Monitoria econometria 2

Camilo Forero - Jhan Andrade - Germán Rodriguez

16/9/2020

Contents

Para las personas que quieran reproducir el documento markdown (extención .md), con el que se genero este PDF, deben tener las bases de datos en el mismo directorio de trabajo donde se encuentre el archivo markdown.

1. Importación y tipo de datos

R studio permite cargar diversos formatos de bases de datos en los cuales se destacan:

- **Archivos CSV:** Archivos delimitados por comas o punto y coma
- **Archivos txt:** Archivos delimitados por tabulaciones
- **Archivos dta:** Bases de datos de STATA
- **Archivos XLS y XLSX**: Archivos de excel

Para importar bases de datos en RStudio es importante en primer lugar configurar el directorio de trabajo

1.1 Configuración del directorio de trabajo

- **Obtener el directorio de trabajo:**

```
WD<-getwd()
WD
```

```
## [1] "/home/germankux/Documents/GitHub/semestre5/econometria_2_personal/monitorias/2020_2/monitoria2_
```

- **Establecer el directorio de trabajo:**

Dentro de las comillas de debe poner la ruta de la carpeta en donde se encuentran guardadas las bases de datos:

Un ejemplo de la estructura del código para usar el comando setwd en RSstudio sería:

setwd("C:/Users/FORERO/Documents/Bases de datos")

```
setwd(WD)
```

Especificando otro directorio de trabajo manualmente:

```
setwd("~/Documents/GitHub/semestre5/econometria_2_personal/monitorias/2020_2/monitoria2_estadistica_en_
```

Una vez se configure el directorio de trabajo, se podrán llamar los archivos por el nombre y formato en el que esta guardado. Es importante indicarle a RStudio en que tipo de formato se encuentra la base de datos, por ejemplo:

- **"Nombre_Base.xls"** para archivos excel
- **"Nombre_Base.xlsx"** para archivos excel
- **"Nombre_Base.csv"** para archivos excel
- **"Nombre_Base.dta"** para archivos excel

Hay que usar diferentes funciones dependiendo del formato en el que se encuentre la base de datos que se va a importar. A continuación, se encuentra las diferentes funciones para importar bases de datos.

1.2 Importar datos en formato csv

```
titanic=read.csv("titanic.csv", sep = ",")
```

Para ver la base de datos usamos el código *View(titanic)*.

Para trabajar con el nombre de los datos de las variables sin necesidad de usar el operador atómico **$**

```
attach(titanic)
```

Para visualizar un resumen de la base de datos importada:

```
summary(titanic)
```

```
##        X                                   Name      PClass         Age
##  Min.   :   1   Carlsson, Mr Frans Olof     :   2   *  :  1   Min.   : 0.17
##  1st Qu.: 329   Connolly, Miss Kate         :   2   1st:322   1st Qu.:21.00
##  Median : 657   Kelly, Mr James             :   2   2nd:279   Median :28.00
##  Mean   : 657   Abbing, Mr Anthony          :   1   3rd:711   Mean   :30.40
##  3rd Qu.: 985   Abbott, Master Eugene Joseph:   1             3rd Qu.:39.00
##  Max.   :1313   Abbott, Mr Rossmore Edward  :   1             Max.   :71.00
##                 (Other)                     :1304             NA's   :557
##      Sex         Survived
##  female:462   Min.   :0.0000
##  male  :851   1st Qu.:0.0000
##                Median :0.0000
##                Mean   :0.3427
##                3rd Qu.:1.0000
##                Max.   :1.0000
##
```

1.3 Importar datos de excel

Para importar datos en formato "xls" o "xlxs" es importante descargar el paquete *readxl*

```
#install.packages("readxl")
library(readxl)
```

Vamos a cargar la base de datos llamada *Datos1 - Script 2* en formato "xlsx"

```
Excel<-read_excel("Datos1 - Script 2.xlsx")
```

Los siguientes códigos nos permiten:

- *Excel*: visualizar la base de datos en la consola
- *attach(Excel)*: trabajar con el nombre de las variables sin necesidad de trabajar con el operador atómico *$*
- *View(Excel)*: Visualizar la base de datos en RStudio en una nueva ventana como si fuera una tabla de excel

1.4 Importar datos provenientes de STATA

Para que RStudio pueda cargar bases de datos en STATA (ya sea que se encuentra guardadas localmente como documentos o cargadas en Internet),es importante instalar el paquete foreign

```
#install.packages("foreign")
library(foreign)
```

El comando para cargar este tipo de bases es:

```
Datos<-read.dta("http://qcpages.qc.cuny.edu/~rvesselinov/statadata/phillips.dta")
```

1.5 Importar datos usando las bibliotecas de tidyverse

La biblioteca tidyverse es un conjunto de paquetes de R que permiten: Importar, modificar y analizar bases de datos. Contiene el objeto: *tibble* que cumple el mismo papel que un dataframe pero con más funcionalidades.

Además, contiene funcionalidades de gráficación.

Paquetes principales:

- dplyr: Para modificar variables dentro de las bases de datos. Esto implica filtrar, modificar variables, agrupar variables y demás dentro una base de datos.
- tidyr: Para modificar la estructura de la base de datos
- ggplot2: Biblioteca de graficación (Posiblemente la principal biblioteca de graficación de R)
- readr: Para importar y trabajar con archivos excel

y mucho otros paquetes. Los interesados en conocer más sobre el conjunto de paquetes del tidyverse pueden revisar el siguiente link: Link al tidyverse

```
#install.packages("tidyverse")
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------------- tidyverse 1.3.0
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------------------------ tidyverse_conflicts()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Importar un archivo csv usando tidyverse

read_csv importa la base de datos en formato csv como un objeto tibble.

```
titanic2 = read_csv("titanic.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   X1 = col_double(),
##   Name = col_character(),
##   PClass = col_character(),
##   Age = col_double(),
##   Sex = col_character(),
##   Survived = col_double()
## )
```

glimpse[1] es un comando que permite visualizar la base de datos de manera compacta en la consola[2]

```
glimpse(titanic2)
```

2. Manipulación de base de datos

RStudio es una de las alternativas mas sencillas para realizar minería de datos, comúnmente lo que uno suele hacer con los datos es ordenarlos, transformarlos, reducir el número de observaciones, etc. Bastante utilizado hoy en día en temas de Big Data y Machine Learning.

[1] este comando hace parte del paquete dplyr
[2] Donde entre otras cosas muestre el tipo o clase de la variable que se está importando

2.1 Remplazar valores de una variable con if else

```
summary(titanic$Sex)
```

```
## female   male
##    462    851
```

Se busca remplazar las variables categóricas del género, de tal manera que se cree una variable dummy que tome el valor de 1 cuando es mujer y 0 en otro caso:

```
titanic$Sex<-ifelse(titanic$Sex=="female",1,0) #ifelse(test lógico, yes/verdadero, no/falso)
summary(titanic$Sex)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.3519  1.0000  1.0000
```

Otro ejemplo, es crear una variable de conteo que indique la clase en la que viajaba el tripulante del Titanic:

```
titanic$PClass<-ifelse(titanic$PClass=="1st",1,
                       ifelse(titanic$PClass=="2nd",2,3))
```

```
table(titanic$PClass)
```

```
##
##   1   2   3
## 322 279 712
```

2.2 Medidas de tendencia central y variabilidad

```
mean(Age, na.rm= T)          #Media
```

```
## [1] 30.39799
```

```
median(Age,na.rm = T)        #Mediana
```

```
## [1] 28
```

```
sd(Age,na.rm = T)            #Desviación estandar
```

```
## [1] 14.25905
```

```
var(Age,na.rm = T)           #Varianza
```

```
## [1] 203.3205
```

El argumento ***na.rm*** *es para que las casillas en las que no hay información no se tengan en cuenta*

2.3 Gráficos

2.3.1 Gráficos usando las funciones de R

Tablas La estructura del código es table(Nombre_variable)

```
table(Sex)
```

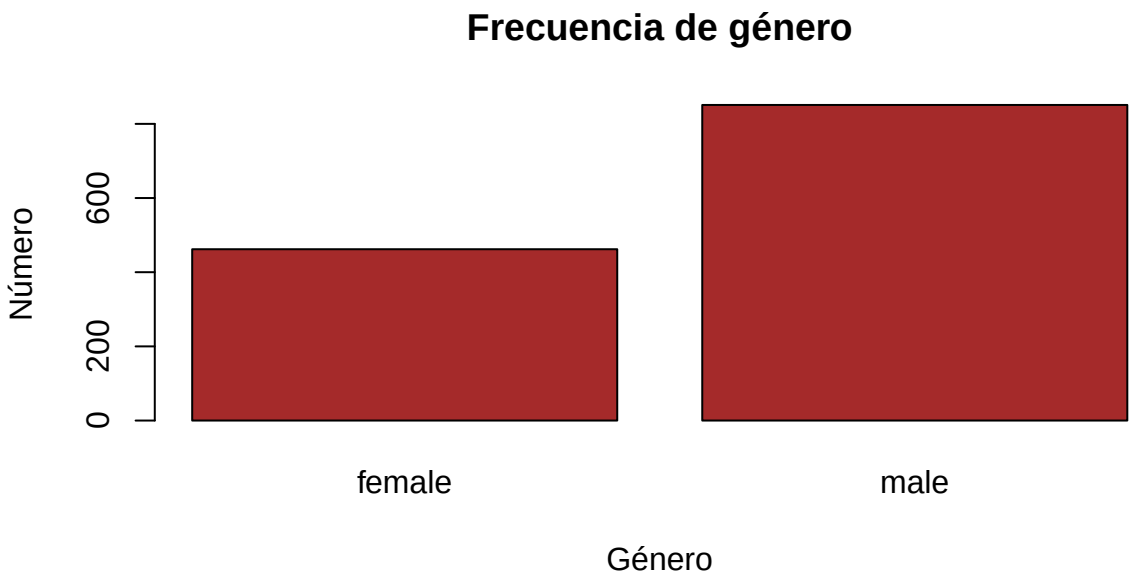
```
## Sex
## female   male
##    462    851
```

```
table(PClass)
```

```
## PClass
##   * 1st 2nd 3rd
##   1 322 279 711
```

Gráfica de barras Para crear un diagrama de barras es importante primero crear una Tabla de frecuencia de la variable que se desee graficar

```
#x11() #x11() permite que las gráficas se desplieguen en ventanas emergentes.
barplot(table(Sex), main ="Frecuencia de género", xlab="Género",
        ylab ="Número", col="brown") # main es para el título, xlab y ylab son para
```




```
                                    # los nombres de las ejes  y col es para el color de las barras
```

Para las personas que prefieran visualizar las gráficas en una pestaña externa pueden usar el comando x11() que permite que las gráficas se desplieguen en ventanas emergentes.

Histogramas

La estructura del código es hist(Nombre_variable, main="...", xlab="...",ylab="...", col="...")

- main: Para el título de la gráfica
- xlab: Para el título del eje x
- ylab: Para el título del eje y
- col: Color de las barras

```
hist(Age, main = "Edad de los tripulantes", xlab="Edad",
     ylab = "Frecuencia", col = "blue3")
```

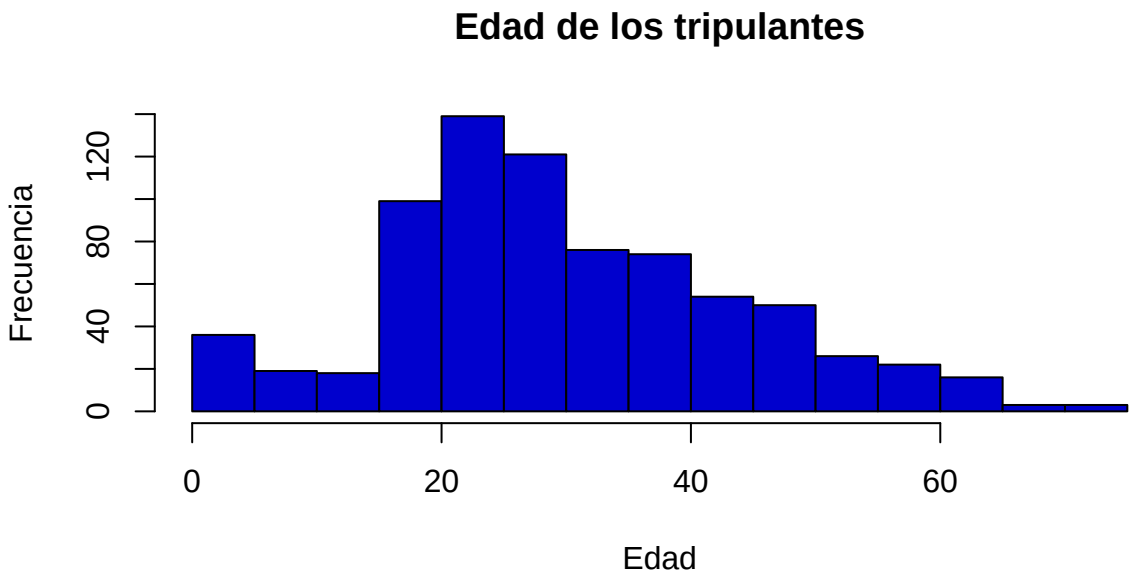
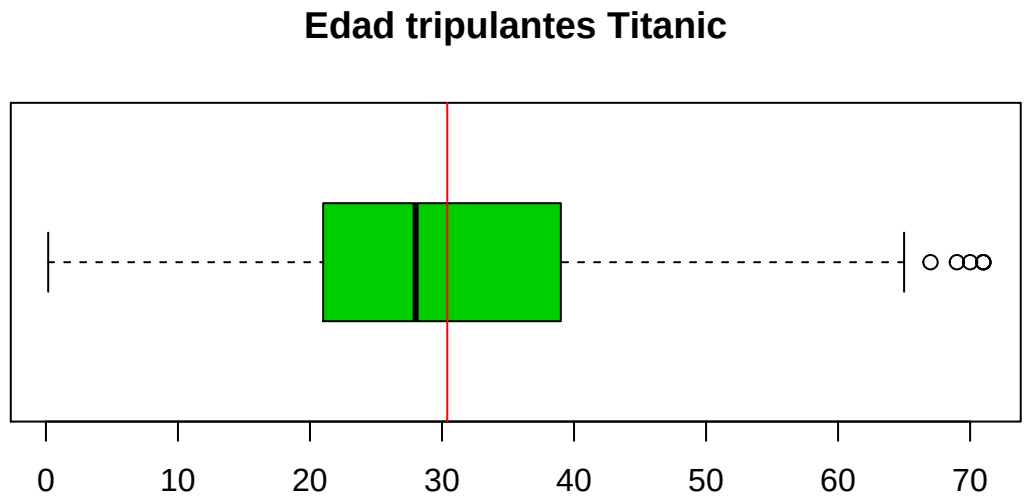


Diagrama de Caja - boxplot

La estructura del código es boxplot(Nombre_variable, horizontal=T, main="...", abline(v=mean(Age,na.rm = T), col="...")

- El argumento *horizontal* permite que el diagrama de caja sea horizontal
- El argumento *abline* agregara una linea vértical al grafico, en este caso indicara la media de la variable

```
boxplot(Age, col=c(3), horizontal = TRUE, main="Edad tripulantes Titanic")
abline(v=mean(Age,na.rm = T),col=c(2))
```




```
summary(Age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.17   21.00   28.00   30.40   39.00   71.00     557
```

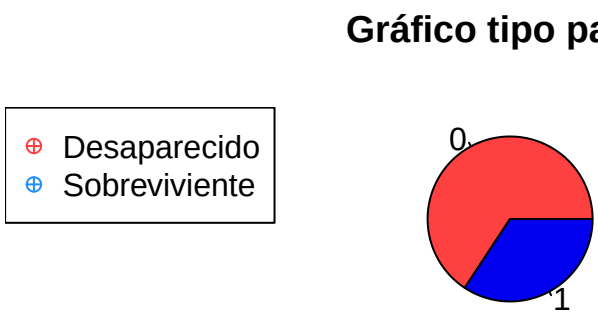
Gráfico tipo pie

De la misma manera que el gráfico de barras, hay que primero crear una tabla de frecuencia acumulada para que R Studio pueda crear el grafico de torta

La estructura del código es pie(table(**VARIABLE**),radius=0.8, main = "...", col = c()),legend("topleft", legend=c("Desaparecido","Sobreviviente"), pch = 10,col =c("brown1","dodgerblue"))

- *radius* indica el tamaño de la gráfica
- *legend* permite agregar un cuadro de leyenda al gráfico (Para mas información pueden leer la documentación de legend)
- *pch* indica la figura de la convención de la leyenda + Hay que tener en cuenta que

```
pie(table(Survived),radius=0.8, main = "Gráfico tipo pastel", col = c("brown1","blue2"))
legend("topleft", legend= c("Desaparecido","Sobreviviente"), pch = 10,col =
         c("brown1","dodgerblue"),)
```

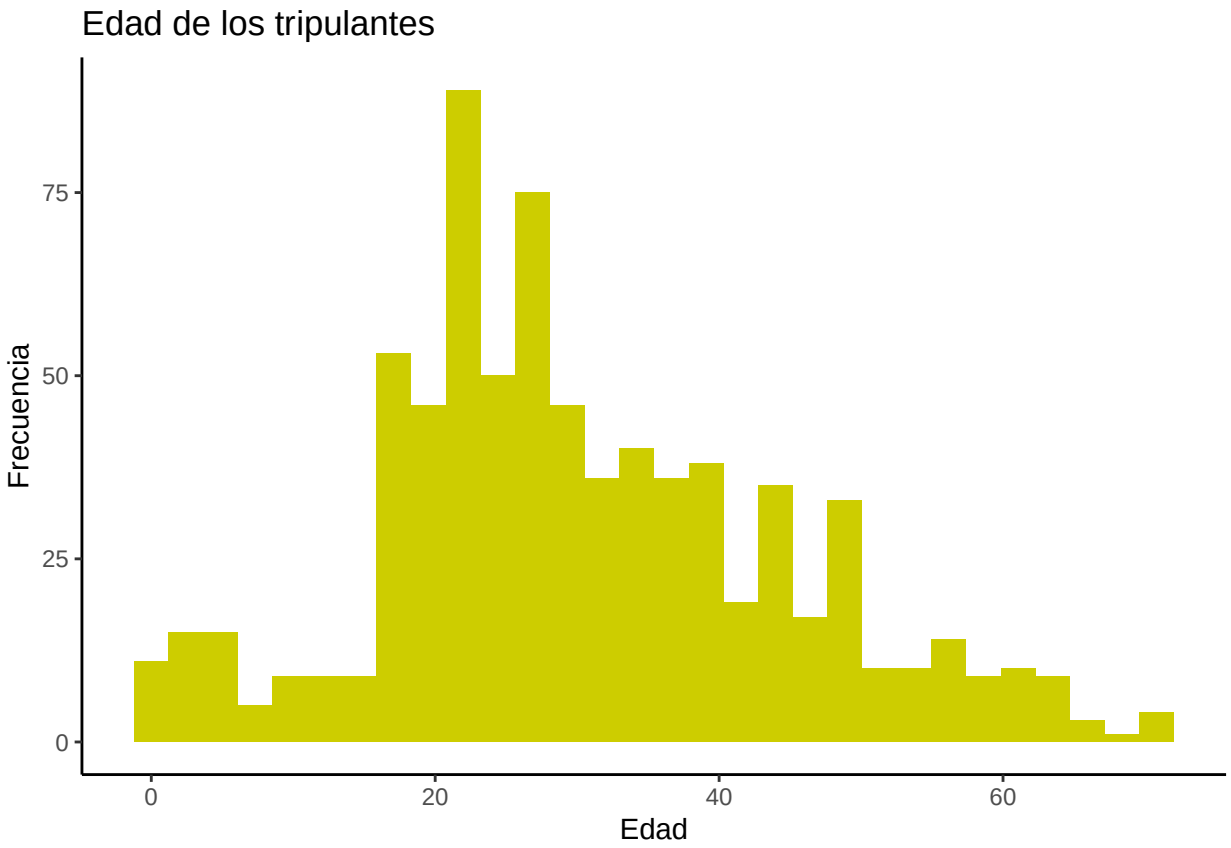


2.2 Gráficos usando la biblioteca ggplot2

La librería ggplot2 permite crear gráficos mucho mas estéticos. Este paquete provee una estructura base de tal manera que se le indica que tipo de variable se va a graficar y como se va a presentar. La función base para hacer cualquier tipo de gráfico es:

```
hist1 = ggplot(data = titanic, aes(Age)) +
        geom_histogram(fill = "yellow3") +
        labs(title = "Edad de los tripulantes") +
        xlab("Edad") +
        ylab("Frecuencia") +
        theme_classic()
hist1
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



ggplot opera a través de layers o capas[3]: 1. La capa principal es la que se conoce como ggplot(*base_datos*, aes(x = var_x, y = var_y)). En ella se especifica primero la *base_datos* con la que se va a trabajar y luego dentro de **aes()** las variables x(independiente) y y(dependiente) para la gráfica. 2. Luego, la capa geom_**gráfica** indica el tipo de gráfica. En el caso del ejemplo anterior se desea hacer un histograma por lo que usa **geom_histogram** 3. A continuación, se especifica la capa de labs en donde se coloca el título de la gráfica. 4. Las siguientes dos capas especifican los nombres para el eje x y para el eje y 5. Finalmente, la capa theme_classic() espeficica

Como pueden observar, toda gráfica de ggplot opera mediante un estructura de capas.

3. Regresión lineal

Para los ejercicios de regresión de esta sección se cargaran los siguientes paquetes[4]:

```
#install.packages("wooldridge")
library(wooldridge)
```

En muchas ocasiones los paquetes tienen bases de datos, las cuales podemos cargar a RStudio. Vamos a importar la base de datos llamada bwght del paquete Wooldridge

```
data("bwght2") # Para importar bases de datos que se encuentran dentro de un paquete previamente instal

attach(bwght2)
```

```
## The following object is masked from package:wooldridge:
##
```

[3]cada capa se conecta mediante un **+**

[4]El paquete wooldridge contiene la mayoría de las bases de datos que se encuentran en la 6 edición del libro Wooldridge

```
##     bwght
```

```
help("bwght2") # Descripción de la base de datos
```

La base de datos *bwght* es una base de datos con 1832 observaciones de 23 variables:

```
glimpse(bwght2)
```

```
## Rows: 1,832
## Columns: 23
## $ mage    <int> 26, 29, 33, 28, 23, 28, 27, 41, 32, 16, 26, 25, 28, 29, 31,...
## $ meduc   <int> 12, 12, 12, 17, 13, 12, 16, 17, 12, 11, 14, 14, 16, 16, 16,...
## $ monpre  <int> 2, 2, 1, 5, 2, 1, 3, 6, 3, 2, 2, 2, 1, 2, 2, 1, 2, 2, 1, 2,...
## $ npvis   <int> 12, 12, 12, 8, 6, 12, 11, 8, 11, 10, 12, 13, 14, 8, 9, 12, ...
## $ fage    <int> 34, 32, 36, 32, 24, 30, 28, 56, 36, 21, 26, 25, 30, 38, 34,...
## $ feduc   <int> 16, 12, 16, 17, 16, 16, 16, NA, 16, 10, 14, 12, 16, 12, 16,...
## $ bwght   <int> 3060, 3730, 2530, 3289, 3590, 3420, 3355, 3459, 3590, 4410,...
## $ omaps   <int> 9, 8, 8, 8, 6, 9, 9, 8, 9, 6, 9, 8, 8, 9, 8, 9, 8, 7, 9, 9,...
## $ fmaps   <int> 9, 9, 9, 9, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 8, 9, 9,...
## $ cigs    <int> 0, NA, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, NA, ...
## $ drink   <int> 0, NA, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, NA, ...
## $ lbw     <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ vlbw    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ male    <int> 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0,...
## $ mwhte   <int> 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,...
## $ mblck   <int> 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ moth    <int> 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ fwhte   <int> 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,...
## $ fblck   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ foth    <int> 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ lbwght  <dbl> 8.026170, 8.224163, 7.835975, 8.098339, 8.185907, 8.137396,...
## $ magesq  <int> 676, 841, 1089, 784, 529, 784, 729, 1681, 1024, 256, 676, 6...
## $ npvissq <int> 144, 144, 144, 64, 36, 144, 121, 64, 121, 100, 144, 169, 19...
```

- mage: mother's age, years
- meduc: mother's educ, years
- monpre: month prenatal care began
- npvis: total number of prenatal visits
- fage: father's age, years
- feduc: father's educ, years
- bwght: birth weight, grams
- omaps: one minute apgar score
- fmaps: five minute apgar score
- cigs: avg cigarettes per day
- drink: avg drinks per week
- lbw: =1 if bwght <= 2000
- vlbw: =1 if bwght <= 1500
- male: =1 if baby male
- mwhte: =1 if mother white
- mblck: =1 if mother black
- moth: =1 if mother is other
- fwhte: =1 if father white
- fblck: =1 if father black
- foth: =1 if father is other
- lbwght: log(bwght)
- magesq: mage^2

- npvissq: npvis^2

Las características o estadística descriptiva de las variables

```
summary(bwght2) # Información más extensa de la base de datos
```

```
##       mage           meduc           monpre          npvis
##  Min.   :16.00   Min.   : 3.00   Min.   :0.000   Min.   : 0.00
##  1st Qu.:26.00   1st Qu.:12.00   1st Qu.:1.000   1st Qu.:10.00
##  Median :29.00   Median :13.00   Median :2.000   Median :12.00
##  Mean   :29.56   Mean   :13.72   Mean   :2.122   Mean   :11.62
##  3rd Qu.:33.00   3rd Qu.:16.00   3rd Qu.:2.000   3rd Qu.:13.00
##  Max.   :44.00   Max.   :17.00   Max.   :9.000   Max.   :40.00
##                  NA's   :30      NA's   :5       NA's   :68
##       fage           feduc           bwght          omaps
##  Min.   :18.00   Min.   : 3.00   Min.   : 360   Min.   : 0.000
##  1st Qu.:28.00   1st Qu.:12.00   1st Qu.:3076   1st Qu.: 8.000
##  Median :31.00   Median :14.00   Median :3425   Median : 9.000
##  Mean   :31.92   Mean   :13.92   Mean   :3401   Mean   : 8.386
##  3rd Qu.:35.00   3rd Qu.:16.00   3rd Qu.:3770   3rd Qu.: 9.000
##  Max.   :64.00   Max.   :17.00   Max.   :5204   Max.   :10.000
##  NA's   :6       NA's   :47                     NA's   :3
##      fmaps             cigs            drink             lbw
##  Min.   : 2.000   Min.   : 0.000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.: 9.000   1st Qu.: 0.000   1st Qu.:0.0000   1st Qu.:0.00000
##  Median : 9.000   Median : 0.000   Median :0.0000   Median :0.00000
##  Mean   : 9.004   Mean   : 1.089   Mean   :0.0198   Mean   :0.01638
##  3rd Qu.: 9.000   3rd Qu.: 0.000   3rd Qu.:0.0000   3rd Qu.:0.00000
##  Max.   :10.000   Max.   :40.000   Max.   :8.0000   Max.   :1.00000
##  NA's   :3        NA's   :110      NA's   :115
##       vlbw              male            mwhte            mblck
##  Min.   :0.000000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.000000   1st Qu.:0.0000   1st Qu.:1.0000   1st Qu.:0.0000
##  Median :0.000000   Median :1.0000   Median :1.0000   Median :0.0000
##  Mean   :0.007096   Mean   :0.5136   Mean   :0.8865   Mean   :0.0595
##  3rd Qu.:0.000000   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:0.0000
##  Max.   :1.000000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##
##       moth             fwhte            fblck             foth
##  Min.   :0.00000   Min.   :0.0000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:1.0000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.00000   Median :1.0000   Median :0.00000   Median :0.00000
##  Mean   :0.05404   Mean   :0.8897   Mean   :0.05841   Mean   :0.05186
##  3rd Qu.:0.00000   3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.0000   Max.   :1.00000   Max.   :1.00000
##
##      lbwght           magesq          npvissq
##  Min.   :5.886   Min.   : 256.0   Min.   :   0.0
##  1st Qu.:8.031   1st Qu.: 676.0   1st Qu.: 100.0
##  Median :8.139   Median : 841.0   Median : 144.0
##  Mean   :8.114   Mean   : 896.4   Mean   : 148.6
##  3rd Qu.:8.235   3rd Qu.:1089.0   3rd Qu.: 169.0
##  Max.   :8.557   Max.   :1936.0   Max.   :1600.0
##                                   NA's   :68
```

```r
#install.packages("tibble")
library(tibble)
#glimpse(bwght2) # Manera compacta de visualizar la base de datos
```

3.1 Estimación

Para hacer una regresión lineal en RStudio el comando es *lm* de tal manera que la sintaxis del comando es *lm(Y ~ X , data="Nombre_Base_Datos")* donde Y es la variable dependiente y X son todas las variables explicativas(regresoras).

Con el *summary(Nombre de la regresión)* se pueden observar los resultados de las regresiones

En el siguiente ejemplo se realizaran 3 regresiones en donde en cada regresión lo que variará serán las variables regresoras.

```r
MODELO1 = lm(bwght ~ cigs + drink + npvis + male + mage +
               mwhte + mblck, data = bwght2)
summary(MODELO1)
```

```
##
## Call:
## lm(formula = bwght ~ cigs + drink + npvis + male + mage + mwhte +
##     mblck, data = bwght2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3039.94  -329.53    21.24   355.72  1803.83
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2944.983    114.269  25.772  < 2e-16 ***
## cigs         -11.070      3.481  -3.180 0.001500 **
## drink        -16.690     48.496  -0.344 0.730776
## npvis         13.526      3.770   3.588 0.000343 ***
## male          81.297     28.123   2.891 0.003894 **
## mage           4.388      2.985   1.470 0.141810
## mwhte        161.235     60.584   2.661 0.007859 **
## mblck        117.499     82.721   1.420 0.155676
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 568.7 on 1640 degrees of freedom
##   (184 observations deleted due to missingness)
## Multiple R-squared:  0.02543,    Adjusted R-squared:  0.02127
## F-statistic: 6.114 on 7 and 1640 DF,  p-value: 4.525e-07
```

```r
MODELO2 = lm(bwght ~ cigs + drink + npvis + male + mage +
               magesq +mwhte + mblck + meduc + feduc , data = bwght2)
summary(MODELO2)
```

```
##
## Call:
## lm(formula = bwght ~ cigs + drink + npvis + male + mage + magesq +
##     mwhte + mblck + meduc + feduc, data = bwght2)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -3032.06  -333.11    17.41   354.30  1769.37
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2037.0532   407.0971   5.004 6.24e-07 ***
## cigs          -8.7128     3.5823  -2.432  0.01512 *
## drink        -16.1868    48.2395  -0.336  0.73725
## npvis         12.4735     3.8234   3.262  0.00113 **
## male          84.0243    28.2243   2.977  0.00295 **
## mage          62.9003    27.4798   2.289  0.02221 *
## magesq        -1.0051     0.4589  -2.191  0.02863 *
## mwhte        170.0040    61.5223   2.763  0.00579 **
## mblck        150.4942    85.4864   1.760  0.07852 .
## meduc         -3.4376     8.6402  -0.398  0.69078
## feduc          8.9771     7.7782   1.154  0.24862
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 564.7 on 1606 degrees of freedom
##   (215 observations deleted due to missingness)
## Multiple R-squared:  0.02736,    Adjusted R-squared:  0.0213
## F-statistic: 4.518 on 10 and 1606 DF,  p-value: 2.499e-06
```

```
mean.educ = (meduc + feduc)/2
MODELO3 = lm(bwght ~ cigs + drink + npvis + male + mage +
               magesq +mwhte + mblck + mean.educ, data = bwght2)
summary(MODELO3)
```

```
##
## Call:
## lm(formula = bwght ~ cigs + drink + npvis + male + mage + magesq +
##     mwhte + mblck + mean.educ, data = bwght2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3035.74  -333.13    17.87   355.63  1779.41
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2044.9636   406.9618   5.025  5.6e-07 ***
## cigs          -8.7357     3.5819  -2.439  0.01484 *
## drink        -17.1603    48.2225  -0.356  0.72200
## npvis         12.5036     3.8230   3.271  0.00110 **
## male          82.9385    28.1941   2.942  0.00331 **
## mage          61.8625    27.4513   2.254  0.02436 *
## magesq        -0.9920     0.4586  -2.163  0.03066 *
## mwhte        170.6822    61.5125   2.775  0.00559 **
## mblck        150.2193    85.4791   1.757  0.07904 .
## mean.educ      6.3898     7.9182   0.807  0.41980
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 564.7 on 1607 degrees of freedom
##   (215 observations deleted due to missingness)
```

```
## Multiple R-squared:  0.02691,    Adjusted R-squared:  0.02146
## F-statistic: 4.938 on 9 and 1607 DF,  p-value: 1.453e-06
```

Para una regresión lineal simple estimada mediante OLS la salida del summary indicaría lo siguiente: - **Call**: indica básicamente el comando utilizado para generar la salida. - **Residuals**: Para indicar los cuantiles de los residuales - **Coefficients:** Para indicar el valor de los coeficientes estimados, el error estándar asociado a cada coeficiente, el t-valor y el p-valor asociado a las pruebas de signicancia. El número de estrellitas indica el nivel de significancia - **Signif. codes**: Indica el significado de cada uno de los códigos para los niveles de significancia. - **Residual standard error**: Indica el valor del error estándar poblacional el σ. - **Multiple R-squared**: indica el R^2 y el R^2_{adj} - **F-statistic**: indica el estadístico F para la significancia global. En particular el valor del F estadístico, el número de grados de libertad y el p-valor.

Supustos de Gauss Markov

1. Linealidad en los paramentros $y = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k + u$
2. La muestra proviene de un muestreo aleatorio
3. No hay multicolinealidad perfecta entre los regresores
4. Media condcional $E[u|x_1, x_2, \cdots, x_k] = 0$
5. Homocedasticidad $var(u|x_1, x_2, \cdots, x_k) = \sigma^2$

Consecuencias de los supuestos de Gauss Markov

- Si se cumplen los primeros 4 supuestos de Gauss Markov el estimador de **MCO** es insesgado y consistente
- Si se cumplen todos los supuestos de Gauss Markov el estimador de **MCO** es **MELI**

Presentación de resultados mediante la función stargazer

Ahora bien, existe el comando *stargazer* el cual genera una presentación de los coeficientes estimados de cada regresión de una manera mucho mas estética de como los genera el *summary*

Para ellos es importante cargar instalar los siguientes paquetes

```
#install.packages("stargazer")
library(stargazer)
```

```
##
## Please cite as:
```

```
##  Hlavac, Marek (2018). stargazer: Well-Formatted Regression and Summary Statistics Tables.
```

```
##  R package version 5.2.2. https://CRAN.R-project.org/package=stargazer
```

La estructura básica de stargazer es indicarle el nombre de los modelos que se quieren presentar, pues este comando permite presentar mas de una regresión en la misma tabla y los diferentes argumentos adicionales que se quieran incluir en la tabla como lo son el R^2, el numero de observaciones entre otros.

Argumentos de Stargazer:

- **title:** introducir un título a la tabla
- **type:** el formato de la tabla (e.g. text o latex)}[5]
- **omit:** si se quiere omitir alguna variable de la tabla
- **style:** específica el estilo de la tabla (p.ej. aer para American Economic Review)
- **all** all statistics
- **"adj.rsq"** adjusted R-squared
- **"aic"** Akaike Information Criterion
- **"bic"** Bayesian Information Criterion
- **"chi2"** chi-squared

[5]Dependiendo del valor del argumento acá Stargazer generará ya sea una tabla de para imprimir en la consla o una tabla ingresar directamente en latex

- **“f”** F statistic
- **“ll”** log-likelihood
- **“logrank”** score (logrank) test
- **“lr”** likelihood ratio (LR) test
- **“max.rsq”** maximum R-squared
- **“n”** number of observations
- **“null.dev”** null deviance
- **“Mills”** Inverse Mills Ratio
- **“res.dev”** residual deviance
- **“rho”** rho
- **“rsq”** R-squared
- **scale** scale
- **“theta”** theta
- **“ser”** standard error of the regression (i.e., residual standard error)
- **“sigma2”** sigma squared
- **“ubre”** Un-Biased Risk Estimator
- **“wald”** Wald test

En el siguiente ejemplo stargazer mostrará las 3 regresiones lineales realizadas anteriormente:

```
stargazer(MODELO1, MODELO2, MODELO3, type="text",
          column.labels = c("REG1","REG2", "REG3"),
          keep.stat = c("n", "rsq","adj.rsq","aic"))
```

```
##
## ================================================
##                        Dependent variable:
##              ------------------------------------
##                               bwght
##                  REG1          REG2          REG3
##                  (1)           (2)           (3)
## ------------------------------------------------
## cigs          -11.070***    -8.713**      -8.736**
##                (3.481)       (3.582)       (3.582)
##
## drink           -16.690       -16.187       -17.160
##                (48.496)      (48.240)      (48.223)
##
## npvis          13.526***     12.474***     12.504***
##                (3.770)       (3.823)       (3.823)
##
## male           81.297***     84.024***     82.938***
##                (28.123)      (28.224)      (28.194)
##
## mage             4.388        62.900**      61.862**
##                (2.985)       (27.480)      (27.451)
##
## magesq                        -1.005**      -0.992**
##                               (0.459)       (0.459)
##
## mwhte          161.235***    170.004***    170.682***
##                (60.584)      (61.522)      (61.512)
##
## mblck           117.499       150.494*      150.219*
##                (82.721)      (85.486)      (85.479)
```

```
##
## meduc                          -3.438
##                               (8.640)
##
## feduc                          8.977
##                               (7.778)
##
## mean.educ                                      6.390
##                                               (7.918)
##
## Constant     2,944.983*** 2,037.053*** 2,044.964***
##               (114.269)    (407.097)    (406.962)
##
## ------------------------------------------------
## Observations    1,648        1,617        1,617
## R2              0.025        0.027        0.027
## Adjusted R2     0.021        0.021        0.021
## ================================================
## Note:                   *p<0.1; **p<0.05; ***p<0.01
```

Si se usa el argumento type=latex la tabla resultante será:

```
stargazer(MODELO1, MODELO2, MODELO3, type="latex",
          column.labels = c("REG1","REG2", "REG3"),
          keep.stat = c("n", "rsq","adj.rsq","aic"))
```

```
##
## % Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harva
## % Date and time: mié, sep 16, 2020 - 16:50:26
## \begin{table}[!htbp] \centering
##   \caption{}
##   \label{}
## \begin{tabular}{@{\extracolsep{5pt}}lccc}
## \\[-1.8ex]\hline
## \hline \\[-1.8ex]
##  & \multicolumn{3}{c}{\textit{Dependent variable:}} \\
## \cline{2-4}
## \\[-1.8ex] & \multicolumn{3}{c}{bwght} \\
##  & REG1 & REG2 & REG3 \\
## \\[-1.8ex] & (1) & (2) & (3)\\
## \hline \\[-1.8ex]
##  cigs & $-$11.070$^{***}$ & $-$8.713$^{**}$ & $-$8.736$^{**}$ \\
##   & (3.481) & (3.582) & (3.582) \\
##   & & & \\
##  drink & $-$16.690 & $-$16.187 & $-$17.160 \\
##   & (48.496) & (48.240) & (48.223) \\
##   & & & \\
##  npvis & 13.526$^{***}$ & 12.474$^{***}$ & 12.504$^{***}$ \\
##   & (3.770) & (3.823) & (3.823) \\
##   & & & \\
##  male & 81.297$^{***}$ & 84.024$^{***}$ & 82.938$^{***}$ \\
##   & (28.123) & (28.224) & (28.194) \\
##   & & & \\
##  mage & 4.388 & 62.900$^{**}$ & 61.862$^{**}$ \\
##   & (2.985) & (27.480) & (27.451) \\
```

```
##   & & & \\
##  magesq &  & $-$1.005$^{**}$ & $-$0.992$^{**}$ \\
##   &  & (0.459) & (0.459) \\
##   & & & \\
##  mwhte & 161.235$^{***}$ & 170.004$^{***}$ & 170.682$^{***}$ \\
##   & (60.584) & (61.522) & (61.512) \\
##   & & & \\
##  mblck & 117.499 & 150.494$^{*}$ & 150.219$^{*}$ \\
##   & (82.721) & (85.486) & (85.479) \\
##   & & & \\
##  meduc &  & $-$3.438 &  \\
##   &  & (8.640) &  \\
##   & & & \\
##  feduc &  & 8.977 &  \\
##   &  & (7.778) &  \\
##   & & & \\
##  mean.educ &  &  & 6.390 \\
##   &  &  & (7.918) \\
##   & & & \\
##  Constant & 2,944.983$^{***}$ & 2,037.053$^{***}$ & 2,044.964$^{***}$ \\
##   & (114.269) & (407.097) & (406.962) \\
##   & & & \\
## \hline \\[-1.8ex]
## Observations & 1,648 & 1,617 & 1,617 \\
## R$^{2}$ & 0.025 & 0.027 & 0.027 \\
## Adjusted R$^{2}$ & 0.021 & 0.021 & 0.021 \\
## \hline
## \hline \\[-1.8ex]
## \textit{Note:}  & \multicolumn{3}{r}{$^{*}$p$<$0.1; $^{**}$p$<$0.05; $^{***}$p$<$0.01} \\
## \end{tabular}
## \end{table}
```

Dicho código generado en la consola se puede copiar y pegar directamente en un documento latex para renderizar la tabla de resultados de la regresión el documento latex.

3.2 Otro ejemplo de estimación

Cargamos la base de datos *Elecciones - Script 2*

```
library(readxl)
Elecciones = read_excel("Elecciones - Script 2.xls")
attach(Elecciones)
#View(Elecciones)
#summary(Elecciones)
```

Realizamos una regresión nivel-nivel:

```
RegresiónA1 = lm(voteA ~ democA +expendA + expendB + prtystrA) #lin-lin
summary(RegresiónA1)
```

```
##
## Call:
## lm(formula = voteA ~ democA + expendA + expendB + prtystrA)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -28.444  -7.937  -0.150   7.055  35.266
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 16.964468   5.217739   3.251  0.00139 **
## democA       9.382202   1.840633   5.097 9.20e-07 ***
## expendA      0.031801   0.003204   9.925  < 2e-16 ***
## expendB     -0.030095   0.002957 -10.177  < 2e-16 ***
## prtystrA     0.555421   0.092108   6.030 1.01e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.38 on 168 degrees of freedom
## Multiple R-squared:  0.6264, Adjusted R-squared:  0.6175
## F-statistic: 70.43 on 4 and 168 DF,  p-value: < 2.2e-16
```

Realizamos una regresión nivel-logaritmo:

```
RegresiónA2 = lm(voteA ~ democA +log(expendA)+log(expendB) + log(prtystrA)) #lin-log
summary(RegresiónA2)
```

```
##
## Call:
## lm(formula = voteA ~ democA + log(expendA) + log(expendB) + log(prtystrA))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.979  -4.925  -1.099   4.931  24.442
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     7.7265    13.2043   0.585 0.559232
## democA          3.4841     1.3821   2.521 0.012635 *
## log(expendA)    5.8093     0.3921  14.817  < 2e-16 ***
## log(expendB)   -6.3065     0.3946 -15.983  < 2e-16 ***
## log(prtystrA)  11.0228     3.2749   3.366 0.000946 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.599 on 168 degrees of freedom
## Multiple R-squared:  0.7998, Adjusted R-squared:  0.7951
## F-statistic: 167.8 on 4 and 168 DF,  p-value: < 2.2e-16
```

Presentación de resultados:

```
stargazer(RegresiónA1, RegresiónA2, type="text",
          column.labels = c("N-N","N-L"),
          keep.stat = c("n", "rsq", "adj.rsq", "aic"))
```

```
##
## ========================================
##                   Dependent variable:
##               ---------------------------
##                           voteA
##                     N-N            N-L
```

```
##                     (1)            (2)
## ----------------------------------------
## democA           9.382***       3.484**
##                  (1.841)        (1.382)
##
## expendA          0.032***
##                  (0.003)
##
## expendB         -0.030***
##                  (0.003)
##
## prtystrA         0.555***
##                  (0.092)
##
## log(expendA)                   5.809***
##                                 (0.392)
##
## log(expendB)                   -6.306***
##                                 (0.395)
##
## log(prtystrA)                  11.023***
##                                 (3.275)
##
## Constant        16.964***        7.727
##                  (5.218)       (13.204)
##
## ----------------------------------------
## Observations       173            173
## R2                0.626          0.800
## Adjusted R2       0.618          0.795
## ========================================
## Note:          *p<0.1; **p<0.05; ***p<0.01
```

3.3 Pruebas de significancia individual para las variables

Para realizar las pruebas de significancia individual es necesario usar el paquete *lmtest*.

```
#install.packages("lmtest")
library("lmtest")
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

El comando *coeftest*[6] nos arroja la significancia estadística de los coeficientes estimados:

```
coeftest(RegresiónA2)
```

```
##
## t test of coefficients:
##
```

[6] del paquete lmtest

```
##                Estimate Std. Error  t value  Pr(>|t|)
## (Intercept)    7.72651   13.20433   0.5851 0.5592324
## democA         3.48410    1.38206   2.5210 0.0126353 *
## log(expendA)   5.80933    0.39207  14.8171 < 2.2e-16 ***
## log(expendB)  -6.30647    0.39458 -15.9826 < 2.2e-16 ***
## log(prtystrA) 11.02277    3.27489   3.3658 0.0009457 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Intervalos de confianza para los coeficientes de la regresión al 90 % y 95 % especificamente:

```
confint(RegresiónA2) #Al 95%
```

```
##                     2.5 %    97.5 %
## (Intercept)   -18.3412875 33.794300
## democA          0.7556668  6.212541
## log(expendA)    5.0353146  6.583353
## log(expendB)   -7.0854505 -5.527486
## log(prtystrA)   4.5575392 17.488005
```

```
confint(RegresiónA2, level = 0.90) #Al 90%
```

```
##                      5 %      95 %
## (Intercept)   -14.113116 29.566128
## democA          1.198217  5.769991
## log(expendA)    5.160860  6.457808
## log(expendB)   -6.959100 -5.653836
## log(prtystrA)   5.606194 16.439351
```

Valores ajustados o estimados usando el comando *fitted.values*

```
Estimados<-fitted.values(RegresiónA2)
```

Residuales de la regresión con el comando *residuals*

```
Residuales<-residuals(RegresiónA2)
RS.Residuals <-rstandard(RegresiónA2) # Residuales estandarizados (divididos por su desviación estándar
```

4. Validación de supuestos

4.1 Test de Ramsey para especificaciones erróneas

```
resettest(RegresiónA2) #Ho = el modelo está bien especificado
```

```
##
##  RESET test
##
## data:  RegresiónA2
## RESET = 29.788, df1 = 2, df2 = 166, p-value = 8.823e-12
```

4.2 Heteroscedasticidad en los residuales

Es necesario instalar el paquete *car*

```
#install.packages("car")
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
## The following object is masked from 'package:dplyr':
##
##     recode
## The following object is masked from 'package:purrr':
##
##     some
```

Test Breusch-Pagan con Ho=Homocedasticidad

```
bptest(RegresiónA2) #Test Breusch-Pagan con Ho=Homocedasticidad
```

```
##
##  studentized Breusch-Pagan test
##
## data:  RegresiónA2
## BP = 9.697, df = 4, p-value = 0.04585
```

Test de Varianza no constante con Ho= Homocedasticidad

```
ncvTest(RegresiónA2) #Test de Varianza no constante con Ho= Homocedasticidad
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 3.01183, Df = 1, p = 0.082659
```

Correlación serial en los residuales

Dado que la muestra es de corte transversal no tiene mucho sentido de hablar de correlación serial. No obstante, vale la pena aclarar que si se puede habalr de correlación espacial en muestras de corte trasnversal. La correlación serial va a tomar mucho más importancia cuando se estudien las series de tiempo.

Durbin Watson test (Ho:No autocorrelación de 1er orden)

```
dwtest(RegresiónA2) #Durbin Watson test (Ho:No autocorrelación de 1er orden)
```

```
##
##  Durbin-Watson test
##
## data:  RegresiónA2
## DW = 1.5476, p-value = 0.001107
## alternative hypothesis: true autocorrelation is greater than 0
```

Prueba Breush-Godfrey (Ho:No autocorrelación de orden p)

```
bgtest(RegresiónA2) #Prueba Breush-Godfrey (Ho:No autocorrelación de orden p)
```

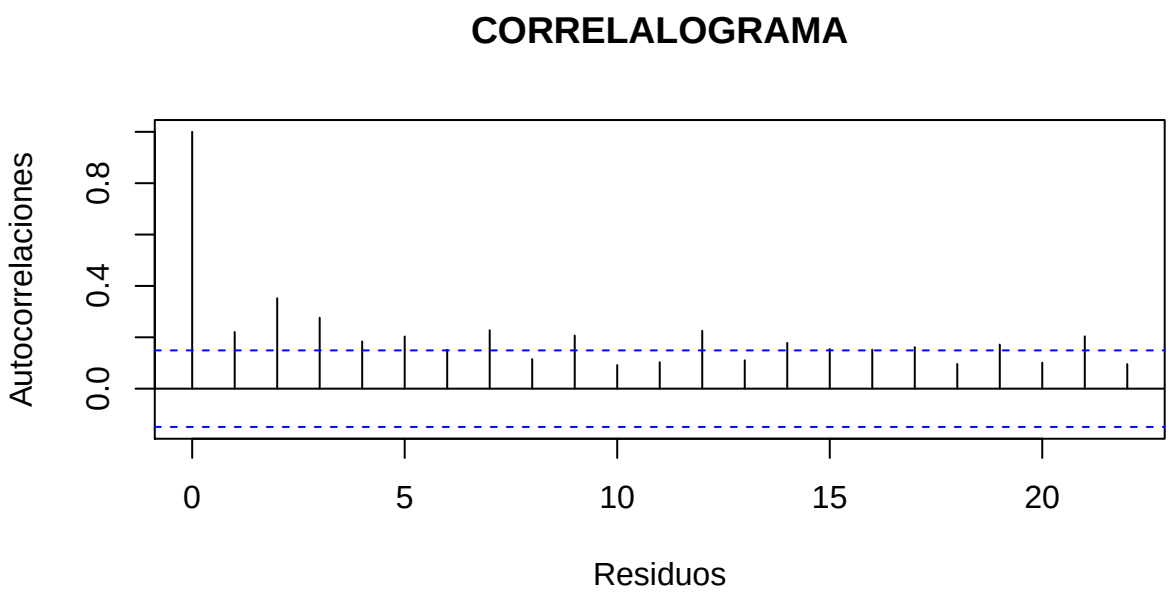
```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  RegresiónA2
```

```
## LM test = 9.6342, df = 1, p-value = 0.00191
```

Gráfico de correlación serial

Una ACF concocida en español como una función de autocorrelación permite investigar si existe autocorrelación o correlación serial en los residuales de una series de tiempo[7].

```
residA2 =rstandard(RegresiónA2)
acf(residA2, xlab="Residuos", ylab="Autocorrelaciones", main= "CORRELALOGRAMA")
```



Errores robustos a la heteroscedasticidad

Para realizar errores robustos en R es muy común usar el paquete *sandwich*.

```
#install.packages("sandwich")
library("sandwich")
```

El siguiente código, muestra:

- **vcovHC**: Para calcular la matriz de varianzas y covarianzas con errores robutos a la heterocedasticidad
- **coeftest**: coeficientes calculados luego de corregir por errores robustos a la heterocedasticidad

```
vcovHC(RegresiónA2) # matriz de varianzas y covarianzas con errores robutos
```

```
##                (Intercept)     democA log(expendA) log(expendB) log(prtystrA)
## (Intercept)    153.891139 -8.4161769   1.23479333  -1.68256748   -37.6094438
## democA          -8.416177  2.0627431  -0.34610460   0.21040202     2.0784100
## log(expendA)     1.234793 -0.3461046   0.32417539  -0.04203953    -0.6545296
## log(expendB)    -1.682567  0.2104020  -0.04203953   0.13241720     0.2883698
## log(prtystrA)  -37.609444  2.0784100  -0.65452961   0.28836984     9.8436567
```

```
coeftest(RegresiónA2, vcov=vcovHC(RegresiónA2)) # coeficientes calculados luego de corregir por errores
```

```
##
## t test of coefficients:
##
```

[7] Nuevamente, en este caso solo es un ejemplo ilustrativo dado que la correlación serial tiene sentido es en series de tiempo o en datos panel

```
##               Estimate Std. Error  t value  Pr(>|t|)
## (Intercept)    7.72651   12.40529   0.6228 0.5342344
## democA         3.48410    1.43623   2.4259 0.0163298 *
## log(expendA)   5.80933    0.56936  10.2032 < 2.2e-16 ***
## log(expendB)  -6.30647    0.36389 -17.3306 < 2.2e-16 ***
## log(prtystrA) 11.02277    3.13746   3.5133 0.0005688 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coeftest(RegresiónA2)
```

```
##
## t test of coefficients:
##
##               Estimate Std. Error  t value  Pr(>|t|)
## (Intercept)    7.72651   13.20433   0.5851 0.5592324
## democA         3.48410    1.38206   2.5210 0.0126353 *
## log(expendA)   5.80933    0.39207  14.8171 < 2.2e-16 ***
## log(expendB)  -6.30647    0.39458 -15.9826 < 2.2e-16 ***
## log(prtystrA) 11.02277    3.27489   3.3658 0.0009457 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Normalidad en los residuales

Se importa el paquete *tseries*[8] para poder utilizar el comando *jarque.bera.test*. El comando *jarque.bera.test* permite realizar una prueba de **Jarque Bera** la cula es una prueba formal para validar si los residuales presentan normalidad[9].

Test formal de Jarque Bera

```
#install.packages("tseries")
library("tseries")
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
resreg=residuals(RegresiónA2) # Calculamos los residuos
shapiro.test(resreg) # Ho= normalidad
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resreg
## W = 0.98648, p-value = 0.09422
```

```
jarque.bera.test(resreg)#Test Jarque Bera  (#Ho= normalidad)/Más apropiado para series de tiempo
```

```
##
##  Jarque Bera Test
##
## data:  resreg
```

[8] El paquete tseries sirve para manejar y manipular series de tiempo en R. Es muy usual emplearlo en series de tiempo

[9] Recuerden que si los residuales no presentan normalidad la inferencia estadistica convncional sin ningún tipo de correción no es correcta

```
## X-squared = 5.6599, df = 2, p-value = 0.05902
```

Histograma de los residuales

Una forma de corroborar si los residuales presentan una distribución normal es a partir de la gráfica del histograma de éstos. Si la forma del histograma parece a la función de densidad de una distribución gaussiana puede ser un fuerte indicativo que los errrores se distribuyen normal[10]

```
hist(resreg, freq=FALSE, main="Distribución de los Residuales", breaks = 20, prob=TRUE)
curve(dnorm(x, mean=mean(resreg), sd=sd(resreg)),col="darkblue", lwd=2, add=TRUE)
```

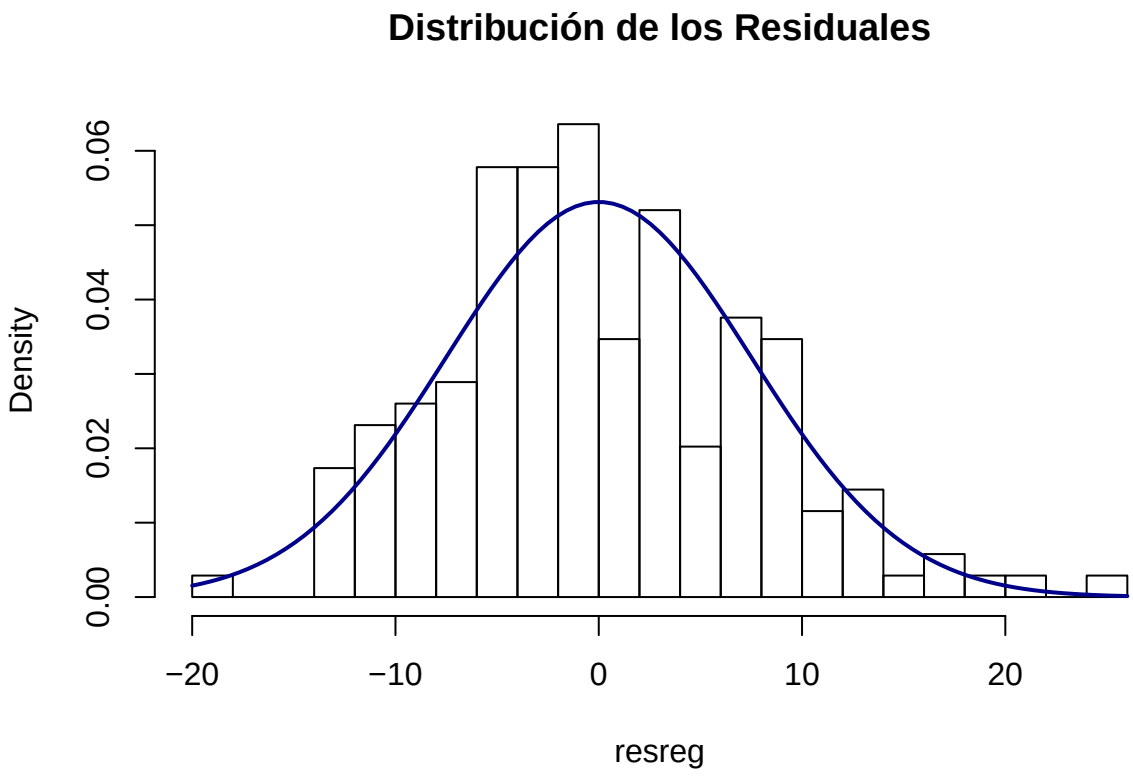


Grafico QQplot

Una gráfica de cuantiles-cuantiles o gráfica Q-Q es una forma de comparar gráficamente dos distribuciones. Por ello, estas gráficas son muy utilizadas para corroborar un supuestos de distribución de alguna muestra de datos. Por ejemplo, para que la inferencia estadistica usual sea correcta en un modelo de regresión lineal estimado por MCO es necesrio que los residuales se comporten como si provinieran de una distribución normal.

Una gráfica tipo QQ-Plot permite comparar el comportamiento/distribución de los residuales, respecto a una distribución normal teórica. Es decir, se comparan los cuantiles teóricos de una distribución normal con los muestrales que resultarian de la distribución de los residuales. Es muy usual utilizar esta herramienta gráfica para validar el supusto de normalidad en los residuales dado que muchos test de normalidad no son muy robustos para algunas muestras de datos.

Gráficamente, entre más cerca se distribuyan los *puntos*, que representan los cuantiles de la distribución muestral de los datos[11], respecto a una línea, que representaría la distribución teórica que se quiere corroborar

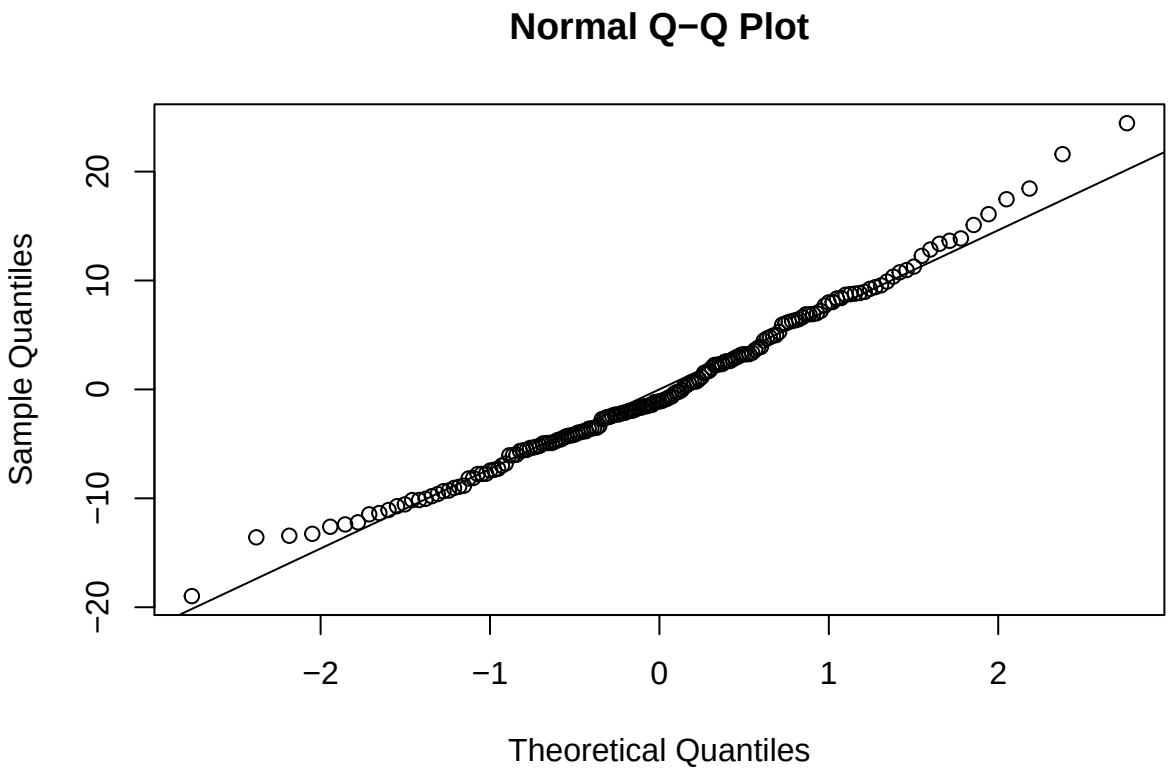
[10] Recordar que la distribución normal se caractriza por tener colas livianas (i.e. por no tener muchos valores atipicos en sus colas y concentrar la mayor parte de los resultados alrededor de su media.)

[11] Nuevamente es muy usual utilizar esta prueba en los residuales de una regresión para validar el supuesto de linealidad

en los datos[12], más cercano será la distribución muestral a la distribución teórica que se quiere validar en los datos.

Una manera rápida de realizar una qqplot es utilizando las funciones qqnorm y qqline que R provee por defecto:

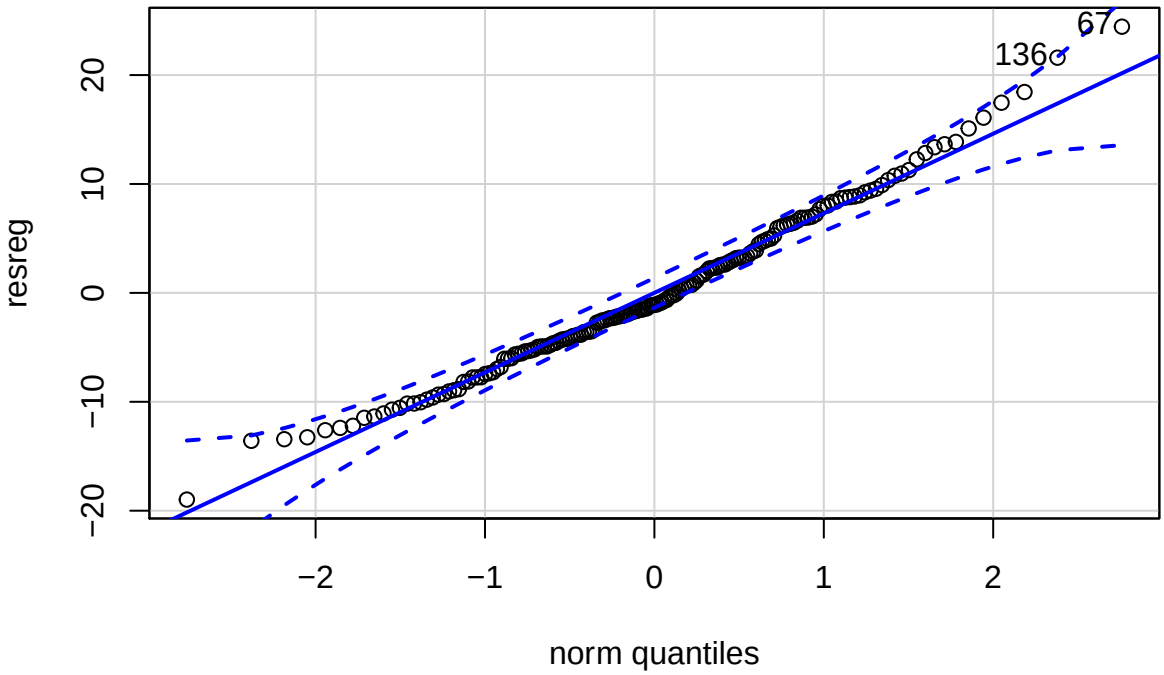
```
qqnorm(resreg)
qqline(resreg)
```



No obstante, una mejor manera de realizar la regresión es mediante el comando *qqPlot* del paquete *car*:

```
#install.packages("car")
library(car)
qqPlot(resreg)
```

[12]Nuevamente, una de las distribuciones de comparación más usuales es la normal




```
## [1]  67 136
```

Como pueden obsrvar en la gráfica realizada mediante el comando qqPlot la línea azul continua representa los cuantiles de la distribución teórica, en este caso normal, mientras que el conjunto de puntos representa la distribución muesral. Las líneas punteadas azules representan los intervalos de confianza, que se interpretan de la manera usual. Por los resultados de la gráfica, se podría decir que la distribución de los residauales es aproximadamente normal, salvo algunos valores atípicos al final de las colas.

Para las personas más interesadas en aprender más sobre qq-plots es recomendable leer el siguiente artículo: QQplots en R!