

Econometría I

Clase 0

Isaura Garcia¹ Elvis Espinal²

¹isauragav@gmail.com

²elvis.espinal95@gmail.com

Mayo, 2018

- 1 Introducción a R
- 2 Operaciones en R
- 3 Gestión de datos en R
- 4 Loops en R
- 5 Asignaciones

Introducción a R

R es un ambiente de programación gratuito con aplicaciones en multiples ámbitos, entre sus posibles usos se destacan:

- Interfáz efectiva para el manejo y almacenamiento de datos.
- Suite para realizar operaciones entre matrices y otros arreglos n-dimensionales.
- **Colección de herramientas para análisis estadístico y visualización**
- Medio para programar prácticamente cualquier cosa ¹²

¹Para mayor información consultar: [r-project](http://r-project.org)

²Ejemplos de aplicaciones desarrolladas en R: Shiny

Ventajas y desventajas de R

Pros

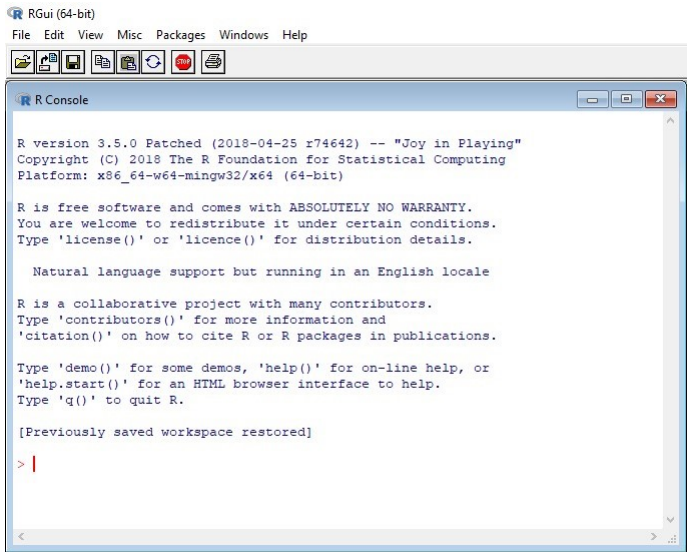
- Flexibilidad
- Desarrollado para análisis estadístico
- Disponibilidad de paquetes y librerías creadas por usuarios
- Comunidad activa, siempre en la frontera de las ciencias

Cons

- Sintaxis poco legible en comparación a otros lenguajes (ej:Python)
- Empresas grandes suelen ver con desdén a softwares open-source (Aunque esto cambia poco a poco)
- Mucha flexibilidad también implica que los usuarios deben tener más control de lo que hacen

Ambiente en R

Al abrir R, el display que podemos ver es el siguiente:



The screenshot shows the RGui (64-bit) application window. The title bar reads "RGui (64-bit)". The menu bar includes "File", "Edit", "View", "Misc", "Packages", "Windows", and "Help". The toolbar contains icons for file operations (open, save, print, etc.) and a red stop button. The main window is titled "R Console" and displays the following text:

```
R version 3.5.0 Patched (2018-04-25 r74642) -- "Joy in Playing"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

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

Natural language support but running in an English locale

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

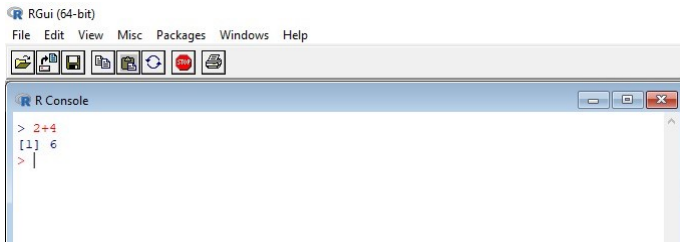
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> |
```

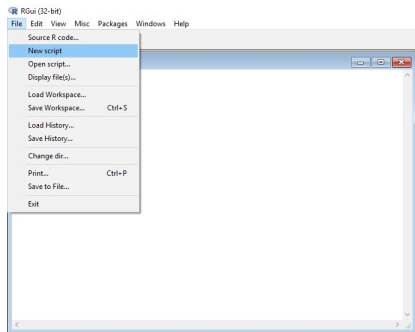
Ambiente en R: Consola

La vista actual es lo que comunmente se conoce como consola, aqui se pueden introducir comandos linea por linea para su ejecución.



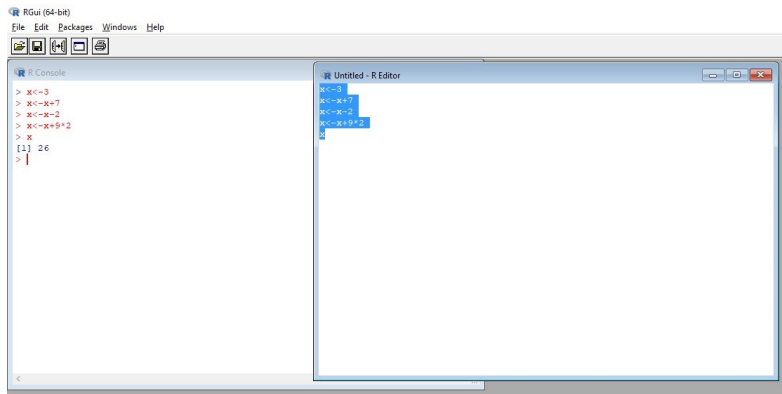
Ambiente en R: Scripts

Una forma más conveniente de trabajar en R cuando se quieren hacer trabajos extensos es abrir un *Script*, que permite al usuario correr bloques de código



Ambiente en R: Corriendo Scripts

Sombreado una parte del código en el script y ingresando *CTRL+R* podemos ver los resultados en la consola



Entre los tipos de elementos en R, se encuentran:

- Carácteres ej. "a","b"
- Numéricos ej. 1,0.17,1e8
- Enteros ej. 1,7,364
- Lógicos ej. TRUE,FALSE,1,0

Entre otros, cada uno de estos tiene su utilidad particular.

En R los objetos se pueden clasificar en dos grupos:

- Atómicos: Conjuntos en los que todos los elementos son del mismo tipo (vectores y matrices).
- Recursivos: Pueden combinar colecciones de elementos de diferentes naturalezas (dataframes, listas).

Operaciones en R

Operadores Matemáticos en R

R puede servir como una calculadora, para realizar calculos entre elementos:

```
x<-c(3,7)  
sum(x)
```

```
## [1] 10
```

```
prod(x)
```

```
## [1] 21
```

```
x[1]-x[2]
```

```
## [1] -4
```

```
log(x)
```

```
## [1] 1.098612 1.945910
```

Operadores Estadísticos en R

Igualmente, se pueden realizar cálculos de estadísticos descriptivos:

```
x<-c(2,7,8,12,14,374)  
mean(x)
```

```
## [1] 69.5
```

```
median(x)
```

```
## [1] 10
```

```
range(x)
```

```
## [1] 2 374
```

```
IQR(x)
```

```
## [1] 6.25
```

Operadores lógicos en R

```
x<-c(2,7,8,12,14,374)  
x[2]!=x[1]
```

```
## [1] TRUE
```

```
x[3]==x[2]
```

```
## [1] FALSE
```

```
x[2]!=x[1] & x[3]==x[2]
```

```
## [1] FALSE
```

```
x[2]!=x[1] | x[3]==x[2]
```

```
## [1] TRUE
```

Operadores lógicos en R

También existe la opción de indexación lógica en R

```
x<-c(2,7,8,12,14,374)  
x[x==2]
```

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

```
x[x>=7]
```

```
## [1] 7 8 12 14 374
```

```
x[x>7 & x<15]
```

```
## [1] 8 12 14
```

²Para más funciones: ref-card

Funciones lógicas en R

Son funciones que se aprovechan de las propiedades de las operaciones lógicas para producir resultados, en R vienen dados como *if*, *else* y/o *else if*.

Funcionan evaluando operaciones y desarrollando las acciones deseadas, por ejemplo:

```
a<-9
if(a>12){
  print("A es mayor que 12")
} else if (a<12){
  print("A es menor a 12")
} else {
  print("A es igual a 12")
}
```

```
## [1] "A es menor a 12"
```

Gestión de datos en R

Dataframes en R

Ya mencionamos los dataframes anteriormente, son estructuras de datos que permiten almacenar diferentes tipos de variables, estas pueden ser obtenidas cargándose de una base disponible, o, simplemente importando una base de tu ordenador:

```
q<-mtcars  
class(q)
```

```
## [1] "data.frame"
```

```
nrow(q)
```

```
## [1] 32
```

```
j<-read.csv(file='dummy_data.csv',header=T,  
            stringsAsFactors = F)  
class(j)
```

```
## [1] "data.frame"
```

Dataframes en R

Para verificar que todo anda bien con nuestros dataframes, podemos verificar la data completa o columnas en particular

```
head(q,1)
```

```
##           mpg cyl  disp  hp  drat    wt  qsec vs am gear carb
## Mazda RX4   21   6  160 110   3.9 2.62 16.46  0  1    4    4
```

```
head(j,1)
```

```
##      Numero Letra Oracion Logico
## 1         1      A      AAA      1
```

```
class(j$Numero)
```

```
## [1] "integer"
```

Variables dentro de dataframes

Podemos identificar variables dentro de un dataframe y hacer cálculos puntuales de esta variable

```
summary(q$mpg)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    10.40   15.43   19.20   20.09   22.80   33.90
```

```
quantile(q$mpg,0.75)
```

```
## 75%
## 22.8
```

Variables dentro de dataframes

También podemos renombrar variables, o añadir variables calculadas al dataframe

```
# Añadiendo variable
```

```
q$new_var<-ifelse(q$mpg>=20,T,F)  
summary(q$new_var)
```

```
##      Mode      FALSE      TRUE  
## logical      18      14
```

```
# Modificando variable
```

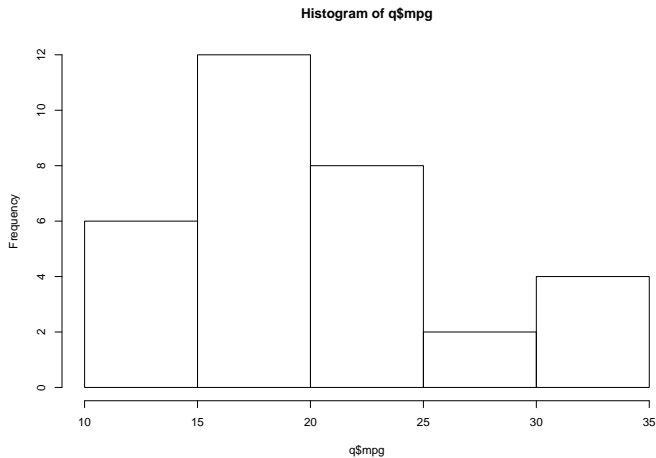
```
names(q)[names(q)=='new_var']<-'Bajo_consumo'  
drop<-c("drat", "gear", "carb")  
q<-q[,!names(q)%in% drop ]  
head(q,1)
```

```
##           mpg cyl disp  hp   wt  qsec vs am Bajo_consumo  
## Mazda RX4   21   6  160 110 2.62 16.46  0  1           TRUE
```

Visualización de datos en R

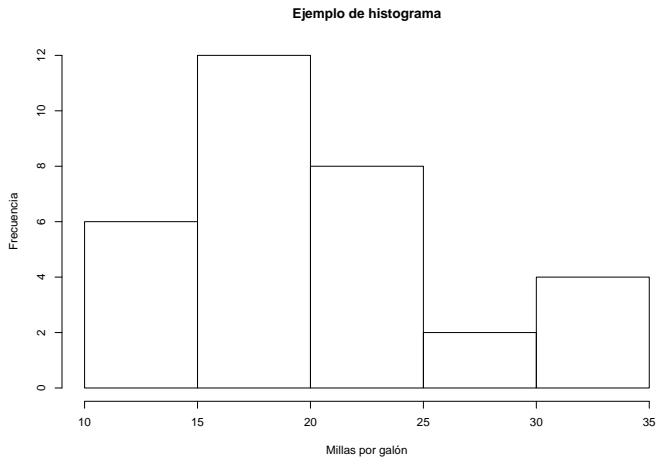
```
# Visualización básica
```

```
hist(q$mpg)
```



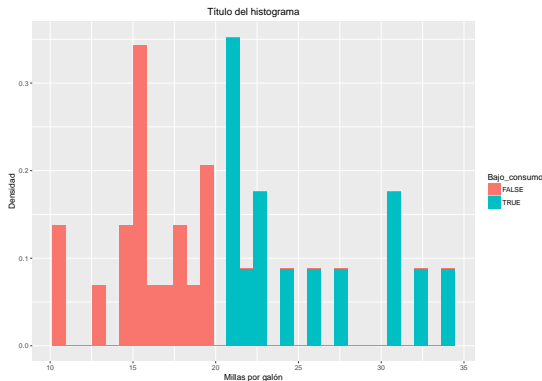
Visualización en R

```
# Visualización básica con modificaciones  
hist(q$mpg, xlab='Millas por galón', ylab='Frecuencia'  
      , main='Ejemplo de histograma')
```



Visualización en R

```
# Visualizacion Avanzada
library(ggplot2)
ggplot(q, aes(x=mpg))+
  geom_histogram(aes(y=..density.., fill=Bajo_consumo), show.legend=T, bins=30)+
  labs(y='Densidad', x='Millas por galón', title='Título del histograma')+
  theme(plot.title = element_text(hjust=0.5))
```



Entre los gráficos que se pueden realizar con R, se encuentran:

- Diagramas de caja (`boxplot()`)
- Diagramas de dispersión (`scatter()`)
- Gráficos de barras (`barplot()`)
- Gráficos de pastel (`pie()` y `pie3D()`)

También son invitados a explorar la librería `ggplot2`, que es una de las librerías de visualización mas usadas en R.

Loops en R

Que es un “Loop”

Un Loop (o Bucle en español) se refiere a una tarea o proceso que, a pesar de ser especificado una sola vez, se realiza varias veces.

En R, veremos 2 tipos de loops:

- Loop While
- Loop For

While loops

Los “while” loops son aquellos en los que se realiza el proceso hasta que uno o varios parametros alcancen una condición pre-determinada

```
m<-7
while (m>1){
print(m)
m=m-2
}
```

```
## [1] 7
## [1] 5
## [1] 3
```

Algo sumamente importante en los while loops es que hay que asegurarse que las condiciones de parada esten bien definidas, pues de no alcanzarse podría iterar para siempre

For loops

Los “For” loops son una versión *beginner-friendly* de los “While” loops, con la salvedad de que realiza el proceso una cantidad pre-determinada de veces, podemos imaginarlo como un “while” loop en el que la condicion de parada es el número de iteraciones

```
n_veces=6
j=0
for(j in 1:n_veces){
  print(j*2)
}
```

```
## [1] 2
## [1] 4
## [1] 6
## [1] 8
## [1] 10
## [1] 12
```

Asignaciones

Asignaciones: Manejo de datos

- 1 Instalar R.
- 2 Dentro de R, cargar el dataframe 'mtcars'.
- 3 Ver las estadísticas descriptivas de todas las variables.
- 4 Hacer un diagrama de dispersion con millas por galón (mpg) en el eje x y caballos de fuerza (hp) en el eje y.
- 5 Crear una variable que tome valor 1 cuando el número de cambios (gear) sea mayor que 3 o los caballos de fuerza (hp) sean mayores a 180, de lo contrario su valor sería 0, llamarla 'rapido'
- 6 Repetir el diagrama del punto 4, solo con las obs que tengan 'rapido'=1

- 1 Crear un For loop que, para los números del 1 al 20 retorne TRUE si es divisible entre 4 y FALSE si no lo es. tip(buscar el operador %%)
- 2 Crear un While loop que tome un valor k, lo inicialice en 5 y en cada iteración reste 0.015 a este valor hasta que sea menor o igual a 4. Para evitar que la computadora itere por demasiado tiempo, defina en el while que solo se puede llegar hasta 50 iteraciones.

- 1 Crear un código que tome una cantidad de dinero (número) entera como parámetro y, dado que solo se tienen solo billetes de 50,20,10,5 y 1, el código deberá calcular la denominación y cantidad de los billetes a devolver para retornar la menor cantidad de billetes posible. Por ejemplo para 61 debería retornar 1 de 50, 1 de 10 y 1 de 1.
- 2 Un matemático famoso propuso hace mucho tiempo una sucesión matemática sumamente reconocida y estudiada. Los primeros dos valores son 1 y del 3er termino en adelante vienen dados por la formula: $fib_i = fib_{i-1} + fib_{i-2}$. Cree un programa que, recibiendo un parámetro entero n, retorne el n-esimo termino de la sucesión.