normalidad con el test de Shapiro-Wilk y la homocedasticidad (es decir que la varianza sea homogénea) con dos métodos diferentes (test de Bartlett y la prueba de Levene).

```
by(goggles$attractiveness, goggles$alcohol, shapiro.test) #normalidad

## goggles$alcohol: None
##
##  Shapiro-Wilk normality test
##
## data:  dd[x, ]
## W = 0.95498, p-value = 0.5725
##
```

```
## ------------------------------------------------------------
## goggles$alcohol: 2 Pints
##
##  Shapiro-Wilk normality test
##
## data:  dd[x, ]
## W = 0.94489, p-value = 0.4132
##
## ------------------------------------------------------------
## goggles$alcohol: 4 Pints
##
##  Shapiro-Wilk normality test
##
## data:  dd[x, ]
## W = 0.952, p-value = 0.522
```

```
bartlett.test(goggles$attractiveness, goggles$alcohol) #varianza homogénea
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  goggles$attractiveness and goggles$alcohol
## Bartlett's K-squared = 4.4295, df = 2, p-value = 0.1092
```

```
#levene.test (goggles$attractiveness, goggles$alcohol) #varianza homogénea
pero más sensible a las desviaciones de la normalidad
```

En todos los casos obtenemos **p > 0.05**, es decir, no encontramos problemas de heterocedasticidad (falta de homogeneidad de varianza) ni de normalidad.

Paso 6: comprobar si existen diferencias entre los grupos realizando la prueba ANOVA.

```
model_anov <- aov(attractiveness ~ alcohol, data= goggles)
summary (model_anov)
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## alcohol      2   3332  1666.1   13.31 2.88e-05 ***
## Residuals   45   5634   125.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Vemos un resultado estadísticamente significativo, ya que sí que hay diferencias entre grupos (p-value < 0.05).

Paso 7: una vez validado el resultado significativo, nos interesa evaluar entre qué grupos encontramos estas diferencias con la prueba de comparaciones múltiples *post hoc* de Tukey.

```
TukeyHSD(model_anov)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
```

```
## Fit: aov(formula = attractiveness ~ alcohol, data = goggles)
##
## $alcohol
##                       diff        lwr       upr     p adj
## 2 Pints-None        0.9375  -8.650654 10.525654 0.9695381
## 4 Pints-None      -17.1875 -26.775654 -7.599346 0.0002283
## 4 Pints-2 Pints -18.1250 -27.713154 -8.536846 0.0001067
```

En los resultados (detallados por parejas de grupos diferentes) podemos observar cómo existen diferencias (p valores < 0.001) entre los que consumen cuatro pintas de alcohol y no consumen alcohol, y también entre los que consumen cuatro pintas de alcohol y dos pintas de alcohol.

En cambio, en los grupos que consumen dos pintas de alcohol y no consumen alcohol no hay diferencia alguna.

En definitiva, la prueba ANOVA de un factor es una técnica estadística que señala si dos variables (una dependiente y una independiente) están relacionadas con base en si las medias de la variable dependiente son diferentes por grupos de la variable independiente. Es decir, si las medias entre dos o más grupos son similares o diferentes.

12. Ejercicios y casos prácticos con R

Antes de practicar con estos ejercicios, aseguraos de que tenéis instalado el paquete **Tidyverse**. Este paquete es muy importante en el análisis estadístico y en la elaboración de gráficos, ya que provee una serie de herramientas destinadas a facilitar estos procesos. Está compuesto, entre otros, de los siguientes paquetes: **readr, dplyr, ggplot2, tidyr**, etc.

Para más información sobre este paquete, podéis consultar la página web:
https://www.tidyverse.org/

Ejercicio 1:

El package de **R datasets** (https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/00Index.html) pone a nuestra disposición una serie de conjuntos de datos con los que poder trabajar como, por ejemplo, **iris**, **cars** o **Titanic**. Escoged un conjunto de datos.

Deberéis:

- Buscar un resumen estadístico de las variables del dataset **Iris** y **Orange**.
- Generar una tabla de frecuencias absolutas y una tabla de frecuencias relativas con el dataset **Iris**. ¿Todas las tablas generadas tienen sentido para vosotros?
- Generar una tabla de frecuencias absolutas con cada una de las variables del conjunto de datos **Orange**. ¿Todas las tablas generadas tienen sentido para vosotros?
- Generar una tabla de doble entrada entre las variables Tree y Age de **Orange**.

Ejercicio 2:

Copiad y ejecutad el código siguiente de los dos vectores:

```
vect1 <- c(1,2,1,2,1,2,1,2,1,2,1,1,1,1,2,2,1,1,2,1)
vect2 <- c(1,1,2,2,2,1,2,1,1,2,1,2,1,1,1,2,1,1,1,1)
```

Responded a los apartados siguientes:

a) Usando el **vect1** creamos un nuevo vector llamado **Bajo_peso** que sea un factor con dos niveles. Las etiquetas corresponden a **1 = Bajo peso** y **2 = Peso normal**.

b) Usando el **vect2** creamos un nuevo vector llamado **Fumador** que sea un factor con dos niveles. Las etiquetas corresponden a **1 = Fuma** y **2 = No fuma**.

c) Creamos una tabla de contingencia con las dos variables anteriores con el nombre Tabla.

d) Miramos la relación de las variables anteriores con la **prueba del ji cuadrado**.

e) Miramos también cómo resulta el **test de Fisher**.

Ejercicio 3:

a) Activad el paquete de datos *airquality* del paquete **datasets** y generad los siguientes gráficos:

- Un gráfico de dispersión de la variable *Ozone* de color azul y con almohadillas (#) en vez de puntos.
- Un gráfico de caja de color rojo con la variable Temp con el título Temperatura (en grados Farenheit).

b) Activad el paquete de datos **airmiles** del paquete **datasets** y generad los siguientes gráficos:

- Un gráfico de líneas de la serie de datos airmiles con el título Datos de pasajeros en vuelos comerciales (en miles) y de color azul (cadetblue2) y la etiqueta del eje x con Miles de pasajeros.
- Un histograma de la serie de datos airmiles de color marrón (chocolate2).

c) Representad los cuatro gráficos en una única imagen donde los veamos juntos.

Ejercicio 4:

Intentad reproducir el **ejemplo 7** de este **LAB** con unos datos simulados por vosotros. No hace falta que sean parecidos, pero es necesario que podáis hacer diferentes gráficos estadísticos con el comando *plot().*

Ejercicio 5:

Para poder practicar la creación de gráficos con **ggplot2**, vamos a crear seis gráficos con diferentes características y con diferentes conjuntos de datos de paquetes trabajados anteriormente.

a) Con los datos *airquality* del paquete datasets cread un gráfico de dispersión simple, pero con la recta de regresión ajustada en color rojo (darkred).

b) Con los datos ***airquality*** del paquete **datasets** cread un gráfico de dispersión con las variables Solar.R y Temp con sus respectivas etiquetas de eje. Los colores y la forma de los puntos deben ser en función de la variable Month convertida a factor.

c) Con los datos de **Birthwt** del paquete **MASS** mostrad la distribución de la variable **age** que tenga el borde de color negro y azul en el interior de las barras.

d) Con los datos de **Birthwt** del paquete **MASS** mostrad la distribución de la variable **age**, pero separada en dos gráficos (fumador/no fumador) por la variable **smoke**.

Ejercicio 6:

Cread un gráfico con unos datos extraídos del paquete Datasets de **R** y guardadlo como imagen (.jpg) con el nombre migrafic1 y, también, como documento en PDF con el nombre migrafic2. Haced una captura de pantalla del fichero generado.

Ejercicio 7:

Para practicar la regresión lineal simple usaremos el conjunto de datos Orange que se encuentra en la librería **tidyverse** y que tiene información sobre tres variables (árbol, edad en días desde que se sembró el árbol y circunferencia del tronco en centímetros) de 35 naranjos.

- Queremos saber qué valor de circunferencia tendrá un árbol seiscientos días después de plantarlo. Cread para ello un modelo lineal y practicad la regresión lineal paso a paso con los pasos que habéis visto en este laboratorio.

Ejercicio 8:

Repetid los gráficos (aquellos que podáis) de la regresión lineal simple del ejercicio anterior pero ahora con **ggplot2**.

Nota: en caso de haber elaborado el ejercicio 7 con gráficos de este tipo, creadlos con gráficos base.

Ejercicio 9:

Activad el conjunto de datos **PlantGrowth** del paquete **datasets** de R. Este archivo tiene los recultados de un experimento para comparar los rendimientos medios por el peso seco de las plantas (**weight**) obtenidos bajo un control y dos condiciones de tratamiento diferentes (**group factor**).

- ¿Creéis que hay diferencias entre tratamientos?
- Se cumplen las condiciones para poder aplicar una ANOVA. ¿Qué pruebas os planteáis?

Ejercicio 10:

Buscad información del paquete **Plotly** para la creación de gráficos interactivos y generad un histograma o un gráfico de barras interactivo. Explicad qué se puede hacer con este gráfico.

Caso práctico

A partir de unos datos bioclínicos o biosanitarios que escojáis y que importéis a R, explicad sus variables (mínimo de ocho variables) y también:

- Realizad un resumen estadístico completo del dataset y explicad los resultados.
- Realizad cinco gráficos básicos con las variables, explicad su significado y guardadlos como imágenes (jpeg o bmp).
- Realizad dos gráficos con el comando *ggplot()*, explicad su significado y guardadlos como imágenes (jpeg o bmp).
- Generad una regresión lineal entre dos de sus variables paso a paso y comentad los resultados obtenidos.

Nota: os dejamos diferentes enlaces con repositorios de datos biosanitarios en abierto:

- Repositorios para aprendizaje y docencia en bioestadística
- Datos abiertos biosanitarios. Biblioteca de Berkeley

Solución a los ejercicios propuestos y casos prácticos con R

Solución del ejercicio 1:

```
#apartado a)
library(datasets)
library(knitr)
data(Orange)
head(Orange)
```

```
##   Tree  age circumference
## 1    1  118            30
## 2    1  484            58
## 3    1  664            87
## 4    1 1004           115
## 5    1 1231           120
## 6    1 1372           142
```

```
summary(Orange)
```

```
##  Tree       age         circumference
##  3:7   Min.   : 118.0   Min.   : 30.0
##  1:7   1st Qu.: 484.0   1st Qu.: 65.5
##  5:7   Median :1004.0   Median :115.0
##  2:7   Mean   : 922.1   Mean   :115.9
##  4:7   3rd Qu.:1372.0   3rd Qu.:161.5
##        Max.   :1582.0   Max.   :214.0
```

```
data(iris)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
```

```
##  virginica :50
##
##
##

#apartado b)
#frecuencias absolutas variables iris
kable(table(iris$Sepal.Length))
```

Var1	Freq
4.3	1
4.4	3
4.5	1
4.6	4
4.7	2
4.8	5
4.9	6
5	10
5.1	9
5.2	4
5.3	1
5.4	6
5.5	7
5.6	6
5.7	8
5.8	7
5.9	3
6	6
6.1	6
6.2	4
6.3	9
6.4	7
6.5	5
6.6	2
6.7	8
6.8	3
6.9	4
7	1
7.1	1

Var1	Freq
7.2	3
7.3	1
7.4	1
7.6	1
7.7	4
7.9	1

```
kable(table(iris$Sepal.Width))
```

Var1	Freq
2	1
2.2	3
2.3	4
2.4	3
2.5	8
2.6	5
2.7	9
2.8	14
2.9	10
3	26
3.1	11
3.2	13
3.3	6
3.4	12
3.5	6
3.6	4
3.7	3
3.8	6
3.9	2
4	1
4.1	1
4.2	1
4.4	1

```
kable(table(iris$Petal.Length))
```

Var1	Freq
1	1
1.1	1
1.2	2
1.3	7
1.4	13
1.5	13
1.6	7
1.7	4
1.9	2
3	1
3.3	2
3.5	2
3.6	1
3.7	1
3.8	1
3.9	3
4	5
4.1	3
4.2	4
4.3	2
4.4	4
4.5	8
4.6	3
4.7	5
4.8	4
4.9	5
5	4
5.1	8
5.2	2
5.3	2
5.4	2
5.5	3
5.6	6

Var1	Freq
5.7	3
5.8	3
5.9	2
6	2
6.1	3
6.3	1
6.4	1
6.6	1
6.7	2
6.9	1

```
kable(table(iris$Petal.Width))
```

Var1	Freq
0.1	5
0.2	29
0.3	7
0.4	7
0.5	1
0.6	1
1	7
1.1	3
1.2	5
1.3	13
1.4	8
1.5	12
1.6	4
1.7	2
1.8	12
1.9	5
2	6
2.1	6
2.2	3
2.3	8
2.4	3
2.5	3

```r
kable(table(iris$Species))
```

Var1	Freq
setosa	50
versicolor	50
virginica	50

```r
#frecuencias relativas variables iris
kable(prop.table(iris$Sepal.Length))

kable(prop.table(iris$Sepal.Width))

kable(prop.table(iris$Petal.Length))

kable(prop.table(iris$Petal.Width))

prop.table(iris$Species)

#apartado c)

kable(table(Orange$Tree))
```

Var1	Freq
3	7
1	7
5	7
2	7
4	7

```r
kable(table(Orange$age))
```

Var1	Freq
118	5
484	5
664	5
1004	5
1231	5
1372	5
1582	5

```
kable(table(Orange$circumference))
```

Var1	Freq
30	3
32	1
33	1
49	1
51	1
58	1
62	1
69	1
75	1
81	1
87	1
108	1
111	1
112	1
115	2
120	1
125	1
139	1
140	1
142	2
145	1
156	1
167	1
172	1
174	1
177	1
179	1
203	2
209	1
214	1

```r
#apartado d)

table(Orange$Tree, Orange$age)
```

```
##
##     118 484 664 1004 1231 1372 1582
##   3   1   1   1    1    1    1    1
##   1   1   1   1    1    1    1    1
##   5   1   1   1    1    1    1    1
##   2   1   1   1    1    1    1    1
##   4   1   1   1    1    1    1    1
```

Solución del ejercicio 2:

```r
#código del enunciado
vect1 <- c(1,2,1,2,1,2,1,2,1,2,1,1,1,1,2,2,1,1,2,1)
vect2 <- c(1,1,2,2,2,1,2,1,1,2,1,2,1,1,1,2,1,1,1,1)

#apartado a)
Bajo_Peso=factor(vect1, levels=c(1,2), labels=c("Bajo Peso","Peso Normal"))
#generamos un factor con sus nuevas etiquetas
Bajo_Peso
```

```
##  [1] Bajo Peso   Peso Normal Bajo Peso   Peso Normal Bajo Peso   Peso
Normal
##  [7] Bajo Peso   Peso Normal Bajo Peso   Peso Normal Bajo Peso   Bajo Peso
## [13] Bajo Peso   Bajo Peso   Peso Normal Peso Normal Bajo Peso   Bajo Peso
## [19] Peso Normal Bajo Peso
## Levels: Bajo Peso Peso Normal
```

```r
#apartado b)

Fumador=factor(vect2, levels=c(1,2), labels=c("Fuma","No Fuma"))  #generamos
un factor con sus nuevas etiquetas
Fumador
```

```
##  [1] Fuma    Fuma    No Fuma No Fuma No Fuma Fuma    No Fuma Fuma    Fuma
## [10] No Fuma Fuma    No Fuma Fuma    Fuma    Fuma    No Fuma Fuma    Fuma
## [19] Fuma    Fuma
## Levels: Fuma No Fuma
```

```r
#apartado c)
tabla<-table(Bajo_Peso,Fumador)
tabla
```

```
##              Fumador
## Bajo_Peso     Fuma No Fuma
##   Bajo Peso      8       4
##   Peso Normal    5       3
```

```r
#apartado d)
chisq.test(tabla)
```

```
## Warning in chisq.test(tabla): Chi-squared approximation may be incorrect
```

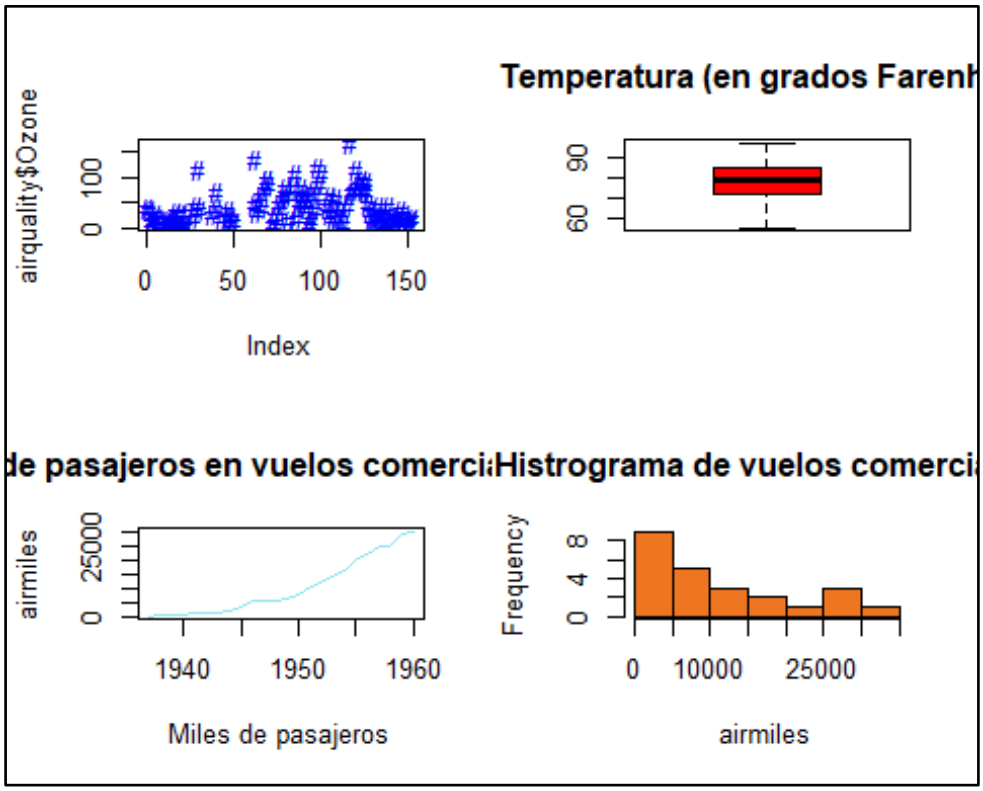
```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  tabla
## X-squared = 0, df = 1, p-value = 1
```

```r
#apartado e)
fisher.test(tabla)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  tabla
## p-value = 1
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##   0.1200513 10.9278345
## sample estimates:
## odds ratio
##   1.189031
```

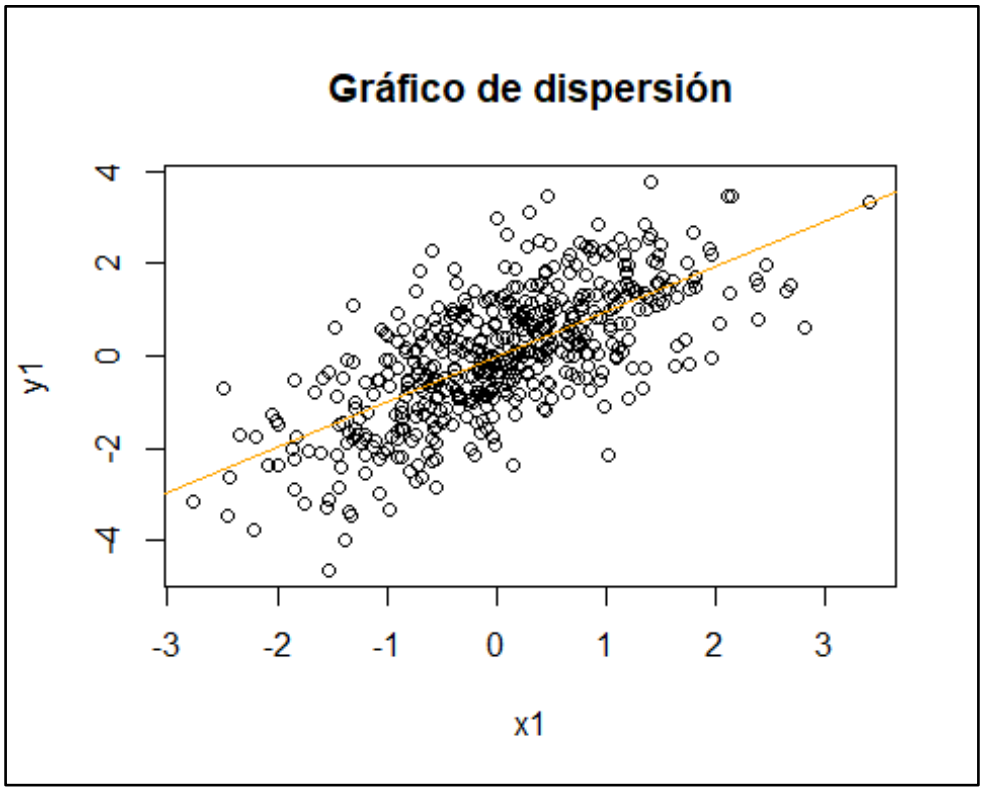
Solución del ejercicio 3:

```r
data("airquality")
data("airmiles")
par(mfrow=c(2,2)) #matriz 2x2
plot(airquality$Ozone, col="blue",pch="#") #apartado a)
boxplot(airquality$Temp, col="red", main="Temperatura (en grados Farenheit)")
plot(airmiles, main = "Datos de pasajeros en vuelos comerciales (miles)",
     xlab = "Miles de pasajeros", col = "cadetblue2")
hist(airmiles, col="chocolate2", main="Histrograma de vuelos comerciales")
```

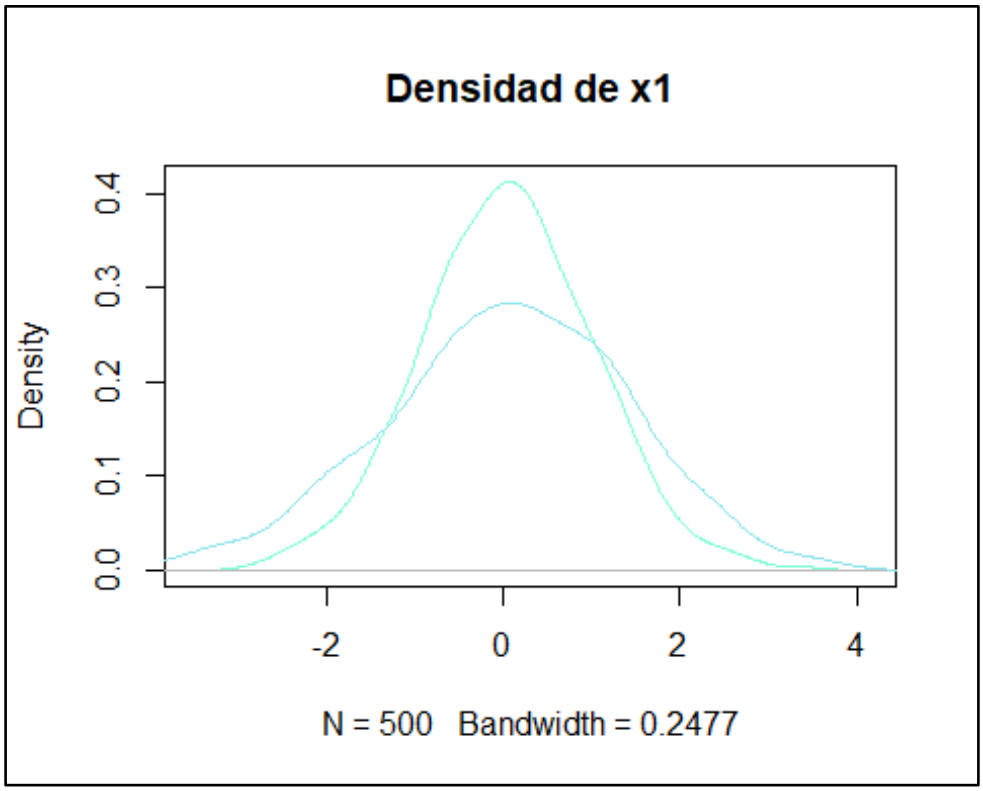


Solución del ejercicio 4:

```
set.seed(333)
x1 <- rnorm(500) #simulamos una variable x1
y1 <- x1 + rnorm(500) #simulamos una variable y1
plot(x1, y1, main="Gráfico de dispersión") #creamos el gráfico de dispersión
de las variables
abline(lm(y1~x1), col="orange") #creamos la recta de regresión de color verde
```




```r
plot(density(x1), main="Densidad de x1", col="aquamarine") #generamos el gráfico de densidad para ver cómo es la distribución
lines(density(y1), col="cadetblue2") #generamos un gráfico con las densidades de las dos variables.
```



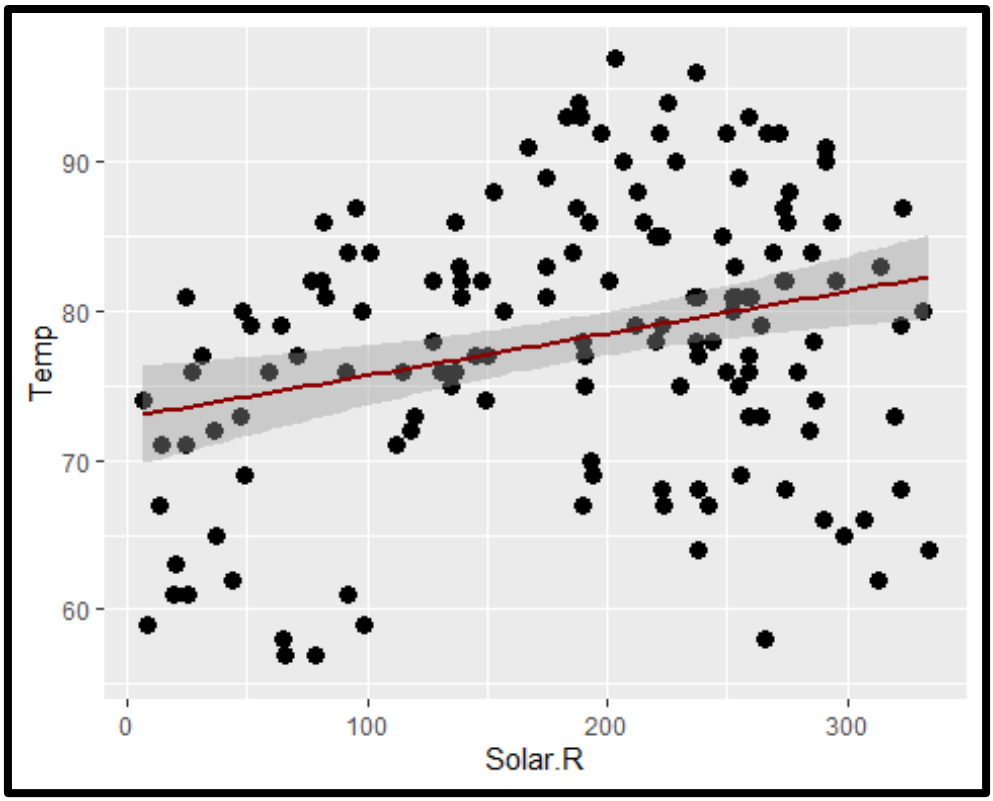
Solución del ejercicio 5:

```
library(ggplot2)
#apartado a)
ggplot(airquality, aes(Solar.R, Temp)) +
  geom_point(size=3) +
  geom_smooth(method="lm", col="darkred")
```

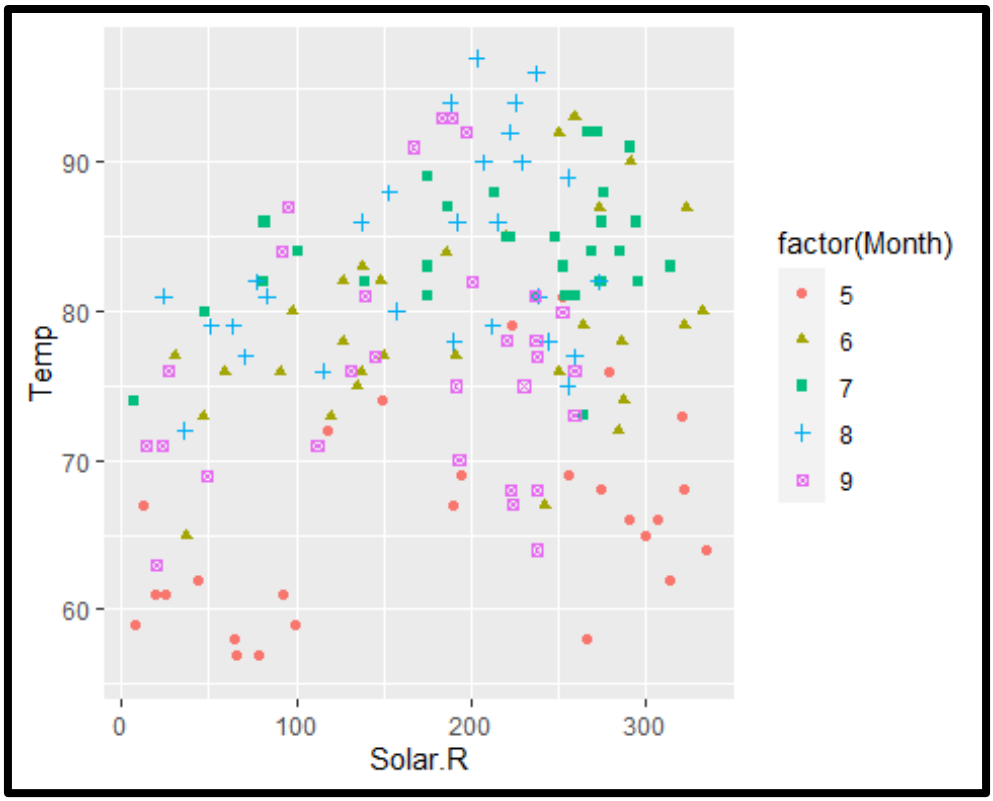
```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## Warning: Removed 7 rows containing non-finite values (`stat_smooth()`).
```

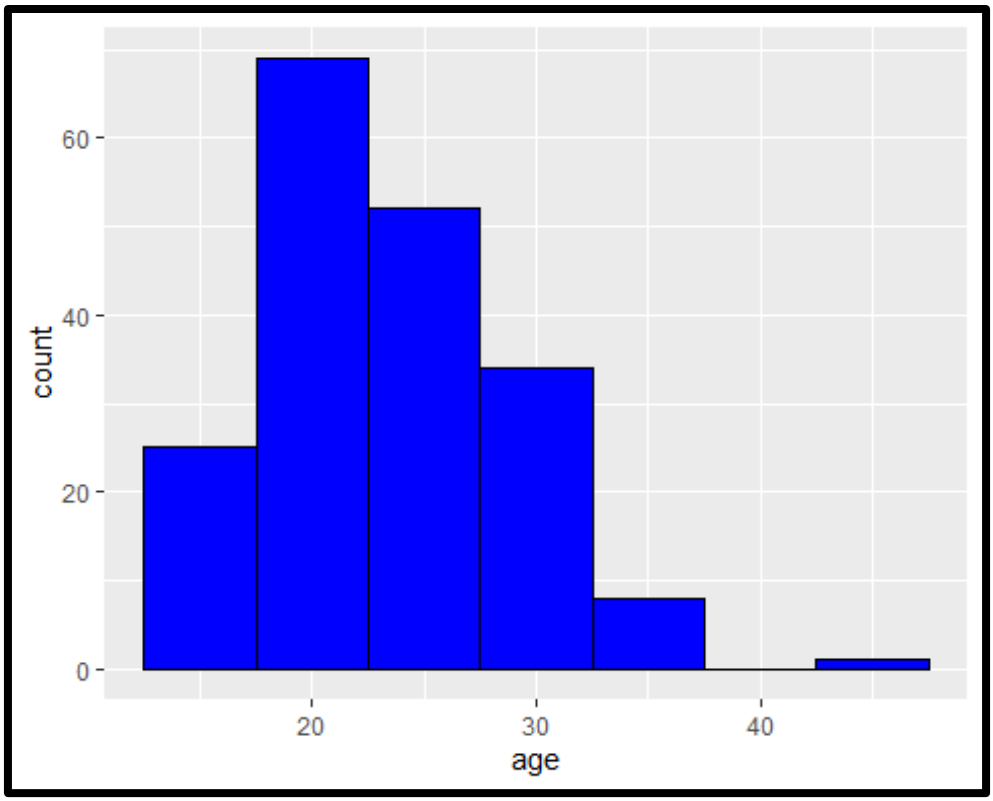
```
## Warning: Removed 7 rows containing missing values (`geom_point()`).
```



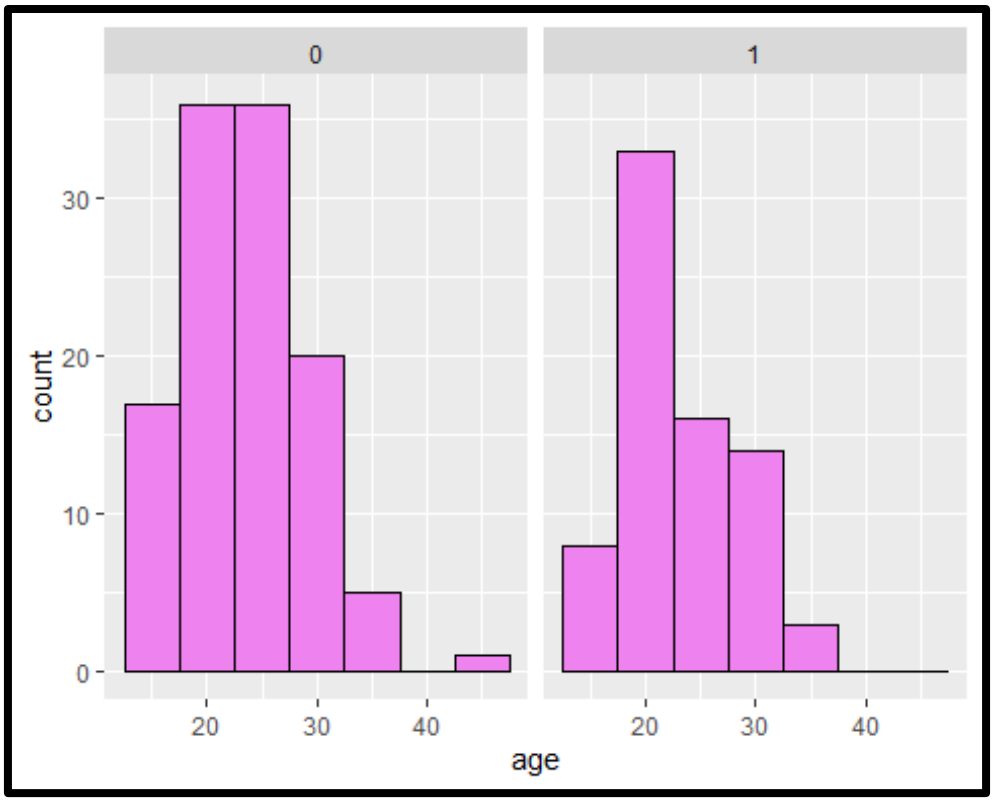

```
#apartado b)
ggplot (data=airquality, aes(x=Solar.R, y=Temp, col=factor(Month),
shape=factor(Month))) +
  geom_point()

## Warning: Removed 7 rows containing missing values (`geom_point()`).
```




```r
#apartado c)
library (MASS)
ggplot(birthwt,aes(age)) +
      geom_histogram(binwidth=5, fill="blue",col="black")
```




```
#apartado d)
ggplot(birthwt,aes(age)) +
      geom_histogram(binwidth=5, fill="violet",col="black")+
      facet_grid(~smoke)
```



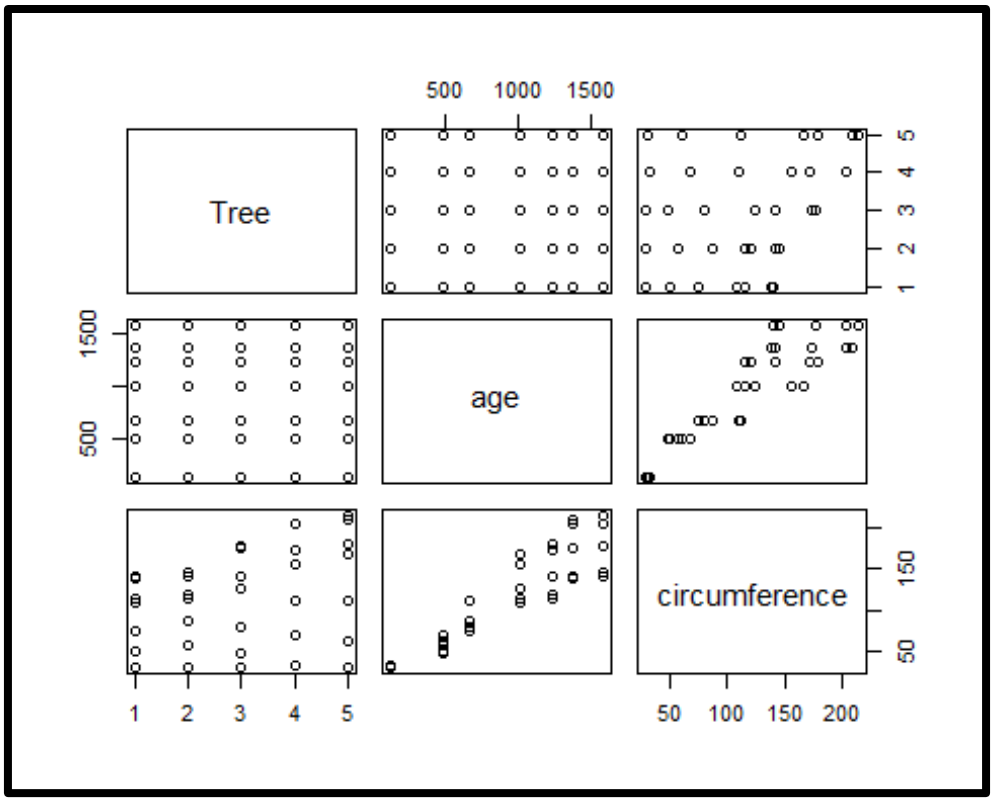
Solución del ejercicio 6:

```
jpeg("migrafic1.jpg") #abrimos el dispositivo gráfico
data(iris)
boxplot(iris$Petal.Width, col="green")
dev.off #cerramos el dispositivo gráfico

## function (which = dev.cur())
## {
##     if (which == 1)
##         stop("cannot shut down device 1 (the null device)")
##     .External(C_devoff, as.integer(which))
##     dev.cur()
## }
## <bytecode: 0x0000024548da0c38>
## <environment: namespace:grDevices>

pdf("migrafic2.pdf") #abrimos el dispositivo gráfico
data(iris)
boxplot(iris$Petal.Width, col="green")
dev.off #cerramos el dispositivo gráfico

## function (which = dev.cur())
## {
##     if (which == 1)
##         stop("cannot shut down device 1 (the null device)")
##     .External(C_devoff, as.integer(which))
```

```
##     dev.cur()
## }
## <bytecode: 0x0000024548da0c38>
## <environment: namespace:grDevices>
```

Solución del ejercicio 7:

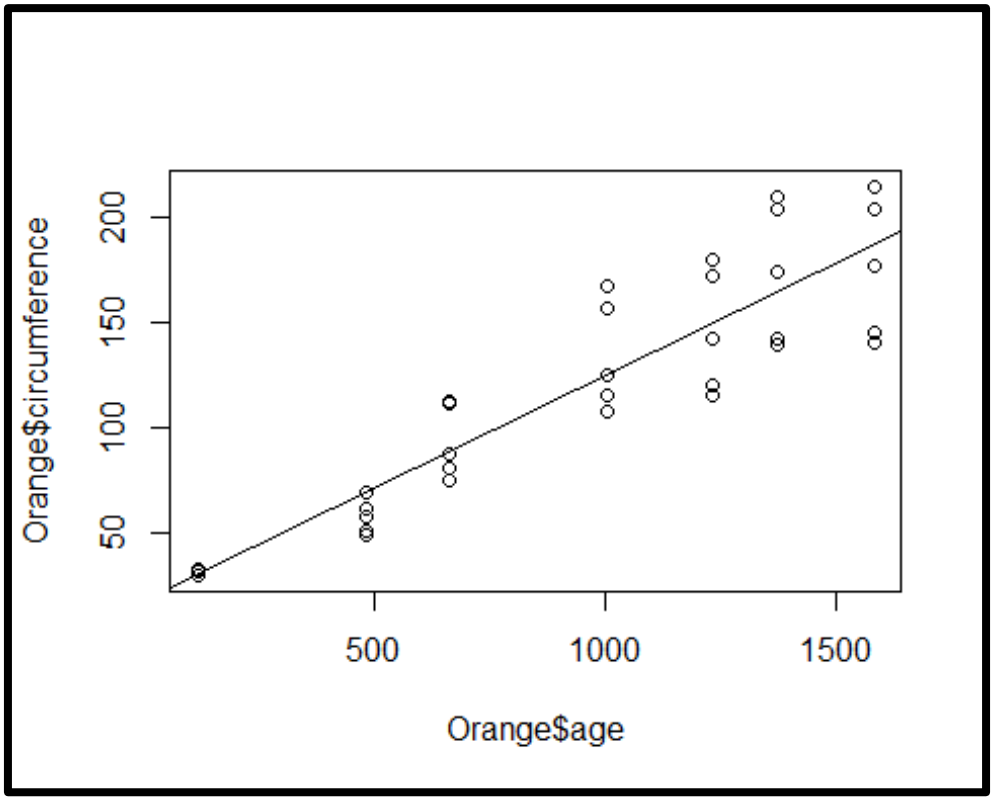
```
#ejercicio de regresión
library(tidyverse)

data("Orange")
head(Orange)

## Grouped Data: circumference ~ age | Tree
##   Tree  age circumference
## 1    1  118            30
## 2    1  484            58
## 3    1  664            87
## 4    1 1004           115
## 5    1 1231           120
## 6    1 1372           142

summary(Orange)

##  Tree       age         circumference
##  3:7   Min.   : 118.0   Min.   : 30.0
##  1:7   1st Qu.: 484.0   1st Qu.: 65.5
##  5:7   Median :1004.0   Median :115.0
##  2:7   Mean   : 922.1   Mean   :115.9
##  4:7   3rd Qu.:1372.0   3rd Qu.:161.5
##        Max.   :1582.0   Max.   :214.0

pairs(Orange)
```




```
cor.test(Orange$age, Orange$circumference)

##
##  Pearson's product-moment correlation
##
## data:  Orange$age and Orange$circumference
## t = 12.9, df = 33, p-value = 1.931e-14
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8342364 0.9557955
## sample estimates:
##       cor
## 0.9135189

mod_reg<-lm(circumference ~ age, data = Orange)
summary(mod_reg)

##
## Call:
## lm(formula = circumference ~ age, data = Orange)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -46.310 -14.946  -0.076  19.697  45.111
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) 17.399650   8.622660   2.018   0.0518 .
## age          0.106770   0.008277  12.900 1.93e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 23.74 on 33 degrees of freedom
## Multiple R-squared:  0.8345, Adjusted R-squared:  0.8295
## F-statistic: 166.4 on 1 and 33 DF,  p-value: 1.931e-14
```

```
plot(Orange$age,Orange$circumference)
abline(mod_reg)
```

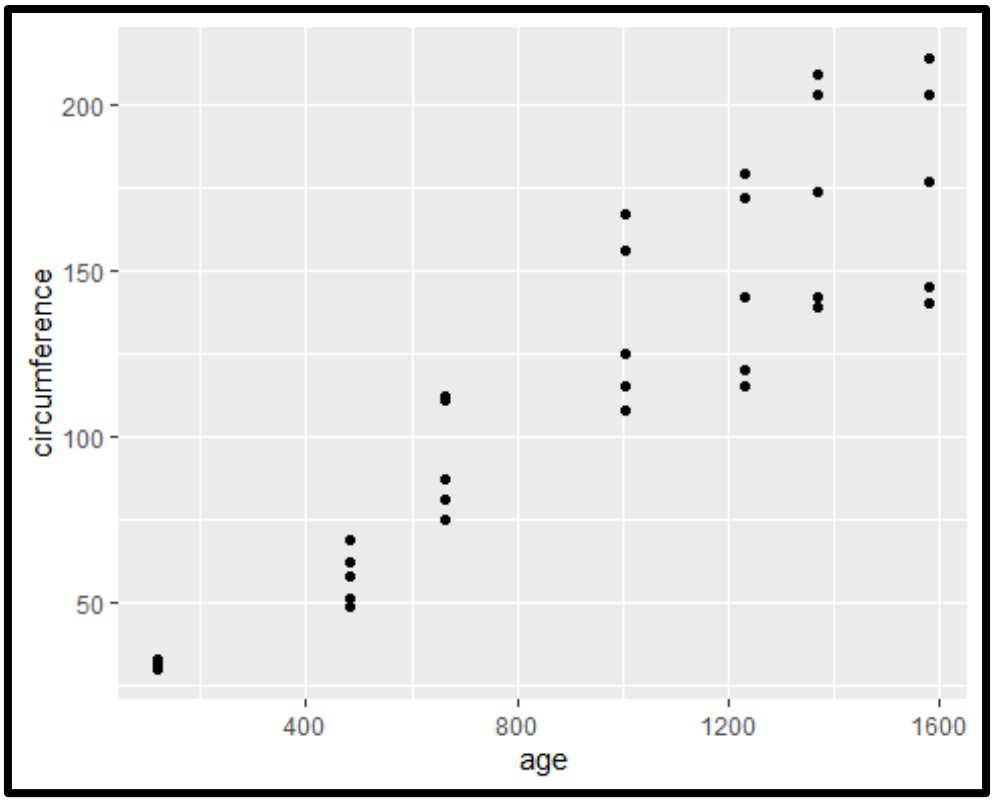



```
#estimación
días <- 600
medida <- 0.1068 * días + 17.3997
print(medida)
```

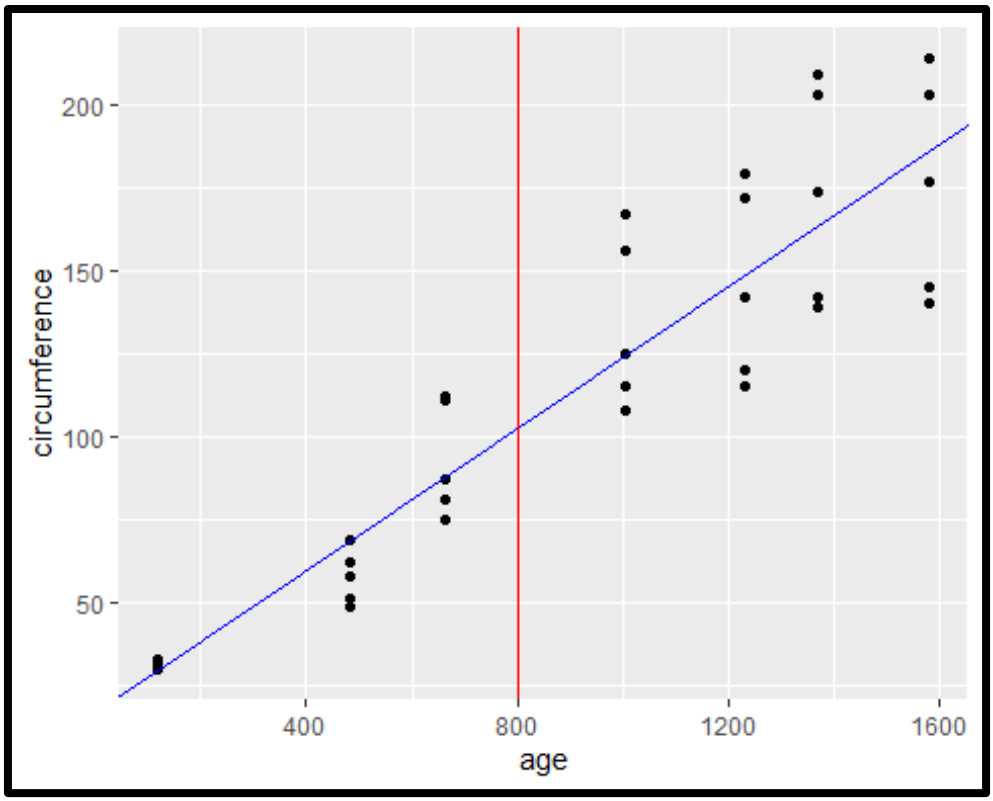
```
## [1] 81.4797
```

Solución del ejercicio 8:

```
Orange %>%
  ggplot(aes(x = age,
             y = circumference)) +
  geom_point()
```




```
Orange %>%
  ggplot(aes(x = age,
             y = circumference)) +
  geom_point() +
  geom_abline(intercept = 17.3997,
              slope = 0.1068,
              col = 'blue') +
  geom_vline(xintercept = 800,
             col = 'red')
```



Solución del ejercicio 9:

```
data("PlantGrowth")
anova(lm(weight ~ group, data = PlantGrowth))

## Analysis of Variance Table
##
## Response: weight
##           Df  Sum Sq Mean Sq F value  Pr(>F)
## group      2  3.7663  1.8832  4.8461 0.01591 *
## Residuals 27 10.4921  0.3886
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Solución del ejercicio 10:

Información sobre el paquete Plotly: https://plotly.com/r/getting-started/

Solución caso práctico:

Diferente solución para cada alumno. Entregar al profesor para su corrección.

Información adicional

Si queréis más información sobre **R Markdown**, **RStudio** o **R**:

R Markdown Oficial

Guía de R Markdown

R Markdown: The Definitive Guide

Trucos de R Markdown en RStudio

R Data for Science

The R Book