Universidad de Valladolid

Facultad de Ciencias Económicas y Empresariales

Trabajo de Fin de Grado

Grado en Administración y Dirección de Empresas

Análisis Estadístico Básico con R

Presentado por:

Jorge Caviedes Fidalgo

Tutelado por:

María Mercedes Prieto Alaiz

Valladolid, 9 de Julio de 2018

RESUMEN

El presente Trabajo de Fin de Grado es una aproximación al análisis estadístico básico realizado con el programa R. Su objetivo principal es realizar una pequeña introducción al análisis estadístico con este software, demostrando su funcionamiento. Partiendo de una introducción a su origen y ventajas e inconvenientes de su uso, hasta dedicar los diferentes capítulos que lo conforman a la forma de instalarlo y trabajar con él, el análisis de cada una de sus estructuras, el estudio de sus funciones, el análisis descriptivo básico con variables de análisis de datos, las formas y funciones con las que presentar de forma gráfica los datos, las funciones y distribuciones de probabilidad, el uso de inferencia estadística para estudiar poblaciones y el estudio de la relación entre variables con análisis de regresión. Todo ello acompañado de imágenes que ejemplifican algunos aspectos básicos de su funcionamiento.

Palabras clave: análisis estadístico básico, R, software.

Códigos de clasificación JEL: C12, C13, C88

ABSTRACT

The present End-of-Degree Project is an approximation to the basic statistical analysis made with the R program. It's main objetive is to make a small introduction to statistical analysis with this software, demonstrating its operation. Starting from an introduction to its origin and advantages and disadvantages of its use, to dedicate the different chapters that form it to the way to install it and work with it, the analysis of each of its structures, the study of its functions, the basic descriptive analysis with data analysis variables, the forms and functions with which to graphically present data, functions and probability distributions, the use of statistical inference to study populations and the study of the relationship between variables with regression analysis. All accompanied by images that exemplify some basic aspects of its operation.

Keywords: basic statistical analysis, R, software.

JEL classification codes: C12, C13, C88.

Índice

Listado de imágenes y cuadros……4

1. Introducción……6

2. Instalación y formas de trabajar en R……7

3. Estructuras de datos en R……..13

    3.1. Vectores……13

    3.2. Matrices……..14

    3.3. Data frames……...16

    3.4. Listas……..18

    3.5. Factores……19

4. Funciones……21

5. Análisis descriptivo básico……23

6. Representación gráfica de datos……..26

7. Distribuciones de probabilidad……..29

8. Inferencia estadística……..33

9. Regresión……..36

10. Conclusiones……..41

11. Bibliografía……..42

12. Anexos……44

LISTADO DE IMÁGENES Y CUADROS

Listado de Imágenes

Imagen 2.1. Logo de R……..8

Imagen 2.2. Ventana principal de R……8

Imagen 2.3. Instalación de paquetes……...……12

Imagen 2.4. R Comander……12

Imagen 3.1. Vectores……..14

Imagen 3.2. Matrices……...16

Imagen 3.3. Data frames……18

Imagen 3.4. Listas……19

Imagen 3.5. Factores……...20

Imagen 4.1. Creación de funciones..……..22

Imagen 4.2. Modificación de funciones 1……22

Imagen 4.3. Modificación de funciones 2……23

Imagen 5.1. Cargar datos……...23

Imagen 5.2. Análisis descriptivo básico……25

Imagen 5.3. Resumen comandos análisis descriptivo……...25

Imagen 5.4. Estadísticos por grupos……26

Imagen 6.1. Funciones gráficas de bajo nivel……27

Imagen 6.2. Funciones gráficas de alto nivel……..28

Imagen 7.1. Distribución de Poisson……..…30

Imagen 7.2. Teorema central del límite 1. Tamaño de la muestra *n*=5……..…31

Imagen 7.3. Teorema central del límite 2. Tamaño de la muestra *n*=10..……32

Imagen 7.4. Teorema central del límite 3 Tamaño de la muestra *n*=100..……32

Imagen 8.1. Inferencia estadística……34

Imagen 8.2. Inferencia estadística para la igualdad de medias……35

Imagen 9.1. Datos ejemplo……37

Imagen 9.2. Diagrama de dispersión……38

Imagen 9.3. Coeficiente de correlación…..……38

Imagen 9.4. Análisis de regresión…….……39

Listado de Cuadros

Cuadro 7.1. Nombres de distribuciones de probabilidad en R……29

1. INTRODUCCIÓN

R es un software de uso y distribución libre que se encuentra bajo Licencia Pública General de GNU, y como indica Collatón (2014), se utiliza para realizar programaciones de análisis estadístico y gráfico. Este programa fue creado en 1993 por los profesores Ross Ihaka y Robert Gentleman, del departamento de Estadística de la Universidad de Auckland, Nueva Zelanda. Denominado así por la inicial del nombre de pila de ambos, viene desarrollándose desde 1997 con aportes de diversas partes del mundo, bajo la coordinación del equipo principal de desarrollo de R (R Core Team Development) (R Project).

Su sintaxis es similar a la de S, un lenguaje de programación estadística desarrollado en 1976, cuyo objetivo según John M. Chambers, uno de sus creadores, es "convertir ideas en software, rápida y fielmente" (*Programming with Data: A Guide to the S Language*, 1998). Con el paso de los años a este lenguaje de cálculo estadístico, como hacen referencia Santana y Mateos (2014), se le han introducido diversos cambios y modificaciones, hasta llegar a los actuales S-PLUS y R.

Las ventajas e inconvenientes, como indica Pech (2015), son las siguientes:

- Ventajas:
    - Es un software de acceso libre, cualquiera puede acceder a él de forma gratuita.
    - Se puede usar en múltiples sistemas operativos, es multiplataforma.
    - Es de código abierto, se pueden crear o modificar sus funciones para adaptarlo a las necesidades de cada usuario.
    - Sus gráficos tienen una gran calidad y versatilidad.
- Desventajas:
    - Utiliza un lenguaje de programación, por lo que su aprendizaje puede resultar bastante complejo.

- Sus mensajes de error en muchas ocasiones no son claros, dificultando localizar el origen de nuestros fallos.
- Debido al carácter de libre disposición de este programa, y tal y como menciona García (2010), nadie se responsabiliza de los resultados obtenidos por el mismo.

El objetivo de este trabajo es realizar una pequeña introducción al análisis estadístico con R con el fin de mostrar su funcionamiento.

Este trabajo se estructurará de la siguiente manera. La Sección 2 estará dedicada a mostrar la manera de realizar la instalación del programa y las formas de trabajar con R; la Sección 3, al análisis de las distintas estructuras de datos: vectores, matrices, data frames, listas y factores; la Sección 4, al estudio de funciones y de cómo crearlas y modificarlas; la Sección 5, al análisis descriptivo básico, en el que se verán, entre otras cosas, distintas variables de análisis de datos; la Sección 6 a las formas y funciones con las que representar de forma gráfica los datos; la Sección 7, a las funciones y distribuciones de probabilidad con sus respectivos nombres dentro del programa y la demostración con ellas del teorema central del límite; la Sección 8, al uso de la inferencia estadística para estudiar poblaciones y la Sección 9, a cómo estudiar la relación entre variables con el análisis de regresión.

2. INSTALACIÓN Y FORMAS DE TRABAJAR EN R

Para instalar el programa, hay que seguir los siguientes pasos:

1. Acceder a la página de R en la siguiente web https://cran.r-project.org. Una vez en ella tenemos que seleccionar el sistema operativo de nuestro ordenador, ya sea Windows, Linux o Mac. En este caso escogeremos Windows.
2. Una vez seleccionado el sistema operativo tenemos que acceder al enlace *"install R for the first time"*. Para terminar, elegiremos "*Download R 3.4.4 for Windows"* y el programa se guardará en nuestro sistema.

3. En la carpeta de Descargas de nuestro ordenador ejecutamos el programa R-3.4.4 y aceptamos que haga cambios en el equipo. Nos aparece una ventana auxiliar donde nos pregunta el idioma de instalación, seleccionamos español, y vamos aceptando todo lo que nos sale a continuación. Una vez hecho esto se nos crea un icono en el escritorio a través del cual podemos acceder al programa como el de la Imagen 2.1.

Imagen 2.1. Logo de R

En la Imagen 2.2. vemos la ventana principal de R. En la parte superior, tenemos la barra de tareas, y debajo unos iconos que nos permiten realizar algunas funciones específicas.

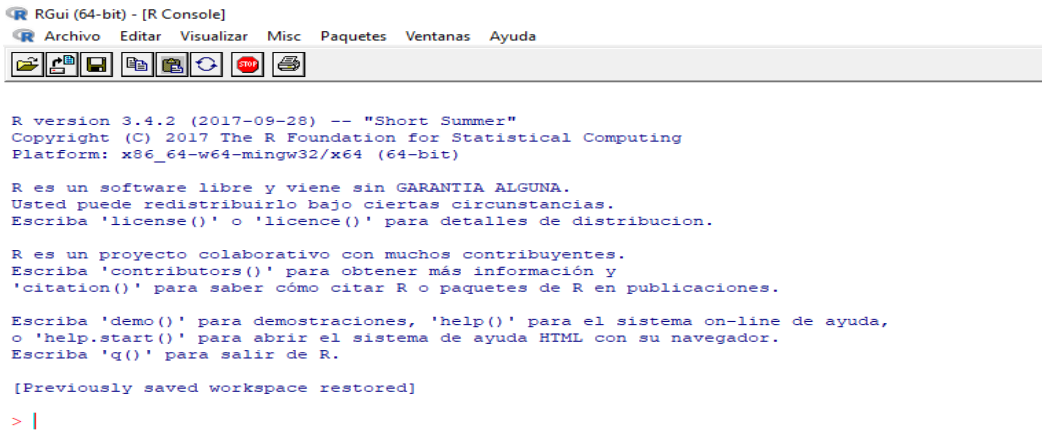

Imagen 2.2. Ventana principal de R

Existen distintas formas de trabajar en R. Una es ejecutar los comandos en la denominada *línea de comandos*, que siempre comienza con el símbolo >, a partir del cual debemos empezar a escribir. Otra forma de trabajar es escribir los comandos en un script, que son editores de texto para una o varias sentencias. Trabajar con scripts nos permite ejecutar todas o una parte de las sentencias. Una tercera forma es cambiando la interfaz (podemos entender una

interfaz como la forma de comunicarse con R) y utilizar otras, siguiendo el estilo de otros programas estadísticos tales como SPSS. Algunos ejemplos son DAS+R o Rcmdr (también llamado R-Commander, que según García (2010) es el más utilizado), aunque es necesario advertir de que tienen muchas limitaciones, pudiendo ejecutarse únicamente métodos estadísticos muy básicos. Es, por esto, por lo que los usuarios de R, por lo general, prefieren usar la línea de comandos. Más adelante veremos cómo descargar y cómo instalar estos interfaces. En este trabajo utilizaremos la primera forma.

Dicho esto, comenzaremos explicando el funcionamiento del software. Como hemos indicado en el párrafo anterior las instrucciones se escriben en la línea de comandos detrás del símbolo >. Estas instrucciones en R se denominan *expresiones*, que una vez escritas se ejecutan con la tecla enter. Si hemos escrito mal la expresión nos aparece un mensaje advirtiéndonos del error. Podemos modificarlo pulsando la tecla con la flecha apuntando hacia arriba, que reproduce la última expresión escrita (cada vez que se presiona esta tecla va pasando a la anterior). Así solo tenemos que corregir el error, no siendo necesario tener que escribirla de nuevo, lo que es especialmente interesante cuando la expresión en cuestión tiene una longitud superior a una línea. También nos aparece un mensaje de error si queremos que nos muestre un objeto que no tenemos definido. Un *objeto* es como llamamos al resultado de ejecutar una expresión precedida por el operador <-, al que denominamos operador *asignación*. Este símbolo es característico de R, antes únicamente se podía utilizar este, pero ahora se puede sustituir por el signo igual =. Hay distintos tipos de objetos, como vectores, matrices, data frames o funciones, entre otros, que veremos con mayor detalle en el apartado 3. Estos objetos, como podemos comprobar en https://cran.r-project.org/doc/manuals/r-release/R-lang.html#Objects, son estructuras de datos especializadas, que podemos crear directamente desde R o, estando ya creados en archivos externos, cargarlos para trabajar con ellos.

A los objetos hay que darles un nombre, que se escribe antes del operador asignación y que puede ser cualquier combinación de letras o números, pero siempre utilizando una letra para empezar. El programa distingue entre

mayúsculas y minúsculas, así que se puede utilizar un mismo nombre para varios objetos.

El programa incorpora de serie funciones, y podemos utilizar muchas más con las librerías que podemos descargar o que nosotros mismos podemos crear. Todas las funciones están compuestas por argumentos, aunque hay algunas en las que no es necesario especificarlos, pues vienen por defecto. Dos funciones con gran utilidad ya incorporadas en el programa, que son ejemplos de funciones que no necesitan argumentos, son *objects()* y *rm()*. La función *objects* nos informa de todos los objetos que hemos creado ordenados alfabéticamente. La otra función, *rm*, sirve para eliminar objetos. Únicamente debemos escribir el nombre del objeto u objetos que queramos eliminar, dentro de los paréntesis, separados entre comas si son varios. Otra forma de eliminar objetos es dándole a otro objeto el mismo nombre del que queramos prescindir.

Al cerrar el programa (que se puede hacer, o bien tradicionalmente, pulsando sobre la X de la esquina superior derecha, o bien ejecutando la expresión q()) aparece una ventana emergente que te da opción de guardar los cálculos realizados a través de la pregunta Guardar imagen de área de trabajo? Si respondemos que sí, podemos usar los objetos usados en la última sesión para la siguiente.

Cabe destacar también que este software además de usarse para realizar diferentes métodos estadísticos, también puede utilizarse como calculadora no sólo con números, sino que también ejecuta cálculos sobre los datos u objetos creados por nosotros en el programa. Realiza desde operaciones tan sencillas como sumas (+), restas (-), multiplicaciones (*) o divisiones (/) hasta otras más complejas, como, por ejemplo, potencias (^), raíces cuadradas o la función exponencial. Estas dos últimas no se realizan de forma tan sencilla como las anteriores, debiendo introducir en la línea de comandos *sqrt(dato)* para la raíz cuadrada y *exp(dato)* para la función exponencial. Además, debemos tener presente que a la hora de introducir los datos para escribir números con decimales no se utilizan comas, sino puntos.

Una de las características más llamativas e importantes de este programa, que hace que destaque por encima de los demás de su categoría, es la capacidad

de incorporar nuevas funciones al programa a partir de los paquetes o librerías. Podemos consultar todos los paquetes existentes en la página de R a través del enlace http://cran.r-project.org/web/packages/ y verlos según áreas en http://cran.r-project.org/web/views/.

Tenemos dos métodos de instalar los paquetes. El primero es a través de la opción Paquetes → Instalar Paquetes, dentro de la barra de tareas. A continuación, nos pregunta por el lugar desde dónde queremos descargarlos, seleccionamos el lugar que tengamos más cerca (en nuestro caso Spain (Madrid)). Después aparece un listado con todos los paquetes y elegimos el que necesitemos.

La otra forma de instalar un paquete, como indican Santana y Hernández (2016) es poniendo en la línea de comandos la función *install.packages ("nombre")*. Para comprobar si tenemos instalado un determinado paquete ejecutamos *library(nombre)*, y si el programa no dice nada es que sí lo está. En caso contrario nos sale error.

La expresión utilizada antes, *library,* es también la que nos permite cargar una librería. Es necesario distinguir entre lo que es instalar un paquete y lo que es cargarlo. Un paquete está instalado instalado si únicamente lo hemos descargado de internet y aparece ubicado en algún directorio de nuestro ordenador en el que R lo puede localizar. Un paquete está cargado si lo hemos ejecutado en la sesión de trabajo actual. Hay que cargar las librerías o paquetes en cada sesión de trabajo para poder usarlas.

Para ejemplificar la segunda forma de instalar paquetes, la Imagen 2.3. muestra la instalación de uno de los paquetes más necesarios, el paquete "foreign", que nos permite abrir en R datos en formato distinto al de este software, como por ejemplo los terminados con la extensión .sav correspondientes al programa SPSS.

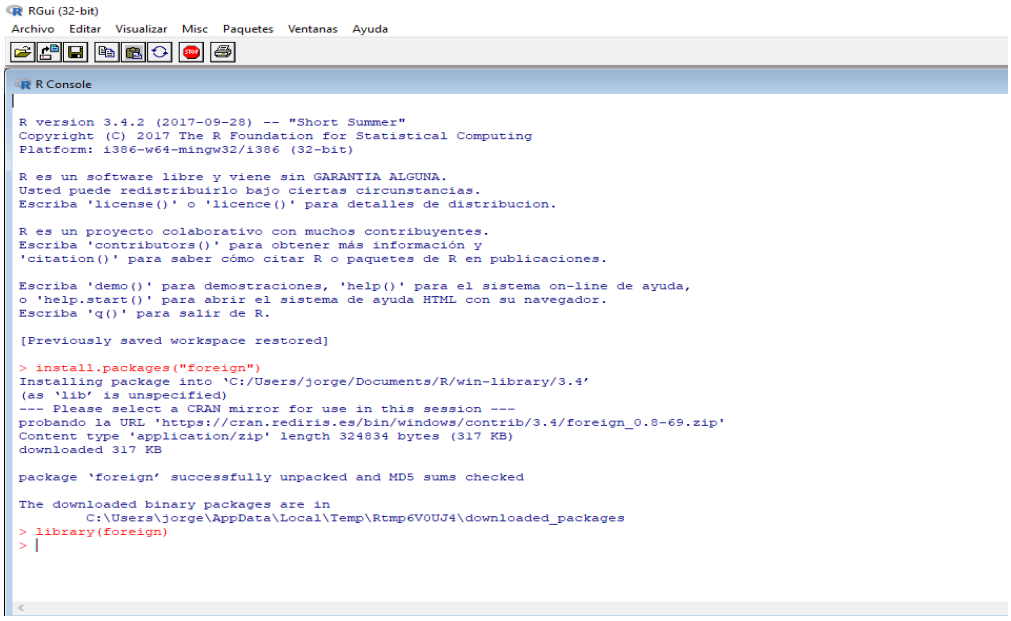

Imagen 2.3. Instalación de paquetes

Como avanzábamos antes, otra de las formas de trabajar en R es cambiando el interfaz a través de otro paquete importante, R-Comander, que nos permite trabajar en una interfaz del programa con una apariencia más sencilla a partir de ventanas y cuadros de diálogos. Lo único que tenemos que hacer es instalar y cargar el paquete Rcmdr. En la Imagen 2.4. vemos la apariencia que presenta:

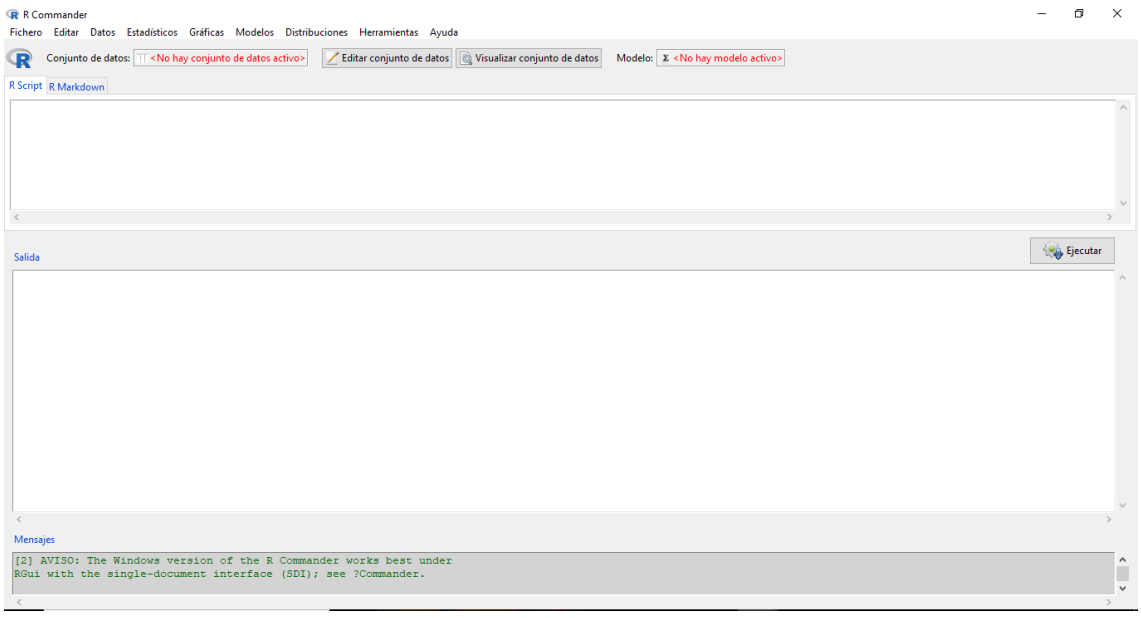

Imagen 2.4. R Comander

3. ESTRUCTURAS DE DATOS EN R

Existen distintas formas de organizar los datos en R, cada una con unas funciones y características. Esta sección está dedicada a revisar algunas de las más empleadas, en concreto, se van a estudiar los vectores, las matrices, los data frames, las listas y los factores (una información más detallada se puede encontrar en http://ocw.uc3m.es/estadistica/aprendizaje-del-software-estadistico-r-un-entorno-para-simulacion-y-computacion-estadistica/estructuras-de-datos-en-r).

3.1. VECTORES

Un vector es la forma de organizar datos más simple que se puede utilizar en R. Para definir vectores lo hacemos a través de la función *c.* Podemos crear tres tipos de vectores: numéricos, lógicos o de carácter.

- Numérico: los valores del vector son números.
- Lógico: vector binario, los valores son T o F (verdadero o falso, según las iniciales en inglés).
- Carácter: los valores del vector están compuestos por caracteres que deben ir separados entre comillas.

El principal inconveniente de los vectores es que no se pueden mezclar los elementos, todos tienen que ser del mismo modo. Es decir, no podemos mezclar en un mismo vector, por ejemplo, números con caracteres, ni con valores lógicos.

Para conocer el modo o la longitud (número de elementos dentro de un vector) podemos hacerlo a través de las funciones *mode* y *length.*

En algunas ocasiones, sobre todo con vectores muy extensos, nos puede interesar conocer los datos de una posición concreta dentro del vector. Para ello podemos extraer los datos escribiendo en la línea de comandos el nombre del vector precedido por, entre corchetes, la función *c,* y entre paréntesis la posición que ocupen los datos que queremos conocer, ya sean solo uno o varios.

También podemos querer excluir uno de los datos, para ello solamente debemos escribir el nombre del vector seguido de la posición que queremos obviar en negativo entre corchetes.

En la Imagen 3.1. podemos ver un ejemplo de cómo se definen vectores y de la utilización de diferentes operaciones realizadas con vectores.

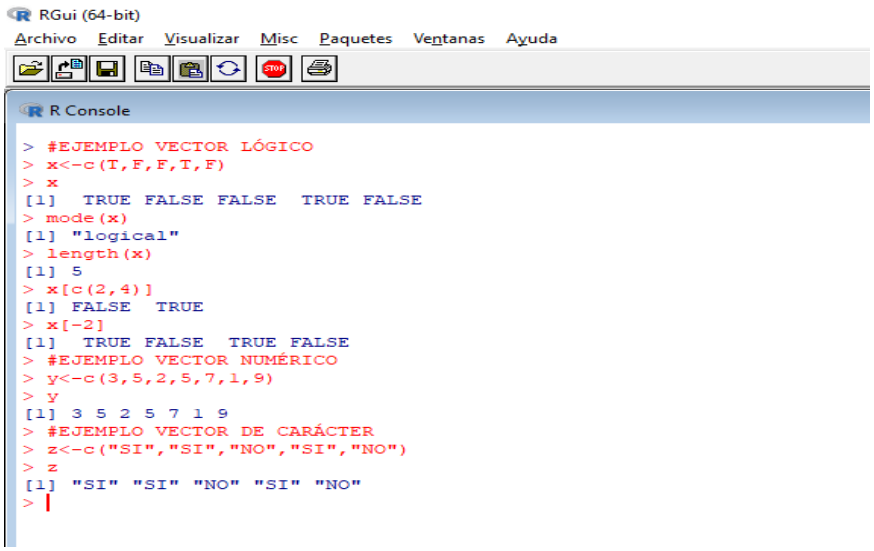

Imagen 3.1. Vectores

3.2. MATRICES

La matriz es un tipo de datos bidimensional, es decir, podemos introducir elementos en dos dimensiones. Igual que sucedía con los vectores, los elementos solo pueden ser de un modo. De hecho, si intentamos mezclar distintos elementos, el programa nos informa de que hemos cometido un error. También, al igual que con los vectores, podemos conocer el modo de los elementos con la función *mode,* y la dimensión con la función *dim.* Esta última función nos devuelve dos números: el primero es el número de filas y el segundo, el número de columnas.

El método para crear una matriz es a través de la función *matrix*, y algo muy importante que se debe tener en cuenta es que las matrices se crean por columnas. Si queremos que se creen por filas debemos utilizar el argumento *byrow=T*. Para determinar el número de columnas que queremos, debemos introducir *ncol=n,* siendo *n* el número de columnas elegido, o bien *nrow=n,* si

preferimos establecer el número de filas. Hay que tener en cuenta que, aunque elijamos determinar el número de filas con *nrow,* los elementos se van a seguir introduciendo por columnas.

Los argumentos *ncol* y *nrow* también se usan para extraer una parte de una matriz ya existente, es decir, para la selección de las primeras filas y columnas de una matriz *x [n,m]*.

Algo que también nos puede interesar es que R nos permite unir varios vectores definidos previamente para crear una matriz. Podemos hacerlo de dos modos, por columnas con la función *cbind,* o por filas, con la función *rbind.*

Una vez creadas las matrices, podemos operar con ellas, siempre y cuando se cumplan las reglas de dimensión para las operaciones entre matrices. En caso contrario nos informa de que hay un error con las dimensiones, como podemos apreciar en el ejemplo de la Imagen 3.2.

Algunas operaciones que se puede realizar con matrices son:

- Suma: +
- Resta: -
- Multiplicación: *
- Cociente de cada uno de los elementos de una matriz entre los elementos correspondientes de otra matriz: /
- Matriz inversa: *solve(x)*
- Matriz transpuesta: *t(x)*

Todo lo que acabamos de mencionar también puede utilizarse para realizar operaciones con vectores numéricos.

Para terminar con el apartado de matrices, vamos a ver cómo dar nombres a las filas y las columnas. Se hace a través del argumento *dimnames* dentro de la función *matrix*, que crea una lista con dos componentes. El primero da los nombres de las filas y el segundo el de las columnas.

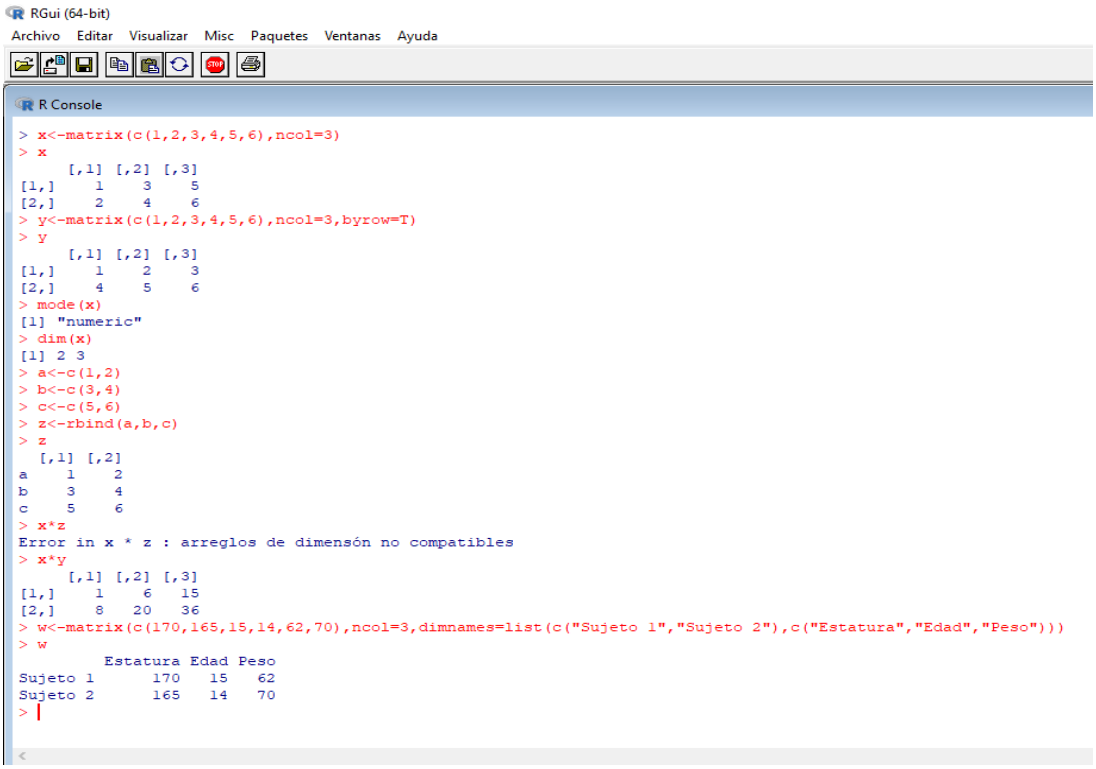

Imagen 3.2. Matrices

3.3. DATA FRAMES

Los data frames son objetos muy similares a las matrices. La diferencia fundamental es que los data frame admiten columnas de distinto modo, aunque todas de la misma longitud.

Para crear un data frame debemos seguir el procedimiento que vimos en el apartado anterior para la creación de matrices, pero teniendo en cuenta que en este caso la función a utilizar es *data.frame*. Una vez creado, si queremos ver la estructura y los nombres de las variables podemos hacerlo a través de las funciones *str()* y *names()*, respectivamente. Podemos ver un ejemplo del trabajo realizado con data frames en la Imagen 3.3.

Para obtener solo determinados datos de un data frame, podemos hacerlo de varias formas, dependiendo de si queremos filtrar datos por filas o por columnas:

- Si queremos solo los datos de alguna fila se pone el nombre del data frame seguido de, entre corchetes, la fila o filas deseadas acabadas en una coma.

- Si queremos los datos de alguna columna, se realiza de la misma forma que la anterior pero la coma debe ponerse al principio, también dentro del corchete. Otra forma es con el nombre del data frame, el símbolo del dólar y el nombre de la variable.

En muchas ocasiones querremos abrir en R datos de ficheros externos para poder trabajar con ellos. Esto se hace a través de la función *read.table,* que convierte esos datos en un data frame. Dentro de la función debemos introducir la ubicación de los datos y el argumento *header=T,* para incorporar la primera línea con los nombres de las variables. Suponiendo que los datos estuvieran en el fichero *d:datos,* deberíamos introducir en la línea de comandos: nombre<-read.table("d:\\datos",header=T). Como ya habíamos avanzado anteriormente, para cargar archivos con un formato distinto al de R debemos hacerlo a través de paquetes, como por ejemplo el paquete *foreign,* que es uno de los paquetes más necesarios.

A la inversa, también podemos convertir datos de R a otros formatos. Dependiendo del formato vamos a necesitar unos paquetes u otros. Para guardarlos en formatos Stata y SPSS basta con el paquete *foreign*, pero para otros como Excel se necesitan otros paquetes más específicos, en este caso el paquete *xlsReadWrite*, como se indica en https://www.r-bloggers.com/lang/uncategorized/302.

Por ejemplo, si queremos guardar un data frame, al que hemos llamado "DATOS", en un archivo con formato .sav (para poder trabajar con el programa SPSS), también llamado "DATOS", tenemos que programar:

write.foreign(DATOS,datafile="DATOS.sav",codefile="código.sps",package="SPSS")

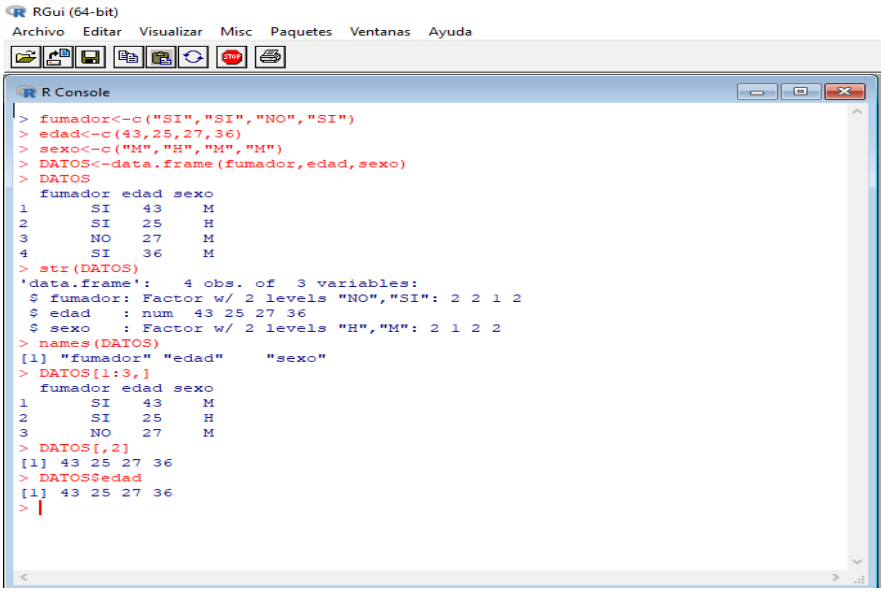

Imagen 3.3. Data frames

3.4. LISTAS.

Dentro de una lista podemos mezclar vectores, matrices, data frames u otras listas. No solo podemos mezclar diferentes modos, también diferentes dimensiones de los datos. Esta versatilidad hace que en la mayoría de las ocasiones, el trabajo final con R sea expresado en listas.

Para crear una lista debemos utilizar la función *list*, seguida por el nombre de los objetos que deseamos introducir en la lista. También podemos crear los objetos directamente dentro de la función *list,* aunque esta segunda opción es mucho más compleja y nos puede llevar a error cuando haya que introducir muchos objetos o sean difíciles de crear debido a su complejidad. En el ejemplo que vemos en la Imagen 3.4., se crea una lista a partir del data frame DATOS que vimos en el ejemplo de la Imagen 3.3. del apartado anterior y una matriz “x”.

También podemos decidir ponerle un nombre a cada uno de los elementos dentro de la lista introduciendo en la función el nombre que queramos darle seguido de un igual y del nombre del elemento. Si no especificamos un nombre los elementos se distinguen por números. Esto va a ser muy importante a la hora de acceder a los objetos que forman parte de la lista. Si tenemos definidos

los nombres, debemos escribir el nombre de la lista, el símbolo del dólar y el nombre del objeto. Si no les hemos puesto nombre, se empieza también con el nombre de la lista pero seguido del número que ocupe el objeto dentro de la misma entre dos corchetes.

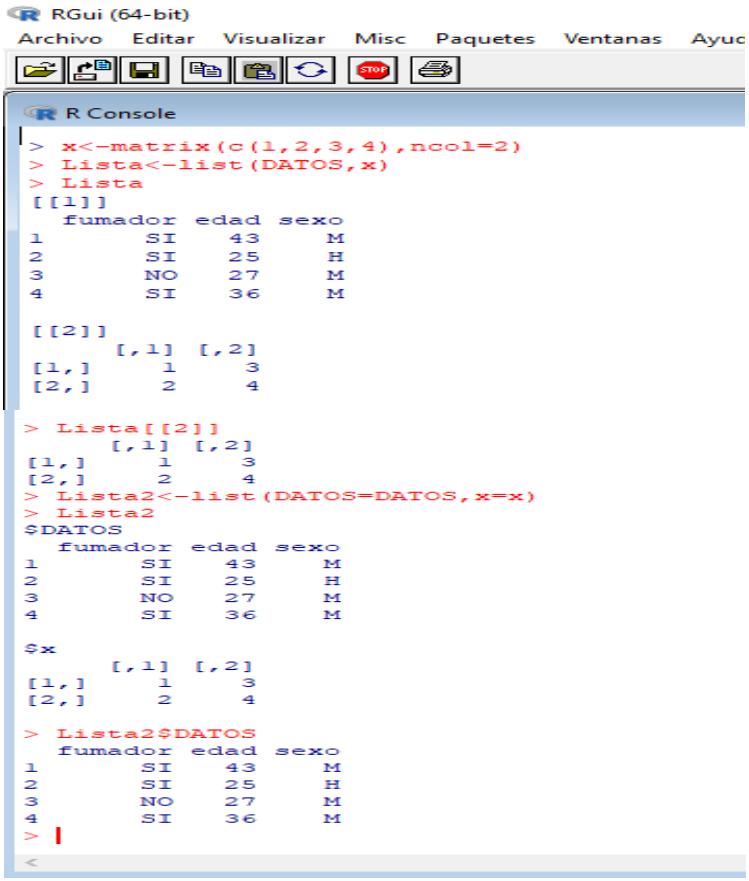

Imagen 3.4. Listas

3.5. FACTORES

El trabajo con variables categóricas requiere que un vector, numérico o carácter, sea declarado en muchas ocasiones como un *factor.* Así, después de crear el vector tenemos que indicarle a R que se trata de un factor, para lo que utilizamos la función *factor*. Si escribimos en la línea de comandos el nombre del factor el programa nos devuelve los datos del mismo, además de indicarnos los distintos niveles por los que está compuesto.

R también nos da la opción de ponerle etiquetas a los factores, especificando dentro de la función que etiqueta ponerle a cada nivel, con las expresiones

labels, y *levels,* respectivamente, como vemos en el ejemplo de la Imagen 3.5. Además, con la función *table*, podemos saber cuántos elementos o individuos pertenecen a cada uno de los niveles.

También tenemos la posibilidad de volver a convertir un factor en una variable numérica por medio de la función *as.numeric*, aunque hemos de tener cuidado porque la conversión nos muestra la codificación interna por la que R guarda los niveles, que suele ser por números enteros desde 1 hasta *n*, siendo *n* el número de niveles. Para corregir esto introducimos *as.character* dentro de la función *as.numeric,* como vemos en la Imagen 3.5.

Algo a tener en cuenta es que la ordenación en la que se presentan las categorías de los factores es alfabética. En el caso del ejemplo de la Imagen 3.5., en el que se nos presenta la frecuencia con que los individuos de la muestra realizan una determinada acción, el orden será "nunca", "ocasionalmente" y "siempre". Puede haber ocasiones en que nos interese tener un orden concreto. Por ejemplo, podemos querer situar en primer lugar "siempre". Para ello tenemos que indicarle a R el orden deseado dentro de la función *factor.*

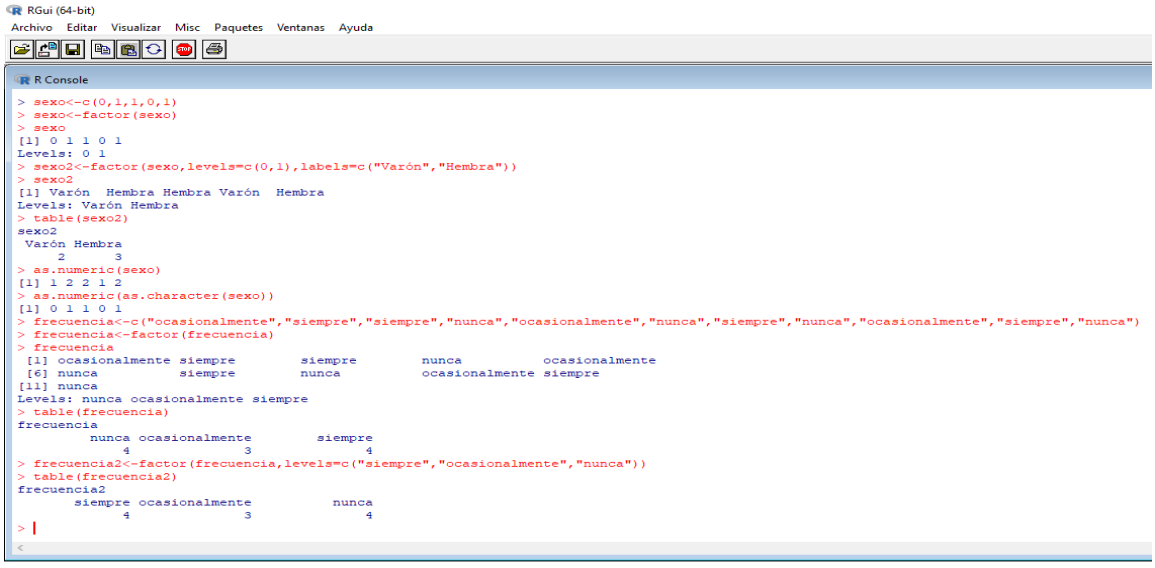

Imagen 3.5. Factores

Podemos ver información más detallada del trabajo con factores en http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/ManualR/intro_estructurasdedatos.html#intro_estructuras_factores.

4. FUNCIONES

Las funciones son uno de los pilares de la programación y del análisis de datos. Durante las primeras secciones ya hemos visto y utilizado algunas funciones para el manejo de R, ahora vamos a verlas y analizarlas con mayor profundidad. En el Anexo 1 podemos encontrar todas las funciones usadas y un pequeño resumen del uso que se les ha dado.

Para utilizar una función debemos escribir su nombre en la línea de comandos, y, entre los paréntesis, se introducen los argumentos. Se puede solicitar ayuda sobre una función de diversas formas:

- Introduciendo ? antes del nombre de la función de R. De esta forma, se abre una ventana en la que entre otras cosas se describe la función, la forma de su uso y ejemplos.

- Si no conocemos la expresión de la función en R, se puede introducir ?? seguido de un texto que haga referencia a la función en inglés en la línea de comandos, y el programa nos ofrece varios recursos relacionados con la misma.

Para crear una nueva función lo más recomendable es hacerlo en un fichero script, para poder escribirla y revisarla sin equivocarnos, aunque también podemos hacerlo directamente en la línea de comandos.

Las funciones están formadas por un nombre (el que queramos darle), los argumentos, el cuerpo de la función y la salida, es decir, el resultado que queremos que nos devuelva la función.

La creación de una función exige escribir el nombre de nuestra nueva función seguida de <- y de *function*, con los argumentos entre paréntesis. A continuación, entre llaves, se escribe el cuerpo de la función, es decir, las sentencias que se quieren ejecutar cada vez que se llame a la función, y los valores que debe devolver la función cuando se ejecute. En la Imagen 4.1. vamos a crear una función llamada e*jemplo*, que sume los dos primeros números y les reste el tercero. Estos tres números constituyen el argumento de la función.

A partir de aquí ya solo nos quedaría revisar que hemos escrito correctamente todo el procedimiento y, una vez comprobado, pasarlo todo a la línea de comandos en el cuadro de texto, y ya podríamos empezar a utilizar nuestra nueva función.

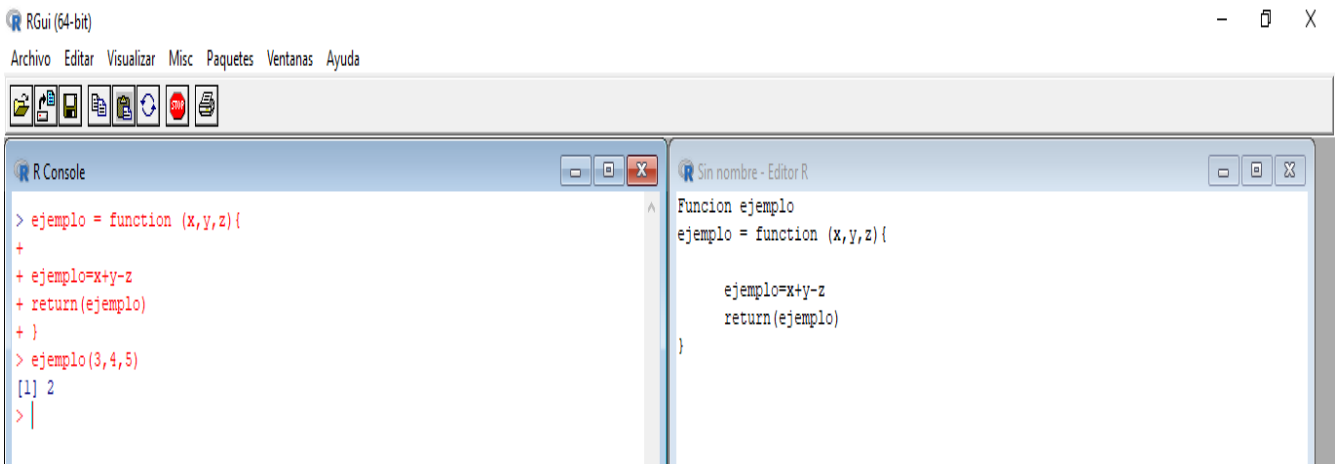

Imagen 4.1. Creación de funciones

También tenemos la posibilidad de editar funciones con la función *fix*. Al introducir la función *fix* se abre el editor de datos, donde nos aparece lo que hace la función que hemos creado, únicamente tendríamos que modificarla para incorporarle los cambios deseados. Ahora la nueva función *ejemplo2,* que podemos ver en la Imagen 4.2., y en la Imagen 4.3., va a multiplicar a la anterior *funcion* que teníamos por una cuarta variable. Cerramos el editor de datos, elegimos la opción de guardar los cambios, y ya podemos utilizarla.

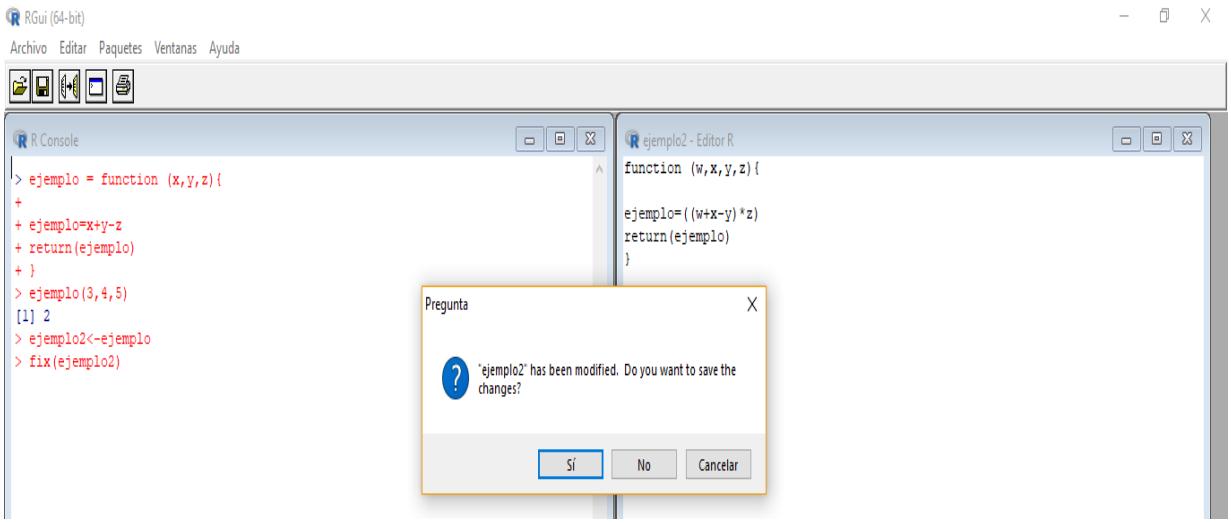

Imagen 4.2. Modificación de funciones 1

Imagen 4.3. Modificación de funciones 2

5. ANÁLISIS DESCRIPTIVO BÁSICO

Para realizar la explicación de cómo hacer un análisis descriptivo vamos a utilizar los datos de una muestra de 260 individuos para los que disponemos de los datos del sexo, nivel de estudios (clasificados en “sin estudios”, “primarios”, “secundaria1”, “secundaria2” y “superior”), renta per cápita y logaritmo neperiano de la renta.

Los datos los tenemos en un archivo en formato spss (extensión .sav). Para abrir este tipo de formato utilizamos el paquete *foreign,* como vemos en la Imagen 5.1.

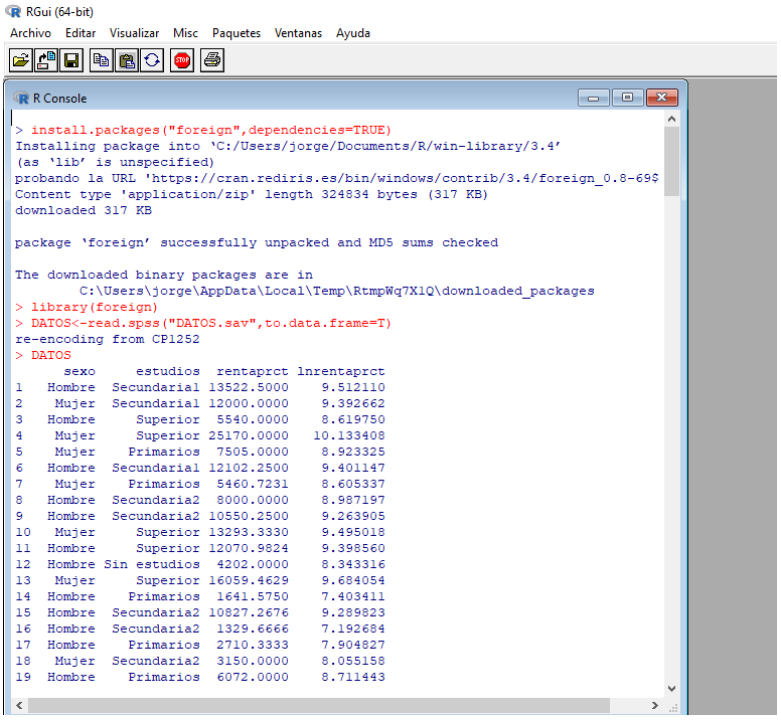

Imagen 5.1. Cargar datos

Para cargar los datos tenemos que introducir en la línea de comandos el nombre que le queremos dar, le indicamos el formato que tiene, el nombre del archivo en el que están recogidos, y le pedimos que nos los transforme en un data frame, pues los datos tienen distintos tipos de elementos (numéricos y caracteres).

Sobre las variables numéricas vamos a ver las funciones más básicas para realizar un análisis descriptivo: media, mediana, varianza y cuantiles.

- La media: es una medida de tendencia central, también llamada promedio, es el cociente entre la suma de los valores de los datos y número de datos. Para R se denomina *mean.*

- La mediana: es el valor que deja la mitad de los valores por debajo y la otra mitad por encima, también es el segundo cuartil. Es la función *median.*

- Cuantiles: dividen la muestra en partes iguales. Reciben distintos nombres dependiendo del número de partes: si es en cuatro cuartiles, si es en diez deciles, y si es en cien centiles. Se obtienen a través de la función *quantile,* seguido del valor, entre 0 y 1, de la posición que queramos saber. Podemos saber también el valor mínimo, el máximo y los cuartiles con la función *fivenum*.

- Varianza: es una medida de dispersión, aunque es más usada la desviación típica, que es la raíz cuadrada de la varianza. Debemos tener cuidado, porque el programa no nos da exactamente la varianza y la desviación típica, sino que nos da la *cuasivarianza* y la *cuasidesviación típica.* Se diferencian entre ellas en que en la cuasivarianza en vez de dividir la suma de los cuadrados de las diferencias entre el tamaño de la muestra *n*, lo divide por *n-1*. Para obtener lo que nosotros conocemos por varianza podemos modificar la función *var* como ya hemos visto para que nos del resultado que nosotros queremos. Para obtener la cuasidesviación típica podemos hacerlo a través de la función *sd* o haciendo la raíz cuadrada del dato que hemos obtenido de la función *var.*

Además, con la función *summary*, podemos obtener directamente los datos que nos proporcionaba *fivenum* y la media a mayores, como podemos ver en la Imagen 5.2. Si son datos no numéricos la función *summary* solo nos dice el número de elementos que hay de cada categoría, pues no podemos hacer un análisis descriptivo sobre ellos.

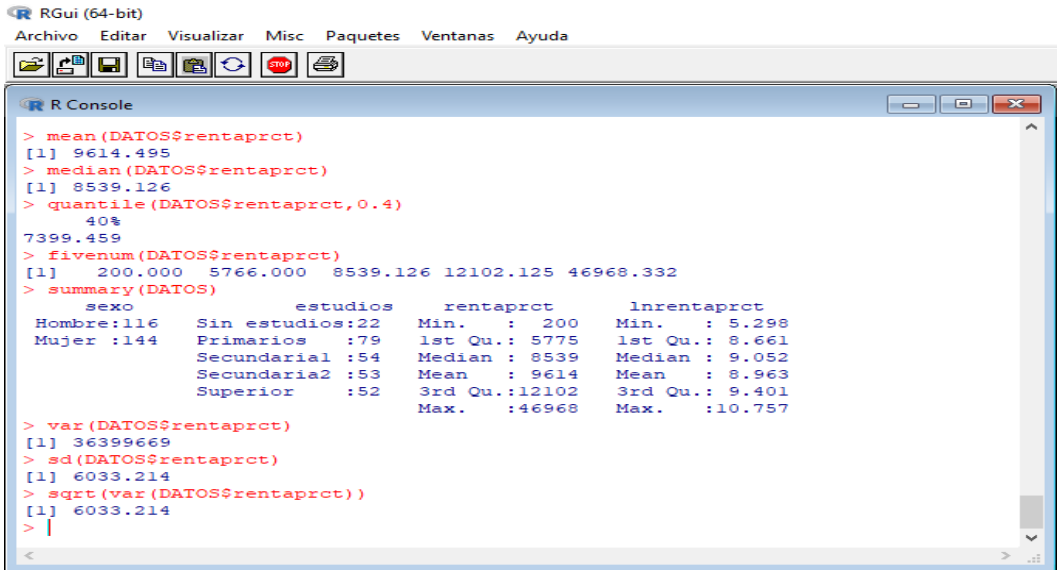

Imagen 5.2. Análisis descriptivo básico

A continuación, se adjunta un cuadro resumen, en la Imagen 5.3., con las expresiones en R para realizar un análisis descriptivo básico.

Nombre Comando	Explicación
summary(*data*)	Resumen estadístico
min(*data*)	Mínimo
max(*data*)	Máximo
range(*data*)	Rango
mean(*data*)	Media aritmética
median(*data*)	Mediana
length(*data*)	Tamaño
sd(*data*)	Desviación típica
var(*data*), cov(*data*)	Varianza
cor(*data*)	Correlación
quantile(*data*, 0.25)	Cuantil Q1
quantile(*data*, 0.75)	Cuantil Q3
IQR(*data*)	Rango intercuartílico
sort(*data*)	Ordenar
table(*data*)	Tabla de frecuencias absolutas

Imagen 5.3. Resumen comandos análisis descriptivo
Fuente: http://www.etsii.upm.es/ingor/estadistica/Grado/rESTDESC.pdf

También podemos obtener los resultados de los estadísticos por grupos, como dicen Santana y Hernández (2016), con la función *aggregate,* en la que tenemos que indicar en primer lugar la variable de la que queremos obtener la información, en segundo lugar la variable de clasificación entre paréntesis precedida de los argumentos *by=list,* y en tercer lugar el dato que queremos conocer. Podemos ver un ejemplo en la Imagen 5.4.

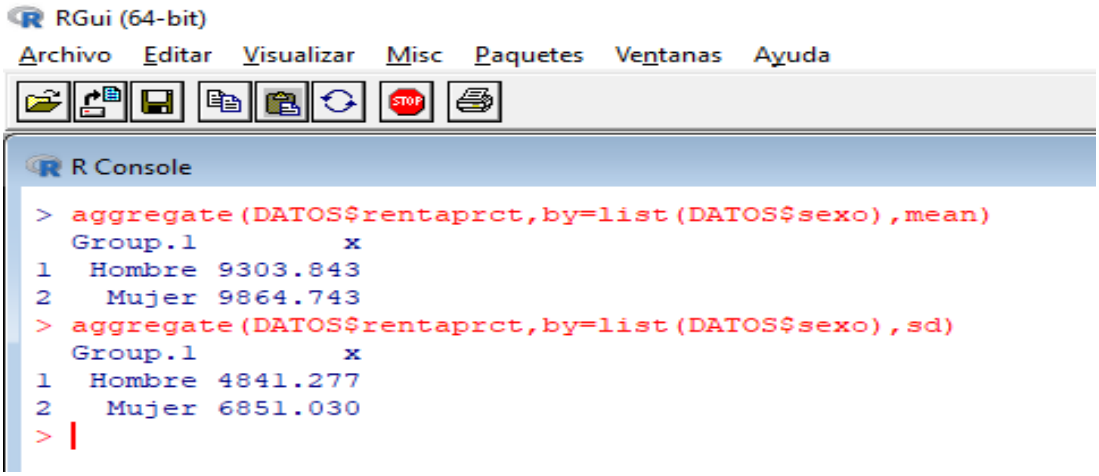

Imagen 5.4. Estadísticos por grupos

Para calcular el coeficiente de asimetría y la curtosis debemos instalar el paquete *moments,* que nos permite utilizar las funciones *skewness* para calcular la asimetría y *kurtosis,* para la curtosis.

6. REPRESENTACIÓN GRÁFICA DE LOS DATOS

El programa R nos ofrece la posibilidad de trabajar de una forma completa y muy intuitiva con los gráficos estableciendo los colores deseados, cambiando los nombres y las leyendas, incorporando comentarios y eligiendo una amplia gama de gráficos. Para trabajar con los gráficos distinguimos entre funciones gráficas de alto nivel y funciones gráficas de bajo nivel.

Como dice García (2010), las primeras son aquellas encargadas de crear un gráfico nuevo con los datos, mientras que las segundas modifican las ya existentes para introducir, como ya avanzábamos antes, elementos como leyendas, líneas, textos, etc.

Dentro de las funciones gráficas de alto nivel, se encuentran algunas funciones como por ejemplo *plot* (diagrama de dispersión), *barplot* (diagrama de barras), *hist* (histograma), *boxplot* (diagrama de caja) *o pie* (diagrama de sectores). Cada una de estas funciones incorporan opciones que permiten, por ejemplo, poner un título al gráfico con *main,* cambiar los colores con *col,* incluir leyendas con *labels* o elegir el tipo de gráfico con *type.* Si utilizamos *type*="*p"* representa puntos, con "*l*" líneas, con *"b"* ambas y con "*n"* no dibuja nada.

En cuanto a las funciones gráficas de bajo nivel, que se añaden una vez creado el gráfico, vamos a destacar *lines,* que une con líneas los puntos de la gráfica, *abline,* que introduce líneas especificando la ordenada en el origen y la pendiente (en ese orden) y *points,* que añade puntos introduciendo primero su situación en el eje de abscisas (eje x) y luego en el eje de ordenadas (eje y). También puede ser muy útil introducir un texto dentro del gráfico con la función *text*, señalando la posición, el texto entre comillas, y el color en el que queramos que aparezca, porque si no por defecto nos aparece en negro. Vemos un ejemplo de lo explicado anteriormente en la Imagen 6.1.

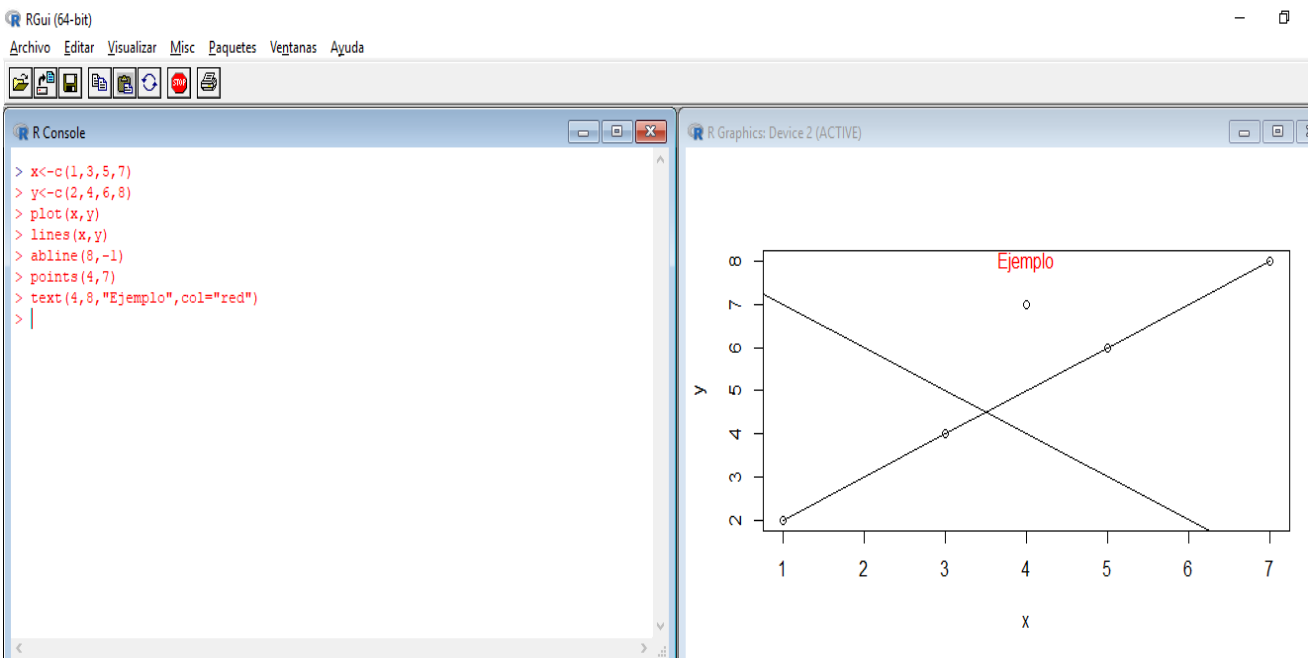

Imagen 6.1. Funciones gráficas de bajo nivel

Cuando ejecutamos un gráfico en R se nos abre una ventana llamada R Graphics, que solo nos permite crear un gráfico, si queremos hacer otro se nos va a borrar el anterior. No obstante, la función *par*, con las opciones *mfrow* y

mfcol, nos van a permitir ejecutar varios gráficos dentro de la misma ventana R Graphics en el orden que queramos, como vemos en el ejemplo de la Imagen 6.2.

Imagen 6.2. Funciones gráficas de alto nivel

Además de todas estas funciones básicas que nos ofrece R, existen algunos paquetes que nos van a permitir realizar gráficos de más calidad y complejidad, como por ejemplo los paquetes *lattice* y *ggplot2*, de los que habla Kabacoff (2017):

- El paquete *lattice* nos ayuda a mejorar los gráficos puesto que proporciona mejores valores predeterminados, además de que admite la creación de gráficos que pueden mostrar la relación entre variables, condicionadas a una o más variables.

- El paquete *ggplot2* cuenta cada vez con más popularidad entre los usuarios de R, pues permite elaborar gráficos de forma más elegante y compleja y representar datos tanto numéricos como categóricos univariados o multivariados de forma directa.

7. DISTRIBUCIONES DE PROBABILIDAD

Como indica Newbold (1997), la distribución de probabilidad de una variable aleatoria es una representación de las probabilidades de todos los resultados posibles. Podemos clasificar las distribuciones de probabilidad en dos grandes grupos, las continuas y las discretas.

Las continuas son aquellas que pueden tomar cualquier valor dentro del intervalo de la variable aleatoria, mientras que las discretas solo pueden tomar ciertos valores dentro del mismo, por lo general nos estaremos refiriendo a números enteros.

Dentro del programa vienen incorporadas múltiples distribuciones de probabilidad con las que podemos trabajar gracias al paquete *stats,* el cual podemos utilizar sin necesidad de cargarlo al venir incorporado por defecto, como nos señalan Santana y Hernández (2016).

Para consultar las distribuciones disponibles podemos hacerlo con la ayuda que nos proporciona el programa, ejecutando, por ejemplo, *help(distributions).* Si quisiéramos trabajar con otras distribuciones más complejas sería necesario hacerlo a través de la instalación de otros paquetes.

Algunas de las distribuciones de probabilidad más comunes, y, por tanto, más utilizadas, aparecen en el Cuadro 7.1.

DISTRIBUCIONES DISCRETAS		**DISTRIBUCIONES CONTINUAS**	
Distribución	**Nombre en R**	**Distribución**	**Nombre en R**
Binomial	*binom*	Beta	*beta*
Geométrica	*geom*	Chi-Cuadrado	*chisq*
Hipergeométrica	*hyper*	Exponencial	*exp*
Binomial Negativa	*nbinom*	F de Fisher	*f*
Poisson	*pois*	Gamma	*gamma*
		Log-Normal	*lnorm*
		Normal	*norm*
		t de Students	*t*
		Uniforme	*Unif*

Cuadro 7.1. Nombres de distribuciones de probabilidad en R

Para cada una de estas distribuciones podemos utilizar cuatro funciones, escribiendo el nombre en R a continuación del nombre de la función que nos van a proporcionar los valores de:

- *d*: la función de densidad o de probabilidad en un punto.
- *p*: la función de distribución en un punto.
- *q*: la función cuantil.
- *r*: la función que genera una muestra de una determinada distribución.

Vemos un ejemplo, en la Imagen 7.1., de todas estas funciones con la distribución de probabilidad de poisson (*pois*) con el parámetro lambda (λ) igual a 1,5. En las funciones de densidad y de probabilidad buscamos el valor en 2 de estas funciones; en la función cuantil, los valores 0,5 y 0,95 y obtenemos una muestra aleatoria de tamaño 10.

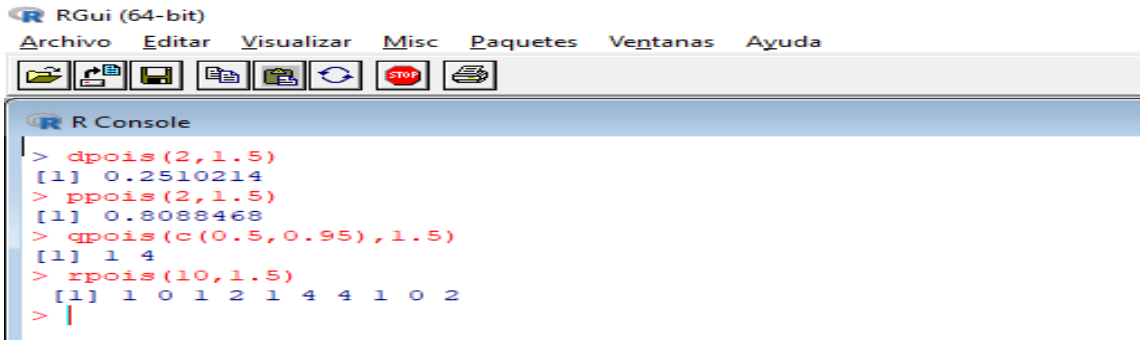

Imagen 7.1. Distribución de poisson

Además, podemos utilizar el programa R para mostrar el cumplimiento del Teorema Central del Límite, que, según Newbold (1997), dice que, sea cual sea la distribución de las variables, la distribución de las variables, la distribución asintótica de la media muestral es una distribución normal. En nuestro caso, vamos a generar un número elevado de muestras (1000) de diferente tamaño (5, 10 y 100) de la misma distribución Poisson del ejemplo anterior con un valor del parámetro lambda de 1,5. Con cada muestra, se obtendrá la media y, para cada tamaño muestral, se representará el histograma de los valores de la media. El aspecto del histograma se irá pareciendo al de una distribución Normal a medida que se incrementa el tamaño muestral.

En primer lugar, generamos una muestra de tamaño 5 (*n*=5) y obtenemos su media. Si generamos diferentes muestras de este mismo tamaño, comprobamos que las medias de dichas muestras cambian. Para agilizar el trabajo, vamos a crear una función *media* cuyo argumento es el tamaño de la muestra, *n,* que nos devuelva la media de una muestra de tamaño *n* de una distribución de Poisson con lambda de 1,5.

También usamos la función *replicate,* que, como su propio nombre indica, se encarga de replicar una función, en nuestro caso la función *media*, tantas veces como necesitemos, en este caso hemos escogido 1000.

Para terminar, le pedimos al programa que nos represente un histograma para poder ver gráficamente en la Imagen 7.2., en la Imagen 7.3., y en la Imagen 7.4., que, a medida que aumentamos el tamaño de la muestra, *n*=5, *n*=10, *n*=100, cada vez se aproxima más a la distribución Normal. También realizamos el contraste de Shapiro-Wilk, que contrasta la hipótesis nula de normalidad frente a la hipótesis alternativa de no normalidad para verificarlo (suponemos un nivel de confianza del 95%).

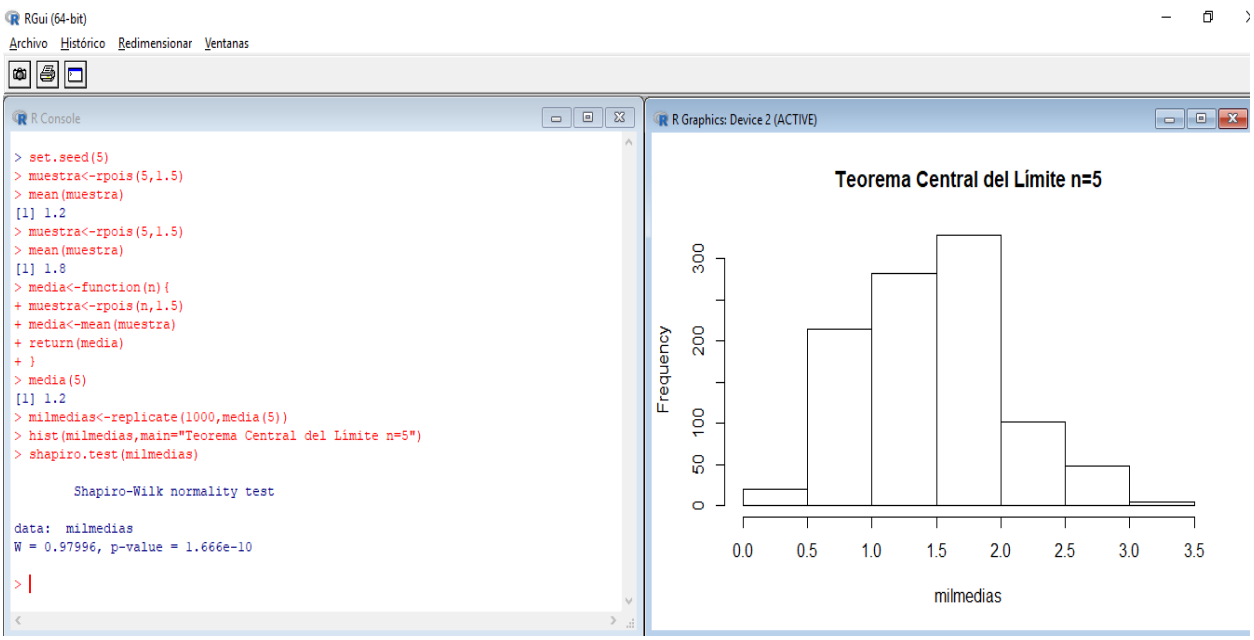

Imagen 7.2. Teorema central del límite 1. Tamaño de la muestra *n*=5

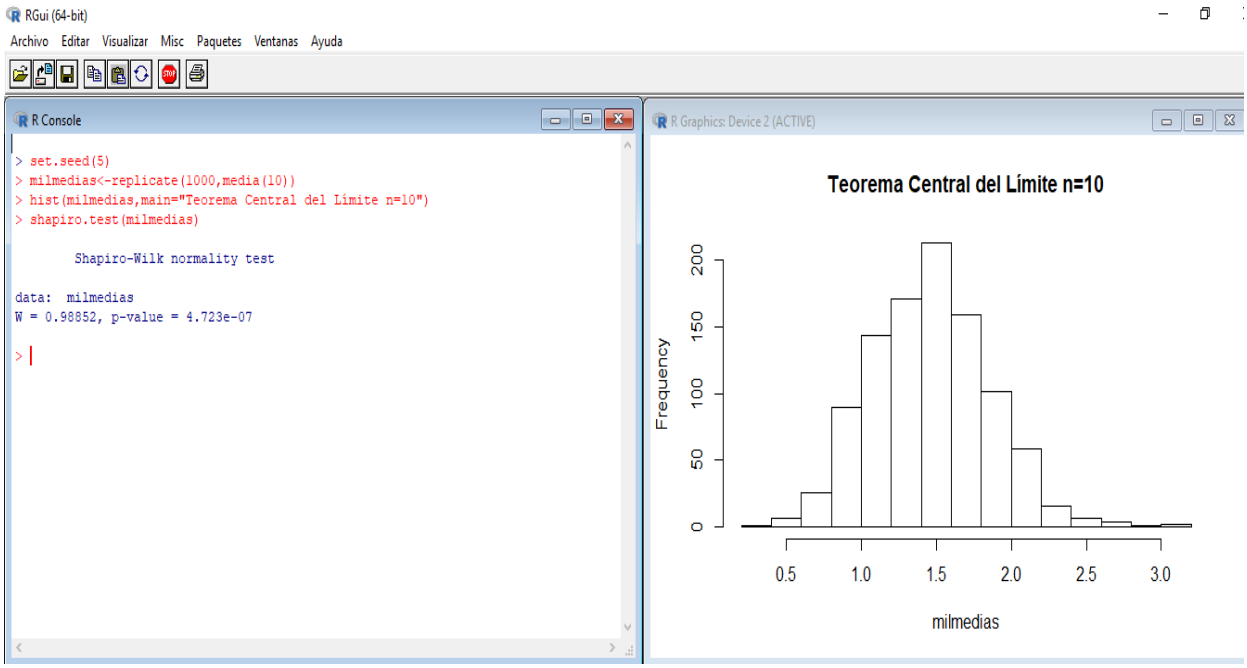

Imagen 7.3. Teorema central del límite 2. Tamaño de la muestra *n*=10

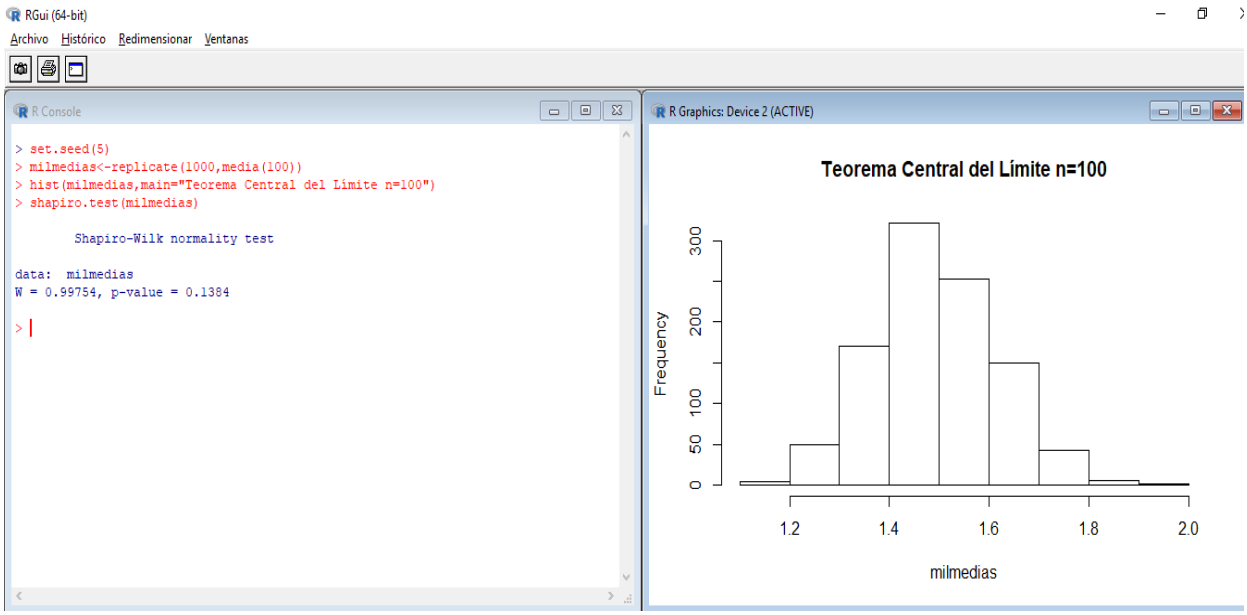

Imagen 7.4. Teorema central del límite 3. Tamaño de la muestra *n*=100

Podemos comprobar que, efectivamente, a medida que aumenta el tamaño de la muestra, el dibujo del histograma que nos muestra se va aproximando más al de la distribución normal. Además, a la vista del contraste de Shapiro-Wilk, en los dos primeros casos de la Imagen 7.2. y de la Imagen 7.3., rechazamos la hipótesis nula de normalidad, mientras que en el caso de la Imagen 7.4. no rechazamos la hipótesis nula, es decir, podemos asumir normalidad sólo en el último de los casos.

8. INFERENCIA ESTADÍSTICA

La función *t.*test nos permite realizar tanto intervalos de confianza como contrastes de hipótesis para poblaciones normales. Esta función nos va a proporcionar los datos del valor del estadístico t (*t*), los grados de libertad (*df*), el p-valor (*p-value*) y el intervalo de confianza (*confidence interval*). Hemos de tener en cuenta que, por defecto, esta función considera que el contraste es bilateral y que el nivel de confianza es del 95%, aunque se pueden cambiar las especificaciones que R tiene asignadas por defecto.

Como nos señala García (2010), para indicar el tipo de contraste de hipótesis tenemos que hacerlo dentro de la función con el argumento *alternative="two.sided",* en el caso de contraste bilateral (recordamos que no es necesario, pues el programa toma esta opción por defecto), *alternative="greater",* cuando la hipótesis nula es menor o igual y la alternativa mayor, y *alternative="less",* para hipótesis nula mayor o igual y alternativa menor.

Para cambiar el nivel de confianza deseado, se hace a través del argumento *conf.level=,* indicándole el valor no en porcentaje, sino en números decimales. Podemos ver un ejemplo de todo esto en la Imagen 8.1.

Con esta función *t.test* podemos realizar:

- Inferencia estadística para la media de una sola población.
- Inferencia estadística para la media de dos poblaciones. En este caso, además, vamos a tener que especificar dentro de la función:
    - Si se trata de muestras pareadas o independientes. Según Newbold (1997), los datos pareados son datos de poblaciones dependienes, en los que los datos van emparejados por individuos, es decir, en cada muestra tenemos los datos de los mismos individuos. R toma por defecto la opción de que son muestras independientes, para indicarle lo contrario tenemos que introducir dentro de la función *t.test* los argumentos *paired=T*.

- Si queremos comparar las medias de poblaciones independientes, también vamos a tener que especificar, dentro de la función, si las varianzas poblacionales son iguales o distintas, con el argumento *var.equal=FALSE,* en caso de ser distintas, y con *var.equal=TRUE,* en caso de ser iguales. Para obtener esta información utilizamos la función *var.test*, donde vamos a tener que aclarar también el tipo de contraste y el nivel de confianza (lo veremos con mayor detalle en el ejemplo de la función *t.test* que se muestra en la Imagen 8.2.).

Vemos un ejemplo en la Imagen 8.1. de la sentencia para realizar el contraste y los resultados del mismo con los datos utilizados para otros apartados. En concreto, con los datos del logaritmo neperiano de la renta, pues se asemejan más a una distribución normal que los datos de la renta, vamos a contrastar si la media del logaritmo neperiano de la renta es igual a 9 frente a una hipótesis alternativa de que es distinto para un nivel de confianza del 90%.

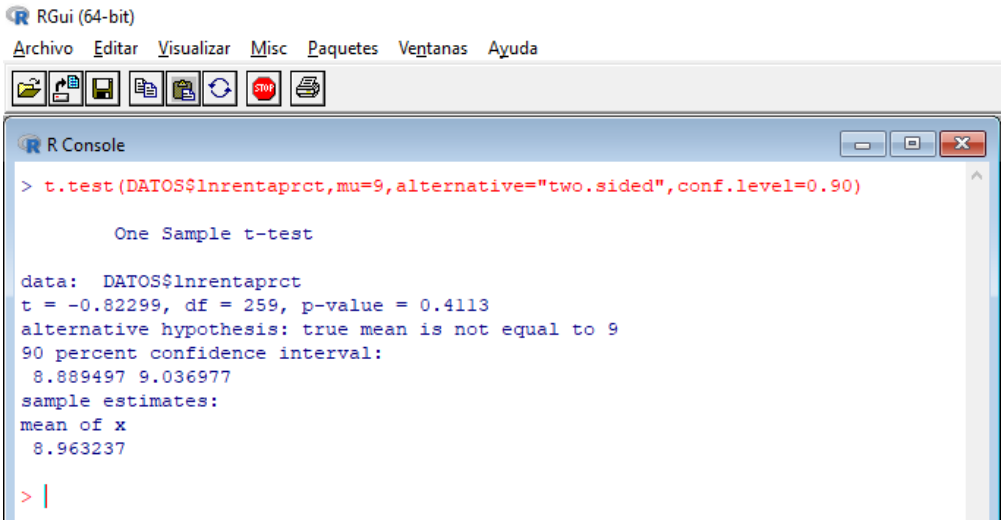

Imagen 8.1. Inferencia estadística

Analizamos los datos que nos devuelve el programa. Para empezar, nos informa de cuáles son los datos que ha estudiado y nos proporciona el valor muestral del estadístico de contraste (-0,82299), los grados de libertad de la distribución t de Student, distribución del estadístico de contraste (259), y el p-valor (0,4113). En este caso como el p-valor es mayor que el nivel de significación fijado (0,1), podemos no rechazar la hipótesis nula de que el valor de la media poblacional del logaritmo neperiano de la renta es igual a 9.

También nos informa de cuál es la hipótesis alternativa (que la media no es igual a 9) y nos da el intervalo de confianza, que aquí también podemos utilizar para ver si se rechaza o no la hipótesis nula. Como 9, que es el valor que suponemos para la media bajo la hipótesis nula, se encuentra dentro del intervalo de confianza, no rechazamos la hipótesis nula, o lo que es lo mismo, rechazamos la hipótesis alternativa. Lo último que nos proporciona es la media muestral.

Vamos a ver ahora un ejemplo en la Imagen 8.2. en el que comparamos las horas diarias medias que se ve la televisión en las viviendas de dos poblaciones distintas. Para realizar el análisis estudiamos dos muestras, una para cada población, de 8 viviendas en cada una de ellas.

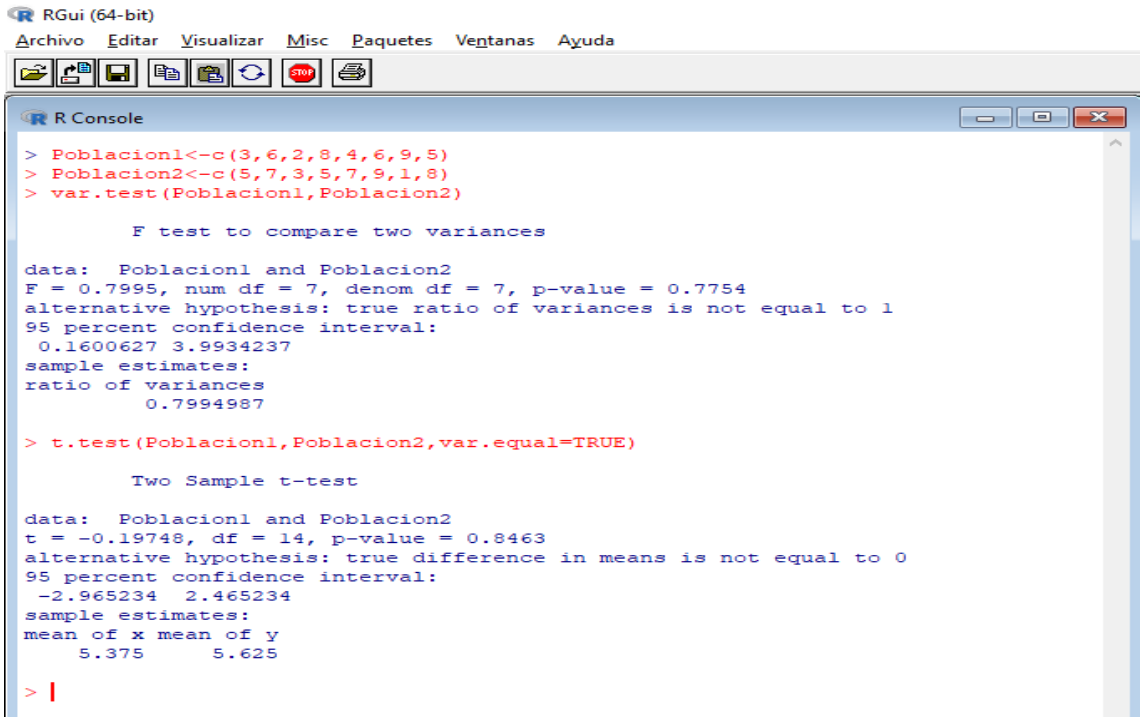

Imagen 8.2. Inferencia estadística para la igualdad de medias

Primero hay que comprobar si las varianzas de las dos poblaciones son iguales. Como nos indica la función *var.test,* sí que lo son, por lo que, al realizar el *t.test,* debemos incorporar en la función esta información con el argumento *var.equal=TRUE*. No hace falta indicar nada más porque escogemos realizar el contraste de hipótesis con los datos por defecto, es decir, contraste bilateral, que la diferencia de medias es igual 0, y para un nivel de confianza del 95%.

Debido a que el p-valor es mayor que el nivel de significación, y a que el valor de la diferencia entre las medias está dentro del intervalo de confianza, no rechazamos la hipótesis nula de igualdad de medias, podemos afirmar para un nivel de confianza del 95% que las horas medias que se ve la televisión en las dos poblaciones son iguales.

9. CORRELACIÓN Y REGRESIÓN

Debemos distinguir entre el análisis de la regresión y el análisis de la correlación, que se utiliza para saber si dos variables están relacionadas linealmente. Una de las formas de analizar la correlación es a través del coeficiente de correlación, que se calcula en R con la función *cor.* Este coeficiente de correlación expresa, en una escala numérica comprendida entre menos uno y uno, el grado de relación lineal entre dos variables. Cuanto más cercana esté a uno o a menos uno, los extremos, mayor relación lineal. El signo del coeficiente determina si la relación entre las variables es directa o inversa.

En cambio, la finalidad del análisis de regresión, como indica Newbold (1997), es encontrar un modelo para explicar la relación entre variables, es decir, tratamos de analizar el comportamiento de una variable (variable dependiente) en función del comportamiento de otra u otras variables (regresores o variables explicativas). Para esto nos servimos de la función *lm,* que sirve para estimar por mínimos cuadrados ordinarios un modelo de regresión (véase Lara, 2017, para un tratamiento más completo del análisis de regresión en R).

Con un nuevo fichero de datos, en el que disponemos de los datos de una región comunitaria relativos al PIB per cápita y al precio de la gasolina en la misma, desde el primer trimestre de 1988 hasta el último trimestre de 2012, vamos a comprobar la relación existente entre el precio de la gasolina (variable dependiente) y el PIC per cápita (variable independiente). Los podemos ver en la Imagen 9.1.

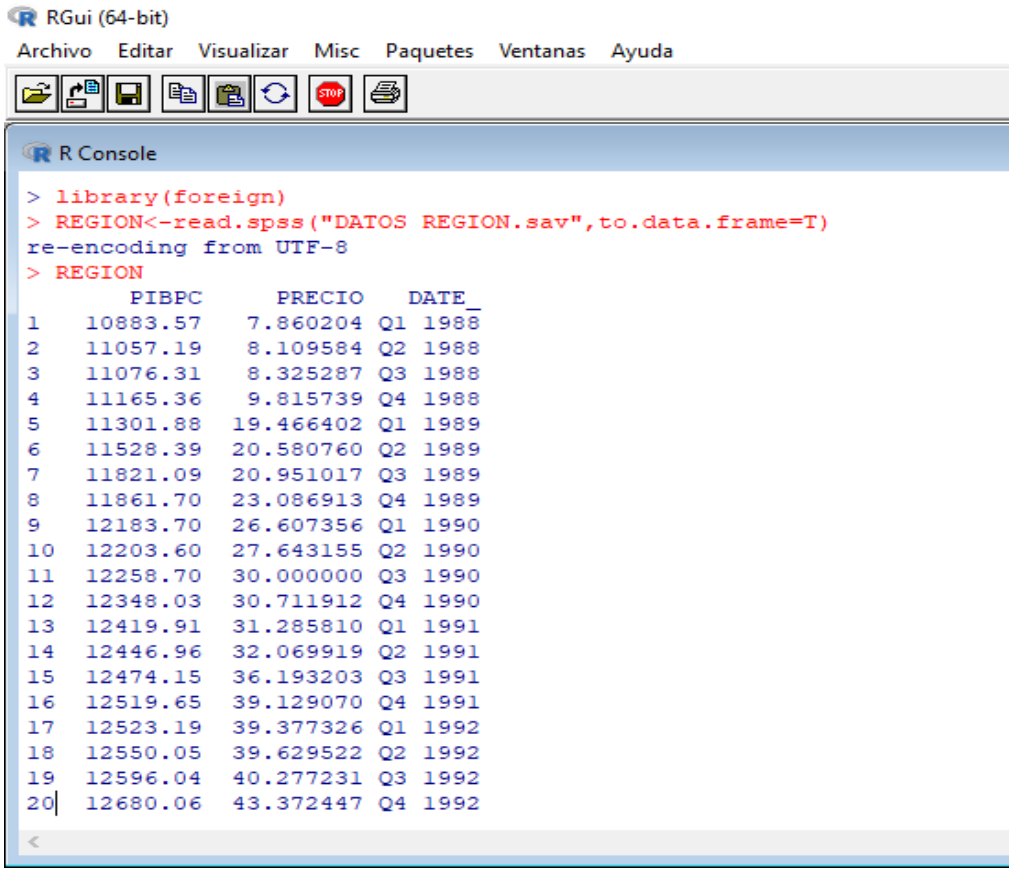

Imagen 9.1. Datos ejemplo

Lo primero es verificar si existe una relación lineal entre ambas variables, representamos el diagrama de dispersión, en el que recordamos que la variable X es el precio y la variable Y el PIB per cápita.

Comprobamos claramente en el gráfico de la Imagen 9.2. que sí existe relación lineal entre el precio y el PIB per cápita. En otros casos podríamos ayudarnos de la función *abline* para dibujar la recta de regresión lineal, pero en este vemos que no es necesario, pues la relación lineal es muy evidente.

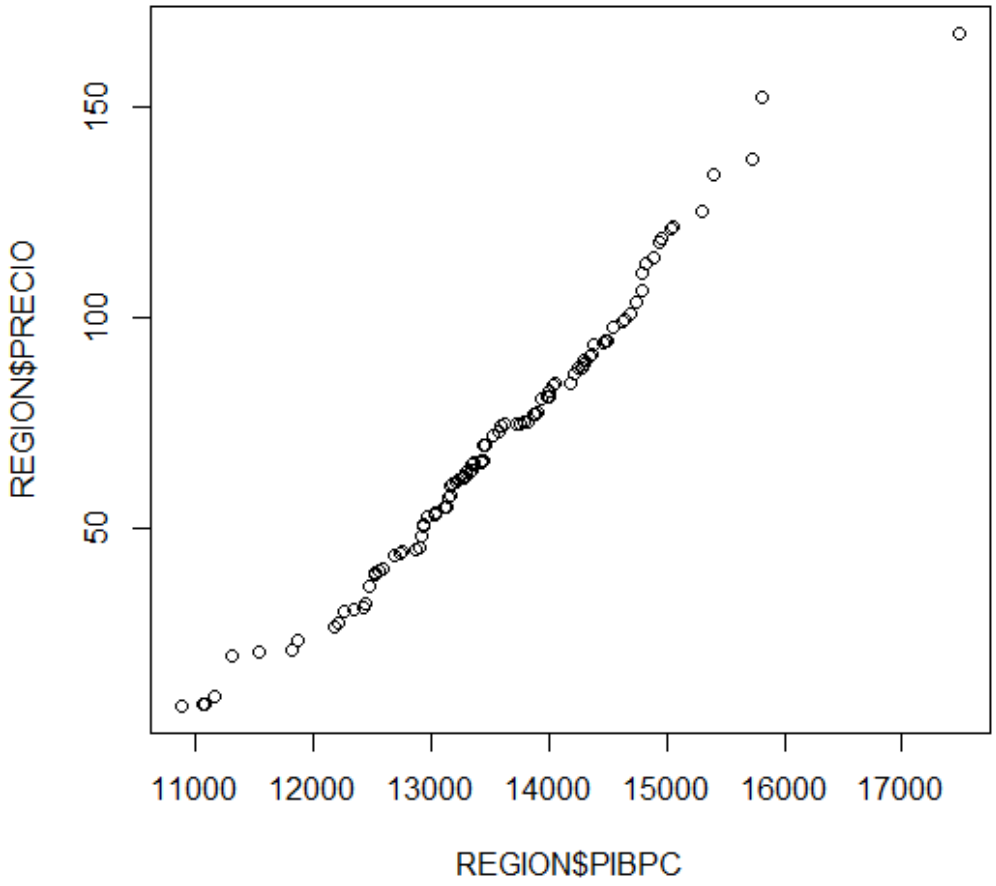

Imagen 9.2. Diagrama de dispersión

Asimismo, calcularemos el valor concreto de la correlación lineal como se muestra en la Imagen 9.3.. Para calcular la correlación entre las variables lo hacemos utilizando la función *cor*, en la podemos elegir entre tres métodos para calcularlo: pearson, kendall y spearman. Si no indicamos nada la opción escogida por defecto es a través del método de Pearson.

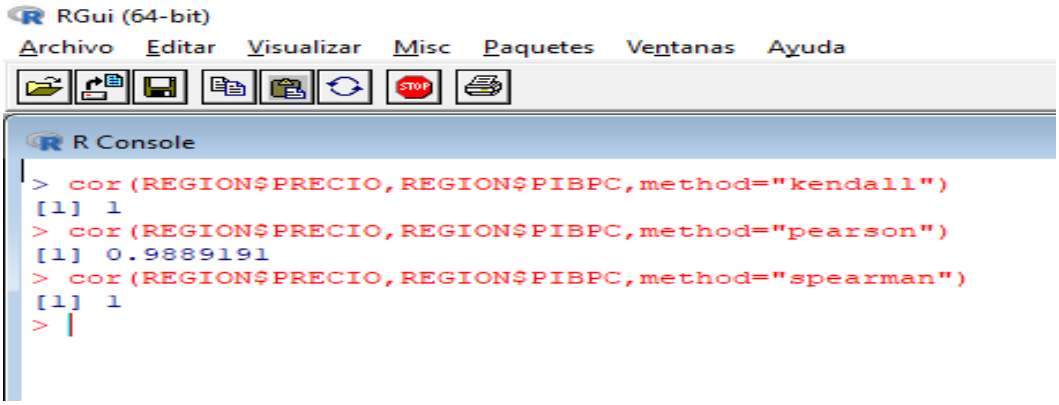

Imagen 9.3. Coeficiente de correlación

Ahora vamos a realizar la regresión como tal, lo que se consigue con la función *lm*, que analizaremos con mayor detalle más adelante a la vista de los resultados de la Imagen 9.4. Damos un nombre al objeto que va a crear esta función: *Regresion.* Dentro de la función tenemos que escribir primero el nombre de la variable dependiente y luego el de la independiente, separadas por el símbolo ~.

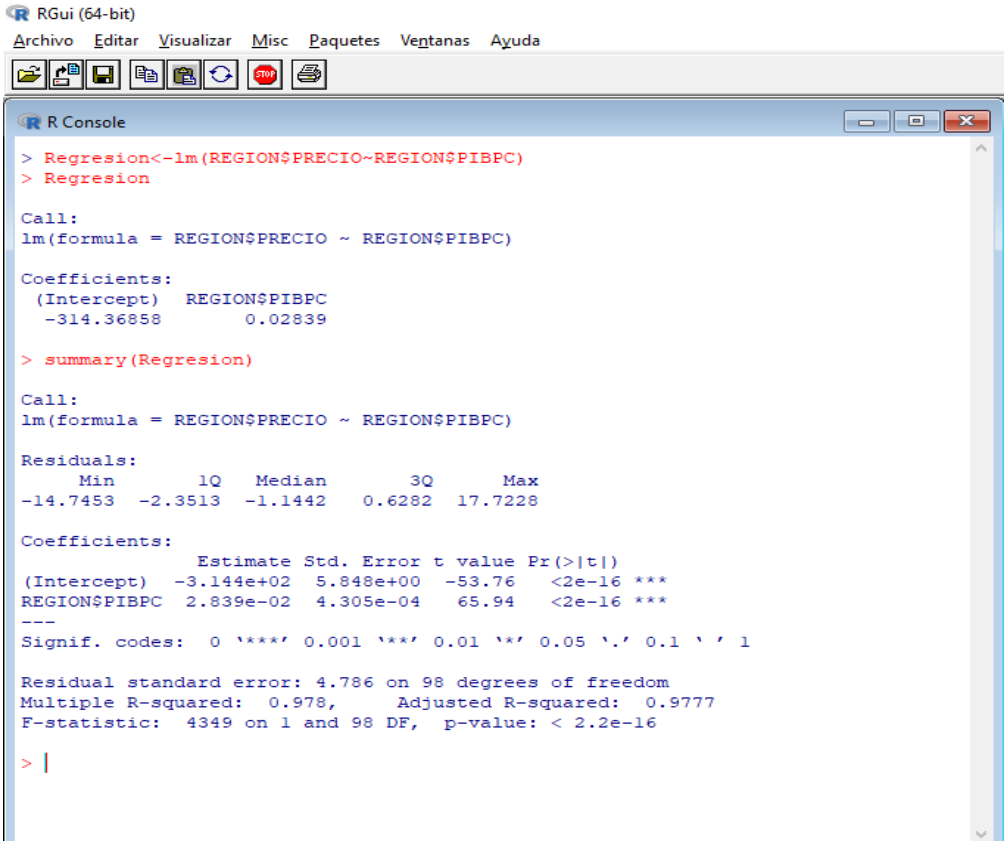

Imagen 9.4. Análisis de regresión

Al ejecutar la función *lm,* como vemos en la Imagen 9.4., obtenemos la estimación de los valores de los coeficientes. En primer lugar nos indica el valor del estimador β_0 (Intercept), cuya interpretación en solitario la mayoría de las ocasiones no tiene sentido, pues lo que nos indica en este caso es que el precio de la gasolina para un valor del PIB de 0 u.m. es de -314,74 u.m., lo que es imposible. También nos da el valor del estimador β_1, el valor del coeficiente asociado al PIB per cápita, 0,02839 (es un valor positivo, por lo que ya podemos afirmar que la correlación entre ambas variables es positiva). Analizamos el conjunto de la ecuación:

Precio $_i$ = -314,36858 + 0,02839 x PIB per cápita $_i$

A partir de esta función podemos determinar el precio estimado de la gasolina a través del PIB per cápita. Por ejemplo, cuando el PIB sea de 13.000 u.m., el precio de la gasolina será de 54,70142 u.m.

Además, podemos conseguir mayor información de la regresión con la función *summary*. Como podemos apreciar en la Imagen 9.4., nos informa del valor de los residuos, que son la diferencia entre el valor real y el valor estimado en el modelo. También nos da más detalles sobre el modelo: los valores de los coeficientes, su desviación típica, el valor del estadístico t y el p-valor. Con estos dos últimos podemos realizar un contraste de significación, es decir, comprobar si las variables explicativas escogidas sirven para explicar el modelo. En este caso vemos que el p-valor de la variable PIB per cápita es muy cercano a cero, menor que 0,05 (pues fijamos un nivel de confianza del 95%), por lo que sí que es significativa para explicar el modelo. Esto es lo que llamamos significación individual de las variables. Para contrastar la significación conjunta podemos hacerlo con lo último que nos ofrece esta función *summary*, el valor del estadístico F y su p-valor. En este caso vemos que el p-valor de este estadístico coincide con el p-valor del estadístico t, porque al haber solo un único regresor o variable independiente la significación individual es igual a la significación conjunta.

Para terminar con la explicación de esta función, nos da también los valores del R2 (o coeficiente de determinación) y del R2 ajustado, con los que vamos a poder analizar la bondad del ajuste. Los valores están entre 0 y 1, cuanto más cercanos a 1 sean mejor será la bondad del modelo. El R2 es 0,978, lo que significa que el 97,80% de la variabilidad de la variable dependiente viene explicada por el modelo. El R2 ajustado, que en este caso es 0,9777, nos sirve para comparar entre modelos con diferente número de regresores, a mayor sea el valor de este mejor será el modelo.

10. CONCLUSIONES

El objetivo del trabajo ha sido realizar una introducción al análisis estadístico con R con el fin de mostrar su funcionamiento. Para ello, hemos introducido los elementos básicos de R necesarios para tal fin. También hemos comprobado la versatilidad y utilidad de R, principalmente por la posibilidad de introducir paquetes creados por otros usuarios, e incluso crear nosotros mismos paquetes nuevos. Esta era su principal ventaja, otras ventajas que ya habíamos mencionado son:

- Es un software de acceso libre, cualquiera puede acceder a él de forma gratuita.
- Se puede usar en múltiples sistemas operativos, es multiplataforma.
- Es de código abierto, se pueden crear o modificar sus funciones para adaptarlo a las necesidades de cada usuario.
- Sus gráficos tienen una gran calidad y versatilidad.

Sus principales desventajas son:

- Utiliza un lenguaje de programación, por lo que su aprendizaje puede resultar bastante complejo.
- Sus mensajes de error en muchas ocasiones no son claros, dificultando localizar el origen de nuestros fallos.
- Nadie se responsabiliza de sus resultados.

A pesar de que el grado de dificultad de R es mayor que el de otros programas convencionales, el hecho de ser un software libre y el de poder trabajar con las funciones creadas por el propio usuario o por otros usuarios hacen que sus ventajas sean superiores a sus inconvenientes.

11. REFERENCIAS BIBLIOGRÁFICAS

Collatón, R. (2014): *Introducción al uso de R y R Commander para el análisis estadístico de datos en ciencias sociales.* Disponible en: https://cran.r-project.org/doc/contrib/Chicana-Introduccion_al_uso_de_R.pdf.

García, A. (2010): *Estadística básica con R*. Madrid (España), UNED.

Santana, J.S. y Mateos, E. (2014): *El arte de programar en R: un lenguaje para la estadística.* Progreso (México), Instituto Mexicano de Tecnología del Agua.

Newbold, P. (1997): *Estadística para los Negocios y la Economía.* Hertfordshire (UK), Prentice Hall.

Pech, E. (2015): "*Ventajas y Desventajas de utilizar R".* Disponible en http://rstadistica.blogspot.com/2015/10/VentajasDesventajasR.html [Consulta: 09/06/2018].

Santana, A. y Hernández, C.N. (2016): "Librerías en R". Disponible en http://www.dma.ulpgc.es/profesores/personal/stat/cursoR4ULPGC/5-librerias.html [Consulta: 17/03/2018].

"R Languaje Definitions". Disponible en: https://cran.r-project.org/doc/manuals/r-release/R-lang.html#Objects [Consulta: 03/07/2018]

"Estructuras de datos en R". Disponible en http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/ManualR/intro_estructurasdedatos.html#intro_estructuras_factores [Consulta: 14/04/2018].

Santana, A. y Hernández, C.N. (2016): "Objetos en R: Data Frames y Listas". Disponible en http://www.dma.ulpgc.es/profesores/personal/stat/cursoR4ULPGC/6g-Data_frames-Listas.html [Consulta: 15/04/2018].

"Exportando bases de datos". Disponible en https://www.r-bloggers.com/lang/uncategorized/302 [Consulta: 10/06/2018].

Santana, A. y Hernández, C.N. (2016): "Objetos en R: Factores". Disponible en http://www.dma.ulpgc.es/profesores/personal/stat/cursoR4ULPGC/6d-Factores.html [Consulta: 15/04/2018].

Santana, A. y Hernández, C.N. (2016): "Estadística Descriptiva con R". Disponible en http://www.dma.ulpgc.es/profesores/personal/stat/cursoR4ULPGC/8-estaDescriptiva.html [Consulta 10/06/2018].

Santana, A. y Hernández, C.N. (2016): "Gráficos en R: introducción". Disponible en http://www.dma.ulpgc.es/profesores/personal/stat/cursoR4ULPGC/9a-graf-Intro.html [Consulta: 28/04/2018].

Kabacoff, R.I. (2017): "R in acción". Disponible en https://www.statmethods.net/ [Consulta: 29/04/2018].

Santana, A. y Hernández, C.N. (2016): "Distribuciones de probabilidad en R". Disponible en http://www.dma.ulpgc.es/profesores/personal/stat/cursoR4ULPGC/10-distribProbabilidad.html [Consulta: 30/04/2018]

Santana, A. y Hernández, C.N. (2016): "Inferencia estadística con R". Disponible en http://www.dma.ulpgc.es/profesores/personal/stat/cursoR4ULPGC/11-inferencia-MediaVar.html [Consulta: 05/05/2018]

Lara, A.M. (2017): "Regresión y correlación". Disponible en: http://wpd.ugr.es/~bioestad/guia-r-studio/practica-3/ [Consulta: 12/05/2018].

12. ANEXOS

ANEXO 1. FUNCIONES UTILIZADAS Y USOS EMPLEADOS

- *objects*: informa de los objetos creados en orden alfabético.
- *rm*: permite eliminar objetos.
- *install.packages*: permite descargar paquetes o librerías.
- *library*: permite cargar paquetes, imprescindible previo a su uso en cada una de las sesiones.
- *c*: permite crear objetos
- *mode*: permite conocer el modo de los objetos.
- *length*: permite conocer la longitud de los objetos.
- *dim*: permite conocer la dimensión de las matrices.
- *matrix*: permite crear matrices.
- *cbind*: permite crear matrices a través de vectores por columnas.
- *rbind*: permite crear matrices a través de vectores por filas.
- *str*: permite conocer la estructura de los data frames.
- *names*: permite conocer los nombres de las variables de los data frames.
- *read.table*: permite trabajar en R con archivos externos en formato distinto al de R.
- *list*: permite crear listas.
- *factor*: permite crear factores.
- *table*: permite conocer los elementos de cada uno de los niveles de un factor.
- *as.numeric*: permite convertir un factor en una variable numérica.
- *fix*: permite editar funciones.
- *mean*: permite calcular la media.
- *median*: permite calcular la mediana.
- *quantile*: permite calcular los cuantiles.

- *fivenum*: informa sobre los cuartiles y los valores mínimo y máximo.
- *var*: permite calcular la cuasivarianza.
- *sd*: permite calcular la cuasidesviación típica.
- *summary*: produce resúmenes de resultados de distintas funciones.
- *aggregate*: permite calcular estadísticos por grupos.
- *skewness*: permite calcular el coeficiente de asimetría.
- *kurtosis*: permite calcular el coeficiente de curtosis.
- *plot*: representa un diagrama de dispersión.
- *barplot*: representa un diagrama de barras.
- *hist*: representa un histograma.
- *boxplot*: representa un diagrama de caja.
- *pie*: representa un diagrama de sectores.
- *lines*: une con líneas puntos en los gráficos.
- *abline*: introduce líneas en los gráficos.
- *points*: introduce puntos en los gráficos.
- *text*: introduce textos en los gráficos.
- *par*: permite ejecutar varios gráficos en la misma ventana de R.
- *d*: calcula el valor de la función de densidad en un punto.
- *p*: calcula el valor de la función de distribución en un punto.
- q: calcula los cuantiles.
- *r*: permite generar muestras aleatorias.
- *replicate*: permite replicar una función tantas veces como necesitemos.
- *t.test*: permite realizar inferencia estadística basada en la distribución normal.
- *var.test*: contrasta si la varianza de dos poblaciones es igual o distinta.
- *cor*: permite calcular la correlación entre dos variables.
- *lm*: permite realizar un análisis de regresión.

ANEXO 2. SCRIPT EMPLEADO PARA DEMOSTRAR EL TEOREMA CENTRAL DEL LÍMITE

C:\Users\jorge\Downloads\tcl.R - Editor R

```
set.seed(5)
muestra<-rpois(5,1.5)
mean(muestra)
muestra<-rpois(5,1.5)
mean(muestra)
media<-function(n){
muestra<-rpois(n,1.5)
media<-mean(muestra)
return(media)
}
media(5)
milmedias<-replicate(1000,media(5))
hist(milmedias,main="Teorema Central del Límite n=5")
shapiro.test(milmedias)

set.seed(5)
milmedias<-replicate(1000,media(10))
hist(milmedias,main="Teorema Central del Límite n=10")
shapiro.test(milmedias)

set.seed(5)
milmedias<-replicate(1000,media(100))
hist(milmedias,main="Teorema Central del Límite n=100")
shapiro.test(milmedias)
```