Universidad de Valladolid

Facultad de Ciencias Económicas y Empresariales

Grado en Administración y Dirección de Empresas

Introducción al análisis econométrico con R

Presentado por:

Gloria Pérez Fuentes

Tutelado por:

María Mercedes Prieto Alaiz

Valladolid, 30 de Junio de 2015

Índice

1. INTRODUCCIÓN

R es un lenguaje y entorno de programación para realizar análisis estadísticos y generar gráficos. Fue desarrollado por Ihaka y Gentleman (1996) del Departamento de Estadística de la Universidad de Auckland (Nueva Zelanda). Dispone de un almacenamiento efectivo de datos en diferentes clases de objetos. Además, cuenta con gran variedad de operadores para realizar un cálculo efectivo sobre variables indexadas, en particular, sobre matrices. En **R** se puede combinar, de manera simple, métodos de análisis estándar (regresión, análisis cluster, análisis de series temporales, etc.) con análisis desarrollados de forma adecuada para una situación específica, además de contar con grandes posibilidades gráficas. Finalmente, **R** es un lenguaje de programación que incluye condicionales, ciclos, funciones recursivas y posibilidad de entradas y salidas. Esta característica permite al usuario tanto escribir sus propios programas (librerías) como contar con los programas escritos por otros usuarios.

Se trata de una implementación del lenguaje **S**, creado por los laboratorios AT&T Bell, en torno al 1979. **S** está disponible como el programa S-PLUS comercializado por Insightful. Sin embargo, existen diferencias importantes en el diseño de **R** y **S**, sobre todo, en la interface de gráficos, A pesar de esto la mayoría de los comandos utilizados con **S** funcionan con **R**.

R se distribuye gratuitamente bajo los términos de la GNU (General Public Licence) para diferentes plataformas, como Linux, Mac y Windows. Su desarrollo y distribución son llevados a cabo por varios estadísticos conocidos como el Grupo Nuclear de Desarrollo de **R**. Además, **R** dispone de múltiples librerías creadas a partir de una amplia comunidad de usuarios que contribuyen a complementar sus capacidades básicas y extender su funcionalidad.

A pesar de la versatilidad y funcionabilidad de **R**, este no dispone de un menú principal donde el usuario pueda acceder a submenús para la lectura de datos, la ejecución de procedimientos estadísticos o la generación de gráficos, sino que estas tareas se realizan mediante un lenguaje de comandos. Esto es una desventaja con respecto a otros programas estadísticos ya conocidos, como son EViews, SPSS o Statgraphics, los cuales presentan un interfaz más amigable para un usuario que este comenzando a trabajar con ellos. No

obstante, se han desarrollado ciertas GUIs (Graphical Users Interfaces) que facilitan esta tarea, entre las que destaca **R**-Commander, desarrollada por John M. Fox en la Mcaster University de Cánada, que presenta un menú para el acceso a los comandos más habituales.

A lo largo de este trabajo se van a exponer las funciones básicas que presenta el programa y cómo se puede realizar un análisis econométrico básico con **R**. En el segundo epígrafe, se va a explicar cómo podemos instalar y descargar **R** en nuestro equipo; las funciones que presenta la pantalla principal y la consola; la opción de crear un sript y la posibilidad de instalar paquetes adicionales. En el tercer epígrafe, se verá la importancia que presentan los objetos en dicho programa y sus funcionalidades. En el cuarto epígrafe se explicarán las formas a través de las cuales podemos introducir los datos con los que se desea trabajar en **R** y cómo podemos generar nuevas variables. El epígrafe cinco está dedicado a presentar las rutinas necesarias para realizar un análisis descriptivo básico. Por último, realizaremos un modelo de regresión lineal y su validación, a través de un sencillo ejemplo. Por lo tanto, en un primer momento se explicará el funcionamiento del programa para posteriormente realizar un análisis econométrico con el objetivo de aprender el funcionamiento básico del programa y sus herramientas para poder aplicarlo en investigaciones y estudios económicos, en este caso, a través de procedimientos estadísticos.

2. INSTALACIÓN Y DESCARGA DE R

Para realizar la instalación de **R** habrá que seguir los siguientes pasos:

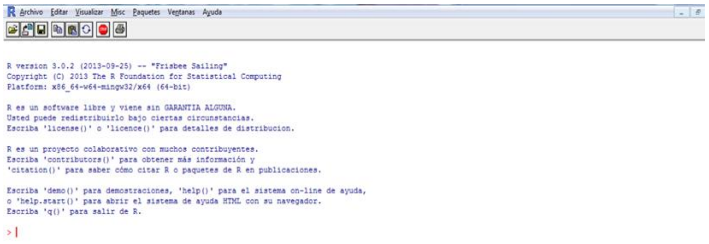
1) Accedemos a la página web de **R** en la siguiente dirección: http://cran.r-project.org. En *Download **R***, seleccionamos el tipo de sistema operativo de nuestro ordenador (Windows, Linux o Mac), en nuestro caso Windows.

2) Elegimos el subdirectorio base, donde se encuentran los ficheros principales de **R**. Seleccionamos *Download **R** 3.1.2 for Windows* que nos permite guardar el fichero ejecutable o ejecutar directamente dicho fichero.

3) Después de aceptar la ejecución del programa en nuestro equipo, indicamos el idioma de instalación y seguimos las instrucciones del asistente de instalación de **R**, aceptando las condiciones de la licencia de **R**.

Una vez instalado **R** en el ordenador podemos entrar en el programa, obteniendo la pantalla principal de **R**, que aparece en la figura 1:

Figura 1: Ventana principal de R



En la parte superior de la pantalla se encuentra la barra de herramientas que presenta los siguientes menus:

- Archivo: en esta opción podemos cargar el archivo con el que vayamos a trabajar, ya sea un sript, un área de trabajo o un histórico que hayamos guardado anteriormente. Además permite cambiar el directorio de trabajo donde deseamos guardar nuestros ficheros.
- Editar: se utiliza para realizar modificaciones en los datos como copiar, pegar, editar datos de una matriz o data.frame y limpiar la consola, además presenta una opción donde elegir las preferencias sobre la interface gráfica (formato de texto, filas, columnas etc.).
- Visualizar: sirve para indicar lo que queremos que aparezca en la pantalla principal (barra de herramientas y barra de status).
- Misc: se trata de una pestaña específica de **R**, que contiene opciones como listar los objetos o el camino de búsqueda, así como parar la computación.
- Paquetes: se trata también de una pestaña específica de **R** y con gran utilidad para el usuario, ya que permite descargar todo tipo de paquetes que el usuario necesite como explicaremos mas adelante.

- Ayuda: en esta pestaña podemos resolver cualquier duda que se nos pueda presentar sobre la utilización de **R**, además contiene enlaces a manuales e incluso a la página principal de **R**.

Debajo de esta barra de tareas, encontramos una serie de iconos que presentan algunas de las funcionalidades que se pueden realizar sin necesidad de entrar en los menus principales.

Finalmente, encontramos la consola de **R**, donde se escribirán los comandos (uno por uno) y se verán los resultados que vayamos obteniendo. Dentro de está, aparece el símbolo “>” a partir del cual introduciremos los comandos e instrucciones para trabajar.

Además **R** presenta una opción muy importante, que es la de crear un fichero sript donde podemos escribir todas las sentencias y luego ejecutarlas a la vez. Para ello seleccionamos *archivo* en la pantalla principal y a continuación *nuevo sript*, escribimos las sentencias con las que deseemos trabajar y guardamos dicho script. Cuando deseemos volver a trabajar con este sript, seleccionamos *archivo* en la pantalla principal y a continuación *abrir script*, elegimos el sript guardado con el que queremos trabajar.

Por otro lado, **R** consta de un “sistema base” y de paquetes adicionales que extienden la funcionalidad. La instalación de paquetes adicionales se realiza seleccionando *paquetes* en el menú superior y, a continuación, *instalar paquetes*. De la lista que aparece, elegimos aquel que más nos interese. Una vez instalado el paquete, es necesario cargarlo cada vez que se vaya a utilizar. Esto se realiza en el menú superior seleccionando *paquetes* y a continuación *cargar paquetes*.

Otra opción para poder trabajar con paquetes adicionales es introducir directamente en la consola o ventana de comandos las sentencias oportuna. Por ejemplo, para instalar y cargar la librería car, necesaria en el análisis de regresión, escribiremos en la ventana de comandos:

```
>install.packages(“car”)
> library (“car”)
```

(1)Nota: De aquí en adelante todas las sentencias utilizadas estarán sombreadas.

La primera función[1] se utiliza para instalar una librería y la segunda para cargarla en una sesión de trabajo. En la siguiente dirección se encuentran todos los paquetes disponibles para **R**:

http://cran.us.r-project.org/web/packages/

3. R: UN PROGRAMA ORIENTADO A OBJETOS

El programa **R** crea y manipula objetos, los cuales tienen nombre y contenido, en decir, todas las operaciones que se realizan en **R** con variables, datos, funciones, resultados etc., se guardan en forma de objetos con un nombre específico. Estos objetos pueden ser modificados o manipulados con operadores (aritméticos, lógicos, y comparativos) y funciones (que a su vez son objetos).

Los objetos también tienen atributos que indican la clase de datos que son representados por el objeto. Un objeto tiene dos atributos intrínsecos: el tipo (*mode*) y la longitud *(lenght)*. El tipo se refiere a la clase básica de los elementos en el objeto (numérico, carácter, complejo y lógico). La longitud representa el número de elementos que hay en un objeto.

Entre las funciones básicas para trabajar con objetos podemos destacar las siguientes:

- getwd(): esta función muestra el directorio en el que se guardan los ficheros creados en **R** por defecto.
- setwd(): se utiliza para cambiar el directorio en el que se guardan los ficheros. Indicando entre paréntesis el nombre de la carpeta donde se desea guardar los ficheros.
- ls(): muestra la lista de objetos que se encuentran en la memoria, los cuales han sido generados por el usuario anteriormente. Solo se muestran los nombres de los mismos. Ejemplo: “x” e “y”.

[1] Cualquier función en **R** debe tener un nombre y unos argumentos que se ponen entre paréntesis. Algunos de estos argumentos están definidos por defecto en la función, aunque el usuario puede modificarlos.

Si solo se desea que aparezcan los objetos que contengan un caracter en particular, se utiliza la función *pattern* o su abreviatura *pat*. Ejemplo:

```
> ls(pat= “z”)
```

Para que se muestren los detalles de los objetos que están en la memoria utilizamos la función:

```
> ls.str()
```

- history(): muestra un sript con todas las sentencias que el usuario ha realizado.
- save.image(): esta función se utiliza para guardar el área de trabajo que hemos creado, si posteriormente le vamos a volver a utilizar.
- load(): se utiliza para abrir un espacio de trabajo, que anteriormente hayamos guardado.

No obstante, si el usuario tiene alguna duda sobre cómo utilizar las funciones, **R** presenta ayuda en línea. Esta ayuda se encuentra disponible directamente para cada función dada, basta con introducir una *interrogación* o *help* seguida de la función de la cual deseamos información. Ejemplo:

```
>?ls o >help(ls)
```

Al indicar la ayuda, se muestra una página donde aparece, en primer lugar, el nombre de la función y del paquete donde se encuentra dicha función y, en segundo lugar, aparece la información sobre la función, donde se muestra la descripción de la misma, su utilidad, los argumentos, detalles, referencias y ejemplos.

4. INTRODUCCIÓN DE LOS DATOS

Se pueden introducir los datos directamente o leyendo ficheros desde cualquier otra aplicación, como EXCEL, SPSS, SAS, etc. A continuación se describen ambas formas de introducir los datos.

4.1 INTRODUCCIÓN DE LOS DATOS DIRECTAMENTE

Se pueden escribir los valores directamente, asignando valores a un vector. Para asignar valores a un objeto se utiliza el símbolo <-. Por ejemplo:

```
> salario<-c(1000,500,2000)
```

Si escribimos el nombre del objeto salario y pulsamos la tecla “Entrar” del teclado, veremos el contenido del vector.

Si quisiéramos seleccionar los elementos en un vector, pondremos entre corchetes el criterio de selección. Por ejemplo, si quisiéramos el segundo elemento del vector salario:

```
>salario[2]
500
```

Si quisiéramos solo los valores inferiores a 1000:

```
> salario[salario<=1000]
1000  500
```

Se distinguen principalmente tres tipos de vectores en función de los valores que lo formen: vectores numéricos, de caracteres y lógicos.

- Los vectores numéricos como su propio nombre indica están formados por números. Para crear estos vectores utilizamos la función “c” explicada anteriormente. Ejemplo:

  ```
  > salario<-c(1000,500,2000)
  ```

- Los vectores lógicos están formados por elementos que solo pueden tomar dos valores: *TRUE* (verdadero) y *FALSE* (falso). Estos valores se representan también por *F* y *T*. Aparecen al utilizar condiciones. Ejemplo:

  ```
  > salmin<-salario<600
  > salmin
  [1] FALSE  TRUE FALSE
  ```

- Los vectores de caracteres están formados por cadenas de caracteres o frases y se construyen escribiendo entre comillas la sucesión de caracteres en la función c. Ejemplo:

  ```
  > sexo<-c("m","m","h")
  ```

4.1.1 Organización de los datos en R: data.frames

R permite organizar los vectores de datos en diferentes estructuras (*data.frame, list, matrix*), la más habitual es el tipo de objetos denominado

data.frame. Un *data.frame* es similar a una matriz, pero puede contener diferentes tipos de datos (atributos, variables númericas, lógicas…).

Para crear un *data.frame* con los datos de salario y sexo escribimos la siguiente función, con la cual obtenemos el resultado que muestra la tabla 1:

```
> datos<-data.frame(sexo,salario)
> datos
```

Tabla 1: Ejemplo data.frame

	sexo	salario
1	m	1000
2	m	500
3	h	2000

Para conocer las variables que forman el *data.frame*, escribimos:

```
>names(datos)
[1] "sexo"    "salario"
```

La estructura de un *data.frame* se puede conocer con:

```
>str(datos)
'data.frame':   3 obs. of  2 variables:
 $ sexo   : Factor w/ 2 levels "h","m": 2 2 1
 $ salario: num  1000 500 2000
```

Para conocer la dimensión de un data.frame introducimos:

```
> dim(datos)
[1] 3 2
```

Se pueden seleccionar elementos de un *data.frame* incorporando el criterio de selección entre paréntesis. Por ejemplo, si queremos seleccionar los datos del de los que tienen un salario mayor a 500 o seleccionar solo los datos de las mujeres:

```
> salario[salario>=500]
[1] 1000  500 2000
```

```
> subset(datos,sexo=="m")
  sexo salario
1    m    1000
2    m     500
```

Si queremos ver un *data.frame*:

```
> data.frame(datos)
  sexo salario
1    m    1000
2    m     500
3    h     2000
```

Para seleccionar cada una de las variables de un data.frame, escribimos nombredataframe$nombredevariable. Por ejemplo,

```
>datos$sexo
[1] m m h
```

Para guardar un *data.frame* tenemos diferentes opciones, dependiendo del formato en el que lo deseamos guardar:

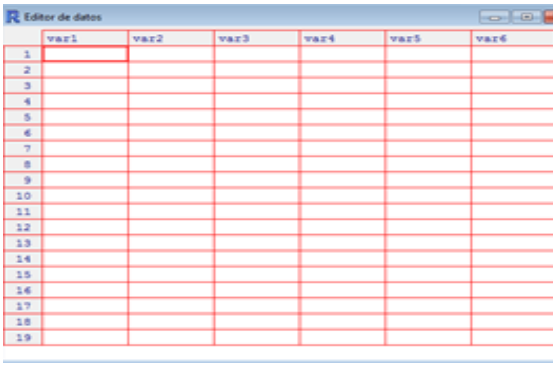
```
> save(datos,file="datos.rda") #guardar en formarto de R
> write.csv(datos,file="datos.csv") #guardar en un archivo csv
```

R dispone de su propio editor de datos, lo que nos permitirá introducir los datos y modificarlos de una manera igualmente sencilla. Para ello le indicamos:

```
>datos <-edit(as.data.frame(NULL))
```

Esto nos traslada a otra pantalla, como vemos en la figura 2, donde podemos introducir los datos:

Figura 2: Editor de datos



4.2 INTRODUCCIÓN DE LOS DATOS INDIRECTAMENTE. LECTURA DE FICHEROS EXTERNOS DE DATOS.

Otra forma de introducir datos en **R** es a través de la lectura de ficheros (ASCII, American Standard Code for Information Interchange). Para ello se pueden utilizar las siguientes funciones: *read.table* y *scan*. También puede leer ficheros en otros formatos como EXCEL, SAS, SPSS, etc. y acceder a bases de datos tipo SQL (Structured Query Language), sin embargo las funciones necesarias no están incluidas en el paquete base.

La función *read.table* lee un *data.frame*, crea un *data.frame* y se utiliza para leer datos en forma tabular. Por ejemplo:

```
>misdatos <- read.table (“datos_población.txt”,header=”T”)
```

Además **R** puede leer ficheros en otros formatos como EXCEL, para ello seleccionamos en EXCEL la hoja o el contenido de la hoja que queremos leer, y activamos la opción *copiar*, a continuación en la consola de **R** escribimos lo siguiente:

```
>misdatos=read.table(“clipboard”)
>mis datos
```

Aparecerá en la consola los datos que habíamos seleccionado en la hoja EXCEL.

También puede leer archivos de datos de SPSS, con el mismo procedimiento que EXCEL. La única diferencia que presenta con respecto a EXCEL es que al copiarse los datos en la consola de **R**, no se conservan los nombres originales de las variables, SPSS les asigna nuevos nombres.

Otra forma de leer archivos es a través de la función read.csv(). Se utiliza para leer archivos con el formato CSV (comma-separated values), es decir, archivos delimitados por comas. Ejemplo:

```
>misdatos<-read.csv (file= “datos.csv”)
```

4.3 GENERACIÓN DE NUEVAS VARIABLES

Se pueden generar gran cantidad de variables en **R**, como podemos observar en los siguientes ejemplos:

- Generar dos vectores “x” e “y”:

```
> x<-c(1,6,3,4,2)
> y<-c(6,1,8,5,7)
```

- Generar la suma de los dos vectores:

```
> sum(x,y)
[1] 43
```

- Sumar por separado los valores de los vectores:

```
> x+y
[1]  7  7 11  9  9
```

- Generar el logaritmo neperiano:

```
> log(x)
[1] 0.0000000 1.7917595 1.0986123 1.3862944 0.6931472
```

- Generar una variable ficticia[2] de la variable sexo:

```
> dummy(as.character(sexo))
     as.character(sexo)h as.character(sexo)m
[1,]                   0                   1
[2,]                   0                   1
[3,]                   1                   0
> dummy(sexo[1:2])
     sexom
[1,]     1
[2,]     1
```

También se pueden generar secuencias regulares de datos. Así si quisiéramos asignar a un vector la secuencia de 1 a 5, se haría de la siguiente forma:

```
> x < - 1:5
```

[2] Es necesario instalar el paquete *dummies* y a continuación cárgalo para poder generar la ficticia.

```
> x
[1] 1 2 3 4 5
```

Además se puede generar una secuencia, de tal forma que la diferencia entre dos valores consecutivos sea de un determinado valor. Para ello, se utiliza la función *seq*. En esta función, el primer argumento indica el principio de la secuencia, el segundo el final y el tercero el incremento que se debe usar para generar la secuencia. Por ejemplo:

```
> seq(1,5,0.5)
[1]  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5  5.0
```

Para crear un vector con elementos idénticos utilizamos la función *rep*. Por ejemplo:

```
> rep(1,10)
[1] 1 1 1 1 1 1 1 1 1 1
```

Para generar números aleatorios usamos: *rFuncionDistribucion* (con sus parámetros). Por ejemplo, para generar una muestra de una normal estandar utilizamos la función *rnorm(n)* o *rnorm(n, mean=0, sd=1)*. Así, si queremos generar una muestra de tamaño 5 de una normal(0,1), escribimos:

```
> rnorm(5)
[1]  1.4362230 -1.0712306  0.2502178 -0.5834647 -1.4662709
```

5. ANÁLISIS DESCRIPTIVO BÁSICO

Para realizar un análisis descriptivo básico en **R** se utiliza la función *summary* que nos permite obtener el valor mínimo (*min*) y el valor máximo (*max*), la media (*mean*), la mediana (median) y el primer y tercer cuartil (*1st Qu., 3rd Qu*.) en el objeto de estudio.

Otra opción de realizar un análisis descriptivo en **R** puede ser a través del paquete *fBasics* ya que es de gran utilidad. Para verlo de forma práctica vamos

a utilizar los datos (datos de corte transversal) que se refieren a la **producción de metal de 27 regiones del sector metalúrgico a través de la medida de trabajo por establecimiento (L) y de la medida de capital por establecimiento (K), de Estados Unidos aportado por el SIC (Standard Industrial Classification Code 33). Estos datos se encuentran recogidos en el CD que contiene el libro "Hands-on Intermediate econometrics using R" (D Vinod, 2008).** Una vez introducido los datos, obtenemos los logaritmos neperianos de los datos, cargamos el paquete ya instalado e introducimos el siguiente comando para generar una matriz con los datos de las variables a utilizar:

```
> descrip<-cbind(Ly,LL,LK)
```

A continuación introducimos la función con la cual vamos a obtener el análisis descriptivo que se muestra en la tabla 1:

```
> basicStats(descrip)
```

Tabla 2: Análisis descriptivo

	Ly	LL	LK
Nobs	27.000000	27.000000	27.000000
NAs	0.000000	0.000000	0.000000
Minimum	6.384941	5.634754	4.919981
Maximum	9.195142	9.546066	7.355532
1. Quartile	6.935645	6.736932	5.360868
3. Quartile	7.928982	8.063829	6.233483
Mean	7.443631	7.445922	5.763652
Median	7.378996	7.436605	5.560335
Sum	200.978045	201.039905	155.618606
SE Mean	0.146484	0.186384	0.126293
LCL Mean	7.142529	7.062804	5.504052
UCL Mean	7.744733	7.829041	6.023252
Variance	0.579354	0.937957	0.430651
Stdev	0.761153	0.968482	0.656240
Skewness	0.613768	0.231518	0.701836
Kurtosis	-0.522245	-0.677601	-0.495534

Además, vamos a calcular el coeficiente de variación. Para ello programamos una función en **R**, teniendo en cuenta que este coeficiente se obtiene calculando el valor absoluto de la desviación típica entre la media.

En primer lugar, introducimos el siguiente comando en la consola:

```
> fix(coef.var)
```

En segundo lugar, aparecerá una nueva pantalla en la que introducimos la nueva función, como se muestra a continuación:

```
function (x)
{m<-mean(x)
d<-var(x)^0.5
coef.var<-abs(d/m)
return(coef.var)}
```

Una vez creada la nueva función, ya podemos utilizarla para calcular el coeficiente de variación de las variables:

```
> coef.var(Ly)
[1] 0.1022556
> coef.var(LL)
[1] 0.1138583
> coef.var(LK)
[1] 0.1300688
```

Para ver el grado de relación entre las variables, podemos utilizar el coeficiente de correlación de Pearson. En R introducimos el siguiente comando:

```
> cor(Ly,LL)
[1] 0.9475284
> cor(Ly,LK)
[1] 0.9431172
> cor(LK,LL)
[1] 0.8945586
```

5.1 REPRESENTACIONES GRÁFICAS

R presenta una amplia variedad de posibilidades gráficas, que se pueden utilizar para mostrar gran cantidad de gráficos estadísticos pero también para construir nuevos tipos de gráficos, todos ellos con numerosas opciones que hacen que tengan una gran flexibilidad en la creación de gráficos.

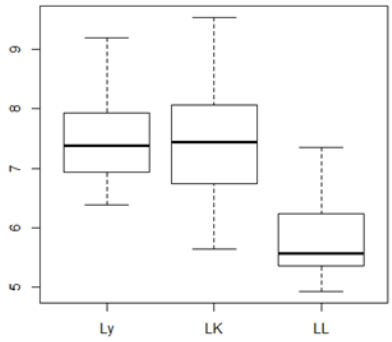
Vamos a ver algunos ejemplos con los datos de la producción de metales utilizados anteriormente en el análisis descriptivo:

- Si deseamos realizar un diagrama de caja o boxplot introducimos:

```
> boxplot(descrip)
```

Obtenemos el siguiente gráfico:

Gráfico 1: Diagrama de caja



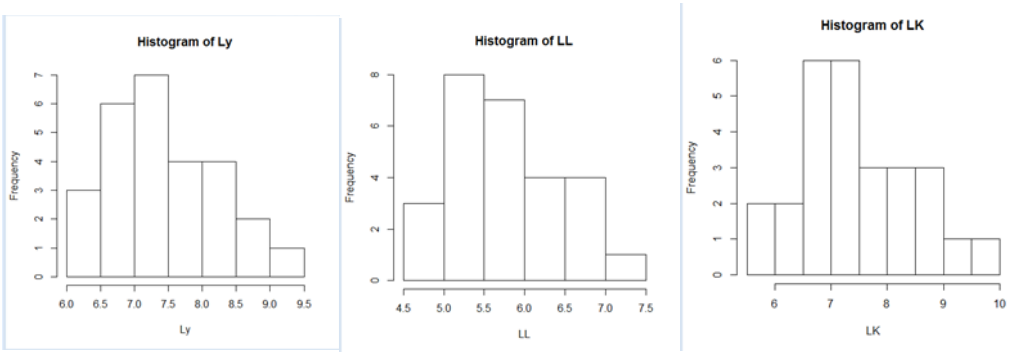
Según nos muestra el gráfico 1, vemos que la variable capital es la que presenta una distribución más simétrica. También observamos que ninguna de las variables presenta valores atípicos.

- Para realizar un histograma indicamos:

```
> hist(Ly)
>hist(LL)
>hist(LK)
```

Obtenemos los siguientes gráficos:

Gráfico 2: Histogramas

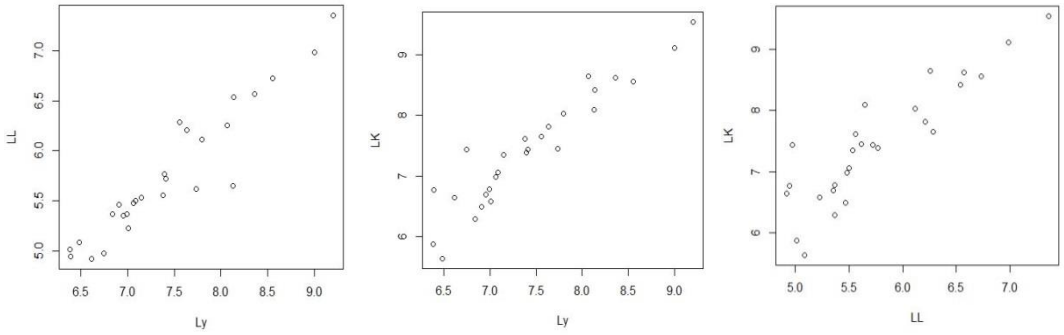


- Para realizar un diagrama de dispersión indicamos:

```
> plot(Ly,LL)
> plot(Ly,LK)
> plot(LL,LK)
```

Obtenemos los siguientes gráficos:

Gráfico 3: Diagramas de dispersión



6. MODELO DE REGRESIÓN LINEAL

En este apartado vamos a presentar las rutinas para realizar un análisis de regresión básico. Además, será necesario instalar algún paquete, entre los que destaca el paquete *car*, ya que presenta muchas funciones necesarias para el análisis de regresión.

6.1 ESPECIFICACIÓN Y ESTIMACIÓN DEL MODELO

La finalidad del modelo de regresión lineal es explicar el comportamiento de una variable (variable dependiente) en función de otras (variables explicativas). Para ello utilizaremos los **datos de la producción de metal del sector metalúrgico de Estados Unidos**, empleados en el análisis descriptivo anteriormente visto.

Consideramos que la producción depende del capital y del trabajo, según una función de producción Cobb Douglas, donde "y" es la producción, "L" es el trabajo y "K" el capital. Tenemos como parámetros desconocidos A, α y β.

$y=f(K, L)= AK^{\alpha}L^{\beta}$

Linealizamos esta función, trabajando con logaritmos neperianos:

Ly: Logaritmo de la variable producción

LL: Logaritmo de la variable trabajo

LK: Logaritmo de la variable capital

El modelo con el que trabajaremos será:

$Ly= \beta_0+\beta_1 LL+\beta_2 LK+\varepsilon_i$

El comando *lm* (linear models) se utiliza para realizar la regresión por mínimos cuadrados ordinarios en **R**. El primer argumento de este comando es la variable dependiente, seguido del símbolo "~" y de las variables independientes separadas estas últimas por el símbolo "+". Finalmente, es interesante crear un objeto con los resultados de la regresión:

```
> prod1<-lm(Ly~LL+LK)
```

El objeto prod1 contiene los resultados de la regresión y lo podemos utilizar en otros análisis.

Para obtener los principales resultados de la regresión introducimos en la consola:

```
> summary(prod1)
```

Obtenemos los resultados que se muestra en la siguiente figura:

Figura 3: Resultados de la regresión

```
Call:
lm(formula = Ly ~ LL + LK)
Residuals:
     Min       1Q       Median       3Q      Max
-0.30385 -0.10119 -0.01819   0.05582  0.50559
Coefficients:
             Estimate    Std. Error   t value    Pr(>|t|)
(Intercept)  1.17064     0.32678      3.582      0.00150 **
LL           0.60300     0.12595      4.787      7.13e-05 ***
LK           0.37571     0.08535      4.402      0.00019 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 0.1884 on 24 degrees of freedom
Multiple R-squared:  0.9435,    Adjusted R-squared:  0.9388
F-statistic: 200.2 on 2 and 24 DF,  p-value: 1.067e-15
```

En primer lugar, nos muestra información sobre las características descriptivas básicas: los valores mínimo y máximo, la mediana y los cuartiles. A continuación, aparece la estimación de los coeficientes asociados a cada una de las variables explicativas de la función (*estimate*), el error estándar de cada variable (*Std. Error*), el estadístico t de significación individual (*t value*) y el p-valor asociado al mismo (*Pr(>|t|).* Finalmente, observamos el coeficiente de determinación (*R-squared y Adjusted R-squared*), el estadístico F (*F-statistic*) y la probabilidad del mismo (*p-value*).

Como podemos observar, la salida que nos muestra **R** es menos completa que la que podemos obtener con otros programas estadísticos como pueden ser EViews o SPSS.

6.2 ANÁLISIS DE LA BONDAD DEL AJUSTE

Este análisis es utilizado para ver que parte de la variabilidad de la variable dependiente es explicada por los regresores y que parte es explicada por los residuos. Una de las formas de realizarlo en **R** es a través de la función *anova* con la cual obtenemos la suma de los cuadrados explicada por los regresores y por los residuos (*Sum sq*) y la media cuadrática o varianza explicada por cada

una de las variables y los residuos (*Mean sq*) que se obtiene dividiendo la suma de cuadrados explicada entre los grados de libertad (*Df*).

```
> anova(prod1)
```

Los resultados se muestran en la figura 4:

Figura 4: Análisis de la varianza

```
Analysis of Variance Table
Response: Ly
           Df  Sum Sq  Mean Sq F value    Pr(>F)
LL          1 13.5239  13.5239 381.118 3.098e-16 ***
LK          1  0.6877   0.6877  19.379 0.0001899 ***
Residuals  24  0.8516   0.0355
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Como vemos en la figura 4, 13.5239 es la suma de cuadrados de la regresión cuando se incorpora la variable trabajo y 0.6877 es el incremento de la suma de cuadrados de la regresión cuando se incorpora la variable capital. Por lo tanto, la suma de cuadrados explicada es 14.2116 y la suma de cuadrados de los residuos es 0.8516. La variabilidad total de los datos será 15.0632 puntos (SCT=13.5239+0.6877+0.8516). La proporción de variabilidad explicada por los dos factores (R^2) será 94.35%, esto es la bondad del ajuste del modelo.

6.3 CONTRASTES DE HIPOTESIS

Supongamos que queremos realizar una restricción lineal exacta sobre los parámetros, los cuales se pueden escribir en notación matricial de la siguiente manera: Rβ= r, donde R es una matriz de orden (H*(K+1)) y rango H, que recoge los coeficientes de cada combinación lineal y r es un vector de H elementos. Por ejemplo:

H_0: $\beta_1 = \beta_2$

H_1: $\beta_1 \neq \beta_2$

Bajo hipótesis clásicas, el estadístico que permite contrastar restricciones como la anterior es el siguiente:

$$\frac{SCR_R - SCR_S}{S^2 * H} \rightarrow F^H_{T-K-1}$$

Para realizar este contraste en **R**, necesitamos cargar el paquete *car* y a continuación introducimos:

```
> linear.hypothesis(prod1,"LL+LK=1")
```

Obtenemos los resultados que se muestran en la figura 5:

Figura 5: Contraste de hipótesis

```
Linear hypothesis test
Hypothesis:
LL  + LK = 1

Model 1: restricted model
Model 2: Ly ~ LL + LK

  Res.Df     RSS  Df Sum of Sq      F   Pr(>F)
1     25   0.85574
2     24   0.85163  1  0.0041075 0.1158 0.7366
```

6.4 PREDICCIÓN

Una predicción es un pronóstico sobre el valor que puede tomar la variable endógena para unos valores dados de las variables explicativas fuera de las observaciones muestrales (Carrascal, Gonzalez y Rodriguez, 2001)

Para hacer una predicción correcta, el modelo propuesto debe ajustarse bien a los datos, debemos tener un conocimiento preciso de los regresores en la observación de la predicción y tenemos que considerar el modelo como válido en la nueva observación. Según el valor a predecir se distinguen dos tipos: predicción de un valor medio y predicción de un valor individual.

Por ejemplo, si queremos predecir (predicción de un valor medio) la producción de metales para valores del trabajo y capital entre 7 y 8. Creamos un fichero de datos con estas nuevas variables y utilizamos el comando *predict*:

```
>pred.frame <- data.frame(LL= seq(7,8 , by =0.5),LK=seq(7,8 , by =0.5))
```

Una vez introducido la sentencia anterior, obtenemos la siguiente tabla:

```
> pc <- predict(prod1, int="c", newdata=pred.frame)
> pc
```

Tabla 3: Predicción

	fit	lwr	upr
1	9.00032	8.498741	9.501899

Como observamos se produce una matriz de predicciones donde lwr y upr muestran los limites inferior y superior del intervalo para la predicción realizada y fit es el valor medio del intervalo.

7. VALIDACIÓN DEL MODELO

Todo modelo de regresión lineal debe cumplir una serie de hipótesis para ser considerado como válido. Algunas de las hipótesis recaen sobre perturbaciones y éstas son desconocidas, por lo que se necesita calcular los residuos, es decir, la diferencia entre los valores observados de la variable dependiente y los valores estimados, para validar las hipótesis del modelo. Los residuos por MCO (e_i) son los más utilizados, ya que hacen mínima la suma de los cuadrados de los residuos. En **R** se calcula de la siguiente manera:

```
e1<-resid(prod1)
```

7.1 NORMALIDAD

Para analizar la normalidad de nuestro modelo se puede hacer mediante métodos gráficos o mediante contrastes de hipótesis.

- Mediante contrastes de hipótesis, entre los que destacan Kolmogorov-Smirnov y Shapiro-Wilks. Por ejemplo, para calcular el contraste de Kolmogorov-Smirknov en **R** introducimos la siguiente función:

```
> ks.test(e1,"pnorm")
```

Obtenemos el siguiente resultado que muestra la figura 6:

Figura 6: Test Kolmogorov-Smirnov

```
One-sample Kolmogorov-Smirnov test
data:  e1
D = 0.3806, p-value = 0.0005016
alternative hypothesis: two-sided
```

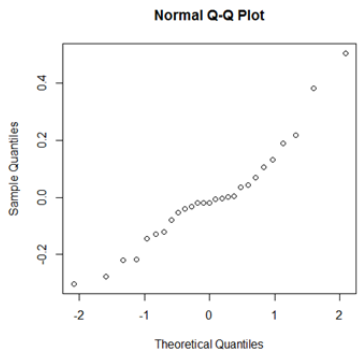
- Mediante métodos gráficos, como pueden ser el gráfico probabilístico normal (p-p plot) y el gráfico cuantil (q-q plot).

  Por ejemplo, para realizar el gráfico cuantil y comparar los residuos con los cuantiles de una normal, introducimos:

```
>qqnorm(e1)
```

Obtenemos el siguiente gráfico:

Gráfico 4: q-q plot



7.2 LINEALIDAD

El análisis de linealidad, al igual que el de normalidad, se puede realizar a través de procedimientos gráficos o mediante un contraste de hipótesis.

- Contrastes de hipótesis: se puede detectar a través del contraste de Ramsey RESET, ya que detecta cualquier falta de especificación en el modelo, para ello realizamos la misma regresión pero incluyendo las potencias de la variable endógena ajustada ($\hat{y}^2, \hat{y}^3 \ldots$) y analizamos la significación conjunta de dichas potencias. Para calcular este contraste en **R** es necesario instalar y cargar el paquete *lmtest* y posteriormente introducir:

```
> resettest(Ly~LL+LK, power=2, type="regressor")
```

Figura 7: Contraste de Ramsey RESET

```
        RESET test

data:  Ly ~ LL + LK
RESET = 2.4534, df1 = 2, df2 = 22, p-value = 0.1092
```

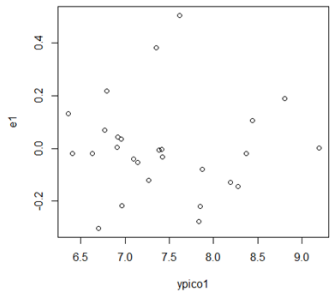
- Procedimientos gráficos, a través de un diagrama de dispersión de los residuos mínimo cuadráticos y los valores ajustados, para ello necesitamos crear los valores ajustados de la función de producción, $\hat{y}$, en **R**:

```
>ypico1<-fitted(prod1)
```

Una vez hecho esto ya podemos realizar la representación gráfica, como se muestra en el gráfico 5:

```
>plot(ypico1,e1)
```

Gráfico 5: Diagrama de dispersión



7.3 HETEROSCEDASTICIDAD

La heteroscedasticidad se produce cuando las varianzas de las perturbaciones del modelo son desiguales. Esto supone que la matriz de varianzas y covarianzas de las perturbaciones presenta varianzas distintas en los elementos de la diagonal principal, es decir no son constantes. Para detectar la presencia de heteroscedasticidad en nuestro modelo vamos a utilizar el contraste asintótico de White, que se basa en analizar la significación de una regresión auxiliar que explica los residuos al cuadrado de la regresión inicial a partir de los regresores, los regresores al cuadrado y los productos de los mismos. La hipótesis a contrastar es:

H_0: Homoscedasticidad

H_1: Heteroscedasticidad

El estadístico de contraste se lleva a cabo obteniendo NR^2_{aux} , donde R^2_{aux}.es el coeficiente de determinación de la región auxiliar. Este estadístico se distribuye asintóticamente, bajo la hipótesis nula, como una $X^2_{k(aux)}$, donde k(aux) es el número de regresores de la regresión auxiliar.

A continuación pasamos a analizar la heteroscedasticidad en, para ello necesitamos instalar y cargar los paquetes *het.test* y *var*:

```
> dataset<-data.frame(Ly, LL,LK)
> reg11<-VAR(dataset,p=1)
> whites.htest(reg11)
```

Obtenemos los siguientes resultados, para el contraste que no contiene términos cruzados:

Figura 8: Contraste de White

```
White's Test for Heteroskedasticity:
====================================
 No Cross Terms
 H0: Homoskedasticity
 H1: Heteroskedasticity
 Test Statistic:
 41.8644
Degrees of Freedom:
 36
 P-value:
 0.2313
```

7.4 AUTOCORRELACIÓN

La autocorrelación se produce cuando las perturbaciones del modelo presentan correlaciones entre ellas. Esto supone que la matriz de varianzas y covarianzas de las perturbaciones presenta valores distintos de cero en los elementos que están fuera de la diagonal principal. Para analizar si existe autocorrelación en nuestro modelo vamos a utilizar el test de Durbin Watson que contrasta si existe relación o no en las perturbaciones.

Se suelen asumir diferentes esquemas que reflejan la relación lineal entre las perturbaciones. Uno de los más utilizados es el esquema autoregresivo de orden 1, que supone que:

$\varepsilon_t = \rho\varepsilon_{t-1} + u_t$

El contraste a realizar es el siguiente:

H_0: $\rho = 0$

H_1: $\rho \neq 0$

El estadístico de Durbin Watson es $d \approx 2(1-\hat{\rho})$ y apoyaremos la hipótesis nula cuando $\hat{\rho}$ sea próximo a cero y d próximo a dos.

Para conocer el valor muestral en **R**, es necesario cargar el paquete *car*, una vez cargado insertamos la siguiente función:

```
> durbinWatsonTest(prod1)
```

Obtenemos los siguientes resultados:

Figura 9: Contraste de Durbin Watson

```
 lag Autocorrelation D-W Statistic p-value
   1      0.04663243      1.885989   0.682
 Alternative hypothesis: rho != 0
```

7.5 VALORES ATÍPICOS

Los valores atípicos son una observación o subconjunto de observaciones que son inconsistentes con respecto a la relación lineal que siguen la mayoría de las observaciones.

Para analizar la presencia de atípicos en **R**, podemos introducir la siguiente función:

```
> influence.measures(prod1)
```

Obtenemos los siguientes resultados mostrados en la tabla 4 (ejemplo de las 10 primeras observaciones):

Tabla 4: Valores atípicos

	dfb.1_	dfb.LL	dfb.LK	dffit	cov.r	cook.d	hat inf
1	0.188504	0.262295	-3.86e-01	0.46460	1.398	7.30e-02	0.2518 *
2	0.052065	0.082809	-1.17e-01	0.14926	1.284	7.70e-03	0.1318
3	0.220847	-0.009566	-8.90e-02	0.32436	1.015	3.44e-02	0.0677
4	-0.027923	0.005130	5.35e-03	-0.059701.	176	1.24e-03	0.0434
5	0.033247	0.006831	-2.23e-02	0.05893	1.201	1.21e-03	0.0607
6	0.175401	-0.101322	1.63e-02	-0.25289	1.151	2.16e-02	0.0909
7	0.035596	0.000560	-2.38e-02	-0.09957	1.169	3.42e-03	0.0510
8	0.022646	-0.006385	-5.50e-03	-0.03282	1.258	3.75e-04	0.0980
9	-0.003234	-0.004821	5.22e-03	-0.03457	1.176	4.15e-04	0.0379
10	-0.063171	0.078442	-6.45e-02	-0.15232	1.132	7.92e-03	0.0509

Esta tabla, es sencilla de analizar, ya que el programa señala con un asterisco el posible valor atípico. Muestra la distancia de Cook y los elementos de la diagonal principal de la matriz sombrero, fundamentales para detectar la presencia de atípicos.

Como se observa en la tabla de las 10 primeras observaciones, la primera observación puede ser un posible valor influyente, un valor muy alejado del resto de los valores de los regresores. **R** señala con un asterisco los valores que presentan un valor en la diagonal principal de la matriz sombrero muy alejado del valor medio.

7.6 MULTICOLINEALIDAD

La multicolinealidad aparece cuando existe algún tipo de relación líneal entre las variables del modelo. Se distinguen dos tipos de multicolinealidad:

- Multicolinealidad perfecta: relación lineal.
- Multicolinealidad imperfecta: alta correlación lineal.

Una de las formas de detectar la presencia de multicolinealidad es a través del factor de inflación de la varianza, que recoge el aumento que experimenta la varianza al incluir en el modelo el resto de variables explicativas:

$FIV_{i} = 1/ (1- R^{2}_{Xi \cdot X1,\ldots Xi-1,\ Xi+1,\ldots Xk})$

En el programa **R** introducimos la siguiente función, para obtener los valores del FIV para cada variable:

```
> vif(prod1)
```

Los resultados se muestran en la siguiente tabla:

Tabla 5: Factor de Inflación de la Varianza

LL	LK
5.005883	5.005883

Como se observa en la tabla 4, los valores del FIV son menores que 10, es decir, podemos asumir que no existe alta colinealidad.

8. CONCLUSIONES

A lo largo de este trabajo se ha mostrado el funcionamiento básico del programa **R** y sus herramientas, para posteriormente aplicarlo en la realización de estudios econométricos sencillos.

Después de realizar el análisis sobre este programa, sus utilidades y haber trabajado con él, se ha comprobado la gran cantidad de funcionalidades que presenta este software para realizar análisis estadísticos y gráficos. Si bien, se puede decir que presenta ciertas limitaciones con respecto a otros programas estadísticos donde no es necesario el lenguaje de comandos y su funcionamiento es más sencillo. Sin embargo, en **R**, a diferencia de en otros programas, el usuario tiene la posibilidad de crear sus propias librerías y compartirlas con el resto de usuarios, esto se convierte en una gran ventaja. Además, al tratarse de un software gratuito muchos usuarios se decantarán por él.

Gracias al ejemplo utilizado, se ha podido realizar por primera vez un análisis econométrico en **R**, utilizando las funciones disponibles en el programa y descargando los paquetes necesarios. Esto ha servido para comprobar que es un programa que se puede adaptar a multitud de estudios tanto para aplicaciones en economía como para muchos otros ámbitos debido a la gran cantidad de funciones que presenta. No obstante, puede resultar más complicado de realizar que en otros programas como SPSS o EViews.

Con todo esto, se puede decir que a pesar de las dificultades que conlleva la utilización de **R,** este tiene aplicaciones prácticas de gran utilidad para las empresas.

9. REFERENCIAS BIBLIOGRÁFICAS

ARRIAZA, A. J.; F. FERNÁNDEZ, F.; LÓPEZ, M. A.; MUÑOZ, M.; PÉREZ, S.; SÁNCHEZ, A. (2008): "*Estadística Básica con R y R–Commander"*. Cádiz, Servicio de Publicaciones de la Universidad de Cádiz. Disponible en: http://gsyc.escet.urjc.es/~herraiz/ebrcmdr.pdf

CARRASCAL, U.; GONZÁLEZ, Y.; RODRÍGUEZ, B. (2001): *"Análisis Econométrico con EViews"*. Madrid, RA-MA.

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

CORREA, J. C.; GONZÁLEZ, N. (2002): *"Gráficos estadísticos con R".* Disponible en: http://cran.r-project.org/doc/contrib/grafi3.pdf

CRAWLEY, M. J. (2007): *"The R Book".* England, John Wiley & Sons, Ltd. Disponible en: http://www.kharms.biology.lsu.edu/CrawleyMJ_TheRBook.pdf

DÍAZ-URIARTE, R. (2003): *"Introducción al uso y programación del sistema estadístico R"*. Disponible en: http://cran.r-project.org/doc/contrib/curso-R.Diaz-Uriarte.pdf

D VINOD, H. (2008): *"Hands-on intermediate econometrics using R".* Singapore, World Scientific Publishing Co. Pte. Ltd.

FARAWAY, J. J. (2002): "*Practical Regression and Anova using R"*. Disponible en: http://cran.r-project.org/doc/contrib/Faraway-PRA.pdf

FEBRERO, M.; GALEANO, P.; GONZÁLEZ, J.; PATEIRO, B. (2008): *"Prácticas de estadística en R".* Disponible en:
http://eio.usc.es/pub/pateiro/files/pubdocentepracticasestadistica.pdf

FOX, J M. (2005): *"The R Commander: A Basic-Statistics Graphical User Interface to R"*. Disponible en: http://www.jstatsoft.org/v14/i09/paper

IHAKA, R. & GENTLEMAN, R. (1996). R: a language for data analysis and graphics. *Journal of Computational and Graphical Statistics* 5:299-314.

LÓPEZ, E.; MARTÍNEZ, J. (2015): *"R desde el principio: curso cero de R".* Disponible en: http://emilio.lcano.com/pub/cero/R-desde-el-principio-curso-cero-V1.0.pdf

PARADIS, E. (2003): "*R for Begginers"*. Disponible en: http://cran.r-project.org/doc/contrib/rdebuts_es.pdf

SÁEZ, A. J. (2009): "*Métodos estadísticos con R y R Commander"*. Disponible en: http://web.udl.es/Biomath/Bioestadistica/R/Introduccion/RRCmdrv20.pdf

SANTANA, A. (2012): *"Introducción al entorno estadístico R".* Disponible en: https://gradocienciasdelmar.files.wordpress.com/2012/03/introduccion_r.pdf

TUSELL, F. (2011): *"Análisis de Regresión. Introducción Teórica y Práctica basada en R"*. Disponible en: http://www.et.bs.ehu.es/~etptupaf/nuevo/ficheros/estad3/nreg1.pdf

VENABLES, W. N.; SMITH D. M.; THE R CORE TEAM. (2015): *"An Introduction to R"*. Disponible en: http://cran.r-project.org/doc/manuals/R-intro.pdf

10. ANEXOS

ANEXO 1: SENTENCIAS UTILIZADAS EN EL ANÁLISIS DESCRIPTIVO BÁSICO.

- Análisis descriptivo

```
descrip<-cbind(Ly,LL,LK)
basicStats(descrip)
```

- Coeficiente de variación

```
fix(coef.var)
coef.var(Ly)
coef.var(LL)
coef.var(LK)
```

- Coeficiente de correlación de Pearson

```
cor(Ly,LL)
cor(Ly,LK)
cor(LK,LL)
```

- Representaciones gráficas:

```
boxplot(descrip)
hist(Ly)
hist(LL)
hist(LK)
plot(Ly,LL)
plot(Ly,LK)
plot(LL,LK)
```

ANEXO 2: SENTENCIAS UTILIZADAS EN EL MODELO DE REGRESIÓN LINEAL.

- Estimación de un modelo de regresión lineal simple

```
Ly=log(met[,1])
LL=log(met[,2])
LK=log(met[,3])
prod1<-lm(Ly~LL+LK)
summary(prod1)
```

- Análisis de la bondad del ajuste

```
anova(prod1)
```

- Contrastes de hipótesis

```
linear.hypothesis(prod1,"LL+LK=1")
```

- Predicción

```
pred.frame <- data.frame(LL= seq(7,8 , by =0.1),LK=seq(7,8 , by =0.1))
pc <- predict(prod1, int="c", newdata=pred.frame)
```

- Residuos y valores ajustados

```
e1<-resid(prod1)
ypico1<-fitted(prod1)
```

- Normalidad

```
ks.test(e1,"pnorm")
library(car)
qqnorm(e1)
```

- Linealidad

```
library(lmtest)
resettest(Ly~LL+LK, power=2, type="regressor")
ypico1<-fitted(prod1)
plot(ypico1,e1)
```

- Heteroscedasticidad

```
library(het.test)
library(var)
dataset<-data.frame(Ly,LL,LK)
reg11<-VAR(dataset,p=1)
whites.htest(reg11)
```

- Autocorrelacion

```
library(car)
durbinWatsonTest(prod1)
```

- Valores atípicos

```
influence.measures(prod1)
```

- Multicolinealidad

```
vif(prod1)
```