



Universidad de Valladolid

Facultad de Ciencias Económicas y Empresariales

Trabajo de Fin de Grado

Grado en Marketing e Investigación de Mercados

Introducción al Análisis Multivariante con R

Presentado por:

Pedro Gordo Gómez

Valladolid, 10 de septiembre de 2020

RESUMEN

R es un software basado en un lenguaje de programación con el mismo nombre muy empleado en el área de la estadística, orientado a objetos y con múltiples herramientas e interfaces desarrolladas para el mismo. Estas interfaces se componen de menús y tienen un apartado gráfico muy simple y estético que acerca R a personas sin conocimiento en programación, permitiendo también el desarrollo de los análisis estadísticos principales de forma simple y gratuita. Los resultados, tablas y gráficos obtenidos permiten tener información muy detallada debido a la multitud de posibilidades de edición que ofrece R.

Palabras Clave: R, análisis estadístico, programación, interfaz

Códigos de clasificación JEL: C65, C80, C88

ABSTRACT

R is a software based on the programming language widely used in statistics, guide to objects, with multiple tools and interfaces developed for it. These interfaces are composed by menus and have a very simple and aesthetic graphical section that brings R to people without knowledge in programming, also allowing the development of the statistical analysis in a simple freeway. The results, tables and graphs obtained allow you to have very detailed information due to the multitude of editing possibilities that R offers.

Keywords: R, statistical analysis, programming, interface

JEL classification codes: C65, C80, C88

ÍNDICE

Listado de imágenes	4
1. Introducción.....	5
2. Los primeros pasos en R.....	8
2.1. Instalación e interfaz.....	8
2.2. Formas de trabajar en R.....	10
2.3. Formas de introducir datos en R.....	12
3. Interfaces gráficas.....	15
4. Tipos de objetos.....	19
4.1. Vectores.....	21
4.2. Matrices.....	21
4.3. Data Frames.....	23
4.4. Funciones.....	24
5. Análisis estadísticos básicos y gráficas.....	26
6. Técnicas multivariantes de análisis de datos.....	31
6.1. Análisis de conglomerados (Clúster).....	32
6.2. Análisis de componentes principales.....	39
7. Conclusiones.....	42
8. Referencias bibliográficas.....	43

LISTADO DE IMÁGENES

Imagen 2.1. Ventana inicial de R.....	8
Imagen 2.2. Script.....	10
Imagen 2.3. Ayuda acerca de las gráficas en R.....	11
Imagen 2.4. Introducción de datos de forma directa.....	12
Imagen 2.5. Función scan.....	13
Imagen 2.6. Función read.table.....	14
Imagen 2.7. Función write.....	14
Imagen 2.8. Lectura de bases de datos.....	15
Imagen 3.1. R Studio.....	16
Imagen 3.2. R Commander.....	18
Imagen 4.1. Función ls.....	20
Imagen 4.2. Función rm.....	20
Imagen 4.3. Vector.....	21
Imagen 4.4. Matriz.....	22
Imagen 4.5. Data Frame.....	24
Imagen 4.6. Ejemplo de función.....	25
Imagen 4.7. Función.....	25
Imagen 5.1. Estadísticos básicos.....	26
Imagen 5.2. Función plot.....	28
Imagen 5.3. Diagrama de caja.....	29
Imagen 5.4. Histograma.....	30
Imagen 5.5. Diagrama de dispersión.....	31
Imagen 6.1. Clúster jerárquico 1.....	35
Imagen 6.2. Clúster jerárquico 2.....	37
Imagen 6.3. Clúster no jerárquico.....	39
Imagen 6.4. Análisis de componentes principales.....	41

1. INTRODUCCIÓN

R es un software basado en un lenguaje de programación con el mismo nombre muy empleado en el análisis de datos. Tal y como se puede leer en su web “R es un sistema para computación estadística y gráfica. Consiste en un lenguaje más un entorno de tiempo de ejecución con gráficos, un depurador y acceso a ciertas funciones del sistema y la capacidad de ejecutar programas almacenados en archivos de script” (Hornik, 2018). De la anterior definición se puede deducir que R es un software con múltiples plugins¹ que permite realizar desde los procedimientos estadísticos más sencillos hasta las técnicas más avanzadas. Cabe destacar que este software es gratuito distribuido mediante una licencia pública GNU (sistema operativo de software libre).

El núcleo de R está basado en funciones e interconectado con los lenguajes de C y C++ (lenguaje de programación basado en una función principal “main” y seguida por el resto de las funciones (Bonet, 2019).

Tal y como dice Hornik (2018) R fue desarrollado inicialmente por Ross Ihaka y Robert Gentleman pertenecientes al departamento de Estadística de la Universidad de Auckland (Nueva Zelanda) en el año 1996. Gracias a que el software es de libre utilización, se puede acceder y modificar el código fuente para corregir errores.

El diseño de R está basado en la implementación de dos lenguajes de programación que ya existían previamente (<https://cran.r-project.org/>):

¹ Según Fox (2019) un Plugin es un paquete de datos que se pueden cargar de forma automática facilitando la generación de funciones, gráficas, etc. <https://www.rdocumentation.org/packages/Rcmdr/versions/2.6-2/topics/Plugins>

- **S** es un lenguaje de programación de alto nivel y un entorno para el análisis de datos y elaborar gráficos desarrollado por los laboratorios AT&T Bell en 1979 dirigido por John M. Chambers. En la actualidad, en lugar de S se emplea S-Plus, ya que genera un valor agregado superior a S, corrigiendo errores y con una interfaz más estética. S-Plus fue desarrollado por TIBCO Software Inc. Las diferencias entre R y S se pueden dividir en tres tipos:
 - Diferencias en el léxico de la programación: En S los valores de las variables libres están determinados por un conjunto de variables globales; mientras que en R están determinados por el entorno en el que se creó la función.
 - Diferencias en el modelado del código: Existen determinadas funciones u operadores que se introducen de forma diferente en R y en S.
 - Otras diferencias: posibles errores de programación al cambiar de un lenguaje a otro.

- El **esquema de Sussman**; lenguaje de programación ilustrado en el libro “*Structure and Interpretation of Computer Programs*” (Abelson et al, 1996).

R contiene un gran número de procedimientos estadísticos. También hay un gran conjunto de funciones que proporcionan un entorno gráfico flexible. Así, existe multitud de paquetes de descarga libre que permiten desarrollar ciertos gráficos que no vienen de forma predeterminada.

El elemento más relevante de la definición de R es que es un lenguaje orientado a objetos, ya que “las variables, datos, funciones, resultados, etc., se guardan en la memoria activa del computador en forma de objetos con un nombre específico. El usuario puede modificar o manipular estos objetos con operadores (aritméticos, lógicos, y comparativos) y funciones (que a su vez son objetos)” (Paradis, 2003).

Hay muchas diferencias entre R y otros softwares dedicados a los análisis estadísticos, como el SPSS, pero son tres las diferencias más relevantes. La primera diferencia es que R es un software gratuito, mientras que para utilizar otros softwares estadísticos como SPSS se requiere licencia de pago. Además, R es más versátil que SPSS, ya que como se ha descrito previamente se puede modificar su código fuente para añadir mejoras fácilmente. Finalmente, el trabajo con paquetes informáticos estadísticos en SPSS es más sencillo puesto que tiene una interfaz más intuitiva y clara que cualquier de las interfaces de R. Así, para realizar un tipo de análisis en R, se necesita conocer las funciones y cómo se deben de introducir, mientras que el diálogo que se establece en las diferentes opciones de los menús de SPSS es bastante claro y fácilmente legible, lo que facilita su uso.

El objetivo fundamental de este trabajo consiste en conocer los procedimientos implementados en R para realizar los análisis estadísticos más básicos y alguna de las técnicas de análisis multivariante más utilizadas, en concreto, el análisis de conglomerados y el análisis de componentes principales.

La estructura del trabajo es la siguiente. En la siguiente sección se tratará de explicar las formas de instalar el programa en los diferentes sistemas operativos y como trabajar e introducir datos en el programa. Seguidamente se presentarán distintas interfaces gráficas que existen para R. En la sección cuarta se presentarán los principales tipos de objetos que hay en R. Más adelante se abordará cómo realizar un análisis estadístico básico y los principales gráficos con R. Seguidamente se llegará al apartado central del trabajo en el que se abordara cómo realizar dos de las técnicas multivariantes más empleadas como son el análisis de conglomerados (clúster) y el análisis de componentes principales. Para finalizar el trabajo se trasladarán unas conclusiones de todo lo realizado en el mismo.

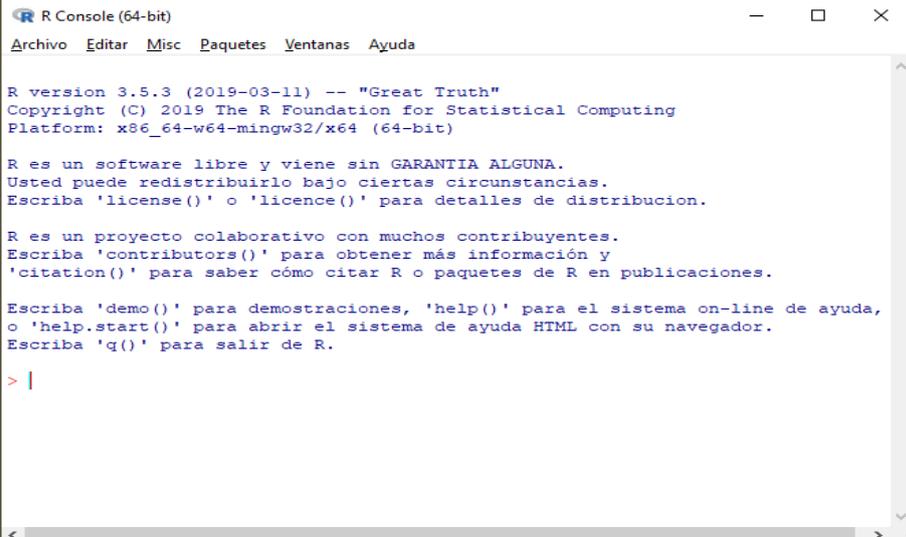
2. LOS PRIMEROS PASOS EN R

En este apartado primeramente se describe la forma de instalar R en un ordenador (ya tenga sistema operativo Windows o Mac). Seguidamente se enumeran las diferentes maneras de trabajar y de introducir datos en R.

2.1 Instalación e interfaz

Según Jiménez (2019) R está disponible tanto para sistemas operativos de Windows, Mac OS y Linux. Para instalar R en Windows hay que seguir los siguientes pasos:

1. Ir a la web de R, en <https://cran.r-project.org/bin/windows/>, y seleccionar instalar R por primera vez, posteriormente descargar R 3.6.1 para Windows.
2. Una vez descargado el programa, se ejecuta R-3.6.1-win. Exe, se selecciona el idioma en el que se quiere instalar, se aceptan las condiciones de uso y licencia y se establece el directorio donde se quiere instalar el software y los ficheros que se quieren instalar.
3. Una vez instalado el software en el ordenador, la interfaz de inicio de R es la siguiente, como se observa en la Imagen 2.1.



```
R Console (64-bit)
Archivo  Editar  Misc  Paquetes  Ventanas  Ayuda

R version 3.5.3 (2019-03-11) -- "Great Truth"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R es un software libre y viene sin GARANTIA ALGUNA.
Usted puede redistribuirlo bajo ciertas circunstancias.
Escriba 'license()' o 'licence()' para detalles de distribucion.

R es un proyecto colaborativo con muchos contribuyentes.
Escriba 'contributors()' para obtener más información y
'citation()' para saber cómo citar R o paquetes de R en publicaciones.

Escriba 'demo()' para demostraciones, 'help()' para el sistema on-line de ayuda,
o 'help.start()' para abrir el sistema de ayuda HTML con su navegador.
Escriba 'q()' para salir de R.

> |
```

Imagen 2.1. Ventana inicial de R

Como se puede apreciar en la Imagen 2.1, dentro de la pantalla de inicio, existen varias pestañas en la parte superior:

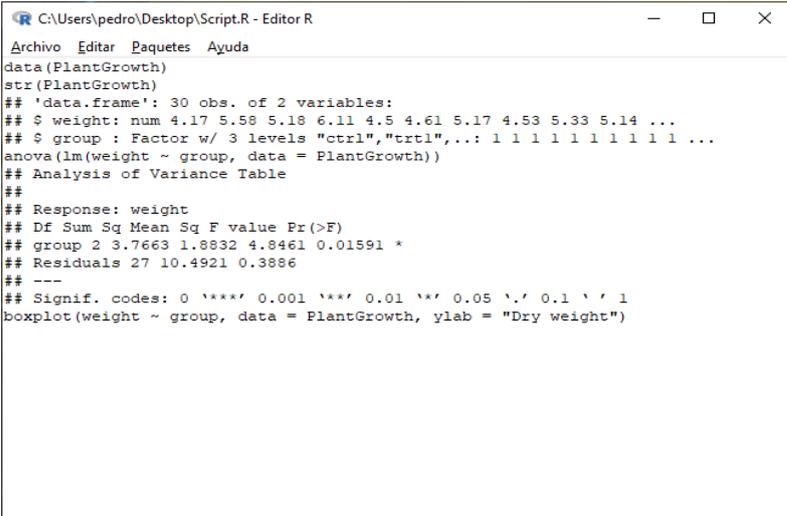
- Archivo: en esta pestaña se puede interpretar el código fuente del programa y también abrir archivos del software mediante scripts, áreas de trabajo o un histórico.
- Editar: permite copiar y pegar datos, además de poder cambiar la interfaz gráfica de R, abrir el editor de datos y limpiar la consola.
- Misc: desde aquí se puede parar la computación de los datos y listar o eliminar todos los objetos introducidos en la consola.
- Paquetes: esta pestaña permite cargar, instalar y actualizar paquetes de la librería que vienen preinstalados junto con el software o que se pueden descargar de la web de R (CRAN Project) <https://cran.r-project.org/web/packages/>
- Ventanas: desde esta pestaña se pueden modificar el modo de visualización de la consola. Hay 5 modos de visualización: cascada, dividida horizontalmente, dividida verticalmente, minimizar el grupo y grupo de restauración.
Ayuda: desde aquí se puede consultar todas las dudas relativas a la utilización de R y su normativa legislativa.

En la Imagen 2.1 también se observa cómo en la pantalla principal del programa hay un signo > de color rojo que describe la línea de comandos, que sirve para introducir las funciones, datos y valores en R.

2.2. Formas de trabajar en R

Existen 2 principales formas de trabajar en R:

- La primera forma es introduciendo directamente las sentencias directamente después del signo > en la línea de comandos. Cada sentencia debe tener un nombre y unos argumentos que se escriben entre paréntesis (elegidos por el usuario). Esta forma es algo arriesgada, ya que hay que programar sin tener el mínimo fallo, puesto que si existe algún error el programa no reconocerá el análisis que estamos realizando y el único error que mostrará la consola será que el objeto no ha sido encontrado.
- La segunda forma es mediante scripts, como se muestra en la Imagen 2.2. Según Paradis (2003) los scripts son archivos de texto simples que incluyen todos los comandos de R realizados. Estos scripts son muy útiles en R, ya que su principal misión es que se pueden escribir varias sentencias y ejecutarlas simultáneamente. Para cargar un script solo hay que abrir la pestaña archivo y después abrir script. Es importante destacar que los scripts deben estar en formato de R o S para que el programa pueda abrirlos.



```
C:\Users\pedro\Desktop\Script.R - Editor R
Archivo Editar Paquetes Ayuda
data(PlantGrowth)
str(PlantGrowth)
## 'data.frame': 30 obs. of 2 variables:
## $ weight: num 4.17 5.58 5.18 6.11 4.5 4.61 5.17 4.53 5.33 5.14 ...
## $ group : Factor w/ 3 levels "ctrl","trtl",...: 1 1 1 1 1 1 1 1 1 1 ...
anova(lm(weight ~ group, data = PlantGrowth))
## Analysis of Variance Table
##
## Response: weight
## Df Sum Sq Mean Sq F value Pr(>F)
## group 2 3.7663 1.8832 4.8461 0.01591 *
## Residuals 27 10.4921 0.3886
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
boxplot(weight ~ group, data = PlantGrowth, ylab = "Dry weight")
```

Imagen 2.2. Script

Es importante señalar que en R se pueden introducir paquetes o librerías que son programas escritos previamente por otros usuarios. Esta característica le confiere a R una gran versatilidad, dado que existe multitud de paquetes, que se pueden descargar e introducir en el software. Tal y como se indica Hornik (2018), para cargar un paquete es necesario abrir la pestaña paquetes y después en cargar un paquete. Otra forma de cargar un paquete es introducir en la línea de comandos: `install.packages()` y después el comando `library()`. El primer comando instala el paquete a elegir y el segundo carga el paquete.

Si se introduce el comando `help()` se puede acceder a la ayuda que proporciona el programa para saber cómo programar en su consola. Por ejemplo, si se quiere buscar ayuda acerca de cómo introducir una gráfica en R se añade la palabra `plot`, con lo que la línea de comandos a introducir sería `help(plot)`. En la Imagen 2.3 se puede observar la ayuda que proporciona R de la función `plot`.

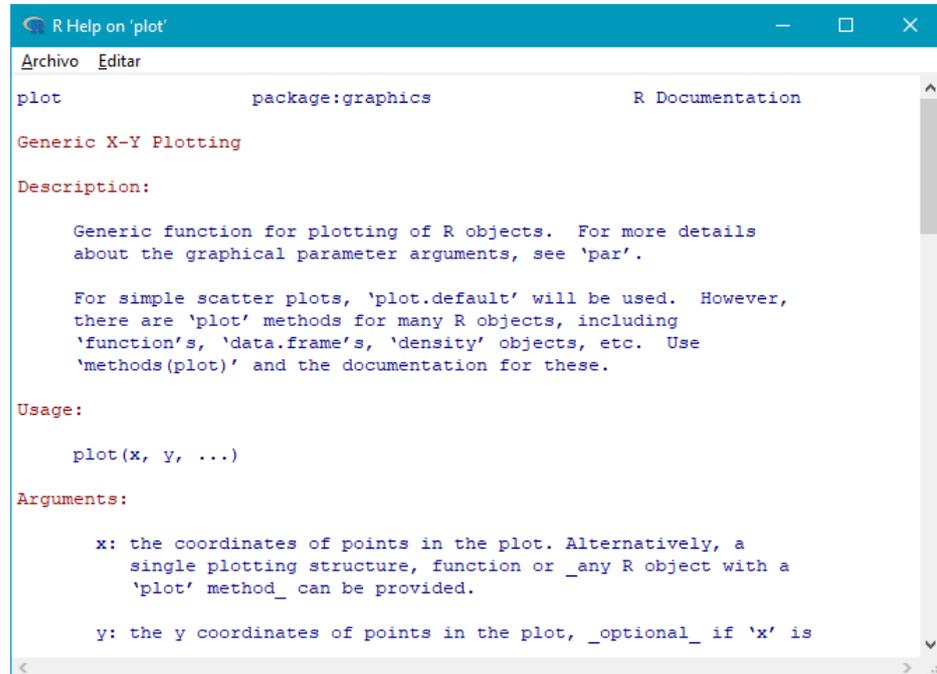


Imagen 2.3. Ayuda acerca de las gráficas en R

2.3. Formas de introducir datos en R

Existen dos formas de introducir datos en R:

- La primera forma de introducir datos en R es introduciendo las variables de forma directa. En el recuadro azul de la Imagen 2.4 se muestra cómo se generan las variables *x*, *y* e *w* con un solo valor numérico en la primera línea de comandos. Para leer el contenido de estas variables es necesario nombrarlas seguido de punto y coma como se observa en la segunda línea de comandos. En el recuadro amarillo de la Imagen 2.4 se observa cómo se genera un vector numérico llamado *z* con cuatro valores (2,4,5345,67). Para poder ver el contenido de ese vector *z* es necesario escribirlo y el programa devuelve los argumentos que forman el vector.

```
> x <- 5 ; y <- 6 ; w <- 9
> x ; y ; w
[1] 5
[1] 6
[1] 9
> z <- c (2,4,5345,67)
> z
[1] 2 4 5345 67
```

Imagen 2.4. Introducción de datos de forma directa

- La segunda forma según Muñoz (2015) es recuperar los datos de un fichero generado previamente gracias a las funciones *scan()* y *read.table()* (importar datos a R).

- Función `scan()`: Esta función permite introducir datos como un vector o una lista desde la consola de R o desde un fichero externo a R. Esta función tiene multitud de argumentos, pero los más destacables son: *file* (introducción de la ruta del fichero entre paréntesis), *what* (que indica el tipo de dato a ser leído, ya sea numérico, lógico...) y *n* (indica el máximo número de valores a ser leído por la consola). En el ejemplo de la Imagen 2.5 dentro de recuadro azul se muestra cómo introducen datos mediante la función `scan()` en un objeto generado llamado `datos`. En el ejemplo de la Imagen 2.5, en el recuadro amarillo, se observa cómo se introducen datos de un documento de texto en R, gracias a la función `scan()` e introduciendo la ruta del fichero entre paréntesis como argumento.

```
> datos <- scan ()
1: 2
2: 4
3: 6
4:
Read 3 items
> datos
[1] 2 4 6

> scan ("C:\\Users\\pedro\\Desktop\\datos.txt")
Read 10 items
[1] 1 2 3 4 5 6 7 8 9 10
> |
```

Imagen 2.5. Función scan

- Función *read.table()*: Con esta función se pueden leer tablas a partir de un archivo texto (fichero). Esta función genera un data frame a partir de los datos contenidos en el archivo. Esta función tiene multitud de argumentos, los más relevantes son: *file* (introducción de la ruta del fichero entre paréntesis), *header* (es un argumento que puede ser true o false, true indica que la primera fila del archivo de texto contiene los nombres de las variables), *row.names* (este argumento contiene el nombre de las filas) y *col.names* (este argumento contiene el nombre de las columnas). En el ejemplo de la Imagen 2.6 se muestra cómo se importan datos de un fichero de texto llamado *datos 1*.

```
> read.table ("C:\\Users\\pedro\\Desktop\\datos 1.txt")
  edad altura
paco  20   174
pepe  22   180
kiko  19   170
```

Imagen 2.6. Función read.table

De forma análoga, también según Muñoz (2015) se puede convertir datos de R a otros formatos (exportar datos de R) para utilizarlos en otros programas. Para ello es necesario utilizar la función *write()* o *write.table()* en el caso de una tabla. La función *write()* tiene como argumentos más relevantes: *x* (el fichero que se va a generar con los datos introducidos), *file* (la ruta del fichero) y *ncolumns* (el número de columnas donde se van a escribir los datos).

```
> x<-c(1,2,3,4,5)
> write(x,"C:\\Users\\pedro\\Desktop\\x.txt")
> |
```

Imagen 2.7. Función write

R cuenta con conjuntos de datos que se pueden utilizar por el usuario. En el ejemplo de la Imagen 2.8 se observa cómo primero con el comando `data (iris)` se cargan los datos de la librería en la memoria. Posteriormente, se asigna los datos que hay en `iris` a un objeto denominado `datos`. Finalmente, se puede visualizar el contenido de este objeto escribiendo su nombre.

```
> data (iris)
> datos = iris
> datos
```

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

Imagen 2.8. Lectura de bases de datos

3. INTERFACES GRÁFICAS

En este apartado se describen las dos interfaces gráficas más interesantes a la hora de utilizar el lenguaje de programación de R para realizar tanto análisis estadísticos como para realizar gráficos.

Debido a que R tiene un aspecto estético muy rudimentario se han desarrollado las interfaces gráficas que permiten observar comandos, datos y gráficos de un modo más sencillo y unificado. Existen multitud de interfaces gráficas, pero las más usadas debido a la facilidad de uso son R Studio y R Commander.

- R Studio: R Studio de acuerdo con González (2016) es una interfaz gráfica completamente gratuita desarrollada por una empresa llamada R-tools technology, que se puede descargar en el siguiente link (<https://www.rstudio.com/products/rstudio/#Desktop>). Esta interfaz es muy usada debido a su facilidad para mostrar simultáneamente datos, gráficos y los comandos generados en la consola. Una vez abierto R Studio se muestra la ventana que se muestra en la Imagen 3.1.

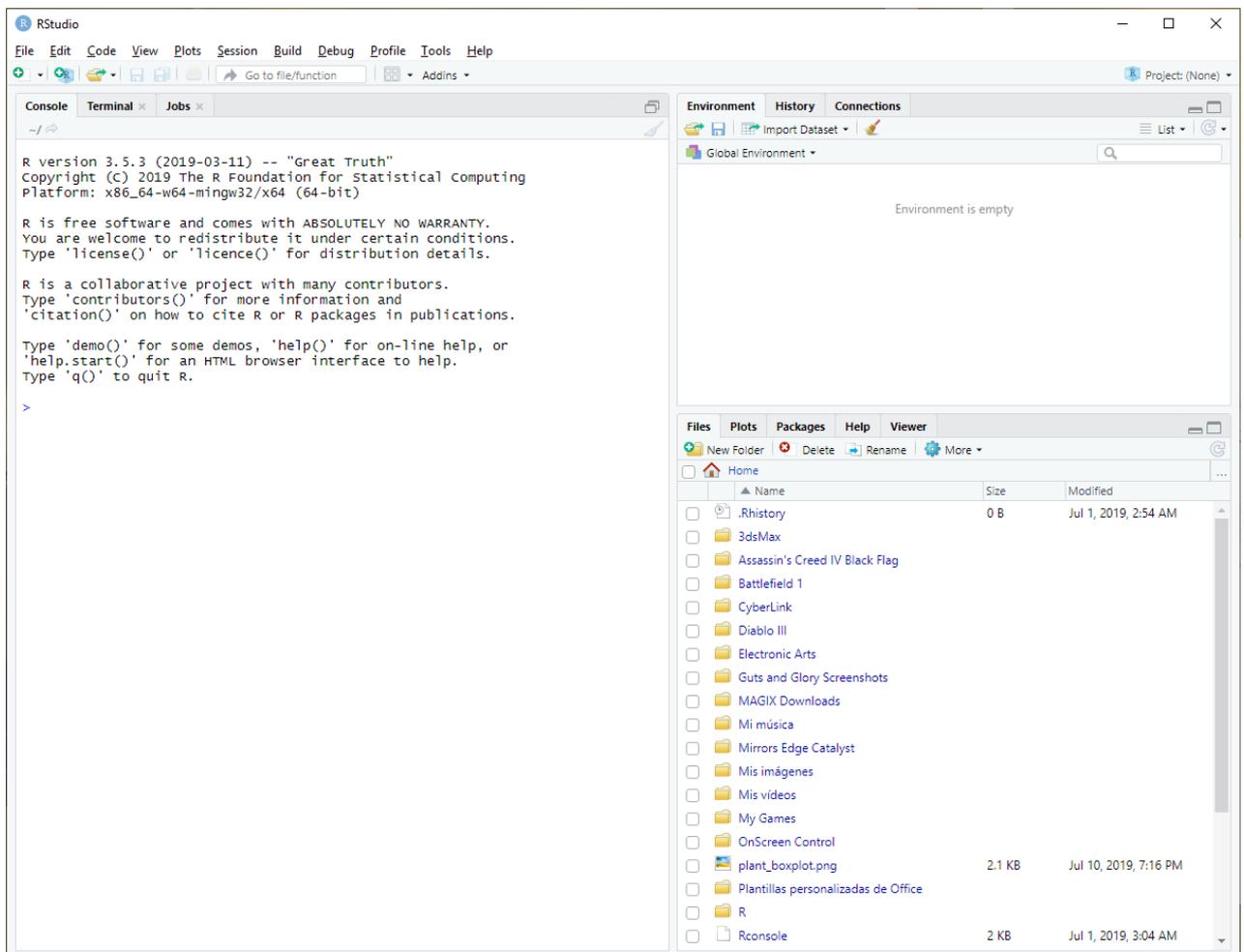


Imagen 3.1. R Studio

La ventana principal de R Studio dispone de una consola a la izquierda donde se escriben los comandos; en la parte superior derecha, se muestra el entorno de trabajo (*environment*) con los datos generados en la consola; y en la parte inferior derecha, se muestra el contenido de nuestro disco o los gráficos generados con los análisis estadísticos realizados. R Studio tiene los siguientes menús:

- File: Se pueden cargar archivos de otra sesión o abrir un nuevo archivo.
 - Edit: Editar las líneas de comando (copiar, pegar, ...)
 - Code: Editar los comandos.
 - View: Formas de visualizar la ventana de R Studio.
 - Plots: Generar gráficas, modificarlas y eliminarlas.
 - Session: Modificación de la sesión en R.
 - Build: Se pueden configurar las herramientas de los comandos
 - Debug: Sirve para notificar posibles errores que puedan surgir en R.
 - Profile: Permite personalizar el espacio de trabajo de R
 - Tools: En este menú se pueden cargar paquetes y modificar los atajos del teclado para R Studio.
 - Help: Ayuda acerca de R Studio.
-
- R Commander: Según Santana (2018) R Commander es una interfaz de licencia abierta y pública creada por John Fox. Se puede iniciar R Commander desde R, introduciendo el comando *library(Rcmdr)* para cargar R Commander. Tal y como se muestra en la Imagen 3.2 se muestra la pantalla principal de la interfaz gráfica de R Commander.

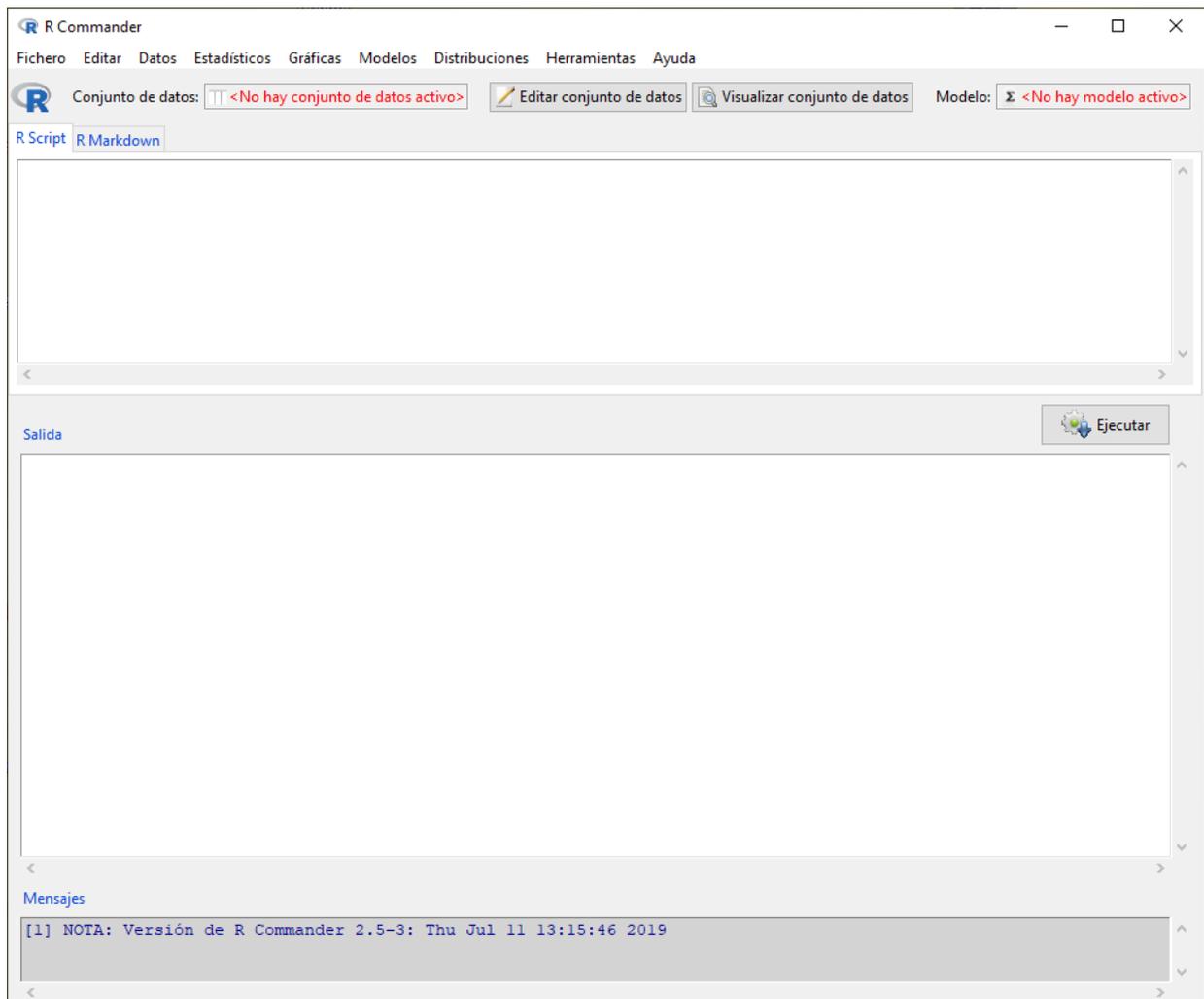


Imagen 3.2 R Commander

La pantalla principal de la interfaz de R Commander se divide en tres zonas, la superior sirve para mostrar el script generado, en la zona intermedia se muestran las líneas de comandos generados (salida) y en la zona inferior se muestran los posibles errores surgidos durante el proceso de compilación de la interfaz.

R Commander dispone de los siguientes menús:

- Fichero: Permite guardar el fichero actual.
- Editar: Aquí se pueden modificar las propiedades básicas del fichero (copiar, pegar, buscar, ...)
- Datos: Permite generar datos, importarlos cargarlos y eliminarlos.
- Estadísticos: Se pueden generar todos los estadísticos (medias, varianzas, ...).
- Gráficas: Se pueden realizar todos los gráficos que se deseen (dispersión, histograma,) e incluso modificar las propiedades de los gráficos.
- Modelos: En este menú se pueden ejecutar test de hipótesis.
- Distribuciones: Aquí se pueden realizar todas las distribuciones que se quieran.
- Herramientas: Aquí se puede cargar otros paquetes, modificar las opciones de R Studio e instalar software adicional.
- Ayuda: Ayuda acerca de R Commander proporcionada por R

4. TIPOS DE OBJETOS

Como se ha comentado anteriormente, R es un lenguaje de programación orientado a objetos. En esta sección se aborda la definición de un objeto en R y los tipos de objetos que existen en R (vectores, matrices, data frames y funciones).

Antes de definir los diferentes objetos que existen en R es necesario definir qué es un objeto y como se aplica en R. De acuerdo con Paradis (2003) un objeto son todas las variables, datos, funciones y resultados que se muestran en la pantalla y se guardan en la memoria activa del ordenador con un nombre específico; cabe destacar que un objeto se obtiene como resultado de una función. El usuario puede modificar estos objetos mediante operadores (aritméticos, lógicos y comparativos) y funciones (que a su vez son objetos). Los objetos, además de nombre y contenido, tienen atributos que especifican el tipo de datos representados por el objeto.

Los objetos tienen dos atributos intrínsecos:

- Tipo: definido mediante la función *mode()*. Este atributo hace referencia a la clase básica de los elementos en el objeto. Existen cuatro tipos principales: numérico, carácter, complejo, y lógico.
- Longitud: definido mediante la función *length()*. Este atributo hace referencia al número de elementos en el objeto.

Un objeto se puede crear con el operador asignar (>). Si el objeto ya existe, su valor anterior es borrado después de la asignación. Cabe destacar que no se puede escribir una expresión sin asignar su valor a un objeto.

En la Imagen 4.1 se muestra la función *ls()*, que permite mostrar los objetos que están en la memoria de la consola con sus respectivos nombres, en el ejemplo de la Imagen 4.1 son x,y e z .

```
> x <- 5; y <- 8; z <- 20
> ls()
[1] "datos" "x"      "y"      "z"
```

Imagen 4.1. Función ls

En la Imagen 4.2 se muestra la función *rm()* tras primeramente generar las variables x,y e z. Esta función permite eliminar objetos almacenados previamente en la memoria. Por ello en la Imagen 4.2 tras utilizar la función *rm()* se elimina la variable x de la memoria quedando como resultado la variable y e z.

```
> x <- 5; y <- 8; z <- 20
> rm(x)
> ls()
[1] "datos" "y"      "z"
```

Imagen 4.2. Función rm

A continuación, se describen algunas características y ejemplos de los principales objetos de R.

4.1 Vectores

Para comenzar hay que definir que un vector en R es la estructura de datos más sencilla en R. De acuerdo con Guillén (2016) el elemento clave de un vector en R es que siempre debe estar formado por el mismo tipo de datos ya sean numéricos o categóricos (mismos argumentos).

Para generar un vector es necesario añadir el comando `c`, tal y cómo se muestra en la Imagen 4.3. En el recuadro azul de la Imagen 4.3 se muestra cómo se genera un vector lógico llamado `y` cuyos argumentos son `T` (true) o `F` (false). En el recuadro amarillo de la Imagen 4.3 se muestra un ejemplo de cómo generar un vector numérico llamado `z`, cuyos elementos son 3, 6 y 8 y posteriormente se observa cómo se extraen los elementos del vector `z`, para ello es necesario introducir la posición del número en el vector que se quiere extraer, en el ejemplo se extrae la segunda y tercera posición del vector `z` que en este caso corresponde con los números 6 y 8.

```
> y <- c (T,T,T,F)
> y
[1] TRUE TRUE TRUE FALSE
>
> z <- c (3,6,8)
> z
[1] 3 6 8
> z [c(2,3)]
[1] 6 8
```

Imagen 4.3. Vector

4.2 Matrices

Según Marín (2017) una matriz es un vector bidimensional formado por filas y columnas. La matriz en R viene definida por la función `matrix ()`. Las matrices pueden estar formadas por variables numéricas, categóricas, ... Los argumentos más relevantes de esta función son: `data` (datos que forman la matriz introducidos en forma de vector), `ncol` (número de columnas que se van a introducir), `nrow` (número de filas a introducir en la matriz) y `byrow` (argumento lógico, sirve para decir si la matriz está formada por filas o por columnas).

En el recuadro azul de la Imagen 4.4 se muestra una matriz llamada *y*, formada a partir de un vector numérico y donde se ha establecido 3 filas mediante el argumento *nrow()*. En el recuadro amarillo de la Imagen 4.4 se muestra una matriz llamada *z* generada a partir de un vector numérico y donde se han establecido que el número de columnas son 3 con el argumento *ncol()*. Posteriormente gracias a la función *dim()* se muestran las dimensiones de la matriz, en este caso 2 filas y 3 columnas. En el recuadro negro de la Imagen 4.4 se genera una matriz llamada *r* formada por los vectores numéricos *q*, *w* y *e* gracias a la función *rbind()* que sirve para unir vectores y formar una matriz. La función *rbin()* sirve para unir vectores por filas y la función *cbin()* sirve para unir vectores por columnas. También se pueden realizar operaciones matriciales como se ve en el recuadro negro, ya que se multiplican la matriz *y* e *z*.

```

> y <- matrix (c(2,4,6,8,10,12), nrow = 3)
> y
      [,1] [,2]
[1,]    2    8
[2,]    4   10
[3,]    6   12

> z <- matrix (c(2,4,6,8,10,12), ncol=3)
> z
      [,1] [,2] [,3]
[1,]    2    6   10
[2,]    4    8   12
> dim (z)
[1] 2 3

> q <- c(1,3)
> w <- c(2,4)
> e <- c(5,7)
> r <- rbind(q,w,e)
> r
      [,1] [,2]
q      1    3
w      2    4
e      5    7
> y*r
      [,1] [,2]
q      2   24
w      8   40
e     30   84

```

Imagen 4.4. Matriz

4.3 Data Frames

Según Marin (2017) un data frame o marco de datos son un tipo de estructuras de datos similares a las matrices, ya que también se componen de columnas y de filas. La diferencia radica en que los datos de las matrices deben ser del mismo tipo, mientras que, en los data frames, se pueden introducir datos de diferente tipo (ya sean numéricos, lógicos o categóricos); las dos únicas limitaciones en los data frames son que todos los elementos de una columna deben ser del mismo tipo y que todas las columnas de un data frame deben ser de la misma longitud. Cabe destacar que un data frame está compuesto por vectores que forman las columnas.

Para generar un data frame primeramente es necesario establecer las variables que la forman (pueden ser categóricas o numéricas), en el ejemplo del recuadro azul de la Imagen 4.5 se genera el data frame llamado *DATA* formado por las variables *sexo* (variable categórica -> hombre o mujer), *conductor* (variable categórica -> Si conductor o No conductor) y *edad* (variable numérica). Para generar el data frame se introduce la función *data.frame()* con todas las variables generadas previamente (*sexo*, *conductor* y *edad*).

Una vez generado el data frame, en el recuadro amarillo de la Imagen 4.5, gracias a la función *str()*, se observan los atributos de las variables y del data frame. En este caso hay 4 observaciones y 3 variables (una variable numérica, dos factores). En este recuadro amarillo de la Imagen 4.5 con la función *dimnames()* del data frame se muestran los valores que toman las variables en las observaciones.

Para extraer los datos de una determinada fila o columna de un data frame se debe de introducir el nombre del data frame y seguidamente entre corchetes las filas o columnas que se desean extraer añadiendo una coma. En el ejemplo del recuadro negro de la Imagen 4.5 se extrae la tercera fila y la primera columna del data frame llamado *DATA*. También se puede extraer una determinada variable del data frame introduciendo el símbolo del dólar y el nombre de la variable. En el ejemplo del recuadro negro de la imagen 4.5, se extraen los valores de la variable *conductor* en el data frame *DATA*.

```

> sexo <- c ("M", "M", "H", "M")
> conductor <- c ("NO","SI", "SI", "NO")
> edad <- c (20,25,30,35)
> DATA <- data.frame (sexo, conductor, edad)
> DATA
  sexo conductor edad
1    M         NO   20
2    M         SI   25
3    H         SI   30
4    M         NO   35

> str (DATA)
'data.frame':  4 obs. of  3 variables:
 $ sexo      : Factor w/ 2 levels "H","M": 2 2 1 2
 $ conductor: Factor w/ 2 levels "NO","SI": 1 2 2 1
 $ edad      : num  20 25 30 35
> dimnames (DATA)
[[1]]
[1] "1" "2" "3" "4"

[[2]]
[1] "sexo"      "conductor" "edad"

> DATA [3,]
  sexo conductor edad
3    H         SI   30
> DATA [,1]
[1] M M H M
Levels: H M
> DATA$conductor
[1] NO SI SI NO
Levels: NO SI

```

Imagen 4.5. Data Frame

4.4 Funciones

Las funciones en R son un tipo de objeto mediante el cual se pueden automatizar algunas tareas (como por ejemplo una suma) que pueden ser simples o complejas según así lo decida el usuario.

Las funciones constan de tres elementos:

- **Cuerpo (*body*):** “Contiene las operaciones que se ejecutan sobre cada uno de los argumentos detallados. Las operaciones se ejecutan cada vez que llamamos a la función” (Ferrero et al, 2017).
- **Argumentos (*Formals*):** “Los argumentos o valores de entrada son los elementos que necesitas para que se ejecute la función” (Ferrero et al, 2017). Los argumentos se separan por una coma dentro de los paréntesis que definen la función.
- **Resultado (*Valor de salida*):** es el valor devuelto por la función generada según las operaciones realizadas en la misma función. Puede ser cualquier tipo de dato.

En la Imagen 4.6 cómo se introducen estos 3 elementos que conforman la función.

```
mifuncion <- function(argumento1, argumento2, ...) {  
  cuerpo  
  resultado  
}
```

Imagen 4.6. Ejemplo de función

Por ejemplo, si se quiere generar la función resta, primeramente habría que denominar a la función como resta (o el nombre elegido por el usuario), después introducir el comando *function()* seguido por los elementos que se quieren restar (*x,y*) (argumentos); más adelante se describe la operación que se asocia a la función (resta de los elementos “*x*” e “*y*”)(cuerpo) y finalmente se ejecuta la operación asociada a la función (resultado) (*x-y*). En la Imagen 4.7 se puede apreciar lo descrito anteriormente.

```
> resta <- function(x,y)  
+ # resta de los elementos "x" e "y"  
+ x-y  
> resta (x=7, y=3)  
[1] 4
```

Imagen 4.7. Función

5. ANÁLISIS ESTADÍSTICOS BÁSICOS Y GRÁFICAS

En este apartado se analiza cómo realizar los estadísticos básicos en R (media, mediana, desviación típica, ...) y como representar los principales gráficos en R (diagrama de caja, histograma y diagrama de dispersión).

R permite obtener de forma individual los principales estadísticos descriptivos. Se muestran a continuación las funciones de R que sirven para obtener los principales estadísticos descriptivos.

- Media: La función `mean()` sirve para calcular la media aritmética. El argumento de esta función es *un vector de datos*.
- Mediana: la función `median()` calcula en R la mediana de un vector numérico. Función formada por un *vector de datos*.
- Varianza: La función `var()` calcula la varianza de un conjunto de números en R. El argumento de la función `var()` es un *vector de datos*.
- Desviación Típica: La función `sd()` ayuda a calcular la desviación típica en R. Esta función tiene como argumento un *vector de datos*.
- Recorrido intercuartílico: La función `IQR()` sirve para calcular el recorrido intercuartílico. El argumento de la función es un *vector de datos*.
- Summary: Gracias al comando `summary()` se pueden obtener varios estadísticos básicos tales como la media, mediana, los cuantiles y el valor máximo y el mínimo.

Todos estos estadísticos se muestran en la Imagen 5.1

```
> datos <- c(1,2,5,6,7,7,7,8,1,9,10,5,4,8,2,9,3)
> mean (datos)
[1] 5.529412
> median (datos)
[1] 6
> var (datos)
[1] 8.639706
> sd (datos)
[1] 2.939338
> IQR (datos)
[1] 5
> summary (datos)
  Min. 1st Qu.  Median    Mean 3rd Qu.   Max.
1.000  3.000   6.000   5.529  8.000  10.000
```

Imagen 5.1. Estadísticos básicos

R es un programa con gran interés en el apartado gráfico de los datos ya que permite realizar y modificar multitud de diferentes gráficos con una gran calidad. En los ejemplos mostrados se utilizará R Studio ya que esta interfaz gráfica permite mostrar a la vez la consola con los comandos y el gráfico generado.

En todos los ejemplos de gráficos se va a utilizar la librería *iris* que es una base de datos con una muestra de 150 flores en las que se encuentran las variables: especie de la flor [*species*] (variable categórica -> *setosa*, *versicolor* y *virginica*), longitud del pétalo [*petal.length*] (valor numérico), anchura del pétalo [*petal.width*] (valor numérico), longitud del sépalo [*sepal.length*] (valor numérico) y anchura del sépalo [*sepal.width*] (valor numérico). [Si se quiere información acerca de los ejemplos utilizados para realizar los gráficos están en la siguiente web: <https://rpubs.com/fdsalgado/graficas>].

La función más simple a la hora de generar un gráfico es *plot()*. Esta función tiene como argumentos más relevantes: *data* (data frame sobre la que se realiza la gráfica), *cex* (argumento que modifica el tamaño de los números), y *main* (argumento que modifica el nombre del gráfico).

Como se puede observar en el ejemplo de la Imagen 5.2 se carga en memoria la base de datos *iris* gracias al comando *data()* y mediante la función *plot()* se genera diagramas de dispersión de todos los pares de variables (*sepal.length*, *sepal.width*, *petal.length*, *petal.width*, *species*).

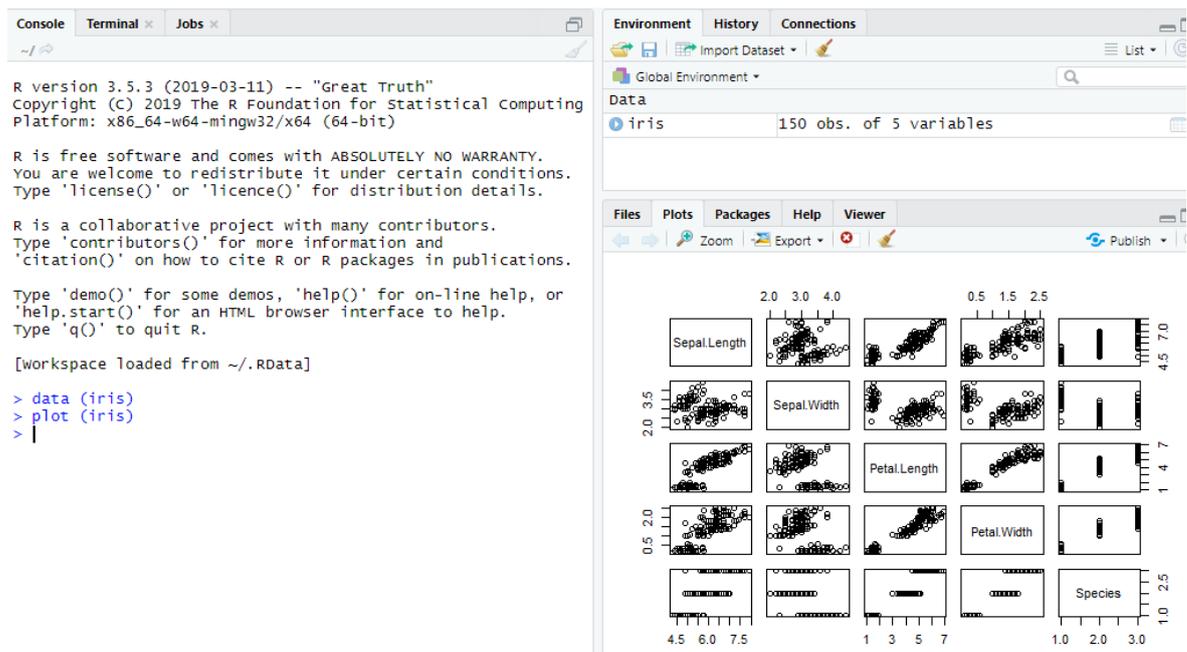


Imagen 5.2. Función plot

De forma complementaria se puede decir que existen una librería que es clave en la generación de gráficos, que es el paquete *ggplot2* ya que permite generar múltiples gráficos en los que intervienen una o múltiples variables y se puede observar la relación entre ellas.

A continuación, se van a analizar lo gráficos más relevantes con relación a el análisis estadístico a abordar en el presente trabajo.

- Diagrama de caja: la función *boxplot()* utiliza para crear un diagrama de caja. Esta función tiene multitud de argumentos, entre los que destacan: *data* (incluye un data frame o lista desde la cual se obtienen los datos para la gráfica), *main* (sirve para nombrar a la función), *xlab* (ese argumento sirve para nombrar el eje x de la gráfica), *ylab* (nombra el eje y del diagrama de cajas), *col* (sirve para dar color a las cajas del gráfico) y *subset* (hace referencia a un vector opcional a incluir si se quieren usar un subconjunto de observaciones en la gráfica).

A continuación, en la Imagen 5.3 se adjunta un ejemplo de un diagrama de caja con la función *boxplot()* y la base de datos *iris*.

Inicialmente se carga la base de datos *iris* con el comando *data()*. Más adelante se emplea la función *boxplot()* con las cuatro variables numéricas de la base de datos anteriormente mencionada (*sepal.width*, *petal.width*, *sepal.length*, *petal.length*) menos la variable categórica (*species*); con el argumento *main* se llama a la gráfica diagramas de caja, con el argumento *xlab* se nombra al eje x como dimensiones y para finalizar con el argumento *col* se colorea de amarillo a las cajas del gráfico.

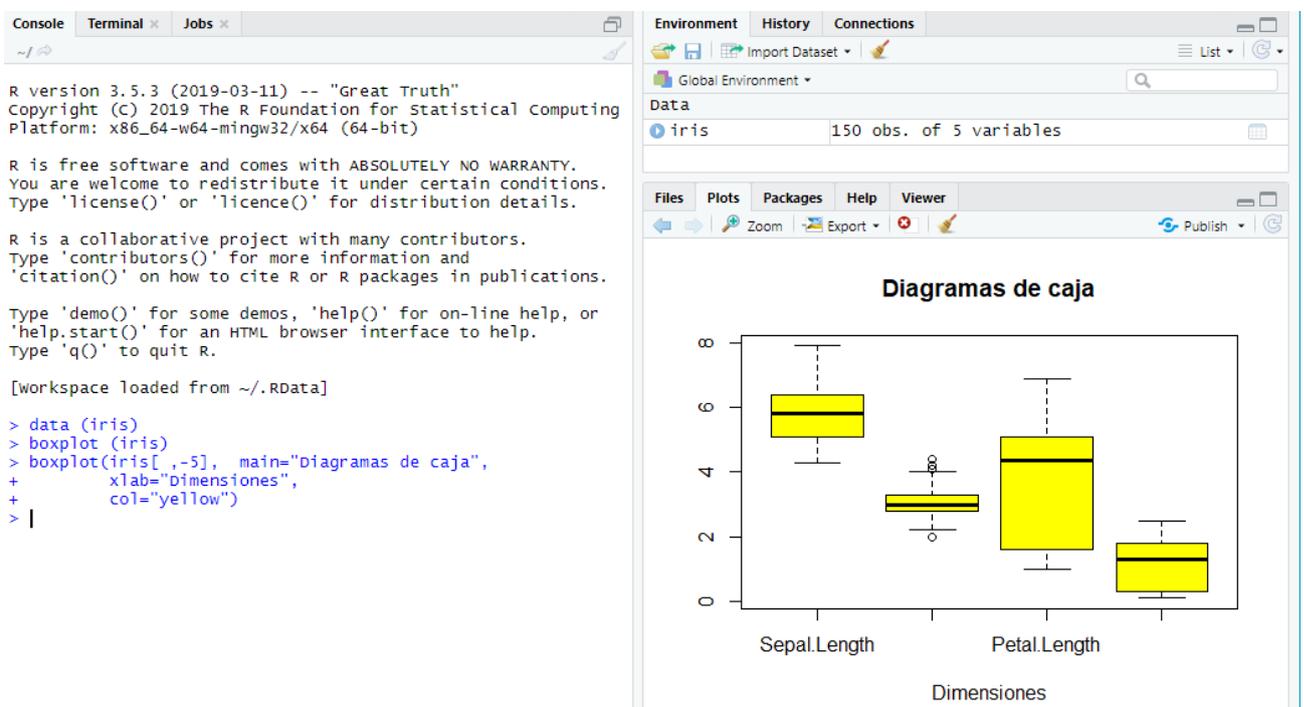


Imagen 5.3. Diagrama de caja

- **Histograma:** para hacer histograma en R es necesario introducir la función *hist()*. Esta función cuenta con los siguientes argumentos relevantes: *x* (vector numérico de la variable de la que se desea hacer un histograma), *main* (nombre del gráfico), *ylab* (nombre del eje y del gráfico), *xlab* (nombre del eje x del gráfico), *col* (color de las barras del histograma) y *freq* (argumento lógico, *true* o *false*, si el argumento es *true* el histograma es una representación de las frecuencias, si el argumento es *false* el histograma es un gráfico de densidad de probabilidad).

En el ejemplo que se muestra en la Imagen 5.4. se realiza un histograma de la variable numérica *Petal.length* de la base de datos *iris*. Después de introducir la base de datos *iris* con el comando *data* se introduce la función *hist()* con el argumento *main* para llamar a la gráfica “Longitud del pétalo”, el argumento *ylab* para llamar al eje y “Frecuencia” y el argumento *col* para que las barras del histograma sean azules (blue)

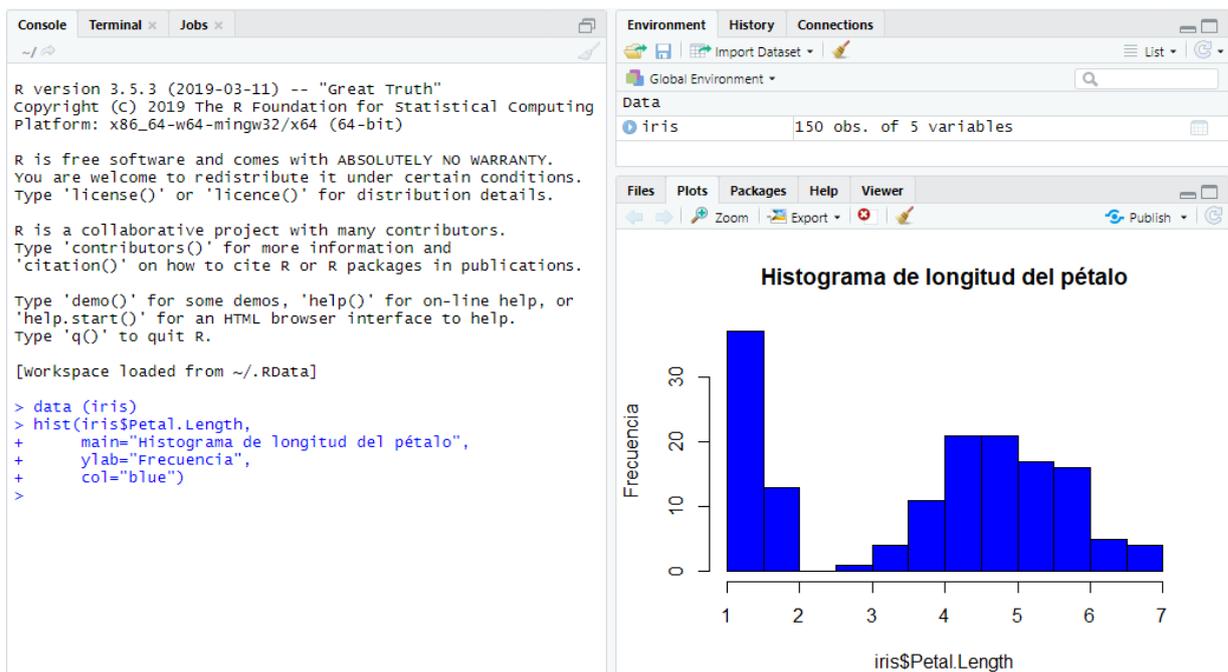


Imagen 5.4. Histograma

- **Diagrama de dispersión:** para generar un diagrama de dispersión en R se utiliza la función *qplot()* contenido en la librería *ggplot2* que se ha mencionado anteriormente. Esta función tiene como argumentos más relevantes: *data* (base de datos en forma de data frame que incluye las variables que se representan en el diagrama de dispersión), *x* (variables empleadas en el diagrama de dispersión), *shape* (argumento que controla la forma de los puntos para los grupos en el diagrama de dispersión), *xlim* (valor límite de x en la gráfica) e *ylim* (valor límite de y en el diagrama de dispersión).

En la Imagen 5.5 se muestra un diagrama de dispersión de las variables *petal.length* y *petal.width* con la base de datos *iris*. Para realizar el gráfico ha sido necesario introducir la base de datos *iris* con el comando *data*, más adelante cargar la librería *ggplot2* con el comando *library()*; y para finalizar ha sido necesario introducir la función *qplot()* cuyos argumentos han sido las dos variables con las que se quiere realizar el gráfico, el argumento *data* donde se incluye la base de datos *iris* y el argumento *shape* con *species* para clasificar las observaciones del diagrama de dispersión según la especie de la flor (*setosa*, *versicolor*, *virginica*).

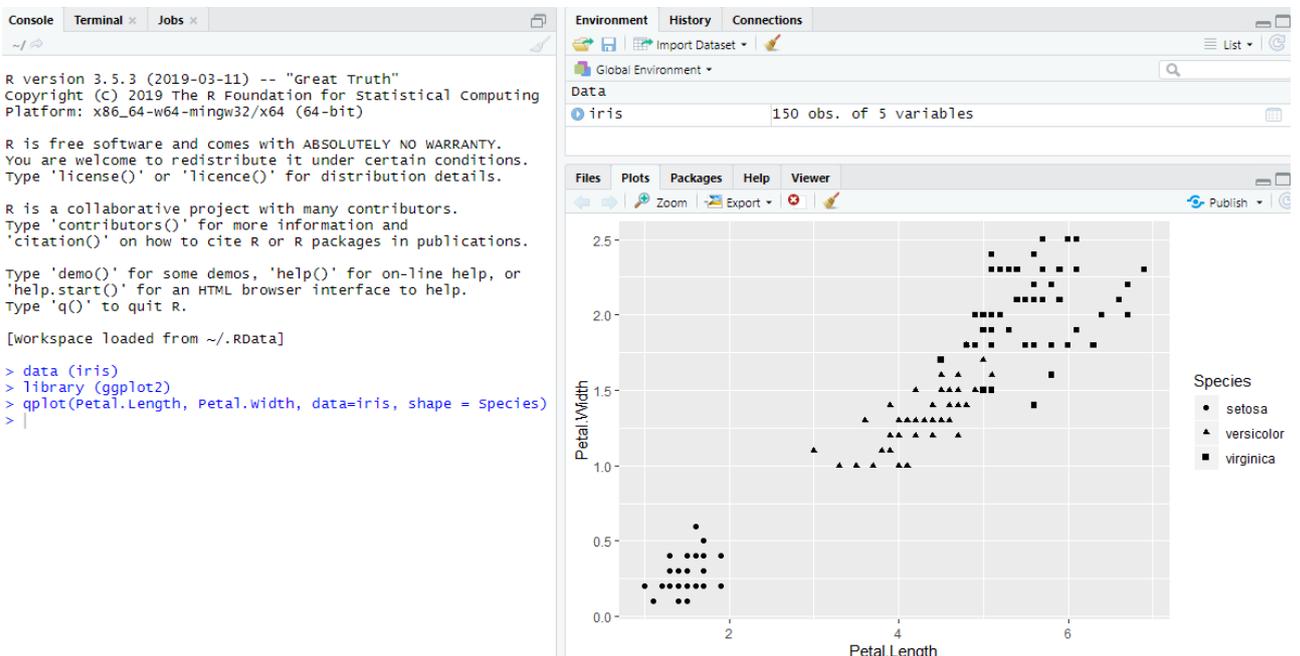


Imagen 5.5. Diagrama de dispersión

6. TECNICAS MULTIVARIANTES DE ANÁLISIS DE DATOS

Este apartado se va a dedicar a presentar y realizar ejemplo del análisis de conglomerados y el análisis de componentes principales. Estos análisis se llevarán a cabo en el entorno de la interfaz gráfica de R Studio debido a la facilidad para observar los comandos realizados en la consola y los gráficos generados de los análisis estadísticos. [Para obtener más información de los análisis estadísticos aplicados se pueden consultar Uriel y Aldás, (2005) Peña (2002) y James et al (2013). Si se quiere información acerca de los ejemplos utilizados para realizar los análisis estadísticos se encuentran en esta web: <https://www.diegocalvo.es/tutorial-de-r-ejemplos-simples/>].

En estos análisis estadísticos se va a utilizar la base de datos *iris*, explicada previamente en el apartado de análisis estadísticos básicos y gráficas. Cabe destacar que en estos análisis se van a utilizar exclusivamente las 4 variables numéricas (*Petal.Length*, *Petal.Width*, *Sepal.Length*, *Sepal.Width*), ya que en estos análisis no se utilizan variables de tipo categórico (*Species*).

6.1 Análisis de Conglomerados (Clúster):

El análisis clúster es una técnica multivariante de datos para clasificar a todos los elementos en grupos homogéneos que deben de cumplir dos condiciones:

- Cada grupo (también llamado clúster o conglomerado) debe tener todos sus elementos lo más parecidos entre sí.
- Los grupos deben ser lo más distintos posible los unos de los otros con relación a las variables a analizar.

Es importante resaltar que, en el análisis de conglomerados, los grupos son desconocidos a priori y es necesario determinarlos según los valores de las variables.

Hay dos métodos a la hora de agrupar a los individuos en un grupo:

- Los métodos jerárquicos: Los grupos se forman de manera secuencial. Hay tantas etapas como observaciones en la muestra menos uno. Estos métodos se dividen en:
 - Métodos divisivos: Parten de un único clúster con todos los datos al inicio y se van haciendo grupos secuencialmente hasta que al final hay tantos clústeres como datos.
 - Métodos aglomerativos: Parten de tantos clústeres como datos hay en la muestra y, secuencialmente, según un criterio se van agrupando hasta que solo quede un clúster con todos los datos.

- Métodos no jerárquicos (procedimiento de agrupación de k-medias): Se determina un número de clúster a priori y se definen unas semillas o puntos de referencia de cada variable. A partir de ahí se agrupa cada caso en el clúster con el que presente la menor distancia. Para realizar este análisis primero hay que seleccionar una serie de centros para los grupos iniciales (k) para posteriormente dividir a todas las observaciones entre todos los grupos según el criterio establecido previamente al análisis. Posteriormente hay que calcular la similitud o diferencia (mediante la distancia) de cada elemento al centro de cada grupo para observar si los grupos han sido creados satisfactoriamente.

Para ver la similitud o la disparidad entre los casos se utilizan medidas de distancias como la distancia euclídea o la distancia de Manhattan o de Minkowski.

Dentro de los métodos jerárquicos aglomerativos, hay multitud de métodos para medir la distancia entre clústeres (es decir lo diferentes que son los clústeres entre sí), pero el más importante es el procedimiento de Ward. Este procedimiento está basado en el cálculo de centroides (el centroide es un vector que incluye la media de cada uno de los grupos), posteriormente se calcula la distancia (diferencia) entre cada elemento y el centroide de su grupo. Más adelante se forman los clústeres teniendo en cuenta que los elementos tengan la menor distancia al centroide (es decir que haya la menor pérdida de información posible).

A la hora de representar los conglomerados aglomerativos y poder interpretar los datos de una manera mucho más visual se utiliza el dendograma que ilustra las fusiones de los grupos hechas en cada etapa.

La forma de realizar un clúster jerárquico en R es mediante la función la función *hclust()* que sirve para agrupar las observaciones en grupos. También son relevantes las funciones *dist()* que genera la matriz de distancias en R y *cutree()* que sirve asignar a cada observación el grupo de pertenencia, siendo necesario determinar el número de grupos.

La función *dist()* tiene como argumentos más importantes: el data frame que contiene los datos y *method* (este argumento hace referencia a la medida de distancia aplicada para medir lo diferentes que son los individuos en los grupos formados. Por defecto se calcula la distancia euclídea (“*euclidean*”), pero se puede modificar en la distancia de Manhattan, Minkowski...).

A la función *hclust()* hay que suministrarle un objeto creado a partir de la matriz de distancias . *Hclust()* tiene un argumento denominado *method* que hace referencia al método de aglomeración empleado en el análisis clúster. Por defecto, se aplica el método de Ward (“*ward.D*”), pero se pueden aplicar otros métodos de aglomeración, como el método del centroide (“*centroid*”).

A la función *cutree()* hay que pasarle un objeto creado con la función *hclust()* y un valor numérico que indica el numero deseado de grupos que se quieren generar con el análisis (*k*).

En la Imagen 6.1 se muestra un ejemplo en R de un análisis clúster jerárquico con la base de datos *iris*. En esta Imagen 6.1 se muestra como inicialmente tras cargar la base de datos con el comando `data()`, se asigna la base de datos (data frame) a un objeto llamado `datos`. Posteriormente, en `datos` se excluye la variable categórica, por eso se incluye un `-5` entre corchetes. Más adelante se tipifica el objeto `datos` gracias a la función `scale`,². Para proseguir se genera el objeto `distancias` que incluye la matriz de distancias con el comando `dist()` cuyos argumentos incluyen el objeto `datos` y como `method` la distancia euclídea ("*euclidean*"). En esta matriz de distancias se observa como el número de clúster adecuado son 3 debido a que hay un salto grande en la matriz de distancia al pasar de 3 a 2 clústeres.

```

R version 3.5.3 (2019-03-11) -- "Great Truth"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[workspace loaded from ~/.RData]

> data(iris)
> datos <- iris
> datos[-5] <- scale(datos[-5])
> distancias <- dist(datos[-5], method = "euclidean")
> distancias
      1      2      3      4      5
2  1.1722914
3  0.8427840 0.5216255
4  1.0999999 0.4325508 0.2829432
5  0.2592702 1.3818560 0.9882608 1.2459861
6  1.0349769 2.1739229 1.8477070 2.0937597 0.8971079
7  0.6591230 0.9953200 0.4955334 0.7029623 0.6790442

```

Imagen 6.1. Clúster jerárquico 1

² En el análisis clúster es conveniente trabajar con las variables tipificadas, para evitar los problemas de las unidades de medida diferentes.

Posteriormente, en la Imagen 6.2 se observa cómo se genera un objeto denominado *agrupamiento* a partir de la función *hclust()* utilizando como argumentos *distancias* que contiene la matriz de distancias generada previamente y *method*, “*Ward.D*”. Posteriormente, se genera un objeto llamado *grupos* que se obtiene a partir de *cutree()*, que contiene los argumentos *agrupamiento*, generada previamente, y un número *k* de grupos, que en este caso son 3.

Más adelante, gracias a la función *plot()*, se realiza el dendograma utilizando como argumento *agrupamiento* (un objeto creado con la función *hclust()*). También se puede fijar el tamaño de los números del gráfico (*cex = 0,7*), y el nombre de la gráfica (*main = Cluster sobre tipos de flor*). Para finalizar con la función *rect.hclust()* se generan recuadros rojos señalando de una manera más visual los clústeres en el dendograma. Esta función *rect.hclust()* tiene tres argumentos: *el objeto generado con la función hclust()*, *k* (número de grupos definidos) y *border* (establece el color de los recuadros). En este ejemplo se emplea la función *rect.hclust()* con los argumentos: *agrupamiento*, 3 clúster a generar (*k*), y los recuadros (*border*) son rojos.

Con el fin de ilustrar como se realiza en R este procedimiento, se va a utilizar de nuevo la base de datos *iris*, tal y como se muestra en la Imagen 6.3. Para comenzar es necesario emplear la función *set.seed()* que sirve para establecer la semilla para generar números aleatorios; esto es importante para que los centroides de los grupos iniciales sean elegidos al azar, vuelvan a ser los mismos y poder reproducir los mismos resultados. Más adelante se carga la base de datos *iris* con el comando *data()* y se genera un objeto llamado *datos* que incluye todas las variables numéricas de la base de datos. Posteriormente se genera un objeto llamado *grupos* a partir de *kmeans()* cuyos argumentos son: *datos* (objeto compuesto por las 4 variables numéricas de la base de datos *iris*) y *centers*, por lo que se decide establecer 3 grupos de acuerdo a lo visto anteriormente en el análisis de clúster jerárquico.

En la salida del análisis clúster no jerárquico se observa cómo se han generado 3 grupos de 150 observaciones totales, el primero está formado por 50 observaciones, el segundo por 62 observaciones y el tercero por 38 observaciones. En el apartado de *Cluster means* se observa cuáles son los centroides en cada uno de los 3 grupos finales. En la sección de *Clustering vector* se muestra a qué grupo va cada observación de la muestra. En el apartado de *Within cluster sum of squares by cluster* se indica la suma de cuadrados dentro de cada grupo. Para finalizar en el apartado de *Available components* se muestran los diferentes elementos del objeto generado con k-means, en este caso ("*cluster*", "*centers*", *etc*).

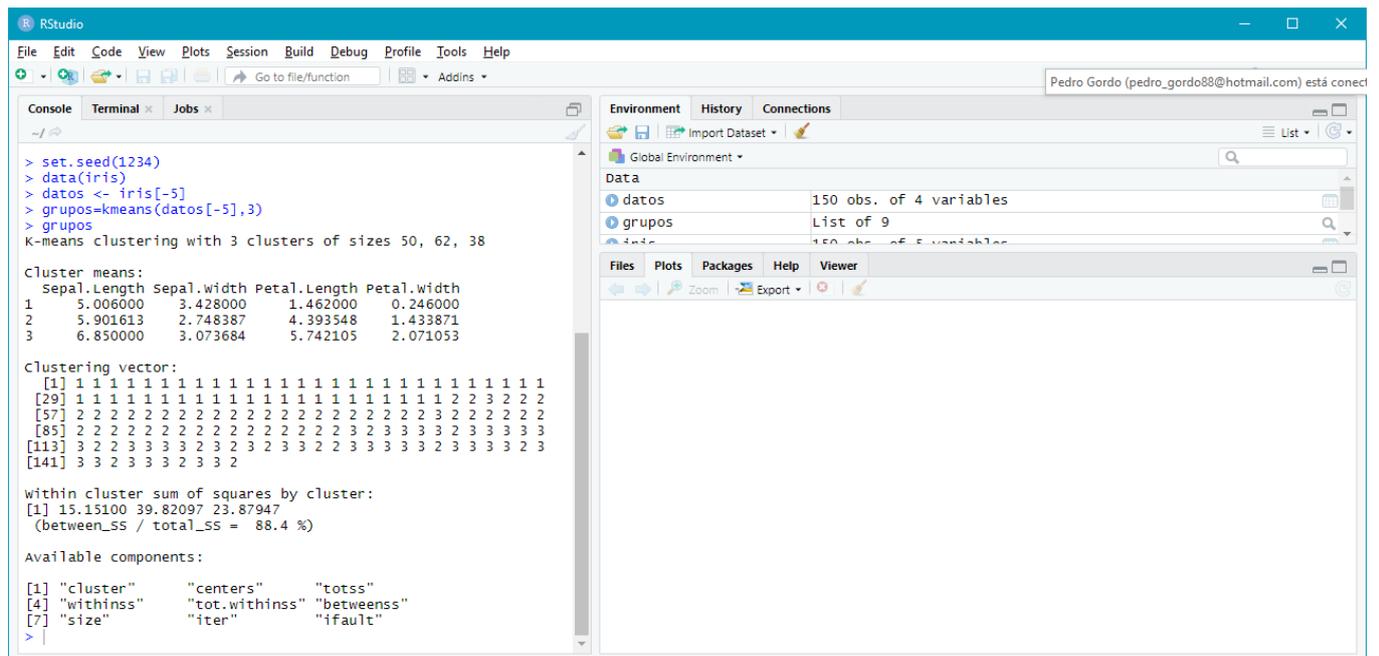


Imagen 6.3. Clúster no jerárquico

6.2 Análisis de Componentes Principales

El análisis de componentes principales tiene como objetivo reducir las dimensiones de un conjunto de variables correlacionadas con el objetivo de perder la menor información de cada variable. La información de cada variable se mide a través de la varianza. Con este análisis se busca generar combinaciones lineales de las variables para que estas nuevas variables tengan información original y estén incorrelacionadas. La forma de deducir las componentes es tal que la primera componente es la que tiene más variabilidad (más información aporta) hasta la última componente que es la que apenas tiene variabilidad (no proporciona casi información).

Para realizar un análisis de componentes en R se aplica la función *prcomp()* cuyos argumentos más importantes son: *data* (si se ha introducido un data frame que contiene las variables a aplicar en el análisis estadístico), *center* (argumento en forma de factor (*True / False*) que proporciona una transformación de las variables; si es *True* indica que las variables se han transformado y tienen media 0) y *scale* (argumento tipo factor (*True / False*) que transforma a las variables de manera que si el argumento es *True* la varianza es igual a 1).

Se va a realizar un análisis de componentes principales con las variables numéricas de la base de datos *iris* (*sepal.length*, *petal.length*, *sepal.width*, *petal.width*) mencionada previamente. El objetivo de este análisis de componentes principales es reducir la dimensión del problema, generando unas nuevas variables llamadas componentes principales que estén incorrelacionadas entre sí y que proporcionan tanta información como las variables originales. Se pueden generar tantas componentes como variables, sin embargo, sólo se suelen considerar las primeras componentes puesto que son las que más información llevan incorporada.

En primer lugar, en el recuadro azul de la Imagen 6.3 se observa cómo se cargan los datos de la librería *iris* con el comando *data* y como posteriormente se genera una función *datos* donde se incluye las cuatro variables de la base de datos *iris* menos la variable *species*.

Más adelante se analiza si con estas cuatro variables es pertinente aplicar el análisis de componentes principales, estudiando la matriz de correlaciones. Para ello se genera un objeto llamado *correlación* que recoge la matriz de correlaciones a partir de la función *cor()*. Se observa como las variables *Sepal.length* y *Petal.length* están relacionadas ya que tienen una correlación elevada (0,87); lo mismo ocurre con *Petal.width* y *Sepal.length* (0,81), *Petal.length* y *Petal.width* (0,96). Con todo esto se deduce que están relacionadas las variables con lo que es adecuado aplicar el análisis de componentes principales.

Posteriormente en el recuadro negro de la Imagen 6.3 se genera un objeto llamado *modelo* a partir de la función *prcomp()* con los siguientes los argumentos: *datos* (data frame generado por las 4 variables), *scale = True* (variables con varianza igual a 1) y *center = True* (variables con media igual a 0), por lo que se trabaja con datos tipificados.

El objeto *modelo* contiene las desviaciones estándar de cada una de las 4 componentes generadas en el análisis de componentes principales (1,70,0,95, 0,38,0,14); cabe destacar que este análisis de componentes principales ha generado una lista de 5 variables. También contiene la matriz de rotación del análisis de componentes principales formada por las cargas factoriales. En esta matriz de rotación se observa que variable tiene más peso (carga factorial) en cada componente. En este caso se observa cómo se han generado 4 componentes (*PC1*, *PC2*, *PC3*, *PC4*) y que en la primera componente la variable *Petal.length* (0,58) tiene un gran peso, seguida de *Petal.width* (0,56) y *Sepal.Length* (0,52). En la segunda componente destaca la carga factorial de *Sepal.Width* (-0,92) sobre las demás, en PC3 destaca la variable *Sepal.Length* (0,71) y en la última componente destaca la carga factorial de *Petal.Length* (-0,80).

Finalmente, en el recuadro amarillo de imagen 6.4 se ven los resultados de *summary(modelo)*, que permiten ver qué porcentaje de variabilidad tiene cada componente sobre la variabilidad total, por lo que se ve que porcentaje de información que aporta cada componente al análisis estadístico. En este caso la componente PC1 explica el 72% de la información total de las 4 variables, PC2 el 22%, PC3 el 3% y PC4 el 0,05%. Con el comando *plot(modelo)* se puede observar gráficamente estos porcentajes.

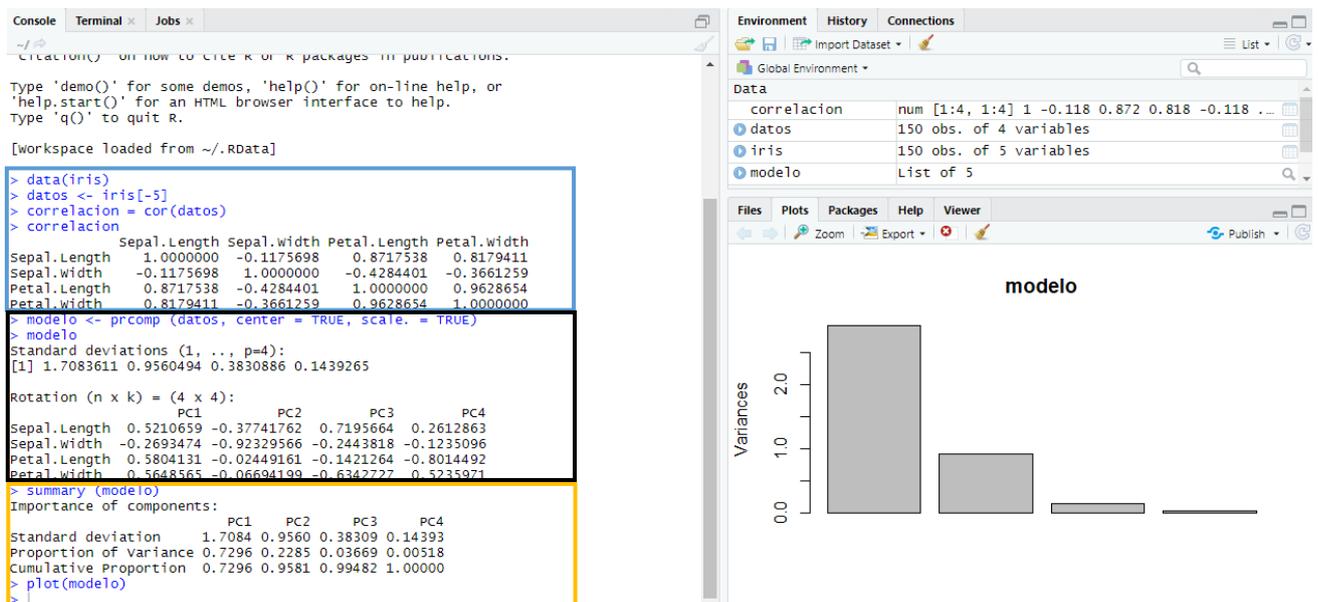


Imagen 6.4. Análisis de componentes principales

7. CONCLUSIONES

Como conclusiones a los objetivos tratados en el apartado de introducción del trabajo se puede decir que:

- R es un lenguaje de programación que pese a no ser muy amigable e intuitivo es muy flexible a la hora de realizar análisis, además de ser un programa fácil de instalar en los distintos sistemas operativos y con licencia abierta por lo que cualquier persona puede acceder a él de forma gratuita.
- R es un lenguaje de programación muy versátil en el que se pueden realizar y modificar multitud de gráficas y análisis estadísticos ya sean simples o complejos, pero es necesario conocer los comandos para generar lo deseado.
- R Studio destaca como interfaz gráfica frente a R Commander y R para realizar análisis debido a su facilidad para mostrar a la vez la consola de R, los gráficos generados y las bases de datos o listas introducidas para realizar el análisis estadístico.

Para resumir todo se puede decir que R es una alternativa real y más que factible para cualquier persona que se quiera adentrar en el análisis de datos y tenga una experiencia previa en el tratamiento de datos en otros programas estadísticos. Para trabajar en R inicialmente es necesaria una breve etapa de aprendizaje de los elementos básicos de R y de los principios básicos de programación.

8. REFERENCIAS BIBLIOGRÁFICAS

Abelson, H; Jay, G. y Sussman, J. (1996): "Structure and Interpretation of Computer Programs". Disponible en <https://web.mit.edu/alexmv/6.037/sicp.pdf> [consulta: 20/05/20].

Bonet, E. (2019): "Lenguaje C". Disponible en <https://informatica.uv.es/estguia/ATD/apuntes/laboratorio/Lenguaje-C.pdf> [consulta 19/05/2019].

Calvo, D. (2016): "Tutorial de R (ejemplos simples)". Disponible en <https://www.diegocalvo.es/tutorial-de-r-ejemplos-simples/> [consulta 20/09/2019].

Ferrero, R. y López, J. (2018): "Crea tu propia función en R paso a paso". Disponible en <https://www.maximaformacion.es/blog-dat/crea-tu-propia-funcion-en-r-paso-a-paso/> [consulta 20/08/2020].

Fox, J. (2019): "Plugin". Disponible en <https://www.rdocumentation.org/packages/Rcmdr/versions/2.6-2/topics/Plugins> [consulta 19/05/2019].

González, M. (2016): "Capítulo I: Introducción a R y R Studio". Disponible en https://rstudio-pubs-static.s3.amazonaws.com/195980_3f4cd84bc3ca434daeec55c6c211d13e.html [consulta 20/05/2019].

Guillén, D. (2016): "Computación y programación en R". Disponible en <https://www.uv.es/conesa/CursoR/material/handout-sesion2.pdf> [consulta 30/06/2019].

Hornik, K. (2018): "R FAQ". Disponible en <https://cran.r-project.org/doc/FAQ/R-FAQ.pdf> [consulta: 15/07/2020].

James, G; Witten,D; Hastie,T. y Tibshirani, R. (2013): “An Introduction to Statistical Learning with Applications in R”, Springer [consulta: 06/09/2020].

Jiménez, J. (2019): “Introducción a R y R Studio”. Disponible en <https://ridda2.utp.ac.pa/bitstream/handle/123456789/9428/manual-introduccion-R.pdf?sequence=1&isAllowed=y> [consulta: 16/07/2019].

Marín, J. (2017): “Estructuras de datos en R”. Disponible en http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/ManualR/intro_e_structurasdedatos.html [consulta 05/06/2019].

Muñoz, A. (2015): “Lectura y escritura de datos en R”. Disponible en <http://ocw.uc3m.es/estadistica/aprendizaje-del-software-estadistico-r-un-entorno-para-simulacion-y-computacion-estadistica/lectura-y-escritura-de-datos-en-r> [consulta 17/08/2020].

Paradis, E. (2003): “R para principiantes”. Disponible en https://cran.r-project.org/doc/contrib/rdebuts_es.pdf [consulta: 16/05/2019].

Peña, D. (2002): “Análisis Multivariantes”, Mc Graw Hill [consulta: 15/07/2020].

Salgado, F. (2018): “Gráficas en R”. Disponible en <https://rpubs.com/fdsalgado/graficas> [consulta: 14/08/2020].

Santana, A. (2018): “Introducción al uso de R Commander”. Disponible en <https://www.uv.es/conesa/CursoR/material/Manual-R-commander.pdf> [consulta: 24/07/2019].

Uriel, E y Aldás, (2005): “Análisis Multivariante Aplicado”, Thomson [consulta: 22/07/2020].