

Guía de Métodos Estadísticos 2020

Pablo Cortes; Francisco Fernández

2020-01-23

Contents

Bienvenidos	5
1 Software estadístico R	7
1.1 Aspectos básicos de R	7
1.2 R Studio	8
1.3 Trabajemos en RStudio	9
2 Tipos de datos en R	15
2.1 Objetos en R	15
2.2 Tipos de Objetos	16
3 Lectura y manipulación de datos en RStudio	21
3.1 Preparemos los datos para R	21
3.2 Importemos los datos a R	22
3.3 Exploremos los datos a R	23
3.4 Exploremos los datos R con la librería dplyr	26
3.5 Reducción de una base de datos	41
4 Visualización de datos en RStudio	43
4.1 Gráfico de dispersión o <i>Scatterplot</i>	43
4.2 Diagrama de cajas o <i>Boxplot</i>	46
4.3 Histogramas	49
4.4 Ejercicios	52

Bienvenidos

La presente Guía tiene el objetivo de acompañarlos en los laboratorios de Métodos Estadísticos durante todo el presente semestre de este año. A través de esta guía tendrán acceso completo a toda la materia que veremos durante el semestre y adicionalmente podrán practicar todo lo que acá veamos.

Chapter 1

Software estadístico R

R es un programa estadístico *open source* de gran versatilidad que permite analizar una amplia gama de problemas cuantitativos. Si bien R incorpora un lenguaje de programación que puede ser extremadamente complejo, vale la pena familiarizarse con esa herramienta que puede llegar a ser muy útil en el futuro, tanto dentro como fuera de la academia.

1.1 Aspectos básicos de R

1. R distingue mayúsculas y minúsculas.
2. Para asignar contenido a un objeto usamos `<-`. Por ejemplo, `x <- 10` asigna a `x` el valor 10. En lugar de `<-` también podemos usar `=`.
3. Para ver el contenido de un objeto simplemente escribimos su nombre.
3. Para obtener ayuda usamos el comando `help`. Por ejemplo, `help(mean)` para obtener ayuda sobre el comando `mean` que calcula la media.
4. El GUI o interfaz gráfica del R tiene dos partes principales: la consola y el script.

[poner foto acá]

La **consola** es el corazón de R, allí podemos pedirle cosas y es donde se nos entregan los resultados. También nos avisa de posibles errores (generalmente en color rojo). La consola es lo primero que observamos cuando abrimos el programa. Cuando la consola tiene el cursor `>` significa que le podemos dar comandos para ejecutar. Si es que tiene el símbolo `+` quiere decir que nos falta completar el comando anterior.

Un **script** corresponde a una hoja para escribir comandos. Nos sirve para escribir solo los comandos, y cuando seleccionamos y presionamos `<Ctrl + R>` (juntos) se ejecuta el comando que hemos escrito y los resultados se visualizarán

en la consola Dependiendo del sistema operativo utilizado, la combinación podría ser `<Ctrl +Enter>`. El script es práctico por que no solo podemos escribir comandos sino también notas personales. Las notas tienen que estar precedidas por el `#`.

1.2 R Studio

RStudio es una interfaz que permite acceder de manera sencilla a todas las funciones de R. Para utilizar RStudio se requiere haber instalado R previamente. La instalación de RStudio se puede realizar desde la página oficial del programa.

1.2.1 Conociendo a Rstudio

Una vez instalados R y RStudio procedemos a ejecutar el programa RStudio desde cualquiera de los iconos que genera y se mostrará la siguiente pantalla:

[Figura 2]

Esta pantalla está dividida en tres partes:

1. La ventana de la izquierda donde está el prompt `>`, llamada Consola, es el espacio de trabajo.
2. La ventana de la derecha se divide en dos:
 - En la ventana superior derecha se encuentra el historial de objetos almacenados en memoria. Desde esta ventana también podemos:
 - a) Limpiar nuestro historial
 - b) Importar datos
 - c) Muestra los comandos y funciones implementadas de los informes con los que se han trabajado.
3. En la ventana inferior de la derecha RStudio muestra el directorio de trabajo, los gráficos que se van generando, paquetes para cargarlos e instalarlos directamente, ayuda y un visor HTML. Estas pestañas se irán describiendo a lo largo del documento

1.3 Trabajemos en RStudio

1.3.1 Objetos de Datos

R es un lenguaje orientado a trabajar con objetos de datos (numéricos, caracteres y lógicos), los cuales se guardan en la memoria activa del ordenador (consola). Dichos objetos pueden ser:

1. *Escalares*: Números reales que sirven para describir un fenómeno físico.
2. *Vectores*: Lista ordenada de elementos, principalmente números reales.
3. *Factores*: Es un tipo especial de vector que almacena datos de caracteres en forma de variables cualitativas.

Para familiarizarnos con la sintaxis de R, escribamos la frase “Metodos estadísticos”. En la consola de RStudio debemos escribir lo siguiente y luego presionar <Ctrl + Enter>:

```
x <- "Metodos"
```

Ahora escribamos x para ver que sucede:

```
x
```

```
## [1] "Metodos"
```

Lo que acabamos de hacer es asignar la palabra Metodos al objeto x mediante el uso del operador <- Ahora, asignaremos la palabra estadísticos al objeto y:

```
y <- "estadísticos"  
y
```

```
## [1] "estadísticos"
```

Por último, combinaremos las dos palabras y las almacenaremos en un nuevo objeto que llamaremos frase:

```
frase <- paste(x,y)  
frase
```

```
## [1] "Metodos estadísticos"
```

IMPORTANTE: Los datos de caracteres siempre deben estar encerrados entre comillas (””)

1.3.2 Datos Numéricos

El objeto mas simple que podemos crear es aquel con contiene solamente un numero real. Para ello, asignamos un valor a un objeto mediante el uso del operador `<-`

Generaremos un objeto de nombre `x` con el valor 5:

```
x<-5  
x
```

```
## [1] 5
```

R permite realizar un sin numero de operaciones algebraicas con nuestros objetos. Dichos operaciones incluyen la adición (+), sustracción (-), multiplicación (*), división (/) y potenciación (^).

```
a<-15  
a
```

```
## [1] 15
```

```
b <- 2  
b
```

```
## [1] 2
```

```
a+b
```

```
## [1] 17
```

```
b-a
```

```
## [1] -13
```

```
a-b
```

```
## [1] 13
```

```
(a+b)/a
```

```
## [1] 1.133333
```

```
(a*b)/(b^b)
```

```
## [1] 7.5
```

Otras funciones matemáticas de importancia para nuestro curso son: la raíz cuadrada (`sqrt`), función exponencial (`exp`) y función logarítmica natural (`log`).

```
sqrt(9)
```

```
## [1] 3
```

```
c <- 9  
c
```

```
## [1] 9
```

```
sqrt(c)
```

```
## [1] 3
```

```
exp(2)
```

```
## [1] 7.389056
```

```
log(10)
```

```
## [1] 2.302585
```

1.3.3 Vectores

Un vector representa una secuencia ordenada de elementos (datos) del mismo tipo. Es posible construir vectores de tipo numérico y caracteres. Para nuestros propósitos, los vectores podrán ser considerados como variables.

Generaremos un vector de nombre `vector1` que contenga tres datos numéricos:

```
vector1<-c(1,5,7)  
vector1
```

```
## [1] 1 5 7
```

Ahora, generaremos un vector de nombre `vector2` que contenga tres caracteres:

```
vector2<-c("cerezo","peral","vid")
vector2
```

```
## [1] "cerezo" "peral" "vid"
```

Intentemos generar un vector de nombre vector3 a partir de los vectores creados:

```
eyJsYW5ndWFnZSI6InIiLCJzYW1wbGUiOiJ2ZWN0b3IxPC1jKDEsNSw3KVxudmVjdG9yMjYhcI
```

IMPORTANTE: Los datos son concatenados (combinados) utilizando el comando `c()`

1.3.4 Funciones para generar vectores

Las funciones `seq` y `rep` nos permiten crear patrones de elementos. `seq` Crea una secuencia de números equiespaciados. Dentro del comando `seq` el comienzo (`from`), el fin (`to`), el espacio entre dos números consecutivos (`by`) o la cantidad de números en la secuencia (`length`) pueden ser especificados.

Por ejemplo:

```
seq(from= 2, to= 8, by=2)
```

```
## [1] 2 4 6 8
```

```
seq(from=2, to= 8,length=3)
```

```
## [1] 2 5 8
```

Por otra parte, el comando `rep` repite un elemento (`x`) una cantidad determinada de veces (`times`) o hasta lograr una longitud especificada (`length.out`).

Por ejemplo:

```
rep (0,5)
```

```
## [1] 0 0 0 0 0
```

```
rep(0:2,3)
```

```
## [1] 0 1 2 0 1 2 0 1 2
```

```
rep(1:2,length.out=7)
```

```
## [1] 1 2 1 2 1 2 1
```

You can write citations, too. For example, we are using the **bookdown** package (?) in this sample book, which was built on top of R Markdown and **knitr** (Xie, 2015).

Chapter 2

Tipos de datos en R

2.1 Objetos en R

Hasta ahora nos hemos dedicado a escribir algunas instrucciones para que R las ejecute. A lo largo del curso aprenderemos muchas funciones, sin embargo existen aspectos críticos que debemos saber antes de seguir avanzando.

Lo primero que debemos saber (y que nos evitará que surjan en la consola los nunca agradables errores) es que todos los elementos que maneja R son objetos: un valor numérico es un objeto, un vector es un objeto, una base de datos es un objeto, una función es un objeto, un gráfico es... un objeto.

En este laboratorio exploraremos algunos tipos de objetos y sus propiedades básicas para trabajar en R.

2.1.1 Propiedades de los objetos

1. Los objetos están compuestos por uno o más elementos. Los elementos pueden ser caracteres alfabéticos y/o numéricos. En este curso, los elementos serán considerados datos.
2. R guarda los objetos en la memoria activa del ordenador con un nombre específico. Para ello, asignamos un valor a un objeto mediante el uso del operador `<-`.

Generemos un objeto de nombre *Asignatura* que contendrá las palabras *Metodos* y *estadísticos*. En el script de RStudio debemos escribir lo siguiente y luego presionar `<Ctrl + enter>`:

```
Asignatura<-c("Metodos", "estadisticos")
```

Ahora escribamos *asignatura* y veamos que sucede:

```
#asignatura
```

Es momento de felicitarnos a nosotros mismos, ya que acabamos de cometer uno de los errores más recurrentes y básicos que se cometen al trabajar en R. ¿Qué fue lo que sucedió?. R discrimina entre letras mayúsculas y minúsculas para el nombre de un objeto, por lo cual no es lo mismo escribir *asignatura* que *Asignatura*.

Ahora escribamos *Asignatura* y veamos que sucede:

```
Asignatura
```

```
## [1] "Metodos"      "estadisticos"
```

IMPORTANTE: Los datos de caracteres siempre deben estar encerrados entre comillas ("") y los datos son concatenados (combinados) utilizando el comando `c()`

2.2 Tipos de Objetos

En este curso nos ocuparemos de aquellos objetos que R utiliza para representar datos: valores, vectores, y dataframes.

2.2.1 Objetos de valores numéricos

El objeto mas simple que podemos crear es aquel con contiene solamente un numero real. Generaremos un objeto de nombre `x` con el valor 5:

```
x<-5
x
```

```
## [1] 5
```

R permite realizar un sin número de operaciones algebraicas con nuestros objetos. Dichos operaciones incluyen la adición (+), sustracción (-), multiplicación (*), división (/) y potenciación (^).


```
# crear objeto a
a<-15
a
```

```
## [1] 15
```

```
# crear objeto b
b<-2
b
```

```
## [1] 2
```

```
# sumar a + b
a+b
```

```
## [1] 17
```

```
# restar b-a
b-a
```

```
## [1] -13
```

```
# restar a - b
a-b
```

```
## [1] 13
```

```
# otras operaciones
(a+b)/a
```

```
## [1] 1.133333
```

```
(a*b)/(b^b)
```

```
## [1] 7.5
```

Otras funciones matemáticas de importancia para nuestro curso son: la raíz cuadrada (`sqrt`), función exponencial (`exp`) y función logarítmica natural (`log`).

```
# Raíz cuadrada  
sqrt(9)
```

```
## [1] 3
```

```
# exponencial  
exp(2)
```

```
## [1] 7.389056
```

```
# logaritmo  
log(10)
```

```
## [1] 2.302585
```

2.2.2 Data frames

Un `data.frame` es un tipo especial de objeto que permite organizar diferentes tipos de vectores (alfanuméricos). La estructura de un `data.frame` es muy similar a una hoja de datos, en donde la información se organiza en filas (observaciones de cada vector) y columnas (vector).

El siguiente ejemplo nos muestra como crear un `data.frame` llamado `Notas` a partir de los datos obtenidos de una muestra a 8 alumnos, para cada una de las cuales se ha registrado su edad, género y nota obtenida en la primera prueba parcial del curso Métodos Estadísticos.

Primero, debemos generar los 3 vectores (variables) que utilizaremos para crear nuestro `data.frame`:

```
edad <- c(22, 21, 21, 25, 19, 22, 23, 24)  
genero <- c("M", "F", "F", "M", "M", "F", "F", "M")  
nota <- c(3.4, 6.0, 5.1, 4.5, 4.6, 6.1, 4.0, 4.5)
```

A continuación, generaremos el `data.frame` llamado `Notas`. Es importante tener presente que el lenguaje de R es bastante intuitivo, a tal punto que la función que permite generar un `data.frame` es `data.frame()`

```
Notas <- data.frame(edad, genero, nota)
```

Un fuerte aplauso para nosotros mismos, acabamos de generar nuestro primer `data.frame`. Para corroborar si lo hicimos bien, y al mismo tiempo ver su contenido, simplemente tenemos que llamarlo su nombre:

Notas

```
##   edad genero nota
## 1   22      M  3.4
## 2   21      F  6.0
## 3   21      F  5.1
## 4   25      M  4.5
## 5   19      M  4.6
## 6   22      F  6.1
## 7   23      F  4.0
## 8   24      M  4.5
```

En caso de que quisiéramos saber la estructura de nuestro `data.frame`, podemos utilizar la función: `str()`

```
str(Notas)
```

```
## 'data.frame':   8 obs. of  3 variables:
## $ edad   : num  22 21 21 25 19 22 23 24
## $ genero : Factor w/ 2 levels "F","M": 2 1 1 2 2 1 1 2
## $ nota   : num  3.4 6 5.1 4.5 4.6 6.1 4 4.5
```

R nos acaba de confirmar que `Notas` es un `data.frame` de tres variables con 8 observaciones cada una. R también nos informa además de que dos variables son numéricas y la tercera, el genero, es un factor con dos valores, “F” y “M”.

En caso de que quisiéramos que R nos entregue el nombre de las variables contenidas en el `Notas` podemos utilizar la función `names()`:

```
names(Notas)
```

```
## [1] "edad" "genero" "nota"
```

2.2.3 Valores Perdidos

En algunos casos los elementos que componen un objeto son desconocidos. En estos casos, debemos especificar que el elemento desconocido se encuentra “not available” (NA), entonces a esa observación le asignamos el valor especial NA. Es importante tener presente que una operación con elementos NA resulta NA, ya que por defecto R no incluyen la especificación que omite o remueve las observaciones faltantes. Para estar seguros de esto, podemos generar un nuevo `data.frame` llamado `Notas1` a partir de los datos obtenidos de una muestra a 8 alumnos.

```
edad1 <- c(22, 21, 21, 25, 19, 22, 23, 24)
genero1 <- c("M", "F", "F", "M", "M", "F", "F", "M")
nota1 <- c(3.4, 6.0, 5.1, 4.5, 4.6, 6.1, 4.0, NA)
Notas1 <- data.frame(edad1, genero1, nota1)
```

Apliquemos una operación básica a la información contenida en `Notas1`. Podemos estimar la nota promedio de los 8 alumnos, para ello utilizaremos la función `mean()`, la cual exploraremos más adelante en nuestro curso:

```
mean(Notas1$nota1)
```

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

Lo que obtuvimos como resultado fue `NA`. Esto se debe a que no le especificamos a R que debe omitir los `NA`s en nuestro `data.frame`. Una forma de solucionar este problema es utilizar la operación `na.rm=TRUE`, la cual especifica que la operación se efectúe con los datos válidos excluyendo los `NA`s:

```
mean(Notas1$nota1, na.rm=TRUE)
```

```
## [1] 4.814286
```

Magia! Con una simple orden acabamos de solucionar un simple pero doloroso problema.

Chapter 3

Lectura y manipulación de datos en RStudio

Hoy en día la información generada a partir del mundo natural se multiplica minuto a minuto. Desde un punto de vista académico y/o profesional, la correcta toma de decisiones demandará la recolección y manejo de grandes cantidades de datos, los cuales nos permitirán describir y determinar propiedades del mundo natural.

Una base de datos se puede definir como un conjunto de información perteneciente a un mismo contexto y almacenada sistemáticamente para su posterior uso. Cada base de datos se compone de una o más tablas que guardan un conjunto de datos. Cada tabla tiene una o más columnas y filas.

En este laboratorio comenzaremos a trabajar con bases de datos. Para ello abordaremos cuatro tópicos centrales:

1. La preparación de los datos para poder trabajarlos en R,
2. La importación de una base de datos a R,
3. La edición de los datos contenidos en la base de datos,
4. La exploración de la información contenida en nuestra base de datos.

3.1 Preparemos los datos para R

Al mencionar que trabajaremos con base de datos, inmediatamente surge el desafío de como podemos ingresar nuestra información a R. Lo primero que debemos tener presente es que a R “le encantan” las bases de datos que tienen cada observación en una fila y que en cada columna incorpora una variable. Por ejemplo, si tenemos una base de datos con la altura y género de los alumnos del curso “métodos estadísticos” de la escuela de Agronomía:

```
genero <- c("Hombre","Hombre","Hombre","Hombre","Hombre","Mujer","Mujer","Mujer","Mujer")
altura<-c(1.75,1.68,1.82,1.71,1.62,1.61,1.55,1.53,1.66,1.58)
base_de_datos <- data.frame(genero,altura)
```

¿Cuáles son las dimensiones de esta base de datos?. La tabla tiene una dimensión de 10x2. Esto quiere decir que está compuesta de diez filas y dos columnas.

IMPORTANTE: * Las filas contienen nuestras observaciones. * Las columnas contienen nuestras variables. * R reconoce puntos (".") y no comas (",") como separador de decimales. * Revisar que nuestra base de datos no contenga filas en blanco (sin valores). Cuando eso ocurra se recomienda rellenar el espacio con las letras NA ("Not available").

3.1.1 ¿Qué formato de archivos reconoce R?

R es muy versátil para reconocer y leer múltiples formatos de archivo. Los archivos de texto plano, tales como TXT o CSV, son la opción más sencilla para importar datos desde hojas de cálculo. Por ejemplo, Excel permite guardar archivos .txt (texto delimitado por tabulaciones) o .csv (texto delimitados por comas). En este curso los alentamos a utilizar el formato CSV. Una ventaja de este tipo de representación es que los datos se pueden visualizar/editar con un editor de texto. CSV representa la información en forma de tabla donde las columnas se delimitan con un carácter (por defecto, con coma)* y las filas con saltos de línea.

3.2 Importemos los datos a R

Para importar los datos, necesitamos especificar a R la localización (ruta) de los datos en el computador. El problema es que la ruta de los archivos generalmente son muy largas y difícil de memorizar. Además, si se equivocan al escribirla, R no la reconocerá. Es por ello que es preferible que RStudio haga ese difícil trabajo.

A lo largo de nuestro curso, el método que utilizaremos para importar la base de datos se basará en la herramienta Import dataset. Rstudio proporciona una función de importación a través de la pestaña Import dataset en la parte superior derecha.

[imagen 3]

Esta función nos permitirá buscar el archivo de datos, seleccionarlo y finalmente para poder cargar la base de datos en R. Al seleccionar la opción From Text (base), podemos navegar en nuestro computador para encontrar nuestro archivo con la base de datos. Una vez lo seleccionamos, solamente debemos hacer clic en el botón Open para que la magia se haga realidad.

- Para aprender sobre otros métodos de importación de datos les sugerimos visitar el siguiente link

Una vez seleccionado el archivo, se abrirá una ventana que entrega una vista preliminar del archivo seleccionado. En la parte superior izquierda, podrán observar la opción name, la cual nos permite asignar un nombre a nuestra base de datos (¿Recuerdan el comando `<-?`). En este caso, nuestra base de datos recibirá el nombre **data**. Lo único que resta por hacer es darle la orden a R para que importe la base de datos. Esto lo podemos hacer habiendo clic en el botón **Import**.

[imagen 4]

3.3 Exploremos los datos a R

Al importar nuestro archivo, lo que buscamos es analizar incluso graficar la información contenida en la base de datos. Sin embargo, antes de eso existe un paso sensible y crítico que jamás debemos olvidar. Es importante asegurarnos que importamos correctamente nuestros datos!

La función `str()` nos mostrará el tipo de datos contenido en nuestra base de datos y enumerará cada variable de columna junto con su tipo de datos.

```
str(data)
```

```
## 'data.frame':    120 obs. of  9 variables:
## $ Cultivo       : Factor w/ 1 level "Arandanos": 1 1 1 1 1 1 1 1 1 ...
## $ Region        : Factor w/ 4 levels "BioBio","La_Araucania",...: 4 4 4 4 4 4 4 4 4 ...
## $ Variedad      : Factor w/ 2 levels "V1","V2": 1 1 1 1 1 1 1 1 1 ...
## $ Hectareas     : num  1030 999 1118 1007 1078 ...
## $ Temperatura  : num  16.8 16.9 15 15.7 15.1 15.9 16.1 15.6 17.4 14.7 ...
## $ costo_jh      : int  13059 13026 12933 13027 13045 12949 13002 12968 12978 13056 ...
## $ rendimiento  : num  6866 7122 7041 6892 6928 ...
## $ Perdida_plaga: num  35.1 34.3 31.7 40.5 35.6 33 38.1 37.9 34.8 34.4 ...
## $ mano_de_obra : int  3080 2728 3120 3135 3276 2966 2725 2874 2785 3193 ...
```

Como podemos ver, nuestra base contiene variables con diferentes tipos de datos.

La función `summary()` nos proporciona para cada variable un conjunto de estadísticas descriptivas, según el tipo de variable:

```
summary(data)
```

```
##      Cultivo      Region Variedad Hectareas
## Arandanos:120 BioBio      :30 V1:60 Min.      : 998.9
##      La_Araucania:30 V2:60 1st Qu.:1477.7
##      Maule      :30 Median :1797.0
##      Ohiggins   :30 Mean   :2090.7
##      3rd Qu.:2352.7
##      Max.      :3788.0
## Temperatura costo_jh rendimiento Perdida_plaga
## Min.      :10.00 Min.      :11893 Min.      :4166 Min.      :24.70
## 1st Qu.:12.50 1st Qu.:11997 1st Qu.:4765 1st Qu.:32.95
## Median :13.75 Median :12490 Median :5457 Median :34.60
## Mean   :13.74 Mean   :12500 Mean   :5617 Mean   :34.66
## 3rd Qu.:15.10 3rd Qu.:13002 3rd Qu.:6285 3rd Qu.:37.30
## Max.   :17.40 Max.   :13121 Max.   :7247 Max.   :42.80
## mano_de_obra
## Min.      :2609
## 1st Qu.:3092
## Median :3312
## Mean   :3384
## 3rd Qu.:3645
## Max.   :4274
```

1. **Variables numéricas:** `summary()` proporciona el rango, los cuartiles, la mediana y la media.
2. **Variables factoriales:** `summary()` proporciona una tabla con frecuencias.
3. **Variables numéricas y factoriales:** `summary()`, en caso de que existan, nos entregará información sobre el número de valores faltantes (NAs).
4. **Variables de caracteres:** `summary()` solo proporciona la longitud de la variable.

La función `head()` nos entregará las primeras filas de nuestra base de datos. Por defecto serán las primeras 6:

```
head(data)
```

```
##      Cultivo Region Variedad Hectareas Temperatura costo_jh rendimiento
## 1 Arandanos Ohiggins      V1    1029.9         16.8    13059     6866.5
## 2 Arandanos Ohiggins      V1     998.9         16.9    13026     7122.1
## 3 Arandanos Ohiggins      V1    1117.8         15.0    12933     7041.2
## 4 Arandanos Ohiggins      V1    1007.2         15.7    13027     6891.5
## 5 Arandanos Ohiggins      V1    1077.6         15.1    13045     6927.7
## 6 Arandanos Ohiggins      V1    1091.3         15.9    12949     6900.4
## Perdida_plaga mano_de_obra
## 1           35.1           3080
```



```
## 2      34.3      2728
## 3      31.7      3120
## 4      40.5      3135
## 5      35.6      3276
## 6      33.0      2966
```

En caso de que quisiéramos ampliar el número de filas:

```
head(data, 10)
```

```
##      Cultivo      Region Variedad Hectareas Temperatura costo_jh rendimiento
## 1 Arandanos Ohiggins      V1    1029.9         16.8    13059    6866.5
## 2 Arandanos Ohiggins      V1     998.9         16.9    13026    7122.1
## 3 Arandanos Ohiggins      V1    1117.8         15.0    12933    7041.2
## 4 Arandanos Ohiggins      V1    1007.2         15.7    13027    6891.5
## 5 Arandanos Ohiggins      V1    1077.6         15.1    13045    6927.7
## 6 Arandanos Ohiggins      V1    1091.3         15.9    12949    6900.4
## 7 Arandanos Ohiggins      V1    1188.9         16.1    13002    6943.9
## 8 Arandanos Ohiggins      V1    1111.6         15.6    12968    6885.7
## 9 Arandanos Ohiggins      V1    1028.8         17.4    12978    6724.3
## 10 Arandanos Ohiggins      V1     1124.2         14.7    13056    6920.4
## Perdida_plaga mano_de_obra
## 1      35.1      3080
## 2      34.3      2728
## 3      31.7      3120
## 4      40.5      3135
## 5      35.6      3276
## 6      33.0      2966
## 7      38.1      2725
## 8      37.9      2874
## 9      34.8      2785
## 10     34.4      3193
```

Por otro lado, la función `tail()` nos entregará las últimas filas de nuestra base de datos. Por defecto serán las últimas 6:

```
tail(data)
```

```
##      Cultivo      Region Variedad Hectareas Temperatura costo_jh
## 115 Arandanos La_Araucania      V2    1857.6         11.1    11974
## 116 Arandanos La_Araucania      V2    1862.7         11.6    11960
## 117 Arandanos La_Araucania      V2    1894.5         10.6    12023
## 118 Arandanos La_Araucania      V2    1873.5         11.1    11971
## 119 Arandanos La_Araucania      V2    1915.4         11.5    12065
```

```
## 120 Arandanos La_Araucania      V2    1932.1      13.5    12084
##      rendimiento Perdida_plaga mano_de_obra
## 115      4591.7          33.2      3439
## 116      4689.4          38.1      3566
## 117      4400.5          24.7      3552
## 118      4380.6          37.6      3449
## 119      4424.1          35.1      3550
## 120      4331.0          38.3      3162
```

La función `names()` nos entregará el nombre de las variables (columnas) contenidas en nuestra base de datos:

```
names(data)
```

```
## [1] "Cultivo"      "Region"      "Variedad"    "Hectareas"
## [5] "Temperatura"  "costo_jh"    "rendimiento" "Perdida_plaga"
## [9] "mano_de_obra"
```

En caso de que quisiéramos conocer las dimensiones (número de columnas y filas) de nuestra base de datos, podemos utilizar la función `dim()`:

```
dim(data)
```

```
## [1] 120  9
```

3.4 Exploremos los datos R con la librería dplyr

Una de las grandes ventajas de R es la ampliación de sus funcionalidades básicas mediante paquetes (packages) o librerías. Las librerías de R se pueden instalar de múltiples formas. En RStudio, lo haremos de siguiendo una serie de sencillos pasos.

Debemos hacer clic en la pestaña `package` ubicada en el panel inferior derecho. Al hacer clic en esta pestaña, aparecerá una pequeña ventana con tres campos principales: `Install from`, `packages`, and `Install to Library`. Solo necesitamos preocuparnos por el campo `packages`, los otros dos los dejaremos en su valor predeterminado.

[figura 5]

Ahora, al escribir las primeras letras del nombre de una librería (en este caso `dplyr`), Rstudio proporcionará una lista de librerías disponibles que coincidan con esta palabra. Después de encontrar la librería, todo lo que tenemos que hacer es hacer clic en el botón `Install` y dejar que Rstudio trabaje.

[fig 6]

A pesar de estar instalada, a menos que lo especifiquemos, R no cargará la librería en la consola. Entonces, debemos “llamar” a la librería que acabamos de instalar con el comando `library(dplyr)`.

```
library(dplyr)
```

La librería `dplyr` incluye un conjunto de comandos que coinciden con las acciones más comunes que se realizan sobre un conjunto de datos (seleccionar filas `filter`, seleccionar columnas `select`, resumir mediante alguna medida numérica `summarise`, entre muchas otras). Para mayor información sobre la librería `dplyr` puedes acceder al siguiente link.

La función `select()` selecciona columnas de nuestra base:

Intentemos seleccionar la columna `Region` en la base de datos llamada `data`

```
select(data, Region)
```

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

```
## 26      Ohiggins
## 27      Ohiggins
## 28      Ohiggins
## 29      Ohiggins
## 30      Ohiggins
## 31      Maule
## 32      Maule
## 33      Maule
## 34      Maule
## 35      Maule
## 36      Maule
## 37      Maule
## 38      Maule
## 39      Maule
## 40      Maule
## 41      Maule
## 42      Maule
## 43      Maule
## 44      Maule
## 45      Maule
## 46      Maule
## 47      Maule
## 48      Maule
## 49      Maule
## 50      Maule
## 51      Maule
## 52      Maule
## 53      Maule
## 54      Maule
## 55      Maule
## 56      Maule
## 57      Maule
## 58      Maule
## 59      Maule
## 60      Maule
## 61      BioBio
## 62      BioBio
## 63      BioBio
## 64      BioBio
## 65      BioBio
## 66      BioBio
## 67      BioBio
## 68      BioBio
## 69      BioBio
## 70      BioBio
## 71      BioBio
```

```
## 72      BioBio
## 73      BioBio
## 74      BioBio
## 75      BioBio
## 76      BioBio
## 77      BioBio
## 78      BioBio
## 79      BioBio
## 80      BioBio
## 81      BioBio
## 82      BioBio
## 83      BioBio
## 84      BioBio
## 85      BioBio
## 86      BioBio
## 87      BioBio
## 88      BioBio
## 89      BioBio
## 90      BioBio
## 91 La_Araucania
## 92 La_Araucania
## 93 La_Araucania
## 94 La_Araucania
## 95 La_Araucania
## 96 La_Araucania
## 97 La_Araucania
## 98 La_Araucania
## 99 La_Araucania
## 100 La_Araucania
## 101 La_Araucania
## 102 La_Araucania
## 103 La_Araucania
## 104 La_Araucania
## 105 La_Araucania
## 106 La_Araucania
## 107 La_Araucania
## 108 La_Araucania
## 109 La_Araucania
## 110 La_Araucania
## 111 La_Araucania
## 112 La_Araucania
## 113 La_Araucania
## 114 La_Araucania
## 115 La_Araucania
## 116 La_Araucania
## 117 La_Araucania
```

```
## 118 La_Araucania
## 119 La_Araucania
## 120 La_Araucania
```

La función `select()` también nos permite seleccionar todas las columnas excepto una:

```
select(data, -Region)
```

##	Cultivo	Variedad	Hectareas	Temperatura	costo_jh	rendimiento
## 1	Arandanos	V1	1029.9	16.8	13059	6866.5
## 2	Arandanos	V1	998.9	16.9	13026	7122.1
## 3	Arandanos	V1	1117.8	15.0	12933	7041.2
## 4	Arandanos	V1	1007.2	15.7	13027	6891.5
## 5	Arandanos	V1	1077.6	15.1	13045	6927.7
## 6	Arandanos	V1	1091.3	15.9	12949	6900.4
## 7	Arandanos	V1	1188.9	16.1	13002	6943.9
## 8	Arandanos	V1	1111.6	15.6	12968	6885.7
## 9	Arandanos	V1	1028.8	17.4	12978	6724.3
## 10	Arandanos	V1	1124.2	14.7	13056	6920.4
## 11	Arandanos	V1	1106.5	15.9	13056	7056.9
## 12	Arandanos	V1	1000.6	15.4	13077	6898.6
## 13	Arandanos	V1	1101.0	16.5	12993	6797.2
## 14	Arandanos	V1	1113.9	14.0	12975	6858.2
## 15	Arandanos	V1	1159.9	14.7	13003	7064.3
## 16	Arandanos	V2	1110.8	14.3	12970	7028.4
## 17	Arandanos	V2	1091.4	13.7	13003	7127.7
## 18	Arandanos	V2	1173.8	15.7	12994	7247.3
## 19	Arandanos	V2	1034.9	15.1	13033	6896.8
## 20	Arandanos	V2	1081.7	14.3	13107	6927.3
## 21	Arandanos	V2	1120.4	14.8	13013	6893.3
## 22	Arandanos	V2	1054.0	16.9	12998	6929.0
## 23	Arandanos	V2	1099.7	15.0	12895	7237.8
## 24	Arandanos	V2	1083.0	15.2	12991	6964.5
## 25	Arandanos	V2	1063.7	15.3	12969	6776.9
## 26	Arandanos	V2	1097.1	13.5	13055	7017.3
## 27	Arandanos	V2	1132.5	14.9	13005	7080.4
## 28	Arandanos	V2	1086.0	14.4	12988	7171.5
## 29	Arandanos	V2	1035.5	12.5	12920	7246.0
## 30	Arandanos	V2	1110.5	16.4	12936	6928.1
## 31	Arandanos	V1	3750.2	15.6	12968	5931.1
## 32	Arandanos	V1	3668.7	11.6	13013	6076.3
## 33	Arandanos	V1	3586.2	15.8	13007	6005.6
## 34	Arandanos	V1	3678.0	15.0	13033	5964.3
## 35	Arandanos	V1	3729.1	12.5	13057	6003.4

## 36	Arandanos	V1	3709.2	15.9	13039	6026.0
## 37	Arandanos	V1	3636.3	15.6	12961	5973.5
## 38	Arandanos	V1	3749.6	13.7	13023	6046.6
## 39	Arandanos	V1	3644.6	12.5	12985	5960.2
## 40	Arandanos	V1	3634.6	14.5	13041	6011.1
## 41	Arandanos	V1	3649.5	15.1	12942	6012.2
## 42	Arandanos	V1	3788.0	13.7	13121	6100.5
## 43	Arandanos	V1	3699.9	15.9	12945	5932.6
## 44	Arandanos	V1	3718.2	14.0	12967	5876.5
## 45	Arandanos	V1	3645.8	13.2	13098	5745.2
## 46	Arandanos	V2	3699.5	14.0	13065	5890.1
## 47	Arandanos	V2	3627.4	13.5	13008	6078.3
## 48	Arandanos	V2	3685.2	16.8	13088	6062.3
## 49	Arandanos	V2	3770.5	14.5	13034	5852.9
## 50	Arandanos	V2	3711.1	15.5	13064	6016.7
## 51	Arandanos	V2	3758.2	13.7	12975	6112.8
## 52	Arandanos	V2	3685.5	12.7	13087	6032.1
## 53	Arandanos	V2	3619.6	13.5	12902	6139.0
## 54	Arandanos	V2	3724.8	14.7	12974	6008.3
## 55	Arandanos	V2	3735.7	13.5	12976	5936.0
## 56	Arandanos	V2	3672.5	15.1	12937	5929.8
## 57	Arandanos	V2	3657.3	15.3	12946	6036.1
## 58	Arandanos	V2	3657.9	13.4	12963	6084.7
## 59	Arandanos	V2	3721.7	15.8	12955	5848.1
## 60	Arandanos	V2	3762.6	16.2	13059	6125.2
## 61	Arandanos	V1	1642.1	13.5	12018	4957.5
## 62	Arandanos	V1	1823.2	13.6	12051	5025.6
## 63	Arandanos	V1	1801.9	12.4	12004	5118.6
## 64	Arandanos	V1	1742.7	12.6	12005	4980.2
## 65	Arandanos	V1	1731.6	13.8	12069	5024.6
## 66	Arandanos	V1	1601.4	11.7	11906	5007.0
## 67	Arandanos	V1	1793.8	13.9	11901	4825.8
## 68	Arandanos	V1	1766.3	14.1	11962	5164.7
## 69	Arandanos	V1	1793.7	14.8	11989	4767.4
## 70	Arandanos	V1	1742.0	11.5	11988	5013.5
## 71	Arandanos	V1	1656.7	14.0	11976	5013.0
## 72	Arandanos	V1	1659.9	14.8	12014	5102.8
## 73	Arandanos	V1	1573.9	14.9	12083	4924.8
## 74	Arandanos	V1	1781.8	13.4	11997	5061.5
## 75	Arandanos	V1	1724.4	12.5	11893	4892.4
## 76	Arandanos	V2	1800.3	13.9	11927	5129.7
## 77	Arandanos	V2	1683.0	11.9	12046	5115.4
## 78	Arandanos	V2	1733.9	13.3	11993	4992.5
## 79	Arandanos	V2	1803.5	13.7	11900	4962.0
## 80	Arandanos	V2	1679.8	14.8	11987	4905.3
## 81	Arandanos	V2	1743.7	15.2	12005	5037.2

32CHAPTER 3. LECTURA Y MANIPULACIÓN DE DATOS EN RSTUDIO

## 82	Arandanos	V2	1664.5	14.8	11906	5107.2
## 83	Arandanos	V2	1715.7	14.5	11962	4914.6
## 84	Arandanos	V2	1706.4	12.4	11985	5058.8
## 85	Arandanos	V2	1698.5	12.1	11919	4934.8
## 86	Arandanos	V2	1734.7	13.8	11997	4965.1
## 87	Arandanos	V2	1785.3	14.0	11967	4970.2
## 88	Arandanos	V2	1694.2	15.7	11967	5137.6
## 89	Arandanos	V2	1680.7	12.1	11938	5140.9
## 90	Arandanos	V2	1681.4	12.7	11988	5168.7
## 91	Arandanos	V1	1907.5	11.3	12005	4166.3
## 92	Arandanos	V1	1896.7	12.4	12015	4439.0
## 93	Arandanos	V1	1867.3	11.9	12011	4374.1
## 94	Arandanos	V1	1922.1	13.0	11995	4506.7
## 95	Arandanos	V1	1941.5	11.2	12018	4466.9
## 96	Arandanos	V1	1846.2	12.8	12023	4469.6
## 97	Arandanos	V1	1854.6	11.9	12084	4517.0
## 98	Arandanos	V1	1898.1	10.0	12069	4538.1
## 99	Arandanos	V1	1783.3	11.9	11923	4456.1
## 100	Arandanos	V1	1833.6	13.1	11970	4444.0
## 101	Arandanos	V1	1801.7	12.8	12007	4525.9
## 102	Arandanos	V1	1931.0	11.5	12023	4632.6
## 103	Arandanos	V1	1865.6	13.3	12022	4327.5
## 104	Arandanos	V1	1819.9	10.3	12063	4526.0
## 105	Arandanos	V1	1855.1	11.6	12042	4523.7
## 106	Arandanos	V2	1910.0	12.6	11924	4447.8
## 107	Arandanos	V2	1708.3	10.4	11962	4421.7
## 108	Arandanos	V2	1823.3	13.2	11951	4554.1
## 109	Arandanos	V2	1780.9	11.6	12050	4344.9
## 110	Arandanos	V2	1886.3	10.1	12020	4456.3
## 111	Arandanos	V2	1838.7	13.1	11961	4528.3
## 112	Arandanos	V2	1858.9	10.1	12038	4651.1
## 113	Arandanos	V2	1761.6	12.0	12037	4758.3
## 114	Arandanos	V2	1908.1	10.6	12055	4539.7
## 115	Arandanos	V2	1857.6	11.1	11974	4591.7
## 116	Arandanos	V2	1862.7	11.6	11960	4689.4
## 117	Arandanos	V2	1894.5	10.6	12023	4400.5
## 118	Arandanos	V2	1873.5	11.1	11971	4380.6
## 119	Arandanos	V2	1915.4	11.5	12065	4424.1
## 120	Arandanos	V2	1932.1	13.5	12084	4331.0
##	Perdida_plaga	mano_de_obra				
## 1		35.1	3080			
## 2		34.3	2728			
## 3		31.7	3120			
## 4		40.5	3135			
## 5		35.6	3276			
## 6		33.0	2966			

## 7	38.1	2725
## 8	37.9	2874
## 9	34.8	2785
## 10	34.4	3193
## 11	37.9	2872
## 12	30.0	2983
## 13	37.3	2932
## 14	33.9	2965
## 15	31.0	2609
## 16	33.0	3104
## 17	36.4	2790
## 18	33.9	2720
## 19	27.2	3099
## 20	37.9	2868
## 21	41.8	3307
## 22	36.9	3155
## 23	36.0	3093
## 24	34.6	2928
## 25	29.5	2996
## 26	29.9	3002
## 27	34.7	3285
## 28	38.2	3068
## 29	28.3	3308
## 30	32.6	2724
## 31	27.5	3446
## 32	34.5	3204
## 33	38.0	3225
## 34	36.9	2979
## 35	33.2	3222
## 36	36.1	3504
## 37	37.7	3349
## 38	36.4	3066
## 39	34.3	3186
## 40	35.6	3205
## 41	38.5	3080
## 42	29.2	3463
## 43	36.9	3410
## 44	31.5	2956
## 45	38.5	3316
## 46	35.5	3027
## 47	34.2	2980
## 48	38.1	3319
## 49	33.5	3640
## 50	32.1	2891
## 51	37.5	3559
## 52	33.5	3278

34CHAPTER 3. LECTURA Y MANIPULACIÓN DE DATOS EN RSTUDIO

## 53	33.3	3356
## 54	37.9	3159
## 55	40.2	3363
## 56	34.2	3341
## 57	33.3	3094
## 58	37.8	3480
## 59	30.7	3088
## 60	30.0	3434
## 61	33.6	3914
## 62	32.8	3901
## 63	27.6	3673
## 64	36.3	3953
## 65	35.7	4171
## 66	32.1	4013
## 67	34.2	3864
## 68	34.2	3947
## 69	34.0	3908
## 70	33.4	3764
## 71	34.6	4029
## 72	29.6	4132
## 73	35.0	4030
## 74	31.4	3882
## 75	41.8	3698
## 76	37.7	3880
## 77	34.6	3785
## 78	35.6	4175
## 79	39.3	3660
## 80	36.7	3832
## 81	34.6	4087
## 82	31.6	4101
## 83	37.1	4205
## 84	41.2	3755
## 85	37.0	3889
## 86	34.6	4036
## 87	37.5	4274
## 88	33.3	4101
## 89	31.7	4250
## 90	33.5	3938
## 91	33.2	3198
## 92	33.6	3303
## 93	37.6	3207
## 94	37.8	3484
## 95	37.1	3212
## 96	33.4	3356
## 97	31.8	3408
## 98	34.9	3319

```
## 99      42.8      3280
## 100     35.4      3105
## 101     37.4      3404
## 102     32.0      3258
## 103     31.4      2927
## 104     33.6      3028
## 105     31.1      3004
## 106     32.8      3403
## 107     37.6      3503
## 108     35.0      3300
## 109     37.3      3325
## 110     29.8      3160
## 111     34.1      3553
## 112     30.4      3614
## 113     32.8      3208
## 114     34.1      3595
## 115     33.2      3439
## 116     38.1      3566
## 117     24.7      3552
## 118     37.6      3449
## 119     35.1      3550
## 120     38.3      3162
```

La función `slice()` selecciona filas de nuestra base. Intentemos seleccionar la quinta fila en la base de datos llamada `data`:

```
slice(data, 5)
```

```
##      Cultivo  Region Variedad Hectareas Temperatura costo_jh rendimiento
## 1 Arandanos Ohiggins      V1    1077.6         15.1    13045    6927.7
##      Perdida_plaga mano_de_obra
## 1           35.6           3276
```

La función `slice()` también nos permite seleccionar una secuencia de filas. Seleccionemos desde la tercera hasta la decima en la base de datos llamada `data`:

```
slice(data, 3:10)
```

```
##      Cultivo  Region Variedad Hectareas Temperatura costo_jh rendimiento
## 1 Arandanos Ohiggins      V1    1117.8         15.0    12933    7041.2
## 2 Arandanos Ohiggins      V1    1007.2         15.7    13027    6891.5
## 3 Arandanos Ohiggins      V1    1077.6         15.1    13045    6927.7
## 4 Arandanos Ohiggins      V1    1091.3         15.9    12949    6900.4
```

```
## 5 Arandanos Ohiggins      V1    1188.9      16.1    13002    6943.9
## 6 Arandanos Ohiggins      V1    1111.6      15.6    12968    6885.7
## 7 Arandanos Ohiggins      V1    1028.8      17.4    12978    6724.3
## 8 Arandanos Ohiggins      V1    1124.2      14.7    13056    6920.4
## Perdida_plaga mano_de_obra
## 1      31.7      3120
## 2      40.5      3135
## 3      35.6      3276
## 4      33.0      2966
## 5      38.1      2725
## 6      37.9      2874
## 7      34.8      2785
## 8      34.4      3193
```

La versatilidad de la función `slice()` es aún mayor. Por ejemplo, podemos seleccionar una secuencia de filas discontinua:

```
slice(data, c(3,10,25,100))
```

```
##      Cultivo      Region Variedad Hectareas Temperatura costo_jh
## 1 Arandanos Ohiggins      V1    1117.8      15.0    12933
## 2 Arandanos Ohiggins      V1    1124.2      14.7    13056
## 3 Arandanos Ohiggins      V2    1063.7      15.3    12969
## 4 Arandanos La_Araucania      V1    1833.6      13.1    11970
## rendimiento Perdida_plaga mano_de_obra
## 1      7041.2      31.7      3120
## 2      6920.4      34.4      3193
## 3      6776.9      29.5      2996
## 4      4444.0      35.4      3105
```

También existe la función ‘`filter()`’, la cual nos permite seleccionar las observaciones (filas) que cumplen las condiciones que nos interesan. En caso de que quisiéramos seleccionar todas aquellas observaciones en donde el rendimiento fue mayor 7000 toneladas/hectárea:

```
filter(data, rendimiento > 7000)
```

```
##      Cultivo      Region Variedad Hectareas Temperatura costo_jh rendimiento
## 1 Arandanos Ohiggins      V1    998.9      16.9    13026    7122.1
## 2 Arandanos Ohiggins      V1    1117.8      15.0    12933    7041.2
## 3 Arandanos Ohiggins      V1    1106.5      15.9    13056    7056.9
## 4 Arandanos Ohiggins      V1    1159.9      14.7    13003    7064.3
## 5 Arandanos Ohiggins      V2    1110.8      14.3    12970    7028.4
## 6 Arandanos Ohiggins      V2    1091.4      13.7    13003    7127.7
```

```
## 7 Arandanos Ohiggins      V2    1173.8      15.7    12994      7247.3
## 8 Arandanos Ohiggins      V2    1099.7      15.0    12895      7237.8
## 9 Arandanos Ohiggins      V2    1097.1      13.5    13055      7017.3
## 10 Arandanos Ohiggins     V2    1132.5      14.9    13005      7080.4
## 11 Arandanos Ohiggins     V2    1086.0      14.4    12988      7171.5
## 12 Arandanos Ohiggins     V2    1035.5      12.5    12920      7246.0
##   Perdida_plaga mano_de_obra
## 1          34.3          2728
## 2          31.7          3120
## 3          37.9          2872
## 4          31.0          2609
## 5          33.0          3104
## 6          36.4          2790
## 7          33.9          2720
## 8          36.0          3093
## 9          29.9          3002
## 10         34.7          3285
## 11         38.2          3068
## 12         28.3          3308
```

Si estuviésemos interesados en seleccionar aquellas observaciones en donde el rendimiento fue mayor 7000 toneladas/hectárea solamente para la variedad 1 de arándanos:

```
filter(data, rendimiento > 7000, Variedad == "V1")
```

```
##   Cultivo   Region Variedad Hectareas Temperatura costo_jh rendimiento
## 1 Arandanos Ohiggins      V1     998.9         16.9    13026      7122.1
## 2 Arandanos Ohiggins      V1    1117.8         15.0    12933      7041.2
## 3 Arandanos Ohiggins      V1    1106.5         15.9    13056      7056.9
## 4 Arandanos Ohiggins      V1    1159.9         14.7    13003      7064.3
##   Perdida_plaga mano_de_obra
## 1          34.3          2728
## 2          31.7          3120
## 3          37.9          2872
## 4          31.0          2609
```

Ahora, intentemos seleccionar aquellas observaciones en donde el rendimiento fue mayor 7000 toneladas/hectárea “o” que sean de la Variedad 1:

```
filter(data, rendimiento > 7000 | Variedad == "V1")
```

```
##   Cultivo   Region Variedad Hectareas Temperatura costo_jh
## 1 Arandanos Ohiggins      V1    1029.9         16.8    13059
```

38CHAPTER 3. LECTURA Y MANIPULACIÓN DE DATOS EN RSTUDIO

## 2	Arandanos	Ohiggins	V1	998.9	16.9	13026
## 3	Arandanos	Ohiggins	V1	1117.8	15.0	12933
## 4	Arandanos	Ohiggins	V1	1007.2	15.7	13027
## 5	Arandanos	Ohiggins	V1	1077.6	15.1	13045
## 6	Arandanos	Ohiggins	V1	1091.3	15.9	12949
## 7	Arandanos	Ohiggins	V1	1188.9	16.1	13002
## 8	Arandanos	Ohiggins	V1	1111.6	15.6	12968
## 9	Arandanos	Ohiggins	V1	1028.8	17.4	12978
## 10	Arandanos	Ohiggins	V1	1124.2	14.7	13056
## 11	Arandanos	Ohiggins	V1	1106.5	15.9	13056
## 12	Arandanos	Ohiggins	V1	1000.6	15.4	13077
## 13	Arandanos	Ohiggins	V1	1101.0	16.5	12993
## 14	Arandanos	Ohiggins	V1	1113.9	14.0	12975
## 15	Arandanos	Ohiggins	V1	1159.9	14.7	13003
## 16	Arandanos	Ohiggins	V2	1110.8	14.3	12970
## 17	Arandanos	Ohiggins	V2	1091.4	13.7	13003
## 18	Arandanos	Ohiggins	V2	1173.8	15.7	12994
## 19	Arandanos	Ohiggins	V2	1099.7	15.0	12895
## 20	Arandanos	Ohiggins	V2	1097.1	13.5	13055
## 21	Arandanos	Ohiggins	V2	1132.5	14.9	13005
## 22	Arandanos	Ohiggins	V2	1086.0	14.4	12988
## 23	Arandanos	Ohiggins	V2	1035.5	12.5	12920
## 24	Arandanos	Maule	V1	3750.2	15.6	12968
## 25	Arandanos	Maule	V1	3668.7	11.6	13013
## 26	Arandanos	Maule	V1	3586.2	15.8	13007
## 27	Arandanos	Maule	V1	3678.0	15.0	13033
## 28	Arandanos	Maule	V1	3729.1	12.5	13057
## 29	Arandanos	Maule	V1	3709.2	15.9	13039
## 30	Arandanos	Maule	V1	3636.3	15.6	12961
## 31	Arandanos	Maule	V1	3749.6	13.7	13023
## 32	Arandanos	Maule	V1	3644.6	12.5	12985
## 33	Arandanos	Maule	V1	3634.6	14.5	13041
## 34	Arandanos	Maule	V1	3649.5	15.1	12942
## 35	Arandanos	Maule	V1	3788.0	13.7	13121
## 36	Arandanos	Maule	V1	3699.9	15.9	12945
## 37	Arandanos	Maule	V1	3718.2	14.0	12967
## 38	Arandanos	Maule	V1	3645.8	13.2	13098
## 39	Arandanos	BioBio	V1	1642.1	13.5	12018
## 40	Arandanos	BioBio	V1	1823.2	13.6	12051
## 41	Arandanos	BioBio	V1	1801.9	12.4	12004
## 42	Arandanos	BioBio	V1	1742.7	12.6	12005
## 43	Arandanos	BioBio	V1	1731.6	13.8	12069
## 44	Arandanos	BioBio	V1	1601.4	11.7	11906
## 45	Arandanos	BioBio	V1	1793.8	13.9	11901
## 46	Arandanos	BioBio	V1	1766.3	14.1	11962
## 47	Arandanos	BioBio	V1	1793.7	14.8	11989

```

## 48 Arandanos      BioBio      V1      1742.0      11.5      11988
## 49 Arandanos      BioBio      V1      1656.7      14.0      11976
## 50 Arandanos      BioBio      V1      1659.9      14.8      12014
## 51 Arandanos      BioBio      V1      1573.9      14.9      12083
## 52 Arandanos      BioBio      V1      1781.8      13.4      11997
## 53 Arandanos      BioBio      V1      1724.4      12.5      11893
## 54 Arandanos La_Araucania V1      1907.5      11.3      12005
## 55 Arandanos La_Araucania V1      1896.7      12.4      12015
## 56 Arandanos La_Araucania V1      1867.3      11.9      12011
## 57 Arandanos La_Araucania V1      1922.1      13.0      11995
## 58 Arandanos La_Araucania V1      1941.5      11.2      12018
## 59 Arandanos La_Araucania V1      1846.2      12.8      12023
## 60 Arandanos La_Araucania V1      1854.6      11.9      12084
## 61 Arandanos La_Araucania V1      1898.1      10.0      12069
## 62 Arandanos La_Araucania V1      1783.3      11.9      11923
## 63 Arandanos La_Araucania V1      1833.6      13.1      11970
## 64 Arandanos La_Araucania V1      1801.7      12.8      12007
## 65 Arandanos La_Araucania V1      1931.0      11.5      12023
## 66 Arandanos La_Araucania V1      1865.6      13.3      12022
## 67 Arandanos La_Araucania V1      1819.9      10.3      12063
## 68 Arandanos La_Araucania V1      1855.1      11.6      12042
##      rendimiento Perdida_plaga mano_de_obra
## 1      6866.5      35.1      3080
## 2      7122.1      34.3      2728
## 3      7041.2      31.7      3120
## 4      6891.5      40.5      3135
## 5      6927.7      35.6      3276
## 6      6900.4      33.0      2966
## 7      6943.9      38.1      2725
## 8      6885.7      37.9      2874
## 9      6724.3      34.8      2785
## 10     6920.4      34.4      3193
## 11     7056.9      37.9      2872
## 12     6898.6      30.0      2983
## 13     6797.2      37.3      2932
## 14     6858.2      33.9      2965
## 15     7064.3      31.0      2609
## 16     7028.4      33.0      3104
## 17     7127.7      36.4      2790
## 18     7247.3      33.9      2720
## 19     7237.8      36.0      3093
## 20     7017.3      29.9      3002
## 21     7080.4      34.7      3285
## 22     7171.5      38.2      3068
## 23     7246.0      28.3      3308
## 24     5931.1      27.5      3446

```

## 25	6076.3	34.5	3204
## 26	6005.6	38.0	3225
## 27	5964.3	36.9	2979
## 28	6003.4	33.2	3222
## 29	6026.0	36.1	3504
## 30	5973.5	37.7	3349
## 31	6046.6	36.4	3066
## 32	5960.2	34.3	3186
## 33	6011.1	35.6	3205
## 34	6012.2	38.5	3080
## 35	6100.5	29.2	3463
## 36	5932.6	36.9	3410
## 37	5876.5	31.5	2956
## 38	5745.2	38.5	3316
## 39	4957.5	33.6	3914
## 40	5025.6	32.8	3901
## 41	5118.6	27.6	3673
## 42	4980.2	36.3	3953
## 43	5024.6	35.7	4171
## 44	5007.0	32.1	4013
## 45	4825.8	34.2	3864
## 46	5164.7	34.2	3947
## 47	4767.4	34.0	3908
## 48	5013.5	33.4	3764
## 49	5013.0	34.6	4029
## 50	5102.8	29.6	4132
## 51	4924.8	35.0	4030
## 52	5061.5	31.4	3882
## 53	4892.4	41.8	3698
## 54	4166.3	33.2	3198
## 55	4439.0	33.6	3303
## 56	4374.1	37.6	3207
## 57	4506.7	37.8	3484
## 58	4466.9	37.1	3212
## 59	4469.6	33.4	3356
## 60	4517.0	31.8	3408
## 61	4538.1	34.9	3319
## 62	4456.1	42.8	3280
## 63	4444.0	35.4	3105
## 64	4525.9	37.4	3404
## 65	4632.6	32.0	3258
## 66	4327.5	31.4	2927
## 67	4526.0	33.6	3028
## 68	4523.7	31.1	3004

3.5 Reducción de una base de datos

La importancia de estas funciones radica en que nos permiten trabajar con subconjuntos de la base de datos original. Por ejemplo, podemos generar una nueva base de datos que solo contenga la información referida a las plantaciones de arándanos provenientes de la región del Maule:

```
data1 <- filter(data, Region == "Maule")
data1
```

##	Cultivo	Region	Variedad	Hectareas	Temperatura	costo_jh	rendimiento
## 1	Arandanos	Maule	V1	3750.2	15.6	12968	5931.1
## 2	Arandanos	Maule	V1	3668.7	11.6	13013	6076.3
## 3	Arandanos	Maule	V1	3586.2	15.8	13007	6005.6
## 4	Arandanos	Maule	V1	3678.0	15.0	13033	5964.3
## 5	Arandanos	Maule	V1	3729.1	12.5	13057	6003.4
## 6	Arandanos	Maule	V1	3709.2	15.9	13039	6026.0
## 7	Arandanos	Maule	V1	3636.3	15.6	12961	5973.5
## 8	Arandanos	Maule	V1	3749.6	13.7	13023	6046.6
## 9	Arandanos	Maule	V1	3644.6	12.5	12985	5960.2
## 10	Arandanos	Maule	V1	3634.6	14.5	13041	6011.1
## 11	Arandanos	Maule	V1	3649.5	15.1	12942	6012.2
## 12	Arandanos	Maule	V1	3788.0	13.7	13121	6100.5
## 13	Arandanos	Maule	V1	3699.9	15.9	12945	5932.6
## 14	Arandanos	Maule	V1	3718.2	14.0	12967	5876.5
## 15	Arandanos	Maule	V1	3645.8	13.2	13098	5745.2
## 16	Arandanos	Maule	V2	3699.5	14.0	13065	5890.1
## 17	Arandanos	Maule	V2	3627.4	13.5	13008	6078.3
## 18	Arandanos	Maule	V2	3685.2	16.8	13088	6062.3
## 19	Arandanos	Maule	V2	3770.5	14.5	13034	5852.9
## 20	Arandanos	Maule	V2	3711.1	15.5	13064	6016.7
## 21	Arandanos	Maule	V2	3758.2	13.7	12975	6112.8
## 22	Arandanos	Maule	V2	3685.5	12.7	13087	6032.1
## 23	Arandanos	Maule	V2	3619.6	13.5	12902	6139.0
## 24	Arandanos	Maule	V2	3724.8	14.7	12974	6008.3
## 25	Arandanos	Maule	V2	3735.7	13.5	12976	5936.0
## 26	Arandanos	Maule	V2	3672.5	15.1	12937	5929.8
## 27	Arandanos	Maule	V2	3657.3	15.3	12946	6036.1
## 28	Arandanos	Maule	V2	3657.9	13.4	12963	6084.7
## 29	Arandanos	Maule	V2	3721.7	15.8	12955	5848.1
## 30	Arandanos	Maule	V2	3762.6	16.2	13059	6125.2
##	Perdida_plaga	mano_de_obra					
## 1		27.5		3446			
## 2		34.5		3204			
## 3		38.0		3225			

## 4	36.9	2979
## 5	33.2	3222
## 6	36.1	3504
## 7	37.7	3349
## 8	36.4	3066
## 9	34.3	3186
## 10	35.6	3205
## 11	38.5	3080
## 12	29.2	3463
## 13	36.9	3410
## 14	31.5	2956
## 15	38.5	3316
## 16	35.5	3027
## 17	34.2	2980
## 18	38.1	3319
## 19	33.5	3640
## 20	32.1	2891
## 21	37.5	3559
## 22	33.5	3278
## 23	33.3	3356
## 24	37.9	3159
## 25	40.2	3363
## 26	34.2	3341
## 27	33.3	3094
## 28	37.8	3480
## 29	30.7	3088
## 30	30.0	3434

Ahora, podemos aplicar todas las funciones que hemos aprendido para explorar la información contenida en una base de datos.

Chapter 4

Visualización de datos en RStudio

En el laboratorio anterior aprendimos a importar una base de datos a R y a realizar una exploración básica de la información contenida en nuestra base de datos. Ahora, ha llegado el momento de seguir explorando nuestros datos pero de una manera diferente. En este laboratorio realizaremos una exploración visual de nuestros datos. Para ello aprenderemos una serie de gráficos básicos en R y luego los “enchularemos”. Al realizar gráficos podremos ver posibles patrones en nuestros datos, formas de las distribuciones de nuestras variables y posibles valores “extraños” o fuera de rango (*outliers*).

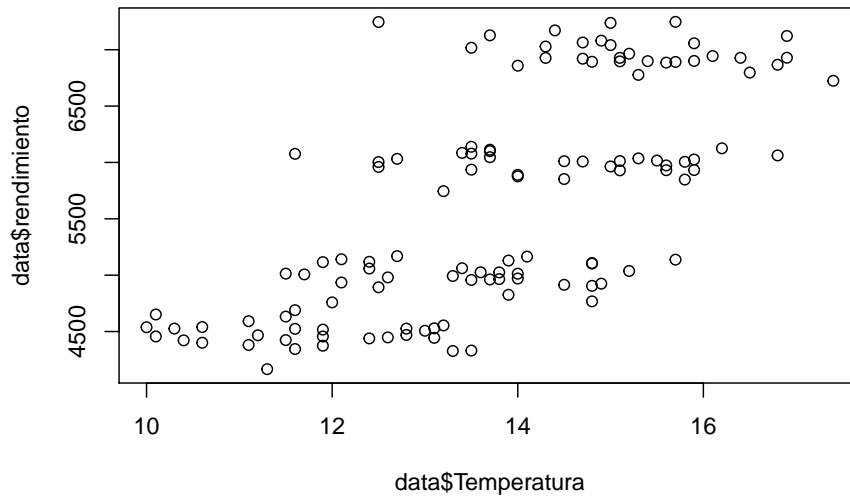
4.1 Gráfico de dispersión o *Scatterplot*

Un gráfico de dispersión es una visualización de datos bidimensionales que usa puntos para representar los valores obtenidos para dos variables diferentes: una graficada a lo largo del eje x y la otra graficada a lo largo del eje y.

Existen muchas maneras de crear un gráfico de dispersión en R. La función básica es `plot(x,y)`.

Por ejemplo visualicemos las variables Temperatura (eje x) y rendimiento (eje y) de la base de datos utilizada en el laboratorio 3:

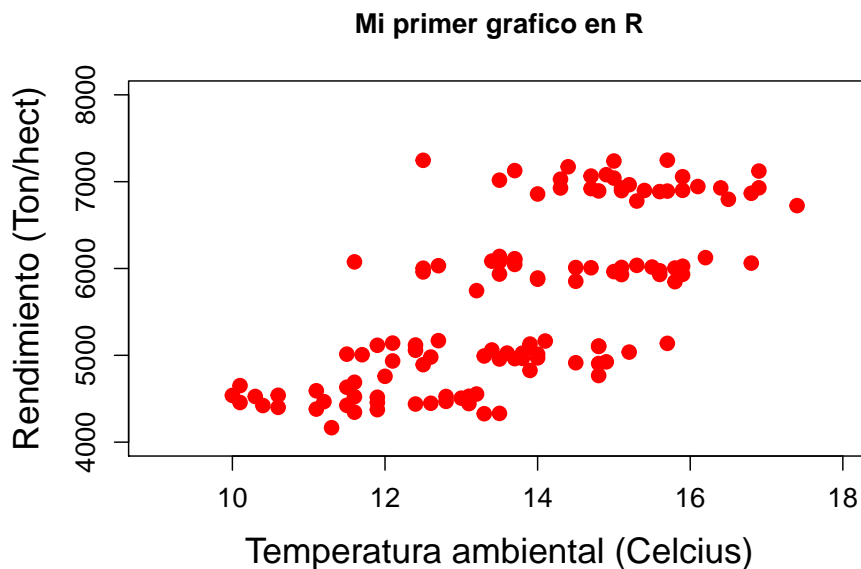
```
data <- read.csv("./datos/dataset.csv")
plot(data$Temperatura, data$rendimiento)
```



Felicitaciones, acaban de generar el primero de muchos gráficos que aprenderán en este curso. Sin embargo, la tarea no está completa. Como podemos apreciar, el gráfico que nos entregó R es bastante “rústico”. Siempre intenten presentar sus datos de una forma atractiva. Esto puede marcar la diferencia entre un buen trabajo y un trabajo sobresaliente.

Afortunadamente, R nos permite “enchular” nuestros gráficos. Para ello debemos ingresar comandos específicos para cada una de las características que queremos modificar:

```
plot(data$Temperatura,data$rendimiento,
      xlab = "Temperatura ambiental (Celcius)",
      ylab = "Rendimiento (Ton/hect)",
      xlim = c(9,18),
      ylim = c(4000,8000),
      main = "Mi primer grafico en R",
      pch=20,
      col="red",
      cex=2,
      cex.lab=1.5,
      cex.axis=1.2)
```



A simple vista, son múltiples los parámetros gráficos que hemos cambiado. A continuación, se detalla la lista de los parámetros utilizados y sus definiciones:

xlab = Permite modificar el nombre del eje x **ylab** = Permite modificar el nombre del eje y **xlim** = Establece los límites del eje x **ylim** = Establece los límites del eje y **main** = Permite agregar un título a nuestro gráfico **pch** = Permite modificar la forma de los puntos en nuestro gráfico **col** = Permite modificar el color **cex** = Permite modificar el tamaño del texto

Para aprender sobre otros comandos que permiten modificar parámetros les sugerimos visitar el siguiente link.

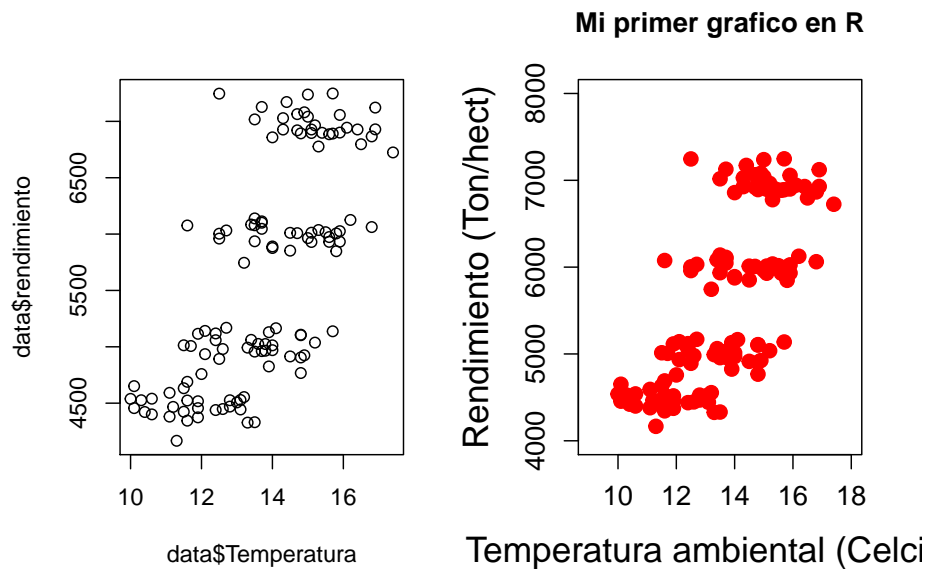
Nunca olviden que aprender R implica un constante prueba y error, por lo cual ahora que tienen una lista de parámetros, comiencen a probar que sucede cuando cambian los valores de cada uno de ellos. No tengan miedo a equivocarse!

Hasta ahora hemos generados dos gráficos, sin embargo solo podemos ver uno a la vez. R proporciona los comandos necesarios para ver más de un gráfico de forma simultánea. Con la función `par()`, podemos incluir la opción `mfrow=c()` para crear una matriz de filas y columnas.

Utilicemos los dos scatterplot anteriores para generar una matriz gráfica que contenga dos gráficos ordenados en una fila y con dos columnas:

```
par(mfrow=c(1,2))  
  
plot(data$Temperatura,data$rendimiento)
```

```
plot(data$Temperatura,data$rendimiento,
     xlab = "Temperatura ambiental (Celcius)",
     ylab = "Rendimiento (Ton/hect)",
     xlim = c(9,18),
     ylim = c(4000,8000),
     main = "Mi primer grafico en R",
     pch=20,
     col="red",
     cex=2,
     cex.lab=1.5,
     cex.axis=1.2)
```



Fantástico! Tenemos ambos scatterplot graficados al mismo tiempo, uno al lado del otro. Ahora bien, en caso de que quisiéramos que estén un scatterplot sobre el otro, ¿¿como lo harían??

4.2 Diagrama de cajas o *Boxplot*

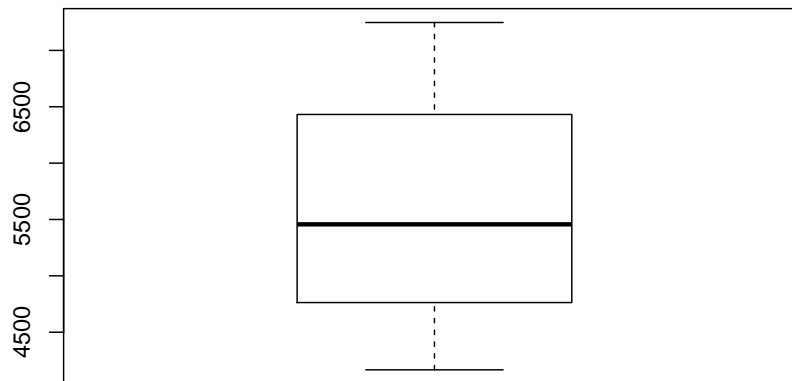
Un diagrama de cajas es un gráfico que está basado en cuartiles y mediante el cual se visualiza la

distribución de un conjunto de datos. Está compuesto por un rectángulo (la «caja») y dos brazos (los «bigotes»). Los diagramas de caja se pueden crear para variables individuales o para grupos de variables.

La función es `boxplot(x)`, donde `x` es nuestra variable a graficar:

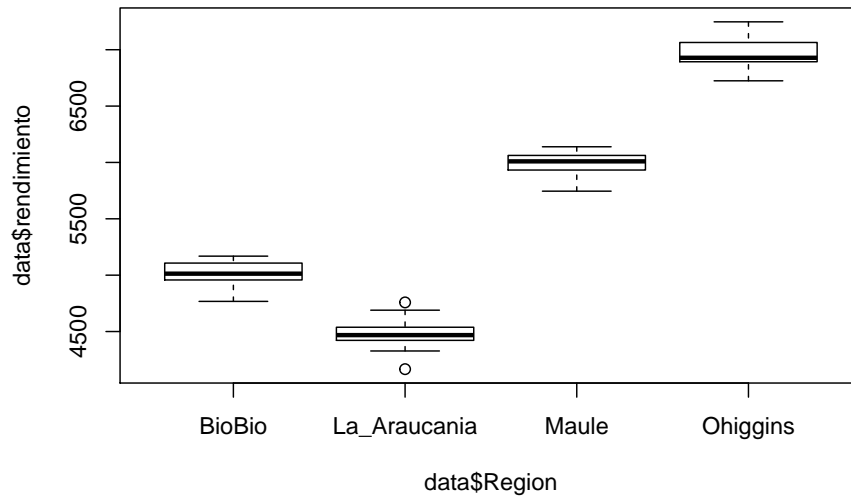
Generemos un gráfico de cajas para la información contenida en la variable rendimiento:

```
boxplot(data$rendimiento)
```



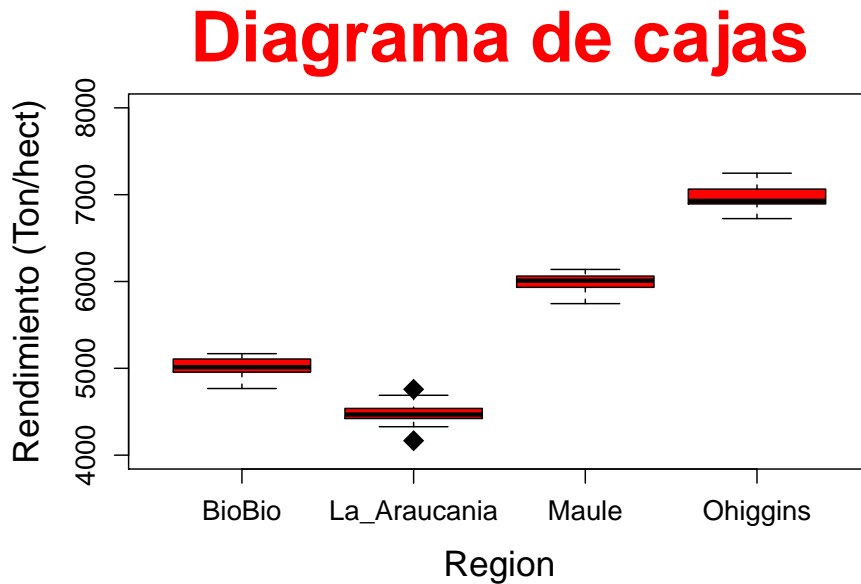
Ahora, generemos el diagrama para la información contenida en la variable rendimiento según la región en la cual se realizó el muestreo (grupo de variable):

```
boxplot(data$rendimiento ~ data$Region)
```



En caso de que quisiéramos encharlar nuestro gráfico de cajas con dos variables, solamente debemos modificar los parámetros gráficos:

```
plot(data$rendimiento ~ data$Region,  
      xlab = "Region",  
      ylab = "Rendimiento (Ton/hect)",  
      ylim = c(4000,8000),  
      main = "Diagrama de cajas",  
      pch=18,  
      col="red",  
      col.main="red",  
      cex=2,  
      cex.main=3,  
      cex.lab=1.5,  
      cex.axis=1.2)
```

4.3 Histogramas

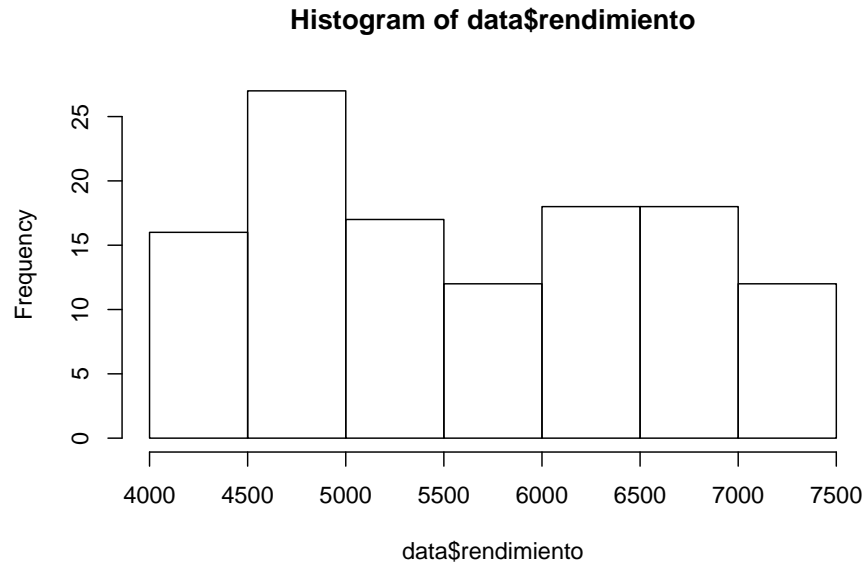
Un histograma es una representación visual de la distribución de un conjunto de datos. Como tal, la forma de un histograma es su característica más obvia e informativa: nos permite ver fácilmente dónde está el medio en su distribución de datos, qué tan cerca están los datos de este medio y dónde se encuentran posibles valores atípicos.

El histograma consiste en un eje x, un eje y, además de varias barras de diferentes alturas. El eje y muestra con qué frecuencia (número de veces que se repite una observación) se producen los valores en el eje, mientras que las barras agrupan rangos de valores o categorías continuas en el eje x (intervalos). Esto explica por qué los histogramas se llaman “histograma de frecuencia”.

La función es `hist(x)`, donde `x` es nuestra variable a graficar.

Generemos un histograma de frecuencias para la información contenida en la variable rendimiento:

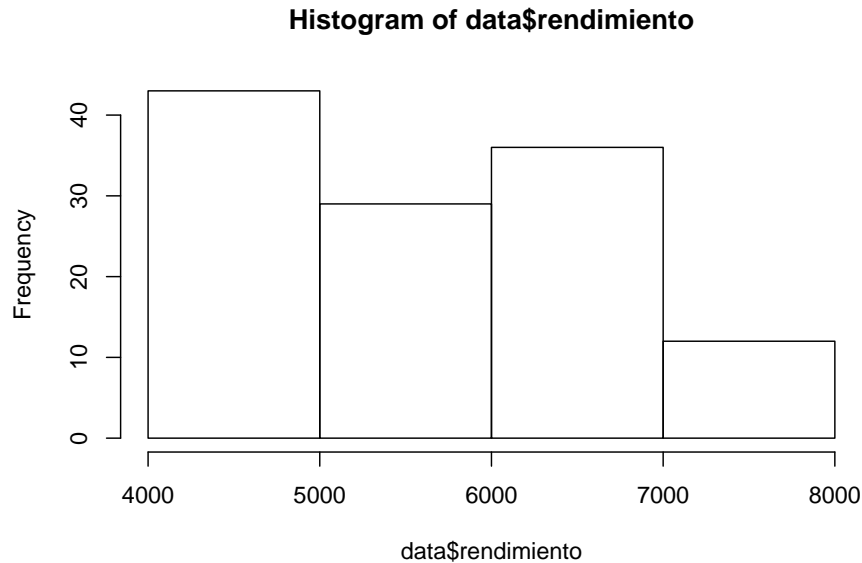
```
hist(data$rendimiento)
```



Al igual que los gráficos anteriores, R nos permite modificar parámetros gráficos de los histograma de frecuencia. Por ejemplo, podemos especificar el número de intervalos en las cuales queremos que se agrupen nuestros datos. Para ello utilizamos el comando `breaks=`.

Generemos un histograma de frecuencias para la información contenida en la variable rendimiento pero que esté agrupada solamente en 4 intervalos:

```
hist(data$rendimiento,breaks=4)
```



Intentemos algo más que nos servirá para los próximos laboratorios. En algunas ocasiones, estaremos más interesados en la densidad (probabilidad relativa de un dato) que en la frecuencia de nuestros datos. En lugar de contar el número de puntos de datos por intervalo, R puede dar las densidades de probabilidad usando la opción `freq= FALSE`, lo cual nos permite normalizar el histograma de modo que el área total es igual a 1. Ahora el eje y está etiquetado como densidad en lugar de frecuencia.

Modifiquemos nuestro histograma de frecuencia anterior por un histograma de densidad y grafiquemos uno al lado del otro:

```
par(mfrow=c(1,2))

hist(data$rendimiento)

hist(data$rendimiento,
      main="Histograma de densidad",
      xlab="Rendimiento (Ton/hect)",
      xlim=c(3500,8500),
      col="red",
      freq=FALSE,
      cex.main=2,
      cex.lab=1.5,
      cex.axis=1.2,
      Freq=FALSE)
```

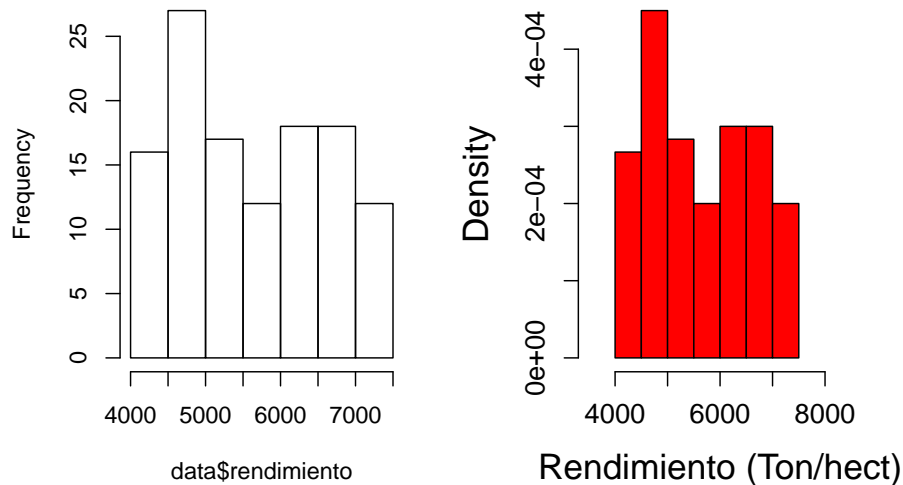
```
## Warning in plot.window(xlim, ylim, "", ...): "Freq" is not a graphical
## parameter

## Warning in title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...):
## "Freq" is not a graphical parameter

## Warning in axis(1, ...): "Freq" is not a graphical parameter

## Warning in axis(2, ...): "Freq" is not a graphical parameter
```

Histogram of data\$rendimiento-histograma de densidad



4.4 Ejercicios

1. Generar un gráfico que permita visualizar la relación entre las variables Temperatura y rendimiento. El gráfico debe contener solamente los datos obtenidos de las regiones del Maule y Araucanía.
2. Generar un histograma de frecuencia con los rendimientos superiores a 5000 Toneladas/hectárea.
3. Generar un gráfico de cajas para la variedad 1 que muestre solamente los rendimientos superiores a 5000 Toneladas/hectárea.

Bibliography

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.