Computación Estadística con R

Clase 1

Rubén Soza

Programa del Curso

- Introducción a R: Instalación, Interfaz y Operatoria básica.
- Manipulación de bases de datos en R.
- Análisis exploratorio y descriptivo en R: Estadísticas de Resumen y Visualización.
- Análisis estadístico en R: Pruebas de hipótesis, regresión y clasificación.
- Creación de reportes utilizando Rmarkdown.
- Otros.

Clases

- Cada clase consistirá de un 25% de exposición y un 90% de ejercitación.
- **Todos** escribiremos código. *Hechando a prender se aprende!*
- Todo el material estará en la página de Educación Continua.

Introducción a R

Preguntas Frequentes

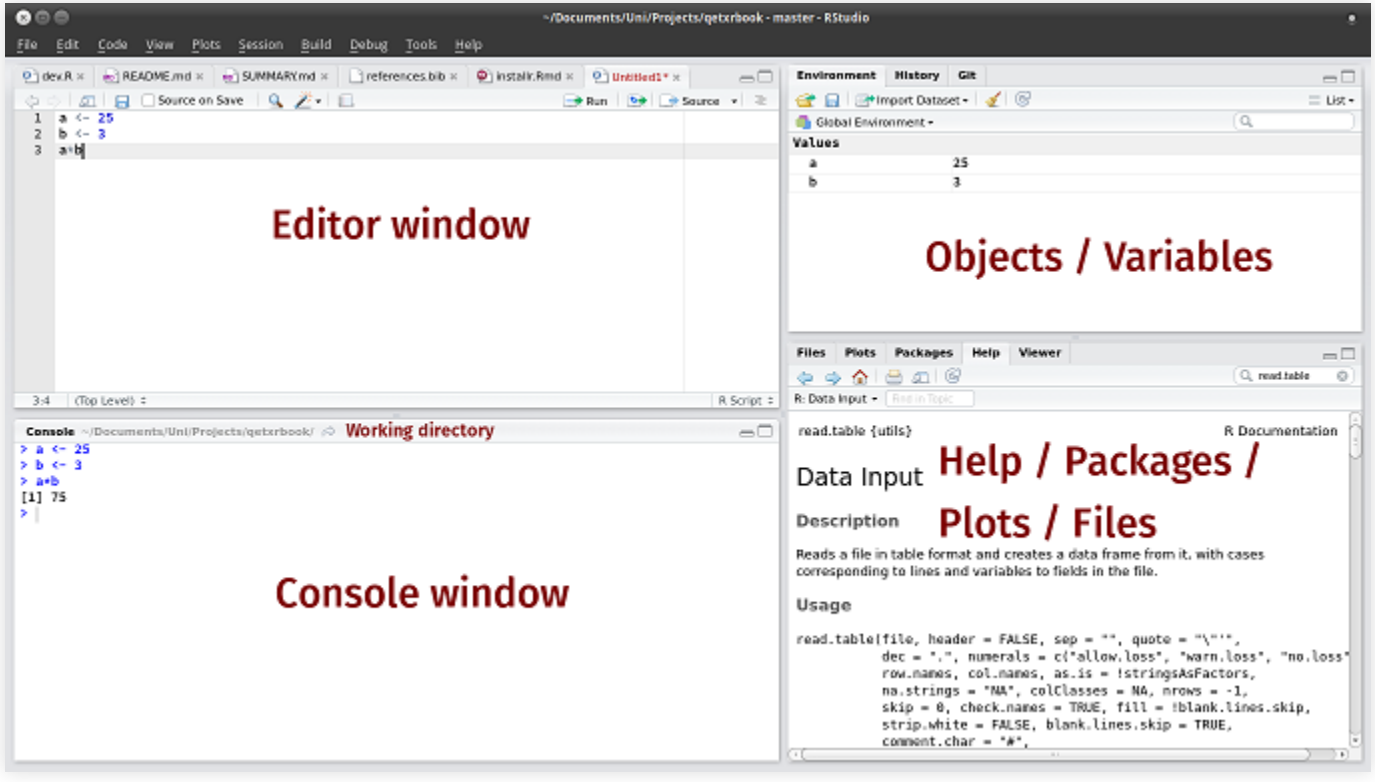
The best thing about R is that it was developed by statisticians. The worst thing about R is that ... it was developed by statisticians.

- Un software *libre* para computación estadística y gráficos con cerca de 30 años.
- ¿Por qué? Un software *libre* para computación estadística y gráficos.
- ¿Lo bueno? Un lenguaje simple. Miles de *paquetes* desarrollados por la comunidad. Además es útil para el desarrollo de la academia.
- ¿Lo meh? No es lo más rápido. No posee la mejor escalabilidad.
- ¿A qué se parece? A otros lenguajes de programación como **python**.

Descarga e Instalación

- R, el software computacional, se descarga desde el CRAN. Debes elegir la opción que corresponda con tu sistema operativo.
- Rstudio, el panel de control, se descarga desde Rstudio. Elegir la primera opción, “RStudio Desktop Open Source License”.
- Instalar ambos programas de la forma usual.
- Además existe una versión online, Rstudio Cloud

Interfaz básica de RStudio



Primeras Interacciones con R

R como calculadora aritmética

En R se pueden realizar todas las operaciones aritméticas:

```
sqrt(2^4 + exp(3)/55 - log(5*8-2))
```

```
## [1] 3.567577
```

Nota: Para correr código desde el editor, utilizar *ctrl + enter*.

R como calculadora lógica

Además, se pueden realizar operaciones lógicas(&,|), las cuales retornan como resultado TRUE o FALSE:

```
38 >= 15
```

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

```
3 < 5 & 6 < 5
```

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

```
3 < 5 | 6 < 5
```

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

Objetos básicos de R

En R podemos guardar objetos utilizando las asignación '<-' o, en su defecto, '='.

```
a <- 20 # Valor numérico
b = 3^2
c <- "Hola Mundo" # String o carácter.
```

Para visualizar un objeto en consola, basta con escribir su nombre en ella, o bien ejecutar la línea deseada desde el editor.

```
a + b
```

```
## [1] 29
```

Objetos básicos de R: Vectores

Para crear un vector se ocupa la función **c()**. Además podemos crear secuencias con la función **seq()**.

```
x <- c(5,b,7,8,-8,20,7,a)
y <- seq(1,10)
mean(x)
```

```
## [1] 8.5
```

```
sum(y)
```

```
## [1] 55
```

Objetos básicos de R: Vectores

Podemos acceder a un elemento de un vector en una posición específica de un vector utilizando '[]'. Algunos ejemplos:

```
x[3] # Elemento en la posición 3
```

```
## [1] 7
```

```
x[2:4] # Elementos en las posiciones 2 y 4 inclusive
```

```
## [1] 9 7 8
```

Objetos básicos de R: Vectores

```
x[c(5,8)] # Elementos en la posición 5 y 8
```

```
## [1] -8 20
```

```
x[-4] # Vector original sin el elemento en la posición 4
```

```
## [1]  5  9  7 -8 20  7 20
```

Objetos básicos de R: Paquetes

Los paquetes de R son el eje central de su funcionamiento. En cada uno de ellos existen funciones desarrolladas para resolver diferentes tipos de problemáticas. En esta ocasión instalaremos y cargaremos el paquete 'Tidyverse': 

Objetos básicos de R: Paquetes

Para utilizar un paquete en R hay que realizar las siguientes operaciones:

- Instalar el paquete, **install.packages('nombre_paquete').**
- Cargar el paquete, **library('nombre_paquete')**.

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

Operador %>%

Permite realizar composición de funciones. Un ejemplo de su utilización es:

```
x %>% mean() %>% log()
```

```
## [1] 2.140066
```

```
log(mean(x))
```

```
## [1] 2.140066
```

Ejemplo en RStudio

Actividad 1

- Genere un vector con los primeros 1000 números impares.
- Del vector anterior, obtenga los impares número 1, 10, 100 y 1000.
- Cálcule la suma de la raíz de los números generados antes utilizando 2 métodos diferentes.

Importación de Base de Datos

Orígen de la BD

Los datos pueden provenir de muchas fuentes:

- Archivos de texto(txt ó csv)
- Excel(xlsx)
- SPSS(sav)
- SQL(sql)
- STATA(dta)
- Una página web.
- etc.

Funciones para Importar

Cada fuente tiene su función de importación en tidyverse.

- Si es csv: **read.csv**.
- Si es texto: **read_delim**.
- Si es excel: **read_excel**.
- Si es spss: **read_sav**.

Para más información, pueden entrar al siguiente torpedo.

Ejemplo: storms.csv

La base de datos storms se encuentra en el siguiente link.

```
url  <- 'https://raw.githubusercontent.com/rstudio/EDAWR/master/data-raw/storms.csv'
download.file(url, "storms.csv", mode = "wb")

library(readr)
storms <- read.csv('storms.csv', header = T)
storms
```

```
##     storm wind pressure       date
## 1 Alberto  110     1007 2000-08-03
## 2    Alex   45     1009 1998-07-27
## 3 Allison   65     1005 1995-06-03
## 4     Ana   40     1013 1997-06-30
## 5  Arlene   50     1010 1999-06-11
## 6  Arthur   45     1010 1996-06-17
```

EJEMPLO: STORMS.CSV

El comando **glimpse()** nos otorga características de las columnas(variables) de la BD.

```
glimpse(storms)
```

```
## Observations: 6
## Variables: 4
## $ storm    <fct> Alberto, Alex, Allison, Ana, Arlene, Arthur
## $ wind     <int> 110, 45, 65, 40, 50, 45
## $ pressure <int> 1007, 1009, 1005, 1013, 1010, 1010
## $ date     <fct> 2000-08-03, 1998-07-27, 1995-06-03, 1997-06-30, 1999-...
```

Ejemplo: numeros.xlsx

```
library(readxl)
numeros <- read_excel('Datasets/numeros.xlsx')
numeros
```

```
## # A tibble: 16 x 3
##    region cuarto valor
##    <chr>  <chr>  <dbl>
##  1 sur    Q1       100
##  2 sur    Q2       150
##  3 sur    Q3       225
##  4 sur    Q4       290
##  5 norte  Q1       150
##  6 norte  Q2       160
##  7 norte  Q3       180
##  8 norte  Q4       300
##  9 este   Q1       180
## 10 este   Q2       200
## 11 este   Q3       200
## 12 este   Q4       240
## 13 oeste  Q1       250
## 14 oeste  Q2       250
## 15 oeste  Q3       300
```

Veamos ahora un ejemplo en RStudio

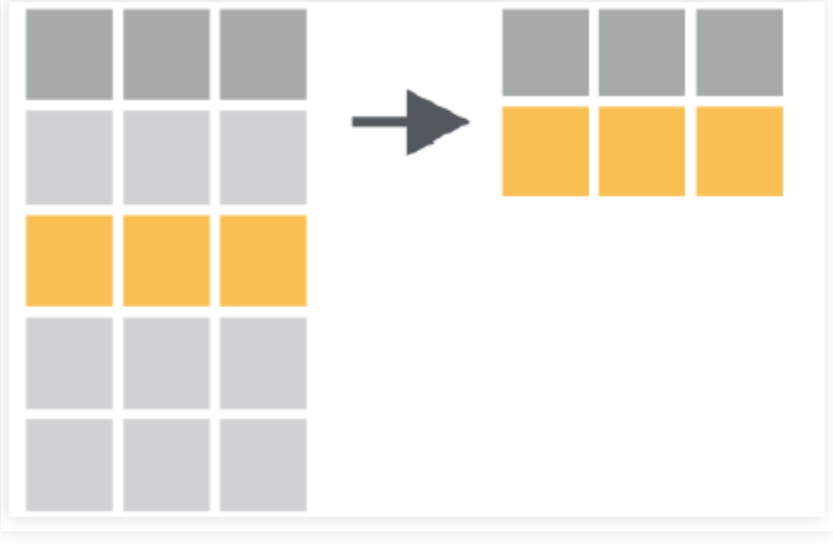
Actividad 2

Dirijase a http://datos.gob.cl y descargue dos bases de datos en diferentes formatos. Lea dichas BD en R y obtenga las características de sus variables.

Manejo de bases de datos utilizando funciones de dplyr

FILTER(): *SELECCIONAR FILAS*

Permite seleccionar filas de cierta BD utilizando un criterio particular.



FILTER(): *CÓDIGO*

```
filter(storms, storm %in% c("Alberto", "Ana"))
```

```
##     storm wind pressure       date
## 1 Alberto  110     1007 2000-08-03
## 2     Ana   40     1013 1997-06-30
```

```
storms %>%
  filter(storm %in% c("Alberto", "Ana"))
```

```
##     storm wind pressure       date
## 1 Alberto  110     1007 2000-08-03
## 2     Ana   40     1013 1997-06-30
```

SELECT(): *SELECCIONAR COLUMNAS*

Permite seleccionar columnas de una base de datos en específico.

SELECT() : *CÓDIGO*

```
select(storms, storm, pressure)
```

```
##     storm pressure
## 1 Alberto     1007
## 2    Alex     1009
## 3 Allison     1005
## 4     Ana     1013
## 5  Arlene     1010
## 6  Arthur     1010
```

```
storms %>%
  select(storm, pressure)
```

```
##     storm pressure
## 1 Alberto     1007
## 2    Alex     1009
## 3 Allison     1005
## 4     Ana     1013
## 5  Arlene     1010
## 6  Arthur     1010
```

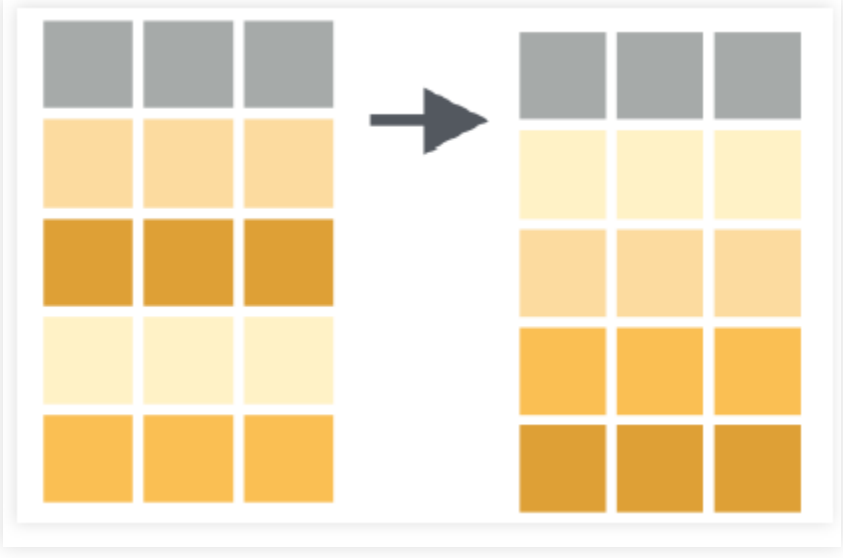
SELECT(): *CÓDIGO ALTERNATIVO*

```
storms %>%
  select(-wind, -date)
```

```
##     storm pressure
## 1 Alberto     1007
## 2    Alex     1009
## 3 Allison     1005
## 4     Ana     1013
## 5  Arlene     1010
## 6  Arthur     1010
```

ARRANGE(): *ORDENAR FILAS*

Permite ordenar de menor a mayor una BD teniendo en consideración una o más variables.



ARRANGE(): *CÓDIGO*

```
arrange(storms, wind)
```

```
##     storm wind pressure       date
## 1     Ana   40     1013 1997-06-30
## 2    Alex   45     1009 1998-07-27
## 3  Arthur   45     1010 1996-06-17
## 4  Arlene   50     1010 1999-06-11
## 5 Allison   65     1005 1995-06-03
## 6 Alberto  110     1007 2000-08-03
```

```
storms %>%
  arrange(wind)
```

```
##     storm wind pressure       date
## 1     Ana   40     1013 1997-06-30
## 2    Alex   45     1009 1998-07-27
## 3  Arthur   45     1010 1996-06-17
## 4  Arlene   50     1010 1999-06-11
## 5 Allison   65     1005 1995-06-03
## 6 Alberto  110     1007 2000-08-03
```

arrange(): *Código de mayor a menor.*

```
storms %>%
  arrange(desc(wind))
```

```
##     storm wind pressure       date
## 1 Alberto  110     1007 2000-08-03
## 2 Allison   65     1005 1995-06-03
## 3  Arlene   50     1010 1999-06-11
## 4    Alex   45     1009 1998-07-27
## 5  Arthur   45     1010 1996-06-17
## 6     Ana   40     1013 1997-06-30
```

mutate(): *Crear/Modificar Columnas*

Permite crear o modificar una columna de la BD.

MUTATE(): *CÓDIGO*

```
storms %>%
  mutate(ratio = pressure/wind, inverse = 1/ratio)
```

```
##     storm wind pressure       date     ratio    inverse
## 1 Alberto  110     1007 2000-08-03  9.154545 0.10923535
## 2    Alex   45     1009 1998-07-27 22.422222 0.04459861
## 3 Allison   65     1005 1995-06-03 15.461538 0.06467662
## 4     Ana   40     1013 1997-06-30 25.325000 0.03948667
## 5  Arlene   50     1010 1999-06-11 20.200000 0.04950495
## 6  Arthur   45     1010 1996-06-17 22.444444 0.04455446
```

SUMMARISE(): *RESUMIR COLUMNAS*

Permite aplicar funciones de resumen en las columnas de una BD.

SUMMARISE(): *CÓDIGO*

```
storms %>%
  summarise(mean = mean(wind), sd = sd(wind))
```

```
##       mean       sd
## 1 59.16667 26.34704
```

Ejemplo en RStudio: storms.csv

- Seleccione las observaciones que tienen pressure mayor o igual a 1010 y wind menor a 50. Encuentre el promedio y la desviación estándar de wind en la BD resultante.
- Divida por 1000 la variable pressure.
- Quite la variable pressure y ordene de mayor a menor según fecha la base de datos resultante.

GROUP_BY + SUMMARISE: *RESUMIR COLUMNAS POR GRUPOS*

Group_by divide la base de datos en grupos, lo cual permite obtener medidas de resumen por grupos utilizando summarise.

GROUP_BY + SUMMARISE: *CÓDIGO*

```
pollution <- read.csv("https://raw.githubusercontent.com/rstudio/EDAWR/master/data-ra
```

```
pollution %>%
  group_by(city) %>%
  summarise(promedio = median(amount),
            suma = sum(amount),
            n = n(),
            max = max(amount))
```

```
## # A tibble: 3 x 5
##    city     promedio  suma     n   max
##    <fct>       <dbl> <int> <int> <int>
## 1 Beijing      88.5   177     2   121
## 2 London       19      38     2    22
## 3 New York     18.5    37     2    23
```

Ejemplo en RStudio: numeros.xlsx

Encuentre el máximo y mínimo por región de la variable valor.

Actividad 3

Teniendo en consideración la BD encuesta.xlsx:

- Seleccione Región, Sexo, Edad, cuánto gastó en seguridad y Score Socioeconómico.
- Seleccione hombres de Valparaíso.
- Ordene de menor a mayor la edad.
- Añada una nueva variable denominada PRSC, calculada como el Score del individuo dividido por el máximo del score observado.
- Obtenga el promedio del PRSC para cada grupo de gasto en seguridad.

Manipulación de BD parte 2