Notas de clase del curso de introducción a Data Science

Diego Kozlowski y Natsumi Shokida

2019-08-20

Contents

Chapter 1

Temario

- **clase 1**: Introducción al entorno R
- **clase 2**: Tidyverse. (limpieza y organización de datos)
- **clase 3**: Visualización de la información
- **clase 4**: Estadística descriptiva
- **clase 5**:
- **clase 6**:
- **clase 7**:
- **clase 8**:
- **clase 9**:
- **clase 10**:

1.0.0.1 Librerias a instalar

```
install.packages(c("tidyverse","openxlsx",'ggplot2','ggthemes', 'ggrepel','ggalt','kableExtra','C
```

Chapter 2

Introducción a R

2.1 Explicación

2.1.1 ¿Que es R?

- Lenguaje para el procesamiento y análisis estadístico de datos
- Software Libre
- Sintaxis Básica: R base
- Sintaxis incremental[1]: El lenguaje se va ampliando por aportes de Universidades, investigadores/as, usuarios/as y empresas privadas, organizados en librerías (o paquetes)
- Comunidad web muy grande para realizar preguntas y despejar dudas.
- Graficos con calidad de publicación

El *entorno* más cómodo para utilizar el *lenguaje* **R** es el *programa* **R studio**

- Rstudio es una empresa que produce productos asociados al lenguaje R, como el programa sobre el que corremos los comandos, y extensiones del lenguaje (librerías).
- El programa es *gratuito* y se puede bajar de la página oficial

2.1.2 Diferencias con STATA y SPSS

- Gratuito
- Funciona principalmente por líneas de código (Aunque ya hay paquetes que permiten ejecutar comandos desde el menú y los botones sin tener que esribir código)
- Trabaja las bases de microdatos de forma virtual y no fisica, lo que permite disponer de varias al mismo tiempo sin mayor dificultad (no requiere

[1] Más allá de los comandos elementales, comandos más sofisticados tienen muchas versiones, y algunas quedan en desuso en el tiempo.

Figure 2.1: https://cran.r-project.org/

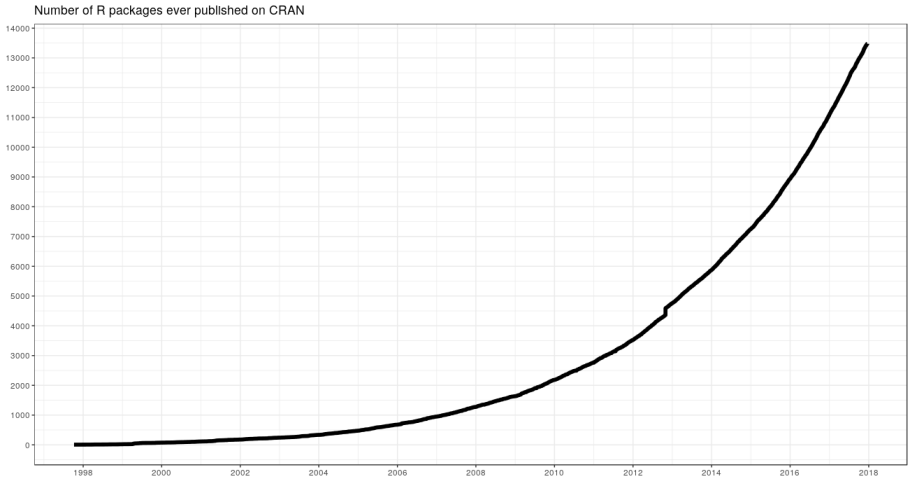

Figure 2.2: fuente: https://gist.github.com/daroczig/3cf06d6db4be2bbe3368

Figure 2.3: https://www.rstudio.com/

Rstudio.bb

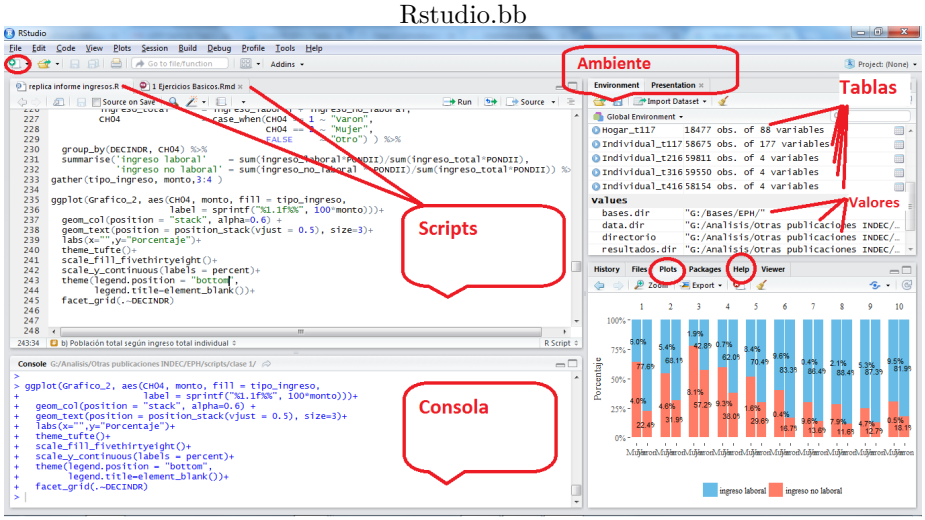

Figure 2.4: Pantalla Rstudio

abrir cada base, trabajarla por separado y luego cerrarla), ni exije guardar fisicamente los cambios.

- Más potente
  - Totalmente automatizable
  - Aportes de usuarias y usuarios
  - Extensible a otros lenguajes y usos (presentación como esta, diseño de aplicaciones)
- Más veloz:

2.1.3 Lógica sintáctica.

2.1.3.1 Definición de objetos

Los **Objetos/Elementos** constituyen la categoría escencial del R. De hecho, todo en R es un objeto, y se almacena con un nombre específico que **no debe poseer espacios**. Un número, un vector, una función, la progresión de letras del abecedario, una base de datos, un gráfico, constituyen para R objetos de distinto tipo. Los objetos que vamos creando a medida que trabajamos pueden visualizarse en la panel derecho superior de la pantalla.

El operador `<-` sirve para definir un objeto. **A la izquierda** del `<-` debe ubicarse el nombre que tomará el elemento a crear. **Del lado derecho** debe ir la definición del mismo

```
A <- 1
```

Al definir un elemento, el mismo queda guardado en el ambiente del programa, y podrá ser utilizado posteriormente para observar su contenido o para realizar

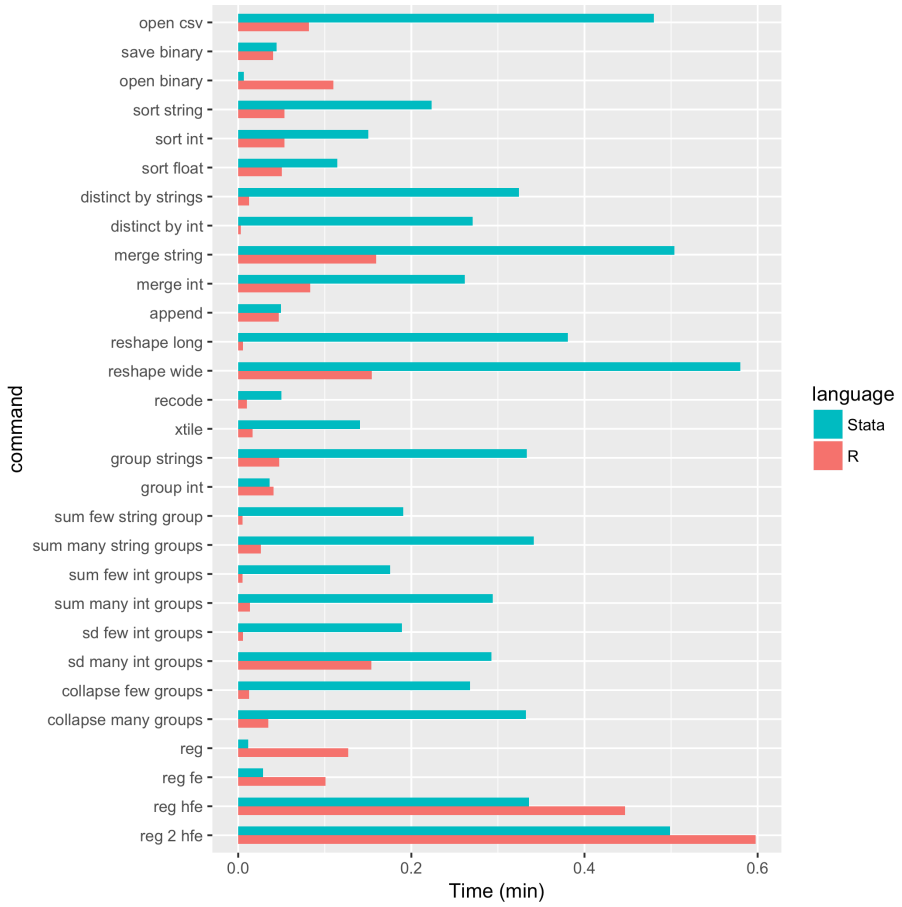

Figure 2.5: fuente: https://github.com/matthieugomez/benchmark-stata-r/blob/master/output/1e7.png

una operación con el mismo

```
A
```

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

```
A+6
```

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

Al correr una linea con el nombre del objeto, la consola del programa nos muestra su contenido. Entre Corchetes Observamos el número de orden del elemento en cuestión

El operador `=` es **equivalente** a `<-`, pero en la práctica no se utiliza para la definición de objetos.

```
B = 2
B
```

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

`<-` es un operador **Unidireccional**, es decir que:
`A <- B` implica que **A** va tomar como valor el contenido del objeto **B**, y no al revés.

```
A <- B
A   #Ahora A toma el valor de B, y B continua conservando el mismo valor
```

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

```
B
```

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

2.1.4 R base

Con *R base* nos referimos a los comandos básicos que vienen incorporados en el R, sin necesidad de cargar librerías.

2.1.4.1 Operadores lógicos:

- $>$
- $>=$
- $<$
- $<=$
- $==$
- $!=$

```
#Redefinimos los valores A y B
A <-  10
B  <-  20
#Realizamos comparaciones lógicas
```

```
A >  B
```

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

```
A >= B
```

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

```
A <  B
```

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

```
A <= B
```

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

```
A == B
```

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

```
A != B
```

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

```
C <- A != B
C
```

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

Como muestra el último ejemplo, el resultado de una operación lógica puede almacenarse como el valor de un objeto.

2.1.4.2 Operadores aritméticos:

```
#suma
A <- 5+6
A
```

```
## [1] 11
```

```
#Resta
B <- 6-8
B
```

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

```
#cociente
C <- 6/2.5
C
```

```
## [1] 2.4
```

```
#multiplicacion
D <- 6*2.5
D
```

```
## [1] 15
```

2.1.4.3 Funciones:

Las funciones son series de procedimientos estandarizados, que toman como imput determinados argumentos a fijar por el usuario, y devuelven un resultado acorde a la aplicación de dichos procedimientos. Su lógica de funcionamiento es:
`funcion(argumento1 = arg1, argumento2 = arg2)`

A lo largo del curso iremos viendo numerosas funciones, según lo requieran los distintos ejercicios. Sin embargo, veamos ahora algunos ejemplos para comprender su funcionamiento:

- paste() : concatena una serie de caracteres, indicando por última instancia como separar a cada uno de ellos

- paste0(): concatena una serie de caracteres sin separar
- sum(): suma de todos los elementos de un vector

- mean() promedio aritmético de todos los elementos de un vector

```
paste("Pega","estas",4,"palabras", sep = " ")
```

```
## [1] "Pega estas 4 palabras"
```

```
#Puedo concatenar caracteres almacenados en objetos
paste(A,B,C,sep = "**")
```

```
## [1] "11**-2**2.4"
```

```
# Paste0 pega los caracteres sin separador
paste0(A,B,C)
```

```
## [1] "11-22.4"
```

```
1:5
```

```
## [1] 1 2 3 4 5
```

```
sum(1:5)
```

```
## [1] 15
```

```
mean(1:5,na.rm = TRUE)
```

```
## [1] 3
```

2.1.4.4 Caracteres especiales

- R es sensible a mayúsculas y minúsculas, tanto para los nombres de las variables, como para las funciones y parámetros.
- Los **espacios en blanco** y los **carriage return** (*enter*) no son considerados por el lenguaje. Los podemos aprovechar para emprolijar el código y que la lectura sea más simple[2].
- El **numeral #** se utiliza para hacer comentarios. Todo lo que se escribe después del # no es interpretado por R. Se debe utilizar un # por cada línea de código que se desea anular
- Los **corchetes** `[]` se utilizan para acceder a un objeto:
  - en un vector[n° orden]
  - en una tabla[fila, columna]
  - en una lista[n° elemento]
- el signo **$** también es un método de acceso. Particularmente, en los dataframes, nos permitira acceder a una determinada columna de una tabla
- Los **paréntesis**`()` se utilizan en las funciones para definir los parámetros.
- Las **comas ,** se utilizan para separar los parametros al interior de una función.

2.1.5 Objetos:

Existen un gran cantidad de objetos distintos en R, en lo que resepcta al curso trabajaremos principalmente con 3 de ellos:

- Valores
- Vectores
- Data Frames
- Listas

2.1.5.1 Valores

Los valores y vectores pueden ser a su vez de distintas *clases*:

Numeric

```
A <-  1
class(A)
```

```
## [1] "numeric"
```

Character

[2] veremos que existen ciertas excepciones con algunos paquetes más adelante.

```
A <-  paste('Soy', 'una', 'concatenación', 'de', 'caracteres', sep = " ")
A
```

```
## [1] "Soy una concatenación de caracteres"
```

```
class(A)
```

```
## [1] "character"
```

Factor

```
A <- factor("Soy un factor, con niveles fijos")
class(A)
```

```
## [1] "factor"
```

La diferencia entre un *character* y un *factor* es que el último tiene solo algunos valores permitidos (levels), con un orden interno predefinido (el cual ,por ejemplo, se respetará a la hora de realizar un gráfico)

2.1.5.2 Vectores

Para crear un **vector** utilizamos el comando `c()`, de combinar.

```
C <- c(1, 3, 4)
C
```

```
## [1] 1 3 4
```

sumarle 2 a cada elemento del **vector** anterior

```
C <- C + 2
C
```

```
## [1] 3 5 6
```

sumarle 1 al primer elemento, 2 al segundo, y 3 al tercer elemento del **vector** anterior

```
D <- C + 1:3 #esto es equivalente a hacer 3+1, 5+2, 6+9
D
```

```
## [1] 4 7 9
```

`1:3` significa que queremos todos los números enteros desde 1 hasta 3.

crear un **vector** que contenga las palabras: “Carlos”,“Federico”,“Pedro”

```
E <- c("Carlos","Federico","Pedro")
E
```

```
## [1] "Carlos"   "Federico" "Pedro"
```

para acceder a algún elemento del vector, podemos buscarlo por su número de orden, entre `[ ]`

```
 E[2]
```

```
## [1] "Federico"
```

Si nos interesa almacenar dicho valor, al buscarlo lo asignamos a un nuevo objeto, dandole el nombre que deseemos

```
elemento2 <-  E[2]
```

```
elemento2
```

```
## [1] "Federico"
```

para **borrar** un objeto del ambiente de trabajo, utilizamos el comando *rm()*

```
rm(elemento2)
elemento2
```

```
## Error in eval(expr, envir, enclos): object 'elemento2' not found
```

También podemos cambiar el texto del segundo elemento de E, por el texto "Pablo"

```
E[2] <- "Pablo"
E
```

```
## [1] "Carlos" "Pablo"  "Pedro"
```

2.1.6 Data Frames

Un Data Frame es una tabla de datos, donde cada columna representa una variable, y cada fila una observación.

Este objeto suele ser central en el proceso de trabajo, y suele ser la forma en que se cargan datos externos para trabajar en el ambiente de R, y en que se exportan los resultados de nuestros trabajo.

También Se puede crear como la combinación de N vectores de igual tamaño. Por ejemplo, tomamos algunos valores del Indice de salarios

```
INDICE  <- c(100,   100,   100,
             101.8, 101.2, 100.73,
             102.9, 102.4, 103.2)

FECHA  <-  c("Oct-16", "Oct-16", "Oct-16",
             "Nov-16", "Nov-16", "Nov-16",
             "Dic-16", "Dic-16", "Dic-16")


GRUPO  <-  c("Privado_Registrado","Público","Privado_No_Registrado",
             "Privado_Registrado","Público","Privado_No_Registrado",
             "Privado_Registrado","Público","Privado_No_Registrado")
```

```
Datos <- data.frame(INDICE, FECHA, GRUPO)
Datos
```

```
##   INDICE  FECHA                 GRUPO
## 1 100.00 Oct-16    Privado_Registrado
## 2 100.00 Oct-16               Público
## 3 100.00 Oct-16 Privado_No_Registrado
## 4 101.80 Nov-16    Privado_Registrado
## 5 101.20 Nov-16               Público
## 6 100.73 Nov-16 Privado_No_Registrado
## 7 102.90 Dic-16    Privado_Registrado
## 8 102.40 Dic-16               Público
## 9 103.20 Dic-16 Privado_No_Registrado
```

Tal como en un **vector** se ubica a los elementos mediante `[ ]`, en un **dataframe** se obtienen sus elementos de la forma **`[fila, columna]`**.

Otra opción es especificar la columna, mediante el operador **`$`**, y luego seleccionar dentro de esa columna el registro deseado mediante el número de orden.

```
Datos$FECHA
```

```
## [1] Oct-16 Oct-16 Oct-16 Nov-16 Nov-16 Nov-16 Dic-16 Dic-16 Dic-16
## Levels: Dic-16 Nov-16 Oct-16
```

```
Datos[3,2]
```

```
## [1] Oct-16
## Levels: Dic-16 Nov-16 Oct-16
```

```
Datos$FECHA[3]
```

```
## [1] Oct-16
## Levels: Dic-16 Nov-16 Oct-16
```

¿que pasa si hacemos `Datos$FECHA[3,2]` ?

```
Datos$FECHA[3,2]
```

```
## Error in `[.default`(Datos$FECHA, 3, 2): incorrect number of dimensions
```

Nótese que el último comando tiene un número incorrecto de dimensiones, porque estamos refiriendonos 2 veces a la columna FECHA.

Acorde a lo visto anteriormente, el acceso a los **dataframes** mediante `[ ]`, puede utilizarse para realizar filtros sobre la base, especificando una condición para las filas. Por ejemplo, puedo utilizar los `[ ]` para conservar del **dataframe** `Datos` unicamente los registros con fecha de Diciembre 2016:

```
Datos[Datos$FECHA=="Dic-16",]
```

```
##   INDICE  FECHA                 GRUPO
## 7  102.9 Dic-16    Privado_Registrado
## 8  102.4 Dic-16               Público
## 9  103.2 Dic-16 Privado_No_Registrado
```

La lógica del paso anterior sería: Accedo al dataframe `Datos`, pidiendo únicamente conservar las filas (por eso la condición se ubica a la *izquierda* de la `,`) que cumplan el requisito de pertenecer a la categoría **"Dic-16"** de la variable **FECHA**.
Aún más, podría aplicar el filtro y al mismo tiempo identificar una variable de interés para luego realizar un cálculo sobre aquella. Por ejemplo, podría calcular la media de los indices en el mes de Diciembre.

```
###Por separado
Indices_Dic <- Datos$INDICE[Datos$FECHA=="Dic-16"]
Indices_Dic
```

```
## [1] 102.9 102.4 103.2
```

```
mean(Indices_Dic)
```

```
## [1] 102.8333
```

```
### Todo junto
mean(Datos$INDICE[Datos$FECHA=="Dic-16"])
```

```
## [1] 102.8333
```

La lógica de esta sintaxis sería: "Me quedó con la variable **INDICE**, cuando la variable FECHA sea igual a **"Dic-16"**, luego calculo la media de dichos valores"

2.1.7 Listas

Contienen una concatenación de objetos de cualquier tipo. Así como un vector contiene valores, un dataframe contiene vectores, una lista puede contener dataframes, pero también vectores, o valores, y *todo ello a la vez*

```
superlista <- list(A,B,C,D,E,FECHA, DF = Datos, INDICE, GRUPO)
superlista
```

```
## [[1]]
## [1] Soy un factor, con niveles fijos
## Levels: Soy un factor, con niveles fijos
##
## [[2]]
## [1] -2
##
## [[3]]
```

```
## [1] 3 5 6
##
## [[4]]
## [1] 4 7 9
##
## [[5]]
## [1] "Carlos" "Pablo"  "Pedro"
##
## [[6]]
## [1] "Oct-16" "Oct-16" "Oct-16" "Nov-16" "Nov-16" "Nov-16" "Dic-16" "Dic-16"
## [9] "Dic-16"
##
## $DF
##   INDICE  FECHA                 GRUPO
## 1 100.00 Oct-16    Privado_Registrado
## 2 100.00 Oct-16               Público
## 3 100.00 Oct-16 Privado_No_Registrado
## 4 101.80 Nov-16    Privado_Registrado
## 5 101.20 Nov-16               Público
## 6 100.73 Nov-16 Privado_No_Registrado
## 7 102.90 Dic-16    Privado_Registrado
## 8 102.40 Dic-16               Público
## 9 103.20 Dic-16 Privado_No_Registrado
##
## [[8]]
## [1] 100.00 100.00 100.00 101.80 101.20 100.73 102.90 102.40 103.20
##
## [[9]]
## [1] "Privado_Registrado"    "Público"               "Privado_No_Registrado"
## [4] "Privado_Registrado"    "Público"               "Privado_No_Registrado"
## [7] "Privado_Registrado"    "Público"               "Privado_No_Registrado"
```

Para acceder un elemento de una lista, podemos utilizar el operador **$**, que se puede usar a su vez de forma iterativa

```
superlista$DF$FECHA[2]
```

```
## [1] Oct-16
## Levels: Dic-16 Nov-16 Oct-16
```

2.1.8 Ambientes de trabajo

Hay algunas cosas que tenemos que tener en cuenta respecto del orden del ambiente en el que trabajamos:

- Working Directory: El directorio de trabajo, pueden ver el suyo con `getwd()`, es *hacia donde apunta el código*, por ejemplo, si quieren leer

Figure 2.6: logo Rpoject

un archivo, la ruta del archivo tiene que estar explicitada como el recorrido desde el Working Directory.
- Environment: Esto engloba tanto la información que tenemos cargada en *Data* y *Values*, como las librerías que tenemos cargadas mientras trabajamos.

Es importante que mantengamos bien delimitadas estas cosas entre diferentes trabajos, sino:

1. El directorio queda referido a un lugar específico en nuestra computadora.

- Si se lo compartimos a otro **se rompe**
- Si cambiamos de computadora **se rompe**
- Si lo cambiamos de lugar **se rompe**
- Si primero abrimos otro script **se rompe**

2. Tenemos mezclados resultados de diferentes trabajos:

- Nunca sabemos si esa variable/tabla/lista se creo en ese script y no otro
- Perdemos espacio de la memoria
- No estamos seguros de que el script cargue todas las librerías que necesita

Rstudio tiene una herramienta muy útil de trabajo que son los **proyectos**. Estos permiten mantener un ambiente de trabajo delimitado por cada uno de nuestros trabajos. Es decir:

- El directorio de trabajo se refiere a donde esta ubicado el archivo .Rproj
- El Environment es específico de nuestro proyecto.

Un proyecto no es un sólo script, sino toda una carpeta de trabajo.

Para crearlo, vamos al logo de nuevo projecto (Arriba a la izquierda de la panatalla), y elegimos la carpeta de trabajo.

2.1.9 Tipos de archivos de R

- **Script**: Es un archivo de texto plano, donde podemos poner el código que utilizamos para preservarlo
- **Rnotebook**: También sirve para guardar el código, pero a diferencia de los scripts, se puede compilar, e intercalar código con resultados (este archivo es un rnotebook)
- **Rproject**: Es un archivo que define la metadata del proyecto
- **RDS y Rdata**: Dos formatos de archivos propios de R para guardar datos.

2.2 Práctica Guiada

2.2.1 Instalación de paquetes complementarios al R Base

Hasta aquí hemos visto múltiples funciones que están contenidas dentro del lenguaje básico de R. Ahora bien, al tratarse de un software libre, los usuarios de R con más experiencia contribuyen sistemáticamente a expandir este lenguaje mediante la creación y actualización de **paquetes** complementarios. Lógicamente, los mismos no están incluidos en la instalación inicial del programa, pero podemos descargarlos e instalarlos al mismo tiempo con el siguiente comando:

```
install.packages("nombre_del_paquete")
```

Resulta recomendable **ejecutar este comando desde la consola** ya que solo necesitaremos correrlo una vez en nuestra computadora. Al ejecutar el mismo, se descargarán de la pagina de CRAN los archivos correspondientes al paquete hacia el directorio en donde hayamos instalado el programa. Típicamente los archivos se encontrarán en `C:\Program Files\R\R-3.5.0\library\`, siempre con la versión del programa correspondiente.
Una vez instalado el paquete, cada vez que abramos una nueva sesión de R y querramos utilizar el mismo debemos **cargarlo al ambiente de trabajo** mediante la siguiente función:

```
library(nombre_del_paquete)
```

Nótese que al cargar/activar el paquete no son necesarias las comillas.

2.2.2 Lectura y escritura de archivos

2.2.2.1 .csv y .txt

Hay **muchas** funciones para leer archivos de tipo *.txt* y *.csv*. La mayoría sólo cambia los parámetros que vienen por default.

Es importante tener en cuenta que una base de datos que proviene de archivos *.txt*, o *.csv* puede presentar diferencias en cuanto a los siguientes parametros:

- encabezado

- delimitador (,, tab, ;)
- separador decimal

```
dataframe <- read.delim(file, header = TRUE, sep = "\t", quote = "\"", dec = ".", fill
```

Ejemplo. Levantar la base de sueldos de funcionarios

En el parametro `file` tengo que especificar el nombre completo del archivo, incluyendo el directorio donde se encuentra. Lo más sencillo es abrir comillas, apretar `Tab` y se despliega el menú de las cosas que tenemos en el directorio de trabajo. Si queremos movernos hacia arriba, agregamos `../`

```
sueldos_funcionarios <- read.table(file = '../fuentes/sueldo_funcionarios_2019.csv',se
sueldos_funcionarios[1:10,]
```

```
##             cuil anio mes funcionario_apellido funcionario_nombre
## 1  20-17692128-6 2019   1    RODRIGUEZ LARRETA    HORACIO ANTONIO
## 2  20-17735449-0 2019   1             SANTILLI        DIEGO CESAR
## 3  27-24483014-0 2019   1                ACUÑA      MARIA SOLEDAD
## 4  20-13872301-2 2019   1             ASTARLOA      GABRIEL MARIA
## 5  20-25641207-2 2019   1             AVOGADRO       ENRIQUE LUIS
## 6  27-13221055-7 2019   1            BOU PEREZ          ANA MARIA
## 7  27-13092400-5 2019   1                FREDA     MONICA BEATRIZ
## 8  20-17110752-1 2019   1         MACCHIAVELLI    EDUARDO ALBERTO
## 9  20-22293873-3 2019   1               MIGUEL       FELIPE OSCAR
## 10 20-14699669-9 2019   1               MOCCIA             FRANCO
##                                      repartición asignacion_por_cargo_i
## 1                              Jefe de Gobierno               197745.8
## 2                         Vicejefatura de Gobierno               197745.8
## 3             Ministerio de Educación e Innovación               224516.6
## 4  Procuración General de la Ciudad de Buenos Aires               224516.6
## 5                           Ministerio de Cultura               224516.6
## 6                             Ministerio de Salud               224516.6
## 7  Sindicatura General de la Ciudad de Buenos Aires               224516.6
## 8         Ministerio de Ambiente y Espacio Público               224516.6
## 9               Jefatura de Gabinete de Ministros               224516.6
## 10    Ministerio de Desarrollo Urbano y Transporte               224516.6
##    aguinaldo_ii total_salario_bruto_i_._ii observaciones
## 1             0                   197745.8
## 2             0                   197745.8
## 3             0                   224516.6
## 4             0                   224516.6
## 5             0                   224516.6
## 6             0                   224516.6
## 7             0                   224516.6
## 8             0                   224516.6
## 9             0                   224516.6
## 10            0                   224516.6
```

Como puede observarse aquí, las bases individuales de la EPH cuentan con más de 58.000 registros y 177 variables. Al trabajar con bases de microdatos, resulta conveniente contar con algunos comandos para tener una mirada rápida de la base, antes de comenzar a realizar los procesamientos que deseemos.

Veamos algunos de ellos:

```
#View(individual_t117)
names(sueldos_funcionarios)
```

```
##  [1] "cuil"                       "anio"
##  [3] "mes"                        "funcionario_apellido"
##  [5] "funcionario_nombre"         "repartición"
##  [7] "asignacion_por_cargo_i"     "aguinaldo_ii"
##  [9] "total_salario_bruto_i_._ii" "observaciones"
```

```
summary(sueldos_funcionarios)
```

```
##             cuil         anio           mes       funcionario_apellido
##  20-13872301-2: 3   Min.   :2019   Min.   :1.00   ACUÑA     : 3
##  20-14699669-9: 3   1st Qu.:2019   1st Qu.:2.00   ASTARLOA  : 3
##  20-16891528-5: 3   Median :2019   Median :3.00   AVELLANEDA: 3
##  20-16891539-0: 3   Mean   :2019   Mean   :3.34   AVOGADRO  : 3
##  20-17110752-1: 3   3rd Qu.:2019   3rd Qu.:5.00   BENEGAS   : 3
##  20-17692128-6: 3   Max.   :2019   Max.   :6.00   BOU PEREZ : 3
##  (Other)      :76                                 (Other)   :76
##        funcionario_nombre
##   ANA MARIA      : 3
##   BRUNO GUIDO    : 3
##   CHRISTIAN      : 3
##   DIEGO CESAR    : 3
##   DIEGO HERNAN   : 3
##   EDUARDO ALBERTO: 3
##  (Other)         :76
##                                                          repartición
##  Consejo de los Derechos de Niñas, Niños y Adoles - Presidencia: 3
##  Ente de Turismo Ley Nº 2627                                   : 3
##  Jefatura de Gabinete de Ministros                             : 3
##  Jefe de Gobierno                                              : 3
##  Ministerio de Ambiente y Espacio Público                      : 3
##  Ministerio de Cultura                                         : 3
##  (Other)                                                       :76
##  asignacion_por_cargo_i  aguinaldo_ii    total_salario_bruto_i_._ii
##  Min.   :197746         Min.   :     0   Min.   :197746
##  1st Qu.:217520         1st Qu.:     0   1st Qu.:217805
##  Median :226866         Median :     0   Median :226866
##  Mean   :224718         Mean   : 14843   Mean   :239560
##  3rd Qu.:231168         3rd Qu.:     0   3rd Qu.:248033
```

```
##  Max.   :249662         Max.   :113433   Max.   :340300
##
##         observaciones
##                :93
##  baja 28/2/2019: 1
##
##
##
##
##
```

```
head(sueldos_funcionarios)[,1:5]
```

```
##            cuil anio mes funcionario_apellido funcionario_nombre
## 1 20-17692128-6 2019   1    RODRIGUEZ LARRETA    HORACIO ANTONIO
## 2 20-17735449-0 2019   1             SANTILLI        DIEGO CESAR
## 3 27-24483014-0 2019   1                ACUÑA      MARIA SOLEDAD
## 4 20-13872301-2 2019   1             ASTARLOA      GABRIEL MARIA
## 5 20-25641207-2 2019   1             AVOGADRO       ENRIQUE LUIS
## 6 27-13221055-7 2019   1            BOU PEREZ           ANA MARIA
```

2.2.2.2 Excel

Para leer y escribir archivos excel debemos utilizar los comandos que vienen con la librería openxlsx

```
# install.packages("openxlsx") # por única vez
library(openxlsx) #activamos la librería

#creamos una tabla cualquiera de prueba
x <- 1:10
y <- 11:20
tabla_de_R <- data.frame(x,y)

# escribimos el archivo
write.xlsx( x = tabla_de_R, file = "../resultados/archivo.xlsx",row.names = FALSE)
#Donde lo guardó? Hay un directorio por default en caso de que no hayamos definido alg

#getwd()

#Si queremos exportar multiples dataframes a un Excel, debemos armar previamente una l
Lista_a_exportar <- list("sueldos funcionarios" = sueldos_funcionarios,
                         "Tabla Numeros" = tabla_de_R)

write.xlsx( x = Lista_a_exportar, file = "../resultados/archivo_2_hojas.xlsx",row.names

#leemos el archivo especificando la ruta (o el directorio por default) y el nombre de
```

```
Indices_Salario <- read.xlsx(xlsxFile = "../resultados/archivo_2_hojas.xlsx",sheet = "sueldos fun
#alternativamente podemos especificar el número de orden de la hoja que deseamos levantar
Indices_Salario <- read.xlsx(xlsxFile = "../resultados/archivo_2_hojas.xlsx",sheet = 1)
Indices_Salario[1:10,]
```

```
##             cuil anio mes funcionario_apellido funcionario_nombre
## 1  20-17692128-6 2019   1    RODRIGUEZ LARRETA    HORACIO ANTONIO
## 2  20-17735449-0 2019   1             SANTILLI        DIEGO CESAR
## 3  27-24483014-0 2019   1                ACUÑA      MARIA SOLEDAD
## 4  20-13872301-2 2019   1             ASTARLOA      GABRIEL MARIA
## 5  20-25641207-2 2019   1             AVOGADRO       ENRIQUE LUIS
## 6  27-13221055-7 2019   1            BOU PEREZ          ANA MARIA
## 7  27-13092400-5 2019   1                FREDA     MONICA BEATRIZ
## 8  20-17110752-1 2019   1         MACCHIAVELLI    EDUARDO ALBERTO
## 9  20-22293873-3 2019   1               MIGUEL       FELIPE OSCAR
## 10 20-14699669-9 2019   1               MOCCIA             FRANCO
##                                      repartición asignacion_por_cargo_i
## 1                                Jefe de Gobierno               197745.8
## 2                           Vicejefatura de Gobierno               197745.8
## 3               Ministerio de Educación e Innovación               224516.6
## 4  Procuración General de la Ciudad de Buenos Aires               224516.6
## 5                             Ministerio de Cultura               224516.6
## 6                               Ministerio de Salud               224516.6
## 7  Sindicatura General de la Ciudad de Buenos Aires               224516.6
## 8          Ministerio de Ambiente y Espacio Público               224516.6
## 9                 Jefatura de Gabinete de Ministros               224516.6
## 10     Ministerio de Desarrollo Urbano y Transporte               224516.6
##    aguinaldo_ii total_salario_bruto_i_._ii observaciones
## 1             0                  197745.8
## 2             0                  197745.8
## 3             0                  224516.6
## 4             0                  224516.6
## 5             0                  224516.6
## 6             0                  224516.6
## 7             0                  224516.6
## 8             0                  224516.6
## 9             0                  224516.6
## 10            0                  224516.6
```