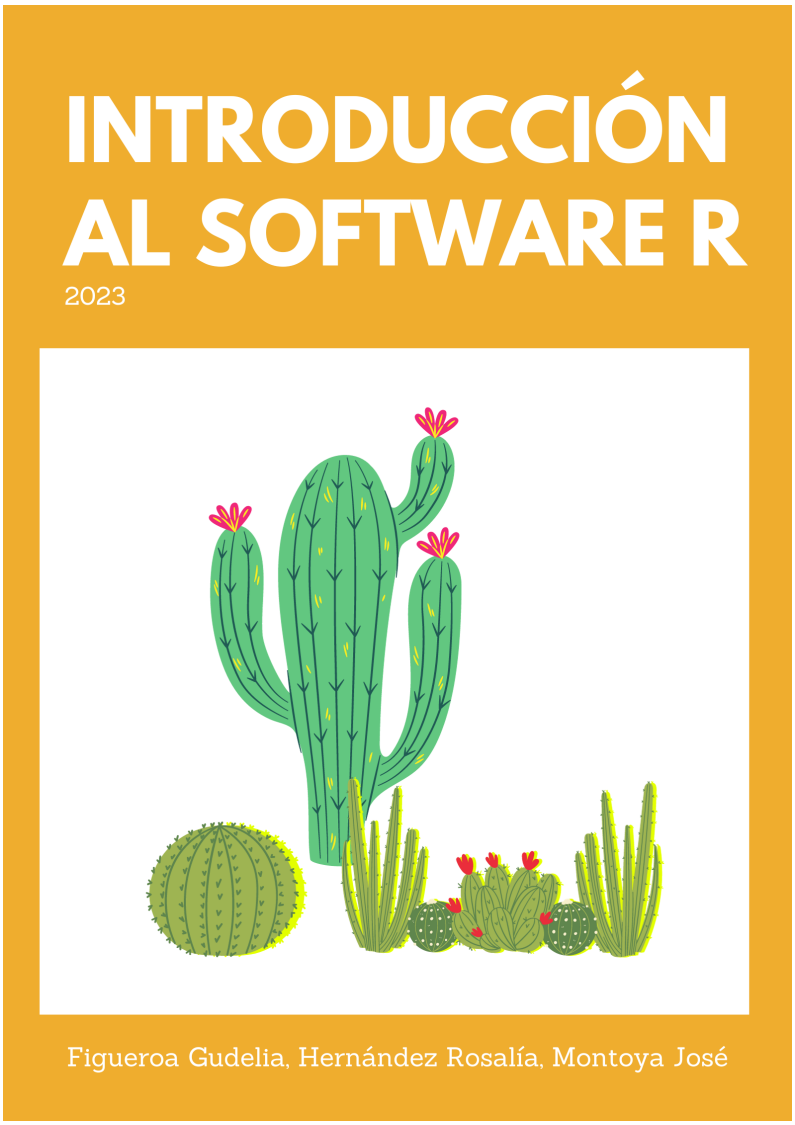
Introducción al software R

Figueroa Preciado Gudelia, Hernández Amador Rosalía Gpe., Montoya Laos José

2023-11-04

Contents

Prefacio



Estas notas tienen el objetivo de proporcionar una introducción práctica al uso del software estadístico R, por medio de la interfaz gráfica RStudio.

Las notas fueron elaboradas como recurso de apoyo en el marco del *Taller de Aplicación de técnicas estadísticas*, impartido a estudiantes del Posgrado en Biociencias de la Universidad de Sonora durante el semestre 2022-2, y fueron adaptadas para utilizarse en cursos de Estadística y Bioestadística durante los semestres 2023-1 y 2023-2 con estudiantes de nivel licenicatura.

Cabe mencionar que para los autores, resulta muy gratificante que las notas hayan sido puestas en práctica durante el mencionado taller por los distintos

asistentes en sus proyectos de tesis, los cuales abarcan una amplia gama de temas en el área de Biociencias en problemáticas de gran relevancia e impacto regional, como estudios relativos a la dinámica y productividad de la flora y fauna en los distintos ecosistemas que se pueden encontrar en el estado de Sonora, desde las zonas desérticas a las costeras.

Los autores pretenden seguir enriqueciendo el texto con el uso subsecuente de las notas, y se espera que este trabajo sea una herramienta útil para los estudiantes que buscan comprender y aplicar las técnicas estadísticas en sus cursos y proyectos, propiciando la conexión entre la teoría y la práctica en esta disciplina.

Chapter 1

El software

Estas notas ofrecen una introducción práctica al uso del software estadístico R mediante la interfaz gráfica RStudio. Es por ésto que para un mejor entendimiento de los conceptos y comandos que aquí se exponen, se exhorta al usuario a poner en práctica las instrucciones aquí documentadas.

1.1 R y RStudio

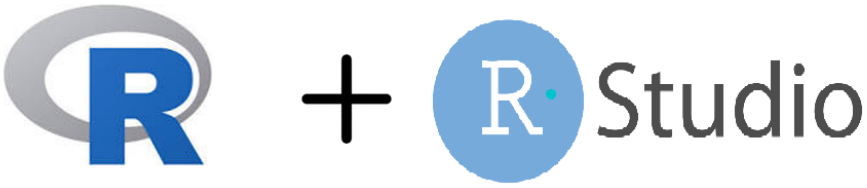
El software **R** es tanto un entorno de programación, como un lenguaje de programación, diseñado para hacer análisis estadístico. Es de código abierto, gratuito y multiplataforma, características que lo han convertido en una herramienta ampliamente utilizada en el campo de la estadística y el análisis de datos.

Por otro lado, **RStudio** es un entorno de desarrollo integrado (IDE) para trabajar con el lenguaje de programación R. Es una plataforma amigable para el usuario, y entre otras propiedades, facilita la escritura de código, visualización, generación de informes reproducibles y la exportación de resultados en diversos formatos.

El entorno RStudio, puede utilizarse de manera gratuita ingresando a través de una cuenta en Posit https://posit.co/ o mediante la instalación previa de R y RStudio en gran variedad de sistemas operativos.

Para instalar R, podemos ingresar a la página https://www.r-project.org y descargar e instalar la última versión del software.

A continuación, para instalar RStudio, ingresamos a la página https://www.posit.co y descargamos e instalamos la interfaz gráfica RStudio.



1.2 RScripts

Un **Rscript** es un archivo de texto compuesto de una serie de comandos e instrucciones escritos en el lenguaje de programación R, que al ser ejecutados realizan tareas específicas como leer y manipular datos, realizar análisis estadísticos o generar gráficos.

Una vez instalado RStudio, podemos abrir un RScript nuevo para escribir y documentar la secuencia de comandos que nos permitirá realizar alguna tarea determinada. Una buena práctica es incluir comentarios en este documento que describan lo que hace cada comando, y que nos permita documentar el código y facilitar su comprensión.

Los Rscripts permiten la reproducibilidad de los análisis resultantes, pues al guardar y compartir un Rscript con otros usuarios, éstos pueden ejecutar los mismos comandos para obtener resultados idénticos.

1.3 Librería del usuario

Al instalar el software R, se instala automáticamente una colección de **paquetes del sistema base**, que se almacenan en una librería. Estos paquetes se componen de funciones básicas del sistema que nos permiten realizar tareas comunes para la manipulación y el análisis de datos.

Adicionalmente, podemos instalar paquetes que nos permiten extender la funcionalidad del sistema base de R. Por ejemplo, podemos ampliar la capacidad de R para la generación de gráficos más atractivos, o para el análisis de datos en áreas especializadas como el estudio de series temporales, o para realizar análisis espaciales con el manejo de datos geoespaciales.

En el repositorio centralizado CRAN (Comprehensive R Archive Network) se albergan miles de paquetes creados por la comunidad de usuarios de R, disponibles para su descarga.

Para utilizar un paquete en R, primero debemos instalarlo y luego cargarlo en la sesión. Una vez cargado, podremos acceder a las funciones y capacidades que proporciona. A lo largo de este texto se explica el uso de una gran variedad de paquetes especializados para ejecutar ciertas tareas.

1.4 Datasets integrados

Existen algunos conjuntos de datos que se incluyen como parte de la instalación del sistema base de R, que se denominan **datasets**, los cuales son proporcionados por los desarrolladores del software y están disponibles para que los usuarios los utilicen directamente en su sesión, sin necesidad de descargar o importar datos desde fuentes externas.

Los datasets integrados en el sistema base de R son útiles para realizar ejemplos y pruebas, así como para aprender y practicar análisis de datos en R.

En estas notas se muestra el uso de algunas técnicas estadísticas aplicadas en datasets cargados en el sistema base de R.

Chapter 2

Tipos de datos y objetos en R

R es un lenguaje de programación orientado a objetos, lo que significa que se centra en la manipulación de objetos y datos. Estos objetos nos permiten almacenar y manipular información.

Por ejemplo, en una **variable** podemos almacenar datos, los cuales pueden ser de distintos tipos como cadenas de caracteres, números enteros, números decimales, números complejos y valores lógicos (verdadero o falso).

Por otro lado, las **funciones** en R, son objetos que nos permiten manipular información para realizar operaciones específicas. Las funciones pueden tomar **argumentos o parámetros** que determinan cómo se ejecutan esas operaciones. Las funciones en R pueden ser funciones propias del lenguaje, o funciones definidas por el mismo usuario.

Se describen a continuación algunos otros objetos comunmente utilizados en R.

2.1 Vectores

Los **vectores** son una de las estructuras de datos más fundamentales en R. Un vector es una secuencia ordenada de elementos del mismo tipo, y pueden ser de tipo numérico, de caracteres, lógicos u otros tipos.

Por ejemplo, (0,1, 2, 3, 25, 23) es un vector numérico, que puede ser creado utilizando la función `c()` del sistema base de R la cual concatena elementos, y puede ser almacenado en la variable `x` mediante el símbolo de asignación `<-`:

```
x <- c(0, 1, 2, 3, 25, 23)
x
```

```
## [1]  0  1  2  3 25 23
```

Podemos utilizar otras funciones para examinar distintas características de los objetos que creamos o que importamos. Por ejemplo, para verificar el tipo de objeto que almacenamos en la variable `x` podemos usar la función `class()`,

```
class(x)
```

```
## [1] "numeric"
```

que indica que `x` es un vector numérico. Si además queremos investigar el tipo de datos que componen al objeto `x`, utilizamos la función `typeof()`,

```
typeof(x)
```

```
## [1] "double"
```

que nos indica que los elementos numéricos son de hecho valores reales.

Otras características que podemos examinar es la longitud del vector,

```
length(x)
```

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

o incluso podemos efectuar operaciones con los elementos almacenados en él, como por ejemplo, podemos sumar sus entradas con el comando `sum()`

```
sum(x)
```

```
## [1] 54
```

Otros ejemplos de vectores son:

```
y <- c("A", "B", "C") # vector de caracteres

z <- c(T, T, F, F) # vector de elementos lógicos

w <- c(5:9) # vector numérico con elementos enteros sucesivos del 5 al 9
```

Podemos también crear nuevos vectores a partir de otros dados, por ejemplo utilizando la función `rep()`

```
rep(y, times = 4)
```

```
##  [1] "A" "B" "C" "A" "B" "C" "A" "B" "C" "A" "B" "C"
```

instrucción con la que se ha concatenado 4 veces el vector `y`.

2.2 Matrices

Las **matrices** son estructuras bidimensionales en la que los datos están organizados en renglones y columnas. Todas las entradas de una matriz deben constar

de elementos del mismo tipo de datos.

Por ejemplo, podemos manipular vectores para crear una matriz. Con la función `cbind()`, se pueden unir dos vectores para formar las columnas de una matriz

```
vec1 <- c(1, 2, 3, 4, 5)
vec2 <- c(-1, -2, -3, -4, -5)
cbind(Columna_1 = vec1, Columna_2= vec2)
```

```
##      Columna_1 Columna_2
## [1,]         1        -1
## [2,]         2        -2
## [3,]         3        -3
## [4,]         4        -4
## [5,]         5        -5
```

o combinar por filas los vectores utilizando la función `rbind()`.

```
rbind(Renglon_1 = vec1,Renglon_2 = vec2)
```

```
##           [,1] [,2] [,3] [,4] [,5]
## Renglon_1    1    2    3    4    5
## Renglon_2   -1   -2   -3   -4   -5
```

2.3 Dataframes

Un **data frame** es una estructura bidimensional que se utiliza generalmente para almacenar datos tabulares. Cada columna de un data frame puede ser de un tipo diferente.

Los data frames son similares a las tablas de una base de datos u hojas de cálculo, donde las columnas representan variables, y los renglones corresponden a observaciones.

Existen algunos comandos que son especialmente útiles para indagar el contenido de una base de datos, especialmente cuando éstas son de gran tamaño. Por ejemplo, supongamos que tenemos los tratamientos 1, 2 y 3, y que para cada uno de los tratamientos se utilizaron distintas unidades experimentales sobre las cuales se tomó una medición:

```
# Almacenamos los datos en dos columnas
datos <- data.frame(Tratamiento = rep(c("1", "2", "3"), 6),
                    Tiempo = c(14, 16, 13,
                               24, 25, 21,
                               18, 19, 14,
                               12, 16, 17,
                               23, 22, 21,
                               25, 23, 22)
```

```
                    )
datos
```

```
##    Tratamiento Tiempo
## 1            1     14
## 2            2     16
## 3            3     13
## 4            1     24
## 5            2     25
## 6            3     21
## 7            1     18
## 8            2     19
## 9            3     14
## 10           1     12
## 11           2     16
## 12           3     17
## 13           1     23
## 14           2     22
## 15           3     21
## 16           1     25
## 17           2     23
## 18           3     22
```

Podemos analizar la estructura de la tabla utilizando la función `str()`,

```
str(datos)
```

```
## 'data.frame':    18 obs. of  2 variables:
##  $ Tratamiento: chr  "1" "2" "3" "1" ...
##  $ Tiempo     : num  14 16 13 24 25 21 18 19 14 12 ...
```

que nos indica las características generales del dataframe, esto es, su tamaño, los nombres de las columnas, y el tipo de datos almacenados en cada columna.

Puede también ser útil visualizar los primeros renglones de la tabla, lo que podemos solicitar utilizando la función `head()`,

```
# Imprimir los primeros tres renglones del dataframe
head(datos,3)
```

```
##   Tratamiento Tiempo
## 1           1     14
## 2           2     16
## 3           3     13
```

y de manera similar, para visualizar lo últimos tres renglones, podmeos utilizar la función `tail()`

```
# Imprimir los últimos tres renglones
tail(datos,3)
```

```
##    Tratamiento Tiempo
## 16           1     25
## 17           2     23
## 18           3     22
```

Al conocer el tipo de contenido de las bases de datos, tendremos un mejor entendiemiento de cómo manipular la información para realizar análisis más específicos.

Chapter 3

Lectura de archivos

En R podemos importar una gran variedad de tipos de archivos, utilizando funciones propias del sistema base o funciones de otros paquetes.

Para leer bases de datos desde un archivo en nuestra computadora, debemos cuidar que éste se encuentre almacenado en el directorio de trabajo.

A continuación se describé cómo importar datos desde formatos de archivos más comunmente utilizados.

3.1 Archivos de texto

Para leer archivos en formato **TXT (texto plano)** podemos usar la función `read.delim()`:

```
Datostxt <- read.delim("analgesics.txt")
head(Datostxt)
```

```
##   Pain_level Drug
## 1          4    A
## 2          5    A
## 3          4    A
## 4          3    A
## 5          2    A
## 6          4    A
```

3.2 Archivos csv

Para leer archivos en formato **CSV (valores separados por comas)** podemos utilizar la función `read.csv`,

```
Datoscsv <- read.csv("calificaciones.csv", header = T)
head(Datoscsv)
```

```
##   X  id female race ses schtyp prog read write math science socst
## 1 1  70      0    4   1      1    1   57    52   41      47    57
## 2 2 121      1    4   2      1    3   68    59   53      63    61
## 3 3  86      0    4   3      1    1   44    33   54      58    31
## 4 4 141      0    4   3      1    3   63    44   47      53    56
## 5 5 172      0    4   2      1    2   47    52   57      53    61
## 6 6 113      0    4   2      1    2   44    52   51      63    61
```

3.3 Archivos de Excel

Para importar datos desde archivos de Excel, en formato **XLSX (valores separados por tabulaciones)** podemos utilizar la función `read_xlsx` del paquete `readxl`, que debemos agregar a la librería de usuario.

Para ésto descargamos primero el paquete, utilizando la función `install.packages()`, y a continuación lo cargamos a la sesión, para poder utilizar las funciones que contiene, utilizando el comando `library()`:

```
# Instalación del paquete readxl
# install.packages("readxl")

# Carga del paquete en la sesión de trabajo
library(readxl)
```

y posteriormente, leemos los datos,

```
DatosXlsx <- read_xlsx("Resultados.xlsx")
```

```
## New names:
## * `` -> `...1`
```

```
head(DatosXlsx)
```

```
## # A tibble: 6 x 12
##    ...1    id female  race   ses schtyp  prog  read write  math science socst
##   <dbl> <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl> <dbl>
## 1     1    70      0     4     1      1     1    57    52    41      47    57
## 2     2   121      1     4     2      1     3    68    59    53      63    61
## 3     3    86      0     4     3      1     1    44    33    54      58    31
## 4     4   141      0     4     3      1     3    63    44    47      53    56
## 5     5   172      0     4     2      1     2    47    52    57      53    61
## 6     6   113      0     4     2      1     2    44    52    51      63    61
```

Para cada tipo de archivo, es posible encontrar funciones o paquetes específicos que facilitan su importación y el procesamiento de datos.

Chapter 4

Análisis exploratorio de datos

Para comprender el comportamiento de un conjunto de datos, antes de aplicar técnicas más especializadas de la estadística, se suele realizar en primer lugar un análisis exploratprio de datos, con el objetivo de visualizar y comprender la distribución de los datos.

4.1 Ejemplo 1

Supongamos que registramos la cantidad de respuestas correctas logradas en un exámen, aplicado a un grupo de 15 estudiantes:

```
respuestas <- c(2, 4, 4, 3, 5, 5, 6, 3, 7, 8, 3, 4, 5, 5, 4)
```

podemos crear una **tabla de frecuencias absolutas** con la siguiente instrucción:

```
table(respuestas)
```

```
## respuestas
## 2 3 4 5 6 7 8
## 1 3 4 4 1 1 1
```

y elaborar una gráfica sencilla, pero de gran utilidad como lo es un **diagrama de barras**

```
barplot(table(respuestas),
        main = "Número de respuestas correctas",
        col = "blue")
```

Además, podemos calcular fácilmente algunas medidas descriptivas. Por ejemplo, podemos examinar el valor mínimo y máximo ordenando las observaciones como sigue:

```
sort(respuestas)
```

```
##  [1] 2 3 3 3 4 4 4 4 5 5 5 5 6 7 8
```

calcular la **media**,

```
mean(respuestas)
```

```
## [1] 4.533333
```

la **mediana**,

```
median(respuestas)
```

```
## [1] 4
```

la **varianza**,

```
var(respuestas)
```

```
## [1] 2.552381
```

la **desviación estándar**,

```
sd(respuestas)
```

```
## [1] 1.597617
```

los **cuartiles**,

```
quantile(respuestas)
```

```
##   0%  25%  50%  75% 100%
##  2.0  3.5  4.0  5.0  8.0
```

especificar otros cuantiles deseados

```
quantile(respuestas, c(0.1,0.5,0.9))
```

```
## 10% 50% 90%
## 3.0 4.0 6.6
```

o podemos solicitar un resumen de las estadísticas descriptivas anteriores con la función `summary()`,

```
summary(respuestas)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   3.500   4.000   4.533   5.000   8.000
```

4.2 Ejemplo 2

Consideremos ahora otro conjunto de datos,

```
datos2 <- c(2.4, 2.6, 1.3, 2.7, 3.1, 3.4, 3.6, 2.5, 3.7,
            3.9, 4.2, 4.1, 4.5, 5.2, 3.3, 1.6, 3.7,7.6)
```

Otra herramienta visual de la estadística descriptiva que podemos realizar en un conjunto de datos para analizar su distribución, es la construcción de un **diagrma de tallo y hojas**. Para elaborarlo, primero ordenamos los datos,

```
sort(datos2)
```

```
##  [1] 1.3 1.6 2.4 2.5 2.6 2.7 3.1 3.3 3.4 3.6 3.7 3.7 3.9 4.1 4.2 4.5 5.2 7.6
```

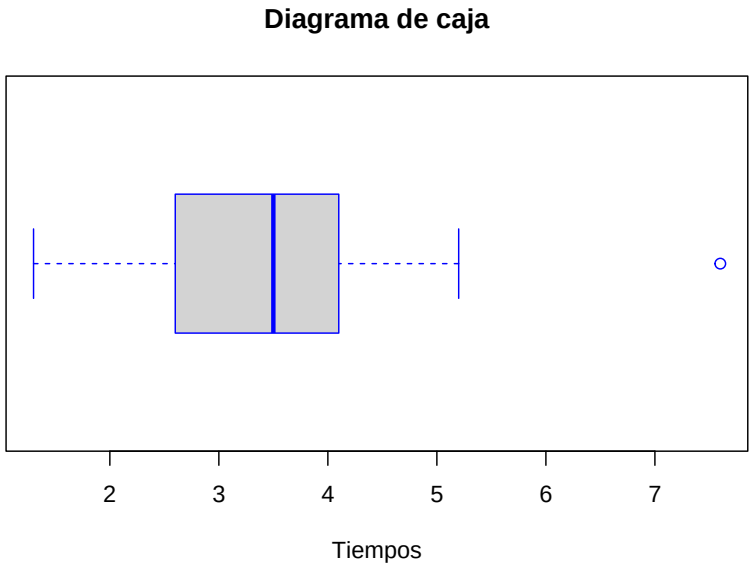
y graficamos el diagrama con los datos ordenados

```
stem(datos2, scale=2)
```

```
##
##   The decimal point is at the |
##
##   1 | 36
##   2 | 4567
##   3 | 1346779
##   4 | 125
##   5 | 2
##   6 |
##   7 | 6
```

o podemos elaborar también **un diagrama de caja**,

```
boxplot(datos2, horizontal = T, border = "blue",
        xlab = "Tiempos", main = "Diagrama de caja")
```



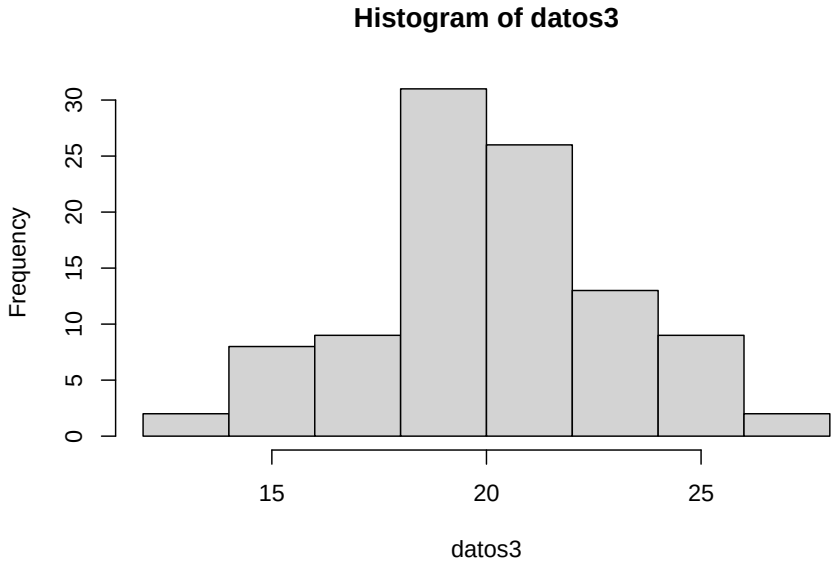
4.3 Ejemplo 3

Simulemos ahora una muestra aleatoria de 100 observaciones provenientes de una distribución normal, con media $\mu = 20$ y desviación estándar $\sigma = 3$,

```
datos3 <- rnorm(100, 20, 3)
```

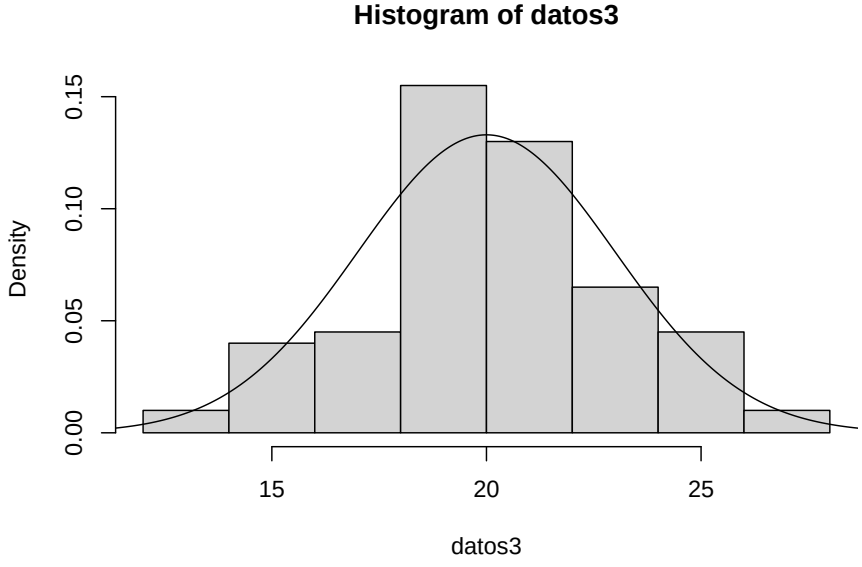
Podemos analizar su distribución por medio de un histograma,

```
hist(datos3)
```



y personalizar la gráfica añadiendo una curva normal,

```
hist(datos3, freq = FALSE)
ejenorm <- seq(10, 30, 0.01)
lines(ejenorm, dnorm(ejenorm, 20,3))
```



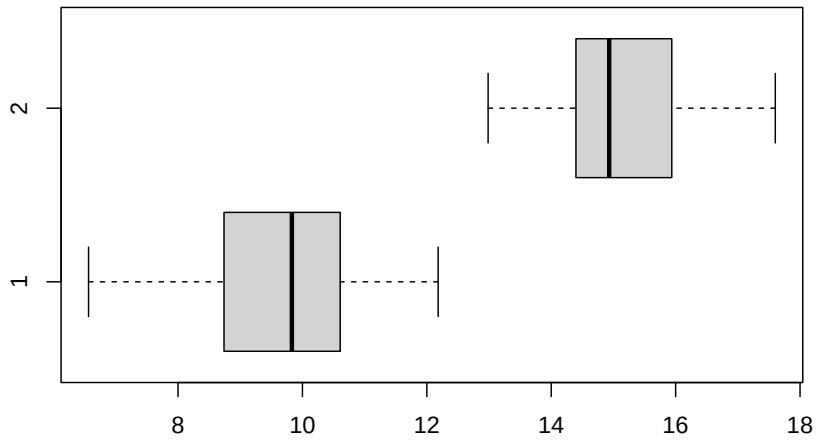
4.4 Ejemplo 4

Comparemos ahora varios conjuntos de datos. Para esto, generemos dos muestras de una distribución normal, y una tercer muestra de una distribución chi cuadarada,

```
x1 <- rnorm(30, 10, 1.5)
x2 <- rnorm(30, 15, 1)
x3 <- rchisq(30, 4)
```

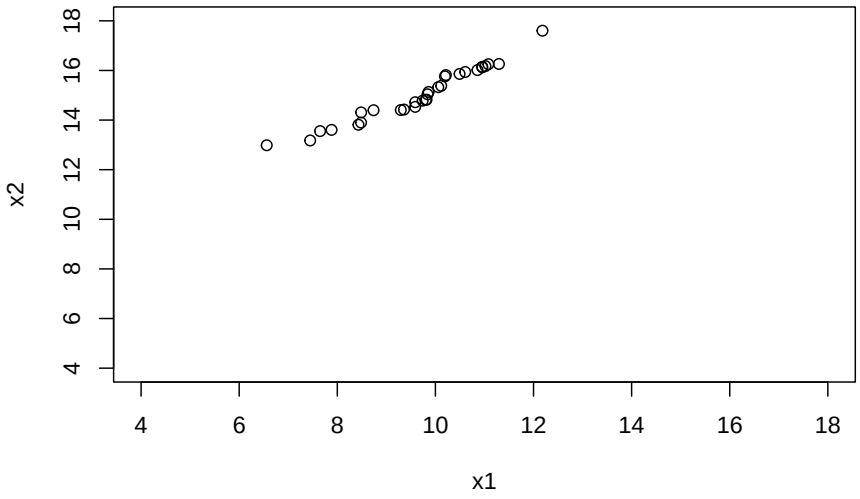
Comparemos los datos de las primeras dos muestras mediante diagramas de cajas

```
boxplot(x1,x2, horizontal = TRUE)
```



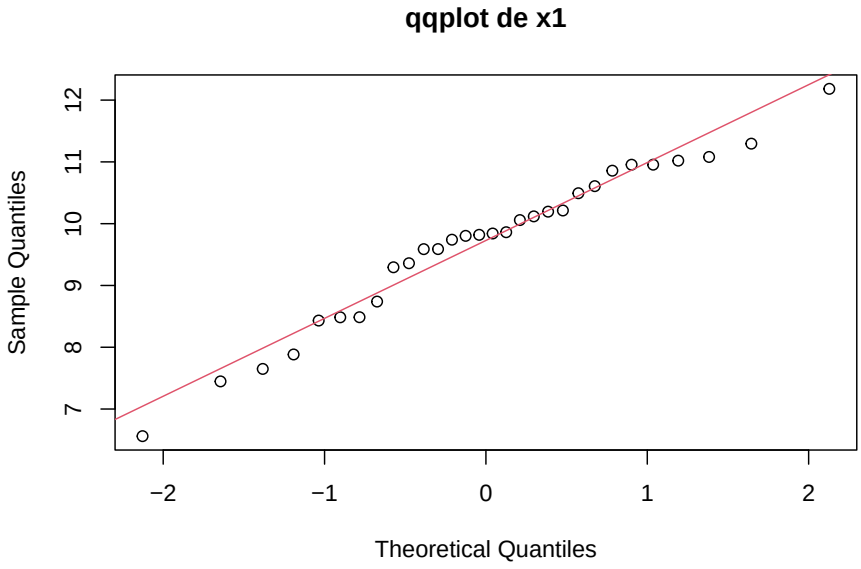
Podemos efectuar un **gráfico cuantil-cuantil (o gráfico Q-Q)**, para comparar si las dos muestras `x1` y `x2` provienen de la misma distribución

```
qqplot(x1, x2, xlim = c(4,18), ylim=c(4,18))
```



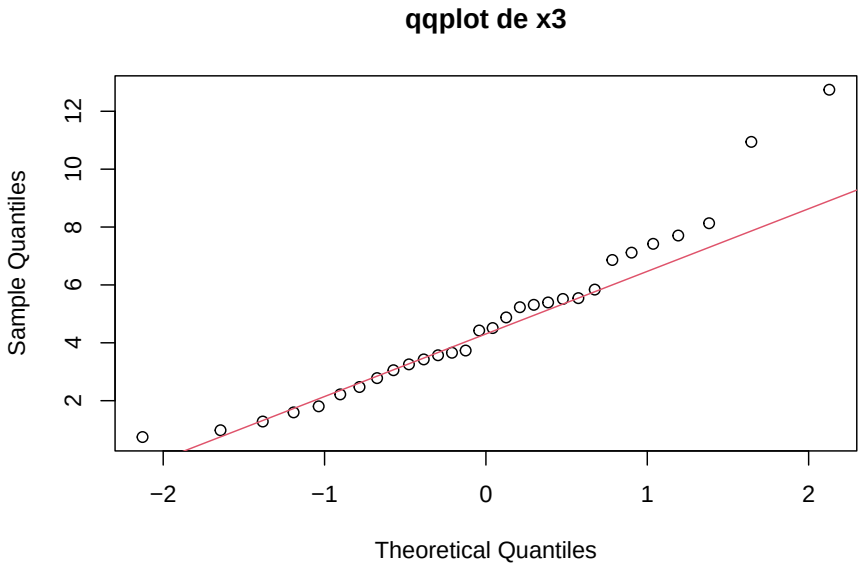
O podemos reemplazar una de las muestras en la comparación anterior, por los cuantiles de una distribución normal, para determinar si cada una de la muestras sigue una distribución normal. Por ejemplo, para la muestra `x1` obtenemos el siguiente gráfico,

```
qqnorm(x1, main="qqplot de x1")
qqline(x1,col=2)
```



y para la muestra `x3` el siguiente,

```
qqnorm(x3,main="qqplot de x3")
qqline(x3,col=2)
```



Chapter 5

Análisis descriptivo para variables correlacionadas

Para comprender la relación entre dos variables y tomar decisiones sobre posibles análisis posteriores, un estudio común en estadística suele consistir en investigar si dos variables se encuentran correlacionadas. Usualmente se calcula un valor numérico de dicha correlación y se complementa el estudio con un análisis gráfico.

5.1 Ejemplo 1

Consideremos un experimento donde se registraron datos de la edad gestacional y peso al nacer de un grupo de 17 bebés.

```
edad_ges <- c(34.7, 36,29.3, 40.1, 35.7, 42.4, 40.3, 37.3, 40.9,
              38.3, 38.5, 41.4, 39.7, 39.7, 41.1, 38, 38.7)

peso_nacer <- c(1895, 2030, 1440, 2835, 3090, 3827, 3260, 2690,
                3885, 2920, 3430, 3657, 3685, 3345, 3260, 2680, 2005)
```

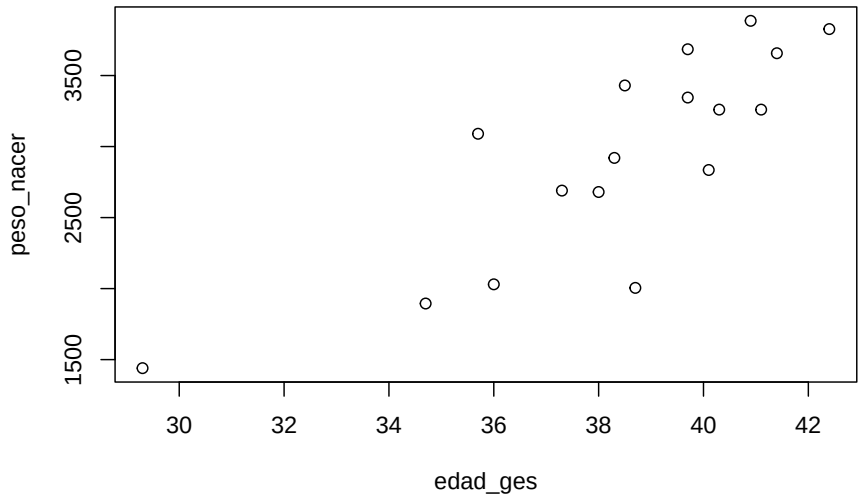
Para comprender la relación entre estas dos variables, se puede calcular su **correlación**,

```
cor(edad_ges,peso_nacer)
```

```
## [1] 0.8197466
```

y reforzar el resultado numérico obtenido, mediante un **diagrama de dispersión**

```
plot(edad_ges,peso_nacer)
```



Podemos ajustar un modelo de **regresión lineal**, para establecer la relación lineal de una variable respecto a la otra,

```
reg1 <- lm(peso_nacer ~ edad_ges)
```

y solicitar los detalles del modelo con la siguiente instrucción,

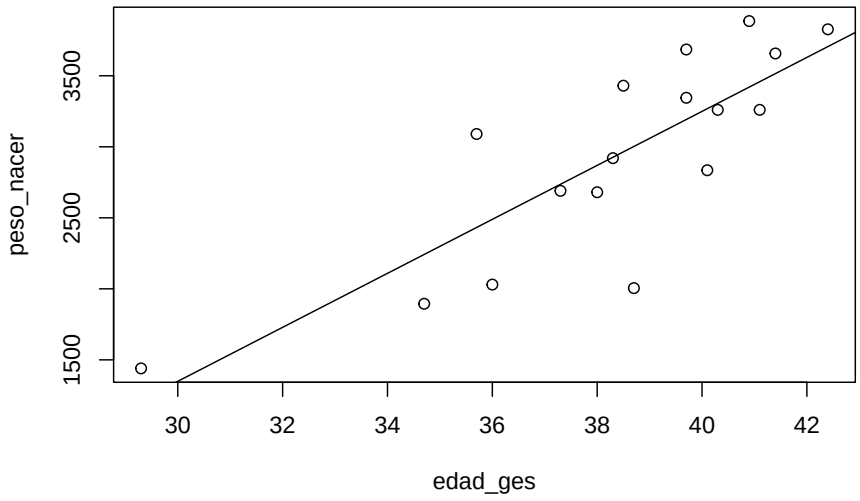
```
summary(reg1)
```

```
##
## Call:
## lm(formula = peso_nacer ~ edad_ges)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -997.12 -198.17   -6.12  224.06  657.93
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4351.62    1319.06  -3.299  0.00487 **
## edad_ges      190.02      34.28   5.543 5.63e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 432.8 on 15 degrees of freedom
## Multiple R-squared:  0.672,  Adjusted R-squared:  0.6501
```

```
## F-statistic: 30.73 on 1 and 15 DF,  p-value: 5.629e-05
```

podemos además ajustar la línea de regresión en el diagrama de dispersión,

```
plot(edad_ges,peso_nacer)
abline(reg1)
```



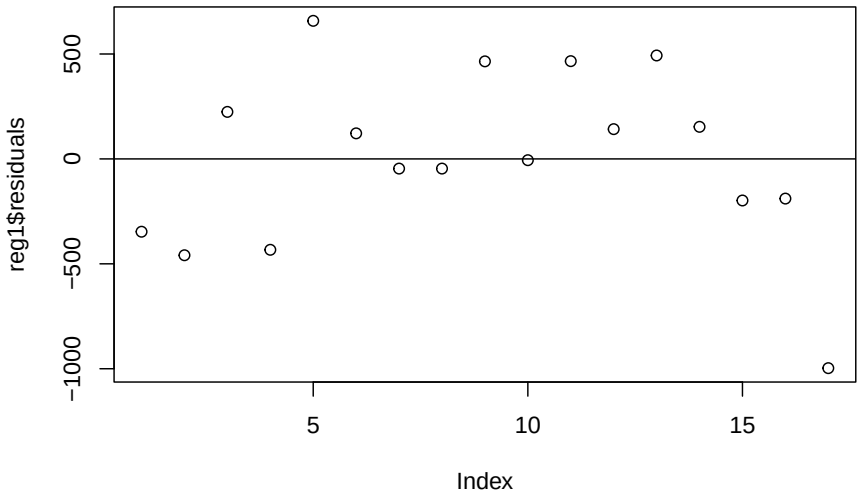
Para extraer información específica asociada al modelo, como por ejemplo sus coeficientes, podemos usar la siguiente instrucción,

```
reg1$coefficients
```

```
## (Intercept)    edad_ges
##  -4351.6240    190.0193
```

y analizar el comportamiento de los residuales gráficamente

```
plot(reg1$residuals)
abline(h=0)
```



5.2 Ejemplo 2

A manera de ilustración del uso de datasets integrados en R, y para destacar la importancia de analizar un conjunto de datos desde diferentes perspectivas, a continuación se analiza el dataset `Anscombre`, el cual fué creado por el estadístico Francis Ascombre y publicado en 1973 en The American Statistician.

En primer lugar cargamos el conjunto de datos Anscombe,

```
data(anscombe)
```

y podemos mostrar el contenido del dataset

```
anscombe
```

```
##    x1 x2 x3 x4    y1   y2    y3    y4
## 1  10 10 10  8  8.04 9.14  7.46  6.58
## 2   8  8  8  8  6.95 8.14  6.77  5.76
## 3  13 13 13  8  7.58 8.74 12.74  7.71
## 4   9  9  9  8  8.81 8.77  7.11  8.84
## 5  11 11 11  8  8.33 9.26  7.81  8.47
## 6  14 14 14  8  9.96 8.10  8.84  7.04
## 7   6  6  6  8  7.24 6.13  6.08  5.25
## 8   4  4  4 19  4.26 3.10  5.39 12.50
## 9  12 12 12  8 10.84 9.13  8.15  5.56
## 10  7  7  7  8  4.82 7.26  6.42  7.91
```

```
## 11  5  5  5  8  5.68 4.74  5.73  6.89
```

Como podemos observar, este dataset consta de cuatro subconjuntos de datos, con 11 pares de observaciones cada uno del tipo $(x_i, y_i), i = 1, ..., 4$.

Para poder utilizar en el análisis siguiente los nombres de las variables tal y como aparecen en la tabla, utilizamos la instrucción `attach()`,

```
rm(list = ls())
attach(anscombe)
```

Como podemos ver con los siguientes cálculos, estos cuatro subconjuntos tienen propiedades estadísticas muy similares, pues si ajustamos un modelo de regresión a cada subconjunto, obtenemos lo siguiente:

```
regresion1 <- lm(y1 ~ x1)
regresion1
```

```
##
## Call:
## lm(formula = y1 ~ x1)
##
## Coefficients:
## (Intercept)           x1
##      3.0001       0.5001
```

```
regresion2 <- lm(y2 ~ x2)
regresion2
```

```
##
## Call:
## lm(formula = y2 ~ x2)
##
## Coefficients:
## (Intercept)           x2
##       3.001        0.500
```

```
regresion3 <- lm(y3~ x3)
regresion3
```

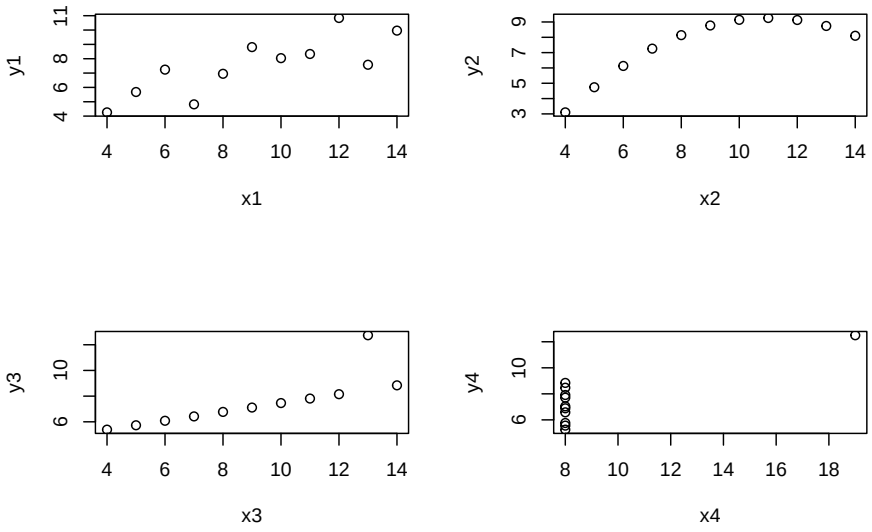
```
##
## Call:
## lm(formula = y3 ~ x3)
##
## Coefficients:
## (Intercept)           x3
##      3.0025       0.4997
```

```
regresion4 <- lm(y4~x4)
regresion4
```

```
##
## Call:
## lm(formula = y4 ~ x4)
##
## Coefficients:
## (Intercept)           x4
##      3.0017       0.4999
```

sin embargo, al momento de representar gráficamente el comportamiento de los cuatro conjuntos de datos mediante diagramas de dispersión, podemos comprobar que difieren significativamente

```
par(mfrow = c(2,2))
plot(x1,y1)
plot(x2,y2)
plot(x3,y3)
plot(x4,y4)
```



Este conjunto de datos, precargado en R, es un ejemplo claro de la importancia de la visualización de datos en el análisis estadístico.

Chapter 6

Intervalos de confianza

Un intervalo de confianza es un rango de valores que se utiliza en estadística para estimar un parámetro desconocido de una población en base a una muestra de datos. Comunmente el parámetro desconocido es la media, una proporción o la varianza. Su objetivo es para proporcionar una medida de la incertidumbre asociada con la estimación del parámetro en cuestión.

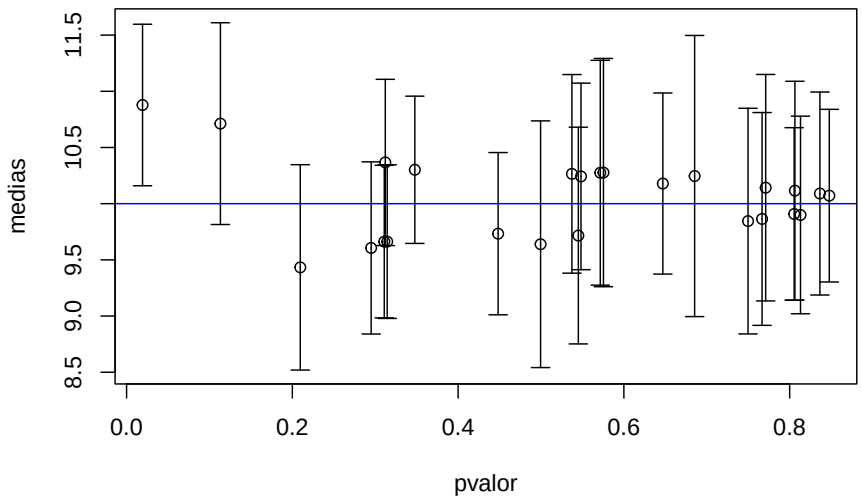
La intención del siguiente ejercicio es comprender el significado de nivel de confianza $1 - \alpha$ de un intervalo de confianza.

Para ésto, se simulan a continuación 25 muestras de tamaño 20 cada una, de datos provenientes de una distribución normal con media $\mu = 10$ y desviación estándar $\sigma = 2$.

Para cada una de las muestras se calcula un **intervalo de confianza** del 95% para la media μ, utilizando un estadístico t-Student. Por medio de una gráfica se explora cuántos de estos intervalos capturaron el verdadero valor de $\mu = 10$:

```
mu <- 10 # media
sigma <- 2 # desviación estándar
n <- 20 # tamaño de muestra
m <- 25 # cantidad de muestras

# inicializamos vectores vacíos
pvalor <- rep(NA,m)
resultado1 <- rep(NA,m)
resultado2 <- rep(NA,m)
medias <- rep(NA,m)

# Repetir m veces el procedimiento
for(i in 1:m)
{
```

```
  x <- rnorm(n,mu,sigma)
  medias[i] <- mean(x)
  pvalor[i] <- t.test(x, alternative = "two.sided",
                      mu=10)$p.value
  resultado1[i] <- t.test(x, alternative = "two.sided",
                          mu=10)$conf.int[1]
  resultado2[i] <- t.test(x, alternative = "two.sided",
                          mu=10)$conf.int[2]
}

# Cargamos el paquete plotrix
library(plotrix)

# Usamos la función plotCI() del paquete plotrix para graficar los IC
plotCI(pvalor, medias, ui = resultado2, li = resultado1,
       yab ="Intervalos al 95% de confianza")
abline(h=mu,col="blue")
```



y podemos detallar la información sobre los 25 intervalos,

```
resultgral <- cbind(pvalor, resultado1,resultado2)
resultgral
```

```
##          pvalor resultado1 resultado2
##  [1,] 0.3477809   9.646192   10.95651
##  [2,] 0.8063723   9.142150   11.08905
```

```
##  [3,] 0.6853893   8.995020   11.49669
##  [4,] 0.5450618   8.751950   10.68031
##  [5,] 0.4482927   9.011083   10.45483
##  [6,] 0.7710948   9.134454   11.14971
##  [7,] 0.8131286   9.020261   10.77839
##  [8,] 0.8478618   9.303247   10.83950
##  [9,] 0.4995012   8.541457   10.73654
## [10,] 0.5753673   9.260954   11.29221
## [11,] 0.0192584  10.159395   11.59655
## [12,] 0.5482903   9.412414   11.07234
## [13,] 0.3144359   8.978627   10.34621
## [14,] 0.8054629   9.140702   10.67609
## [15,] 0.8365133   9.187201   10.99333
## [16,] 0.1132281   9.814496   11.60982
## [17,] 0.7667446   8.916962   10.81076
## [18,] 0.7498114   8.840841   10.84876
## [19,] 0.2951013   8.839982   10.37195
## [20,] 0.2095819   8.519360   10.34675
## [21,] 0.3109084   8.983693   10.34112
## [22,] 0.6470200   9.373389   10.98484
## [23,] 0.5715428   9.274720   11.27571
## [24,] 0.5373050   9.381629   11.14884
## [25,] 0.3121481   9.627332   11.10645
```

Si contamos la cantidad de veces que los intervalos capturaron el valor real $\mu = 10$,

```
cont=0
for(i in 1:m)
{
  if(mu> resultado1[i] && mu < resultado2[i])
    cont=cont+1
}
cont
```

```
## [1] 24
```

y calculamos la proporción de veces que ésto ocurre,

```
cont/25
```

```
## [1] 0.96
```

podemos ver es un valor muy cercano al 95%.

Chapter 7

Intervalo de confianza y prueba de hipótesis para comparar dos medias

La comparación de dos medias es un procedimiento común en estadística que se utiliza cuando se desea determinar si existen diferencias significativas entre las medias de dos grupos o poblaciones. El siguiente ejercicio muestra dos enfoques para la comparación de medias, mediante intervalos de confianza y mediante pruebas de hipótesis.

Para este tipo de procedimientos, debemos verificar primeramente que se cumplen los supuestos de normalidad en las muestras que se asumen independientes, y debemos verificar que no sea rechazado el supuesto de homogeneidad de varianzas.

Consideremos dos muestras independientes de datos:

```
rm(list=ls())
metodoA <- c(23.2,26.6,24.4, 23.5, 22.6, 25.7, 25.5, 22.3,
             22.5, 23.1, 24.6, 25.2, 23.7)

metodoB <- c(25.7, 27.7, 26.2, 27.9, 25.0, 27.9, 26.1, 25.3,
             26.2, 27.4, 27.1, 25.8, 26.4, 27.2)
```

Para determinar normalidad en los conjuntos de datos, podemos aplicar la prueba de Shapiro-Wilk en cada una de éstas,

```
shapiro.test(metodoA)
```

```
##
##  Shapiro-Wilk normality test
```

```
##
## data:  metodoA
## W = 0.94428, p-value = 0.5147
```

```
shapiro.test(metodoB)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  metodoB
## W = 0.93744, p-value = 0.3866
```

Para efectuar una **prueba de hipótesis** para comparar las varianzas σ_1^2 y σ_2^2, la hipótesis nula puede considerarse como $H_0 : \sigma_1^2 = \sigma_2^2$ o equivalentemente $H_0 : \sigma_1^2/\sigma_2^2 = 1$, y la hipótesis alternativa será $H_1 : \sigma_1^2/\sigma_2^2 \neq 1$.

```
var.test(metodoB,metodoA,alternative= "two.sided")
```

```
##
##  F test to compare two variances
##
## data:  metodoB and metodoA
## F = 0.49361, num df = 13, denom df = 12, p-value = 0.2212
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.1523838 1.5564429
## sample estimates:
## ratio of variances
##          0.4936113
```

Podemos notar que no importa el orden en que se ingresa cada uno de los métodos en la prueba de homogeneidad de varianzas.

```
var.test(metodoA,metodoB,alternative="two.sided")
```

```
##
##  F test to compare two variances
##
## data:  metodoA and metodoB
## F = 2.0259, num df = 12, denom df = 13, p-value = 0.2212
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.6424907 6.5623769
## sample estimates:
## ratio of variances
##           2.025886
```

El p-valor resultante de 0.2212 indica que no hay evidencia en la muestra para rechazar la hipótesis nula.

Por lo tanto, satisfechos los supuestos de normalidad y homogeneidad de varianzas, podemos solicitar el intervalo de confianza y la prueba de hipótesis para comparar las medias poblacionales utilizando el comando `t.test()` de R.

En este caso las hipótesis que se plantean son:

$$H_0 : \mu_1 = \mu_2$$
$$H_1 : \mu_1 \neq \mu_2$$

```
t.test(metodoA, metodoB, alternative = "two.sided", var.equal=TRUE)
```

```
##
##  Two Sample t-test
##
## data:  metodoA and metodoB
## t = -5.489, df = 25, p-value = 1.058e-05
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.431235 -1.558875
## sample estimates:
## mean of x mean of y
##  24.06923  26.56429
```

En este caso la muestra da evidencia para rechazar la hipótesis nula, al nivel de significancia del 5%, conclusión que también se obtiene con el intervalo de confianza construido, pues recordemos que si el intervalo de confianza no incluye el cero, se puede concluir que hay una diferencia significativa entre las medias.

Chapter 8

Análisis de varianza en una clasificación

Para comparar las medias de más de dos grupos, podemos utilizar un **ANOVA (Analysis of variance)**. El ANOVA compara las medias de tres o más grupos para determinar si al menos uno de los grupos es significativamente diferente de los demás. Si el ANOVA muestra diferencias significativas, se pueden realizar pruebas post hoc, como la prueba de Tukey, para identificar qué grupos específicos son diferentes entre sí.

En el siguiente ejemplo se ilustra el uso de un ANOVA en una clasificación, el cual se utiliza para comparar las medias cuando solo se tiene un factor o variable independiente.

Por ejemplo, supongamos que se desea evaluar si la dosis suministrada de un tratamiento produce cambios significativos en el crecimiento de cierta clase de planta, y se cuenta con los siguientes registros, que se asumen independientes:

```
altura <- c(12.4, 12.8, 12.2, 13, 14, 14.2, 11.6, 15, 12, 13.2,
            16, 12.6, 14.8, 13, 14, 15, 14, 17, 18, 19, 17.8, 14.4,
            20, 15.8, 17.0, 20.0, 19.6, 18.0, 20.2, 18.0, 21, 14.8,
            19.1, 15.8, 18, 20, 21.1, 22, 19, 18.2)

dosis <- c(rep("dosis_50", 10), rep("dosis_100", 10),
           rep("dosis_200", 10), rep("dosis_400", 10))

rendimiento = data.frame(altura, dosis)
rendimiento
```

```
##    altura     dosis
## 1    12.4  dosis_50
## 2    12.8  dosis_50
```

```
## 3    12.2  dosis_50
## 4    13.0  dosis_50
## 5    14.0  dosis_50
## 6    14.2  dosis_50
## 7    11.6  dosis_50
## 8    15.0  dosis_50
## 9    12.0  dosis_50
## 10   13.2  dosis_50
## 11   16.0 dosis_100
## 12   12.6 dosis_100
## 13   14.8 dosis_100
## 14   13.0 dosis_100
## 15   14.0 dosis_100
## 16   15.0 dosis_100
## 17   14.0 dosis_100
## 18   17.0 dosis_100
## 19   18.0 dosis_100
## 20   19.0 dosis_100
## 21   17.8 dosis_200
## 22   14.4 dosis_200
## 23   20.0 dosis_200
## 24   15.8 dosis_200
## 25   17.0 dosis_200
## 26   20.0 dosis_200
## 27   19.6 dosis_200
## 28   18.0 dosis_200
## 29   20.2 dosis_200
## 30   18.0 dosis_200
## 31   21.0 dosis_400
## 32   14.8 dosis_400
## 33   19.1 dosis_400
## 34   15.8 dosis_400
## 35   18.0 dosis_400
## 36   20.0 dosis_400
## 37   21.1 dosis_400
## 38   22.0 dosis_400
## 39   19.0 dosis_400
## 40   18.2 dosis_400
```

Podemos explorar el comportamiento de la variable `altura`,

```
summary(altura, data = rendimiento)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   11.60   14.00   15.90   16.34   19.00   22.00
```

y comparar el comportamiento por grupos,

```r
summary(altura[dosis == "dosis_100"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   12.60   14.00   14.90   15.34   16.75   19.00
```

```r
summary(altura[dosis == "dosis_50"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   11.60   12.25   12.90   13.04   13.80   15.00
```

```r
summary(altura[dosis == "dosis_200"])
```

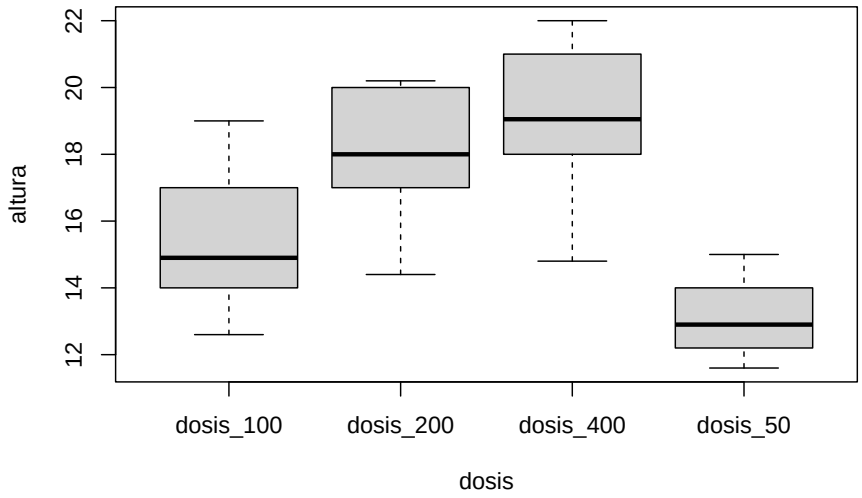
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   14.40   17.20   18.00   18.08   19.90   20.20
```

```r
summary(altura[dosis == "dosis_400"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   14.80   18.05   19.05   18.90   20.75   22.00
```

Además, podemos resumir la información, representada gráficamente por grupos como sigue,

```r
boxplot(altura ~ dosis, data = rendimiento)
```



Para evaluar si las varianzas de los distintos grupos son homogéneas, podemos aplicar la prueba de homogeneidad de varianzas de Bartlett, con la siguiente hipótesis nula H_0 e hipótesis alternativa H_1:

$$H_0 : \sigma_1^2 = ... = \sigma_k^2$$
$$H_1 : \sigma_i^2 \neq \sigma_j^2, \ \forall i \neq j$$

Solicitamos a R que se aplique la prueba con la siguiente instrucción:

```
bartlett.test(altura ~ dosis, data = rendimiento)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  altura by dosis
## Bartlett's K-squared = 4.8838, df = 3, p-value = 0.1805
```

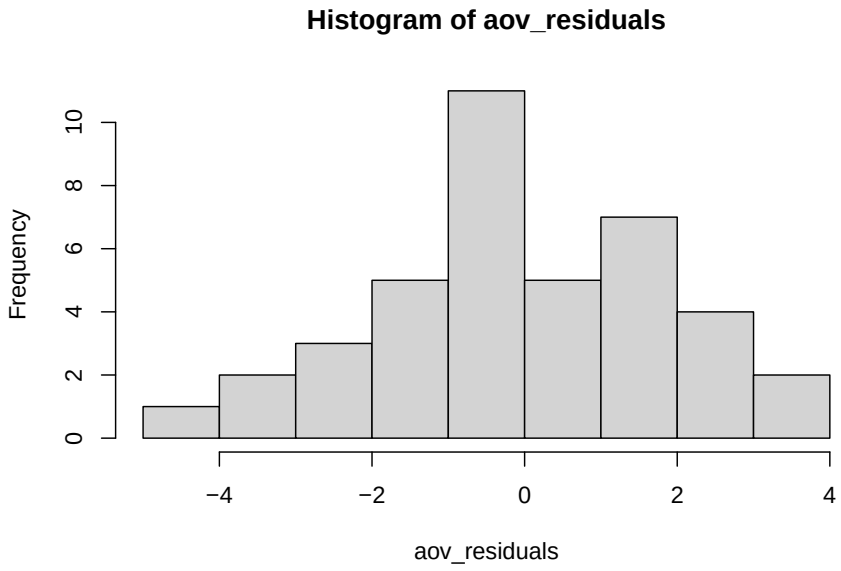
Una vez verificado el supuesto de homogeneidad de varianzas, podemos solicitar el análisis de varianza en una clasificación de la siguiente manera:

```
aov.out <- aov(altura ~ dosis, data = rendimiento)
summary(aov.out)
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## dosis        3  214.7   71.57   19.35 1.21e-07 ***
## Residuals   36  133.1    3.70
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

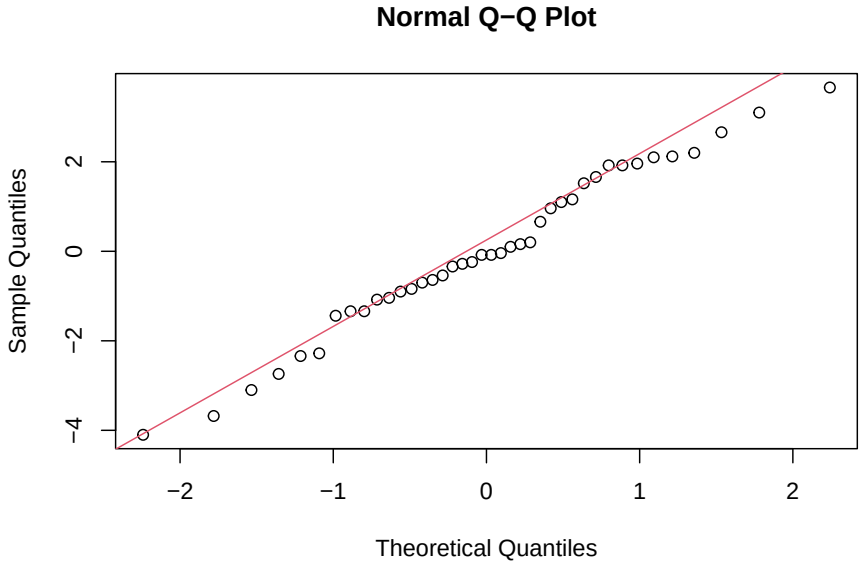
Extraemos ahora los residuales y los analizamos de manera gráfica, verificando su normalidad:

```
aov_residuals <- aov.out$residuals
hist(aov_residuals)
```



o utilizando una prueba cuantil-cuantil

```
qqnorm(aov_residuals)
qqline(aov_residuals,col=2)
```



o más formalmente, podemos verificar la normalidad de residuales a través de una prueba de Shapiro-Wilk:

```
shapiro.test(x=aov_residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  aov_residuals
## W = 0.98177, p-value = 0.7547
```

Finalmente, podemos solicitar una prueba post Hoc, como la de Tukey, de la siguiente manera:

```
TukeyHSD(aov.out, conf.level = 0.95)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = altura ~ dosis, data = rendimiento)
##
## $dosis
##                       diff        lwr         upr     p adj
## dosis_200-dosis_100  2.74  0.4238594  5.05614056 0.0150760
## dosis_400-dosis_100  3.56  1.2438594  5.87614056 0.0011022
## dosis_50-dosis_100  -2.30 -4.6161406  0.01614056 0.0521923
## dosis_400-dosis_200  0.82 -1.4961406  3.13614056 0.7762839
## dosis_50-dosis_200  -5.04 -7.3561406 -2.72385944 0.0000062
## dosis_50-dosis_400  -5.86 -8.1761406 -3.54385944 0.0000003
```