Clase 1

Secretaría de Investigaciones Derecho - UBA

Demian Zayat

10/15/2019

Presentacion

Las diapositivas están en http://www.presentacionsi.netlify.com.

Primeros pasos

Para abrir R en RStudio debemos abrir el programa RStudio directamente.

Si vamos a trabajar en el R cloud, debemos ingresar en rstudio.cloud, y allí nos logueamos (o creamos una cuenta si es la primera vez).

Lo primero que vamos a hacer es crear un nuevo proyecto. Ahí podemos elegir en qué carpeta crearlo. Luego, iniciamos un `script` nuevo, que es donde vamos a ir escribiendo nuestro código.

El RStudio tiene 4 areas de trabajo. En el panel principal, arriba, va el `script`, donde escribrimos el código (Panel de edición). Esto es lo que se guarda en un archivo .R y es lo que se ejecuta una y otra vez. El panel principal, abajo, tiene la `consola`, que es donde se producen los cómputos. Ahí es donde van las intrucciones, y la computadora las procesa. Luego, en el panel de la derecha, arriba, vamos a encontrar los objetos que vayamos guardando en la memoria, y en el panel de abajo, está la ruta de los archivos, visualizaciones, paquetes. etc....

Entonces, en general, escribimos en el `script` y con `Ctrl+Enter` lo ejecuta en la consola. Si apretamos `Ctrl + Enter` en cualquier lugar de la fila, ejecuta esa fila.

Los espacios no cuentan, uno puede dejar mucho espacio y funciona lo mismo. Si comenzamos con `#` esa fila queda comentada, y no se se ejecuta.

R es una gran calculadora. Prueben en su script

```
4 + 4
```

```
## [1] 8
```

```
sqrt(25)
```

```
## [1] 5
```

```
8/6 + 23/116
```

```
## [1] 1.531609
```

Y van surgiendo los resultados. Pero como R es un lenguaje *orientado a objetos* podemos ir guardando esos resultados o valores en objetos. Para ello, le ponemos un nombre y se lo asignamos. Para asignar un nombre usamos `Alt + guion`. En general se pone `nombre <- valor`:

```
# Calculamos la dimensión de una habitación
ancho <- 4
largo <- 22
superficie <- ancho * largo
superficie
```

```
## [1] 88
```

Cada vez que llame al objeto `superficie`, va a tener el valor guardado (`ancho * largo`).

Nuestro primer proyecto

Vamos a comenzar con el capítulo 2 del libro de Antonio Vázquez Brust, disponible en https://bitsandbricks.github.io/ciencia_de_datos_gente_sociable/una-presentacion-a-toda-marcha-de-r.html

Para ello, debemos cargar el paquete tidyverse. La primera vez que se usa un paquete, hay que instalarlo. Para ello, usamos el comando `install.packages("tidyverse")`. Fíjense que el nombre del paquete va entre comillas. Una vez que está descargado e instalado en la computadora, lo debemos cargar en la memoria para usarlo. Eso lo hacemos con el comando `library(tidyverse)`. Fíjense que el nombre del paquete **no** va entre comillas.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.2.0     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.1
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

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

Si aparece todo eso está ok. Si dice "Error" hubo un error, y habría que ver qué paso. Si dice "Warning" es que se cargó, pero que hay algunas cosas que deberíamos estar al tanto... Pero, *Warning no es error*.

Con esto cargado, vamos a bajar una base de datos. Vamos a utilizar la base de datos de Mortalidad infantil de la CABA. Podríamos bajarla del sitio oficial, pero para este primer ejercicio mejor hacerlo de este repositorio, y se lo asignamos al objeto que llamaremos `mortalidad`. Luego podemos explorar un poco la base.

```
mortalidad <- read_csv2("https://raw.githubusercontent.com/Demzayat/derecho_y_datos/master/Clase1/mortal
```

```
## Using ',' as decimal and '.' as grouping mark. Use read_delim() for more control.
```

```
## Parsed with column specification:
## cols(
##   Comuna = col_double(),
##   `2010` = col_double(),
##   `2011` = col_double(),
##   `2012` = col_double(),
##   `2013` = col_double(),
##   `2014` = col_double(),
##   `2015` = col_double(),
##   `2016` = col_double(),
##   `2017` = col_double(),
##   `2018` = col_double()
## )
```

```
str(mortalidad)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 15 obs. of  10 variables:
##  $ Comuna: num  1 2 3 4 5 6 7 8 9 10 ...
##  $ 2010  : num  9.9 6.7 4.9 9.9 3.6 3.2 8.1 7.9 8.1 7.5 ...
##  $ 2011  : num  9.1 6.6 10.8 11.1 7.2 5.2 11 9.7 10.4 7.1 ...
```

```
##  $ 2012  : num  10.7 5.1 7.2 11.8 5.7 9 8.9 10.8 8.8 7.3 ...
##  $ 2013  : num  8.9 7.5 11.9 11.8 8.7 6.8 11.9 10.9 9.3 12.2 ...
##  $ 2014  : num  4.9 9.2 9.4 12.3 6.7 7.5 8.5 12.7 6.5 5.3 ...
##  $ 2015  : num  7.6 4.5 5.4 7.2 6.7 5.7 6.4 8.2 6.5 5.8 ...
##  $ 2016  : num  9.5 3.6 8 11.9 8.5 2.4 8.5 9.7 10.1 3.6 ...
##  $ 2017  : num  5.3 3.1 7.5 7.6 7.8 5.8 7.8 8.9 5.2 6.3 ...
##  $ 2018  : num  6.3 5 4.5 6.9 3.4 6.8 4 7.6 9.9 4.9 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Comuna = col_double(),
##   ..   `2010` = col_double(),
##   ..   `2011` = col_double(),
##   ..   `2012` = col_double(),
##   ..   `2013` = col_double(),
##   ..   `2014` = col_double(),
##   ..   `2015` = col_double(),
##   ..   `2016` = col_double(),
##   ..   `2017` = col_double(),
##   ..   `2018` = col_double()
##   .. )
```

```
names(mortalidad)
```

```
##  [1] "Comuna" "2010"   "2011"   "2012"   "2013"   "2014"   "2015"
##  [8] "2016"   "2017"   "2018"
```

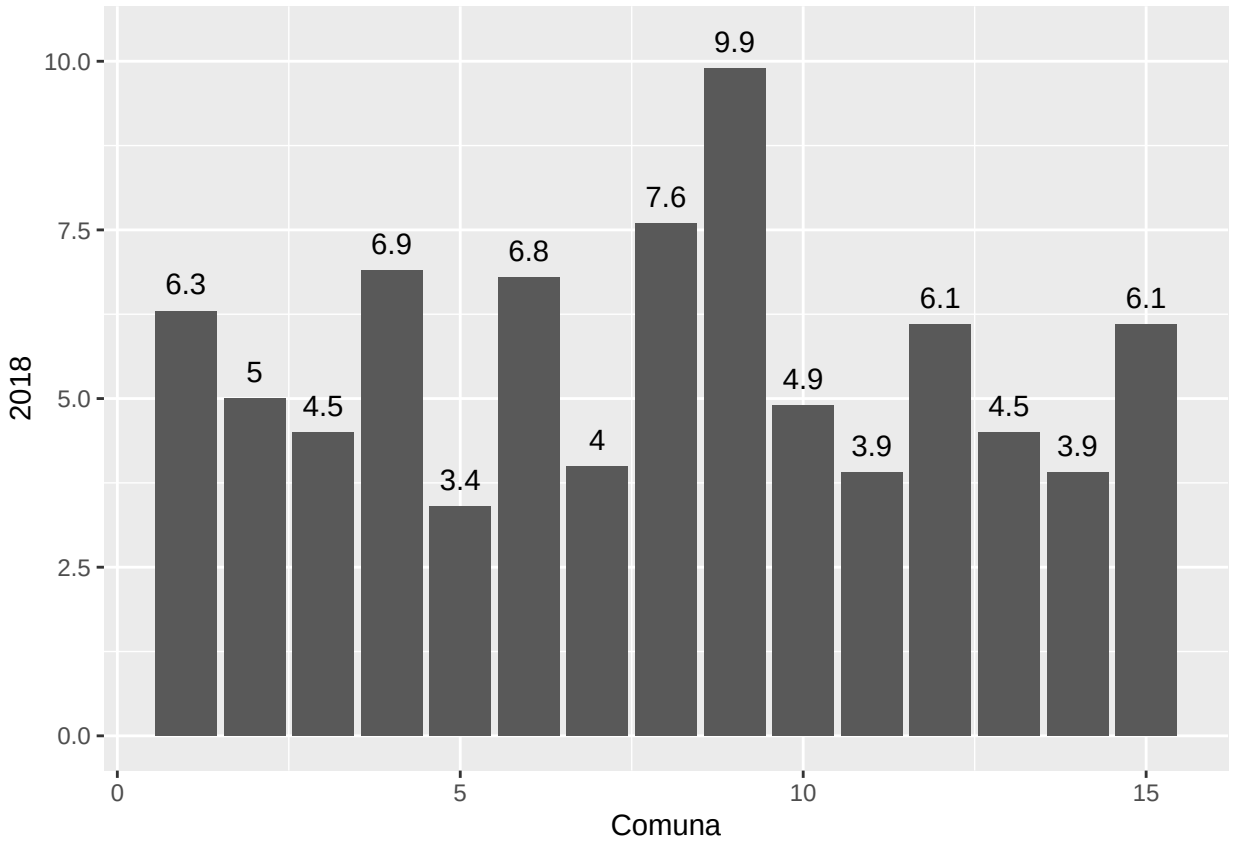
```
dim(mortalidad)
```

```
## [1] 15 10
```

Si hacemos click sobre el objeto (en el panel de Entorno) se abre tabla para que la podamos ver.

Podemos hacer algún gráfico, y comparar la mortalidad del 2018, por comuna Para ello, usaremos el paquete `ggplot` que ya tenemos cargado en la constelación `tidyverse`

```
ggplot(data = mortalidad, aes(x = Comuna, y = `2018`))+
  geom_col()+
  geom_text(aes(label = `2018`), nudge_y = 0.4)
```

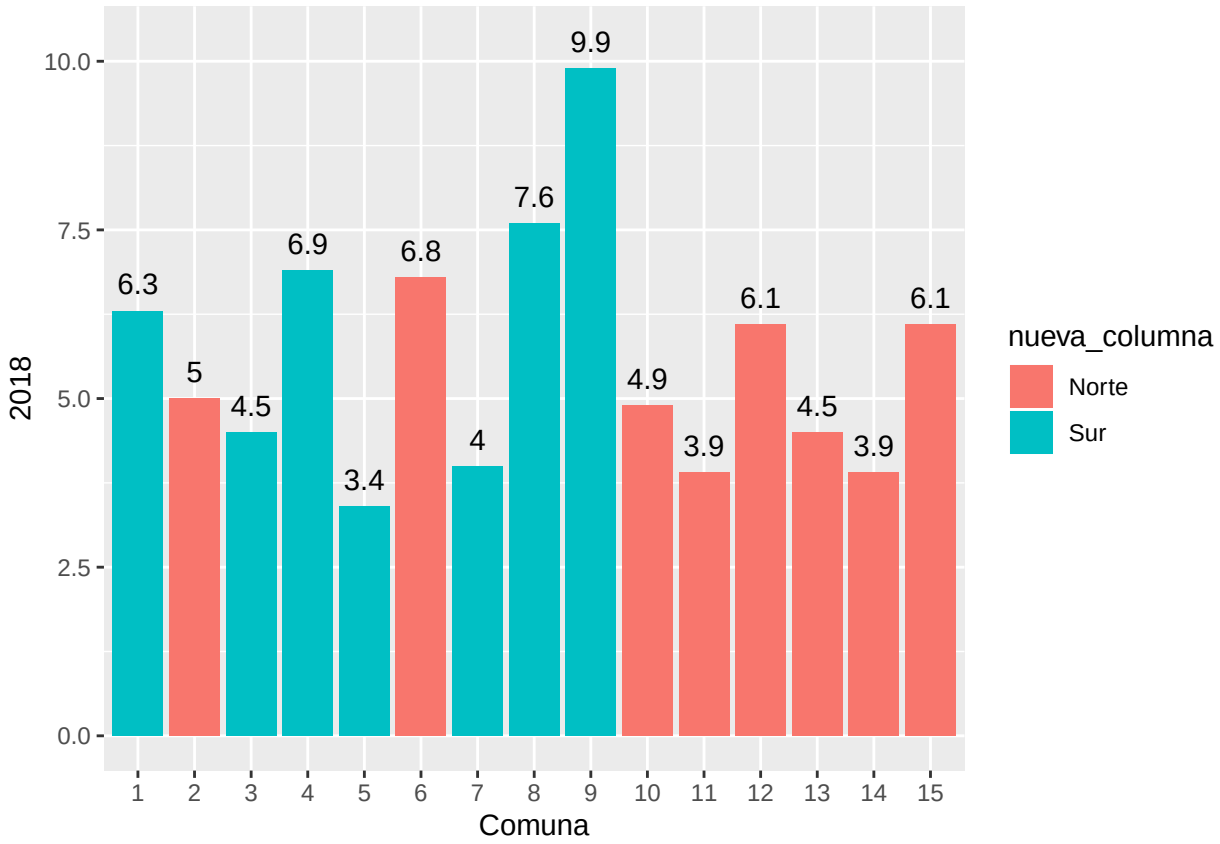


Ahora podemos ver qué comunas son del sur, y cuáles del norte, y lo graficamos.

```
nueva_columna <- c("Sur", "Norte", "Sur", "Sur", "Sur", "Norte", "Sur", "Sur", "Sur", "Norte", "Norte",

mortalidad <- mortalidad %>%
  mutate(Comuna = as.factor(Comuna),
         ubicacion = nueva_columna)

ggplot(data = mortalidad, aes(x = Comuna, y = `2018`))+
  geom_col(aes(fill = nueva_columna))+
  geom_text(aes(label = `2018`), nudge_y = 0.4)
```



Y podemos también hacer un mapa!! Para ello, debemos instalar el paquete `sf`

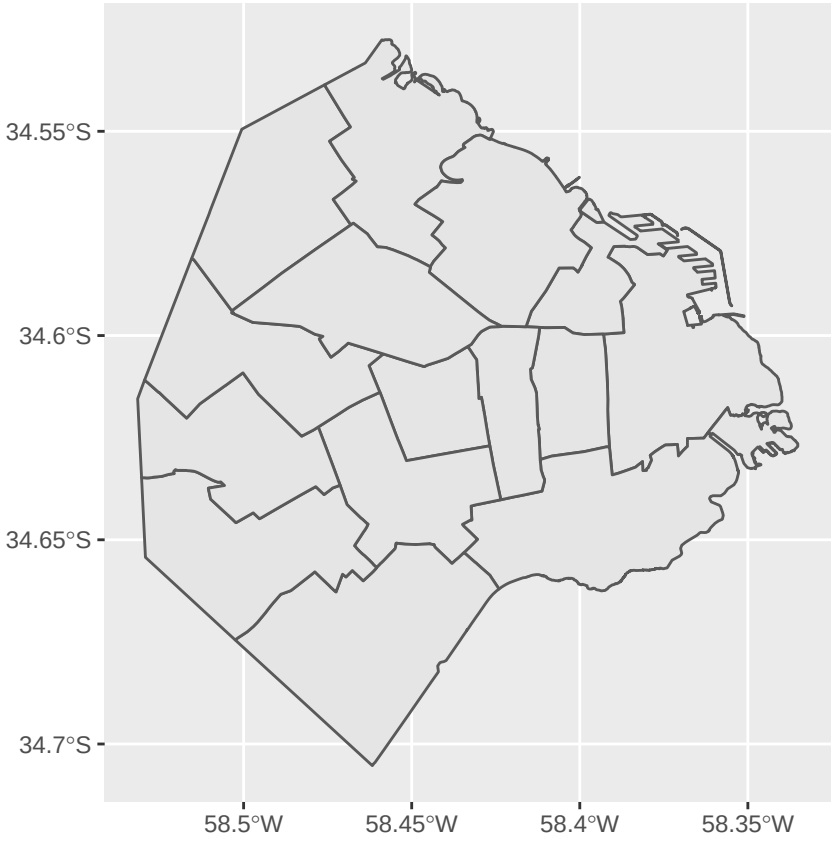
```
# install.package("sf")
library(sf)
```

```
## Linking to GEOS 3.6.1, GDAL 2.1.3, PROJ 4.9.3
```

```
comunas <- st_read('https://bitsandbricks.github.io/data/CABA_comunas.geojson')
```

```
## Reading layer `OGRGeoJSON' from data source `https://bitsandbricks.github.io/data/CABA_comunas.geojs
## Simple feature collection with 15 features and 4 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -58.53152 ymin: -34.70529 xmax: -58.33514 ymax: -34.52754
## epsg (SRID):    4326
## proj4string:    +proj=longlat +datum=WGS84 +no_defs
```

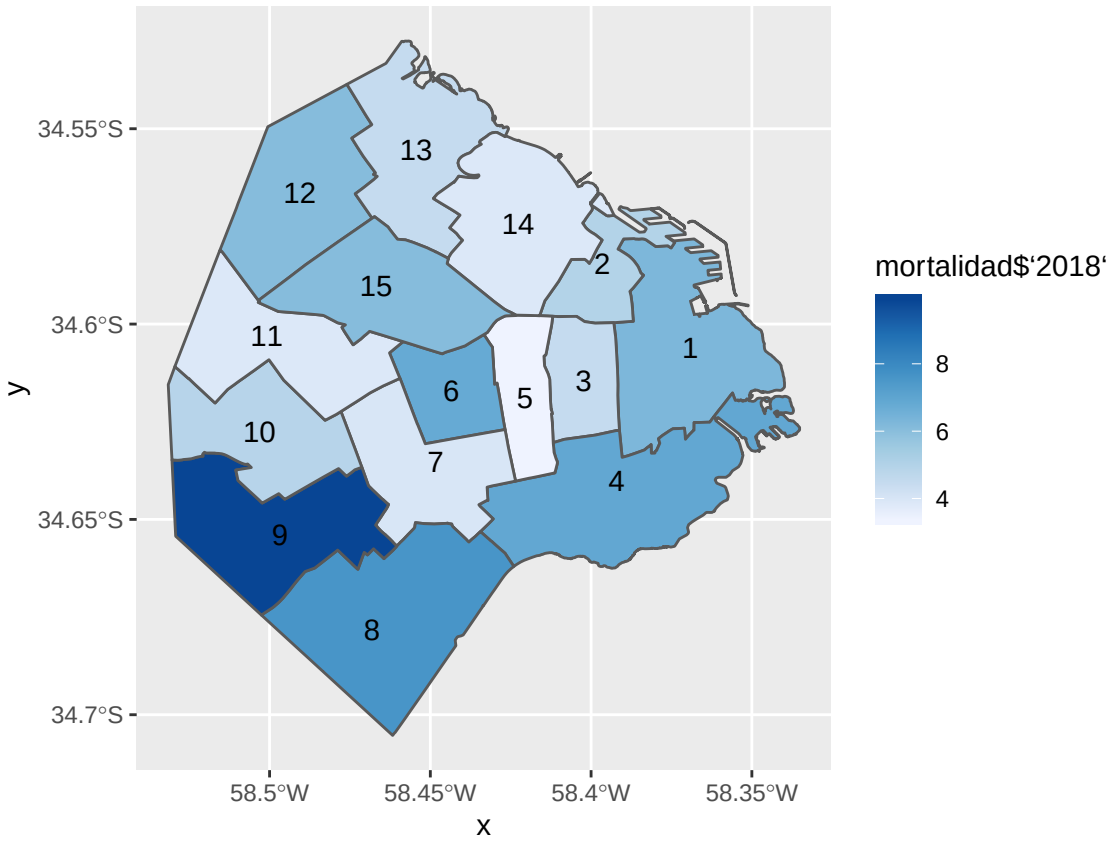
```
ggplot()+
  geom_sf(data = comunas)
```



y pintamos de acuerdo a la mortalidad del 2018

```
ggplot()+
  geom_sf(data= comunas, aes(fill = mortalidad$`2018`))+
  geom_sf_text(aes(label = comunas), data = comunas)+
  scale_fill_distiller(direction = 1)
```

```
## Warning in st_point_on_surface.sfc(sf::st_zm(x)): st_point_on_surface may
## not give correct results for longitude/latitude data
```


34.55°S
34.6°S
34.65°S
34.7°S
y
58.5°W
58.45°W
58.4°W
58.35°W
x
12
13
14
2
15
11
1
6
5
3
10
7
4
9
8
mortalidad$'2018'
8
6
4