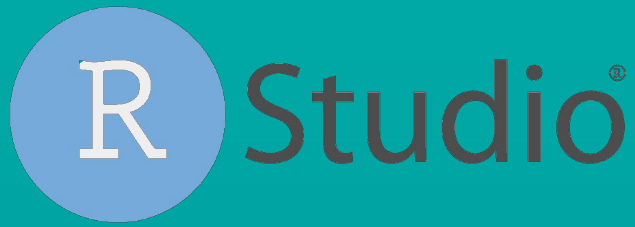
XIV Jornadas de **Informática en Salud**

HOSPITAL ITALIANO de Buenos Aires

Departamento de Informática en Salud



Workshop

Fernando Binder

XIV Jornadas de Informática en Salud

HOSPITAL ITALIANO de Buenos Aires
Departamento de Informática en Salud

R y Análisis de datos

- Ciencia de Datos y sus tareas
- R y R Studio: consola y paneles principales
- Objetos y funciones:
    - Variables, vectores, tablas (data.frames y tibbles)
- Carga de datos
- Ejercicios:
    - Exploración de una base de datos
    - Cálculos de estadísticas descriptivas / de resumen
    - Visualización de datos (paquete ggplot2)
- Recursos para aprendizaje y uso de R

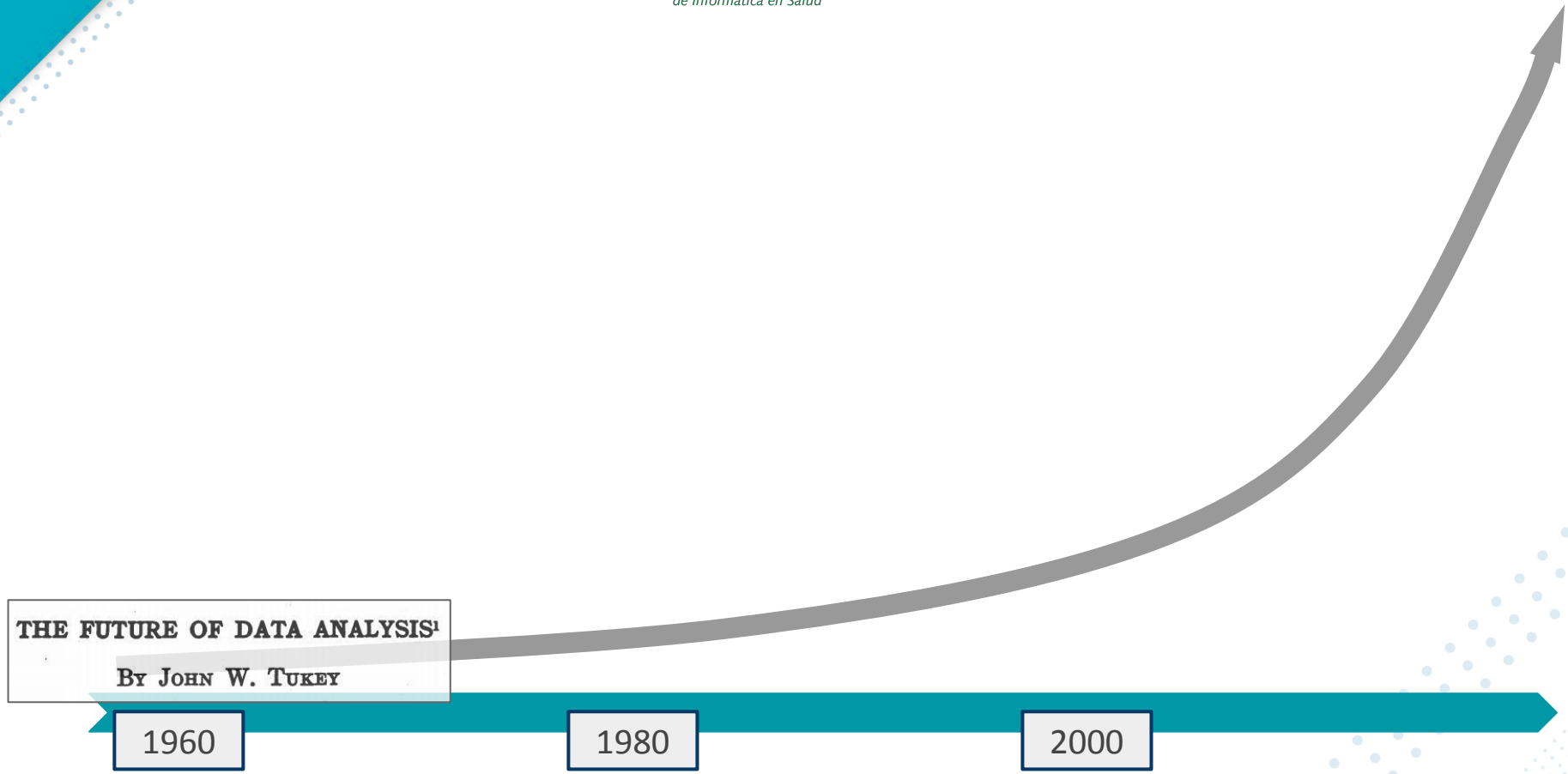
XIV Jornadas de **Informática en Salud**

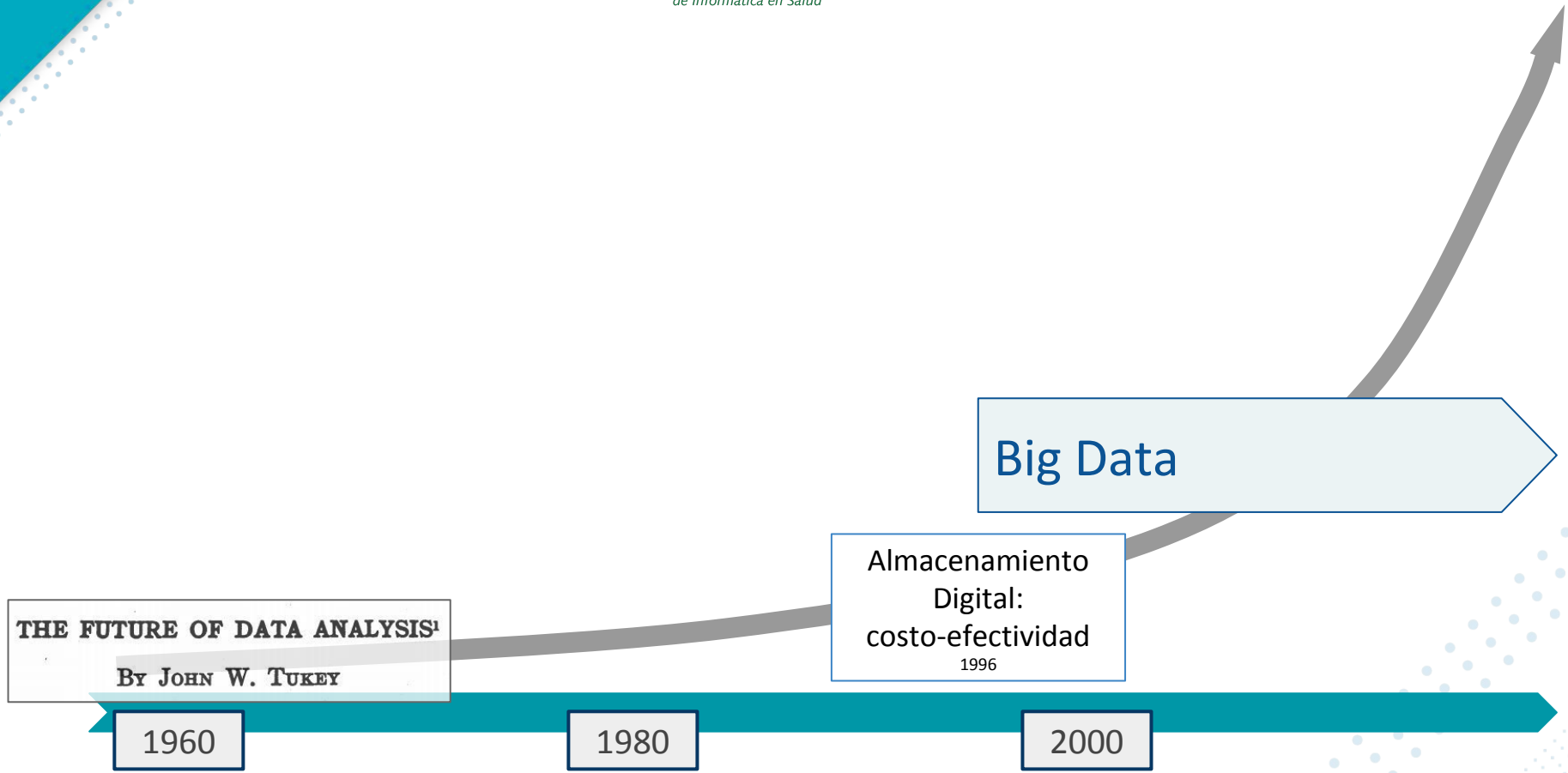
HOSPITAL ITALIANO de Buenos Aires

Departamento de Informática en Salud

Material

https://github.com/f-binder/r_jis_2019


THE FUTURE OF DATA ANALYSIS[1]
By John W. Tukey
1960
1980
2000


THE FUTURE OF DATA ANALYSIS[1]
By John W. Tukey
Almacenamiento Digital: costo-efectividad
1996
Big Data
1960
1980
2000

Workflows
gamechangers

Big Data

Almacenamiento Digital: costo-efectividad
1996

THE FUTURE OF DATA ANALYSIS[1]
By John W. Tukey

1960

1980

2000

Ciencia de Datos como disciplina

- Adquisición y almacenamiento de datos
- Gestión y limpieza de datos (wrangling)
- Análisis de datos
- Reporte de resultados, visualización

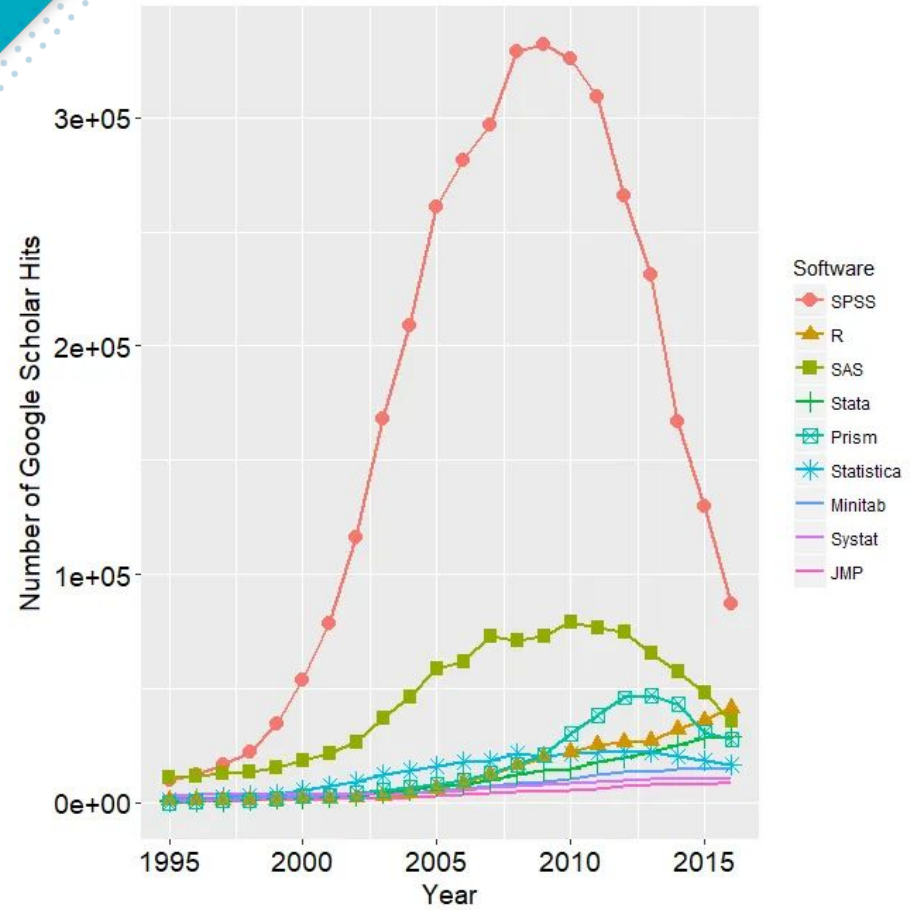
Además:

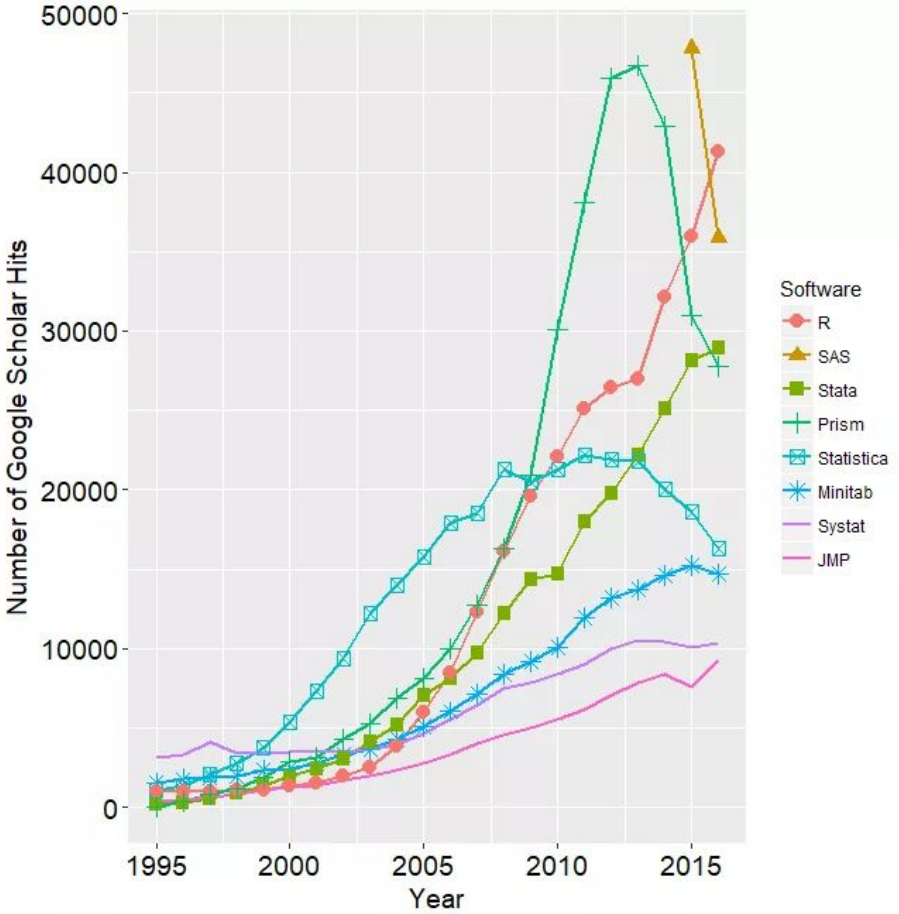
- Estrategias para manejo y computación de grandes datos (bases de datos, herramientas para cloud y cluster computing, etc.)

R para Ciencia de Datos

- Lenguaje de programación y entorno de trabajo para análisis estadístico y visualización de gráficos
- Código abierto
- Flexible y extensible
- Comunidad activa

Principal entorno de programación cuantitativa en **estadística académica** y creciente popularidad en **análisis de datos en salud**.


Number of Google Scholar Hits
3e+05
2e+05
1e+05
0e+00
1995
2000
2005
2010
2015
Year
Software
SPSS
R
SAS
Stata
Prism
Statistica
Minitab
Systat
JMP

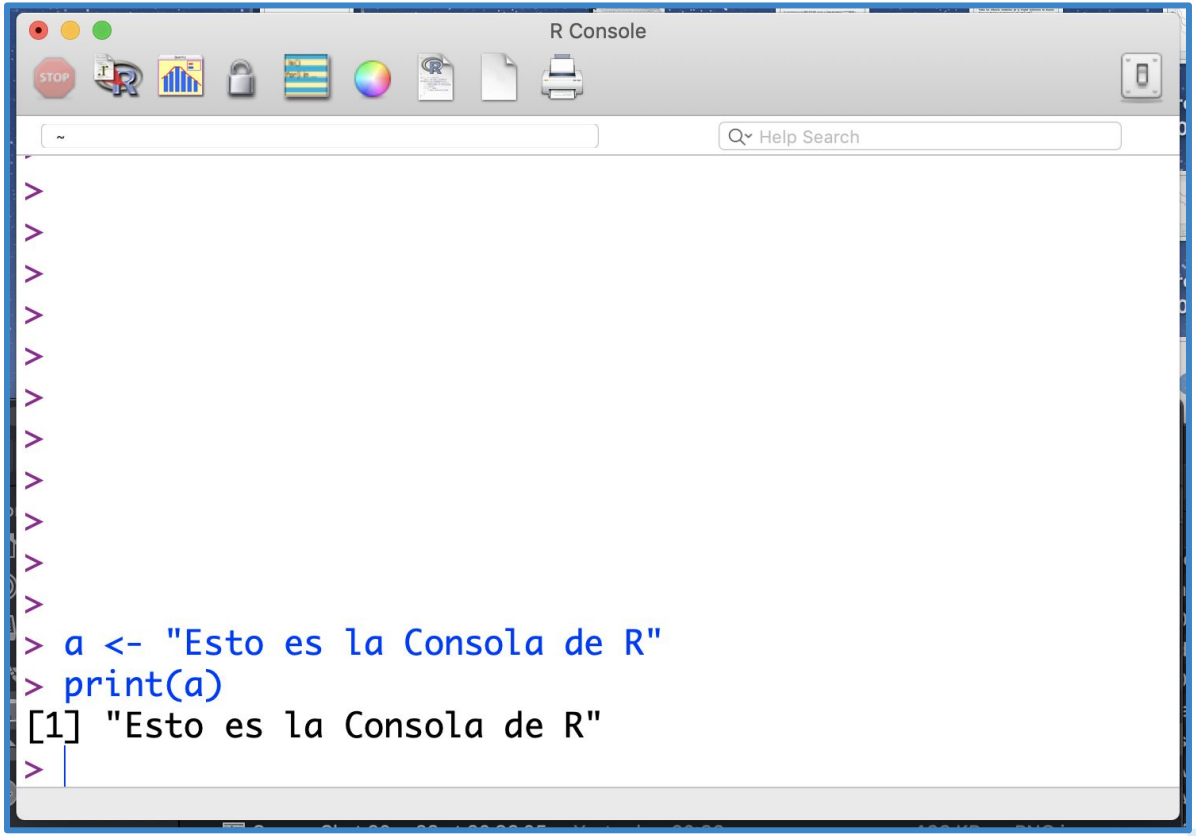

Number of Google Scholar Hits
50000
40000
30000
20000
10000
0
1995
2000
2005
2010
2015
Year
Software
R
SAS
Stata
Prism
Statistica
Minitab
Systat
JMP

Ciencia de Datos como disciplina

- ☐ Adquisición y almacenamiento de datos
- ☑ Gestión y limpieza de datos (wrangling)
- ☑ Análisis de datos
- ☑ Reporte de resultados, visualización
- ☐ Otras tareas

R y R Studio

- Consola R
- Entorno de desarrollo integrado (IDE): R Studio


R Console
~
Help Search
> a <- "Esto es la Consola de R"
> print(a)
[1] "Esto es la Consola de R"
>

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Untitled1*

```
1
2  x <- data.frame(n=c(rnorm(5e4), rnorm(5e4, mean=2)), c=c(rep('a', 5e4), rep('b', 5e4)))
3
4  library(ggplot2)
5  library(ggthemes)
6
7  ggplot(x) +
8    geom_density(aes(x=n, fill=c), alpha=0.5) +
9    theme_tufte(16)
10
```

10:1 (Top Level) R Script

Console /spin1/home/linux/wresch/

```
> library(ggthemes)

> ggplot(x) +
+   geom_density(aes(x=n)) +
+   theme_tufte(16)
> x <- data.frame(n=c(rnorm(5e4), rnorm(5e4, mean=2)), c=c(rep(1, 5e4), rep(2, 5e4)))
> ggplot(x) +
+   geom_density(aes(x=n, fill=c)) +
+   theme_tufte(16)
> x <- data.frame(n=c(rnorm(5e4), rnorm(5e4, mean=2)), c=c(rep('a', 5e4), rep('b', 5e4))
)
> ggplot(x) +
+   geom_density(aes(x=n, fill=c)) +
+   theme_tufte(16)
> ggplot(x) +
+   geom_density(aes(x=n, fill=c), alpha=0.5) +
+   theme_tufte(16)
>
```

Environment History Connections

Import Dataset List

Global Environment

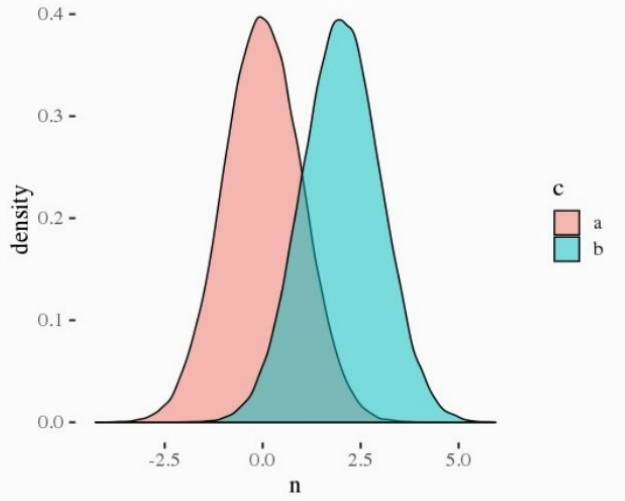
Data

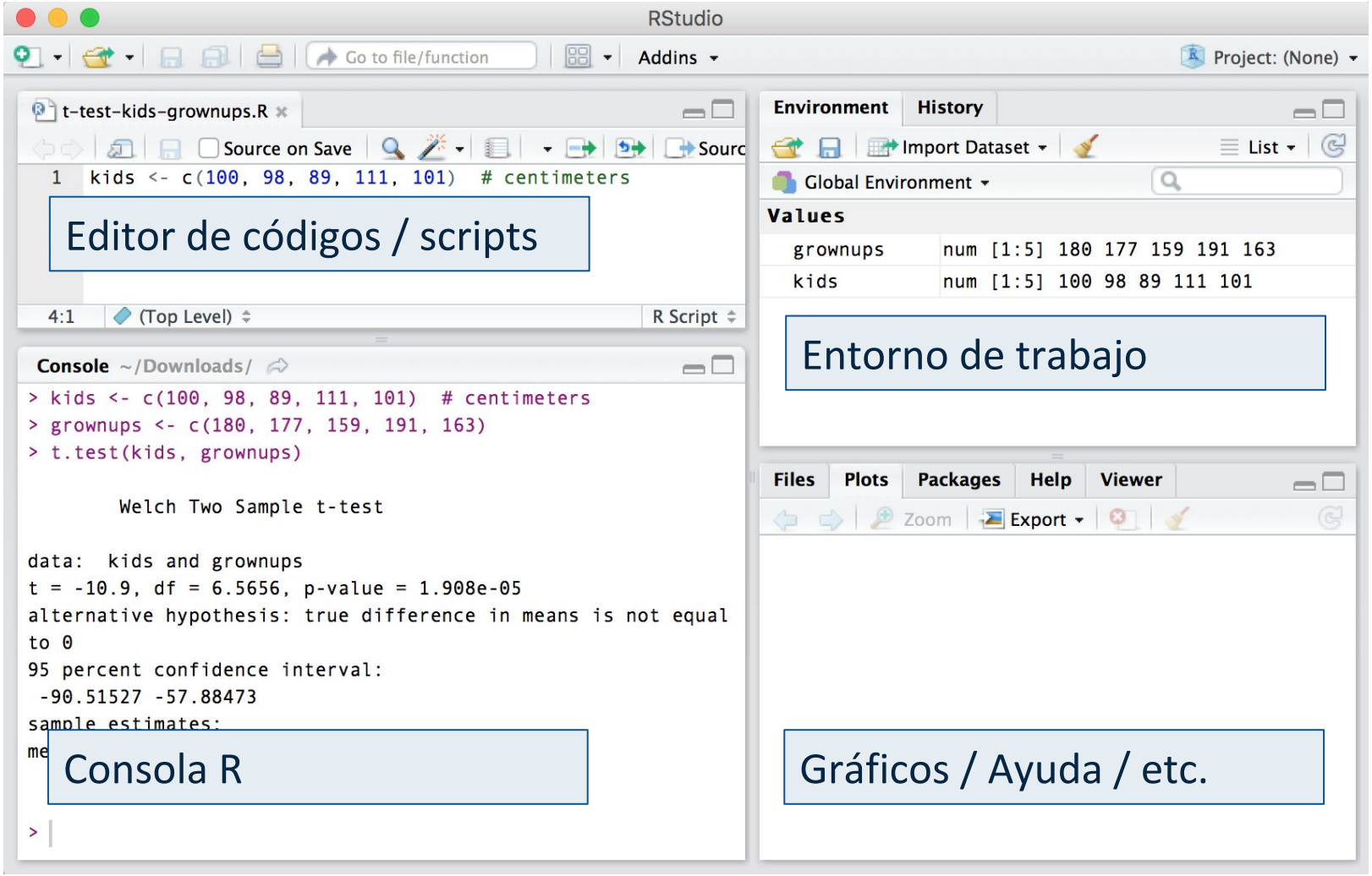
x 100000 obs. of 2 variables

Values

Files Plots Packages Help Viewer

Zoom Export Publish





Paquetes

- Flexibilidad y Extensibilidad de R
- `install.packages()` y `library()`

```
> install.packages("dplyr")
> install.packages("ggplot2")

> library(dplyr)
> library(ggplot2)
```

Objetos: Variables, Vectores & data.frames

Objetos: Variables

- Principios de codificación prolija
- “Asignar”: operador <-

```
> a <- 5
> a
[1] 5
> print(a)
[1] 5
```

Clases de objetos

Clases principales			
numeric		character	logical
integer	double		
1 25	34.6 0.18	“texto”	TRUE FALSE

Objetos: Variables

```
> a <- 5
> b <- 3

> a * b
[1] 15

> sqrt(b)
[1] 1.732051
```

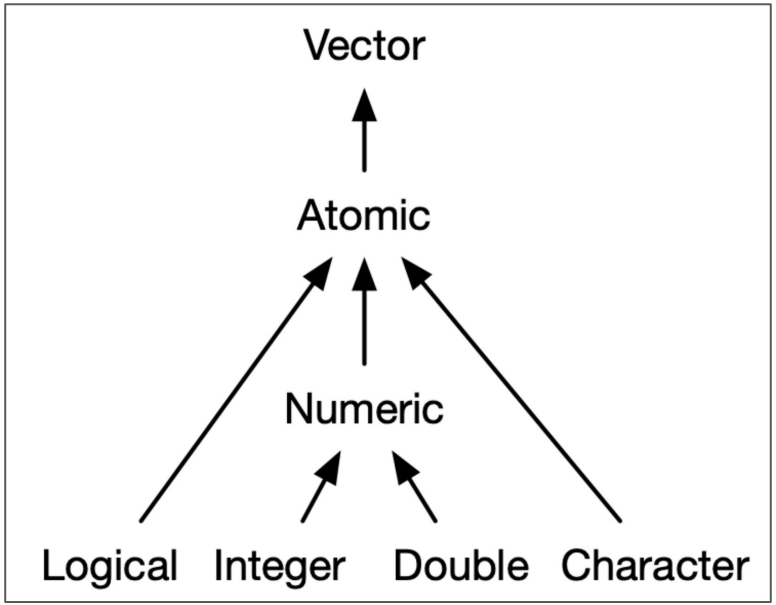
Funciones

```
> funcion()     # ejs. mean(), max(), read_csv()

> funcion(argumento1, argumento2, ...)

> ?funcion      # Abre ventana de ayuda
```

```
> b <- 3
> sqrt(b)
[1] 1.732051
```

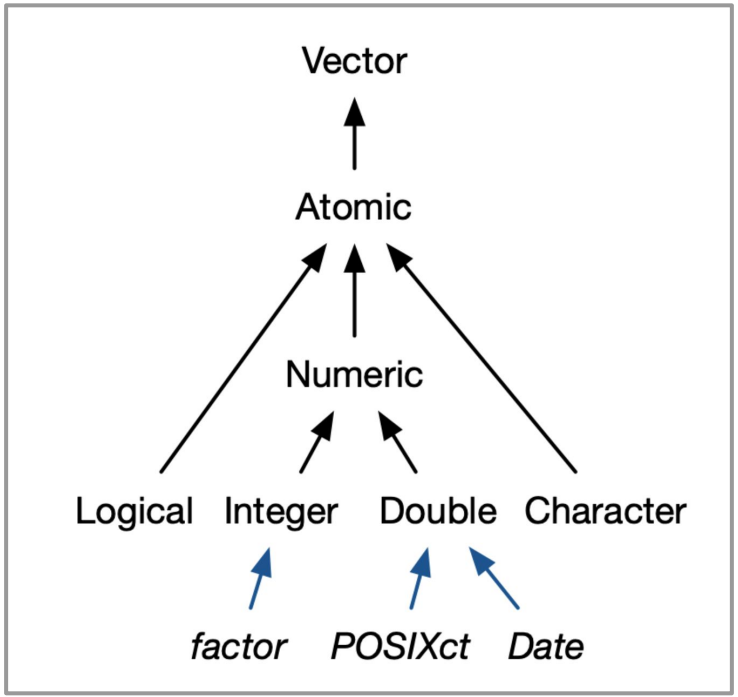
Objetos: Vectores

- **Vectores atómicos** y listas



Advanced R, Second Edition
H Wickham
Chapman & Hall/CRC

Vectores atómicos del sistema de objetos S3



Fechas y variables categóricas

Advanced R, Second Edition
H Wickham
Chapman & Hall/CRC

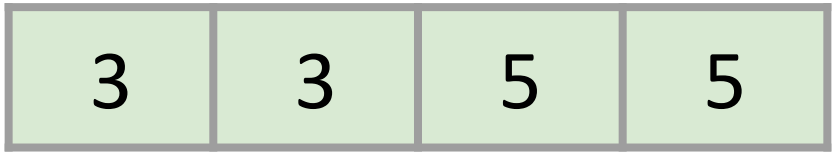
Vectores atómicos del sistema de objetos S3

- Factores: variables categóricas
- Fechas
- Fecha-hora (datetime)

In contrast to a character vector, when you tabulate a factor you'll get counts of all categories, even unobserved ones.
In base R you tend to encounter factors very frequently because many base R functions (like `read.csv()` and `data.frame()`) automatically convert character vectors to factors.
This is suboptimal because there's no way for those functions to know the set of all possible levels or their correct order. Use the argument `stringsAsFactors = FALSE` to suppress this behaviour.

Advanced R, Second Edition
H Wickham
Chapman & Hall/CRC

Crear Vectores: función concatenar c( )




```
> v1 <- c(3,3,5,5)
> v1
[1] 3 3 5 5
```

Crear Vectores

“Lucia”	“Rocamadour”	“Ronald”	“Babs”

```
> v2 <- c(“Lucia”, “Rocamadour”, “Ronald”, “Babs”)
> v2
[1] “Lucia”  “Rocamadour”  “Ronald”  “Babs”
```

Crear Vectores

```
> v1 <- c(3,4,5,6,7)
> v1
[1] 3 4 5 6 7

> v2 <- 1:4
> v2
[1] 1 2 3 4

> v3 <- c("items", "de", "un", "vector")
> v3
[1] "items"  "de"     "un"     "vector"
```

Vectores

1	“m”	182.5
2	“f”	163
3	“f”	157
4	“m”	177.5

Acceder a elementos de un vector [ ]


```
> vect <- c(3,3,4,5,9)
> vect
[1] 3 3 4 5 9
```

Acceder a elementos de un vector [ ]


```
> vect[4]
[1] 5

> vect[1:3]
[1] 3 3 4
```

```
> vect[c(1,4)]
[1] 3 5

> vect[-4]
[1] 3 3 4 9
```

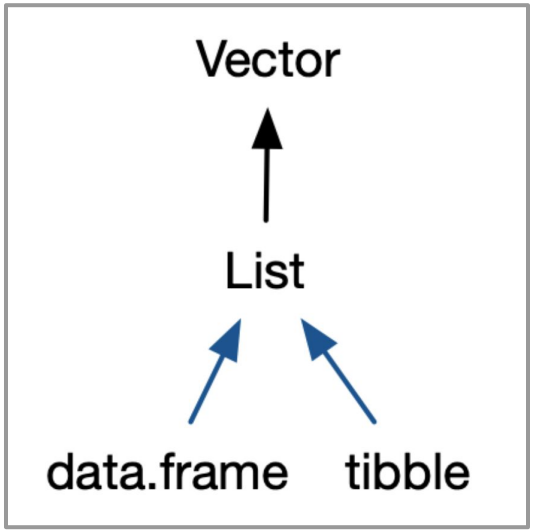
Acceder a elementos de un vector [ ]


```
> vect[c(TRUE, TRUE, FALSE, FALSE, FALSE)]
[1] 3 3
```

Acceder a elementos de un vector [ ]


```
> vect[c(TRUE, TRUE, FALSE, FALSE, FALSE)]
[1] 3 3

> vect[vect == 3]  # Usando condiciones
[1] 3 3
> vect[vect > 4]
[1] 5 9
```

Tablas: Data.frames y tibbles



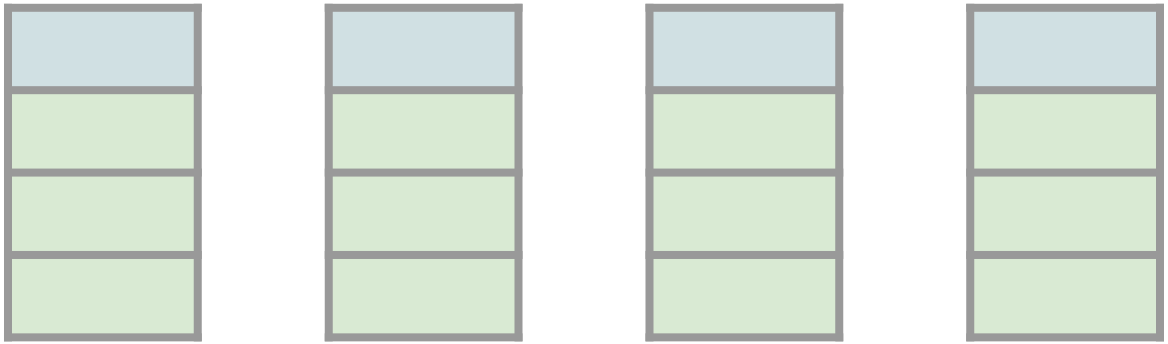
The two most important S3 vectors built on top of lists are data frames and tibbles

Advanced R, Second Edition
H Wickham
Chapman & Hall/CRC

Tablas: Data.frames y tibbles

- Objetos columnares:

  Vectores del mismo largo (columnas de una tabla)



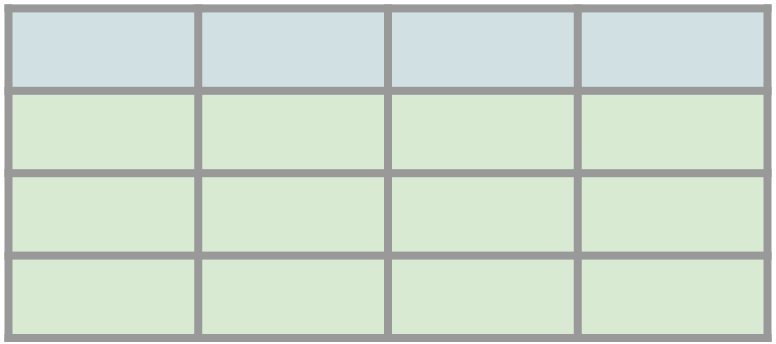
Es recomendable no usar row.names

Advanced R, Second Edition
H Wickham
Chapman & Hall/CRC

Tablas: Data.frames y tibbles

- Objetos columnares:

  Vectores del mismo largo (columnas de una tabla)



Es recomendable no usar row.names

Advanced R, Second Edition
H Wickham
Chapman & Hall/CRC

Crear un data.frame

columna1	3	3	2	3	3

```
columna1 = c(3, 3, 2, 3, 3)
```

Crear un data.frame

columna1	3	3	2	3	3

columna2	“m”	“f”	“f”	“f”	“f”

columna3	2013	2014	2015	2016	2017

Crear un data.frame

```
data.frame(columna1 = c(3, 3, 2, 3, 3),
           columna2 = c("m", "f", "f", "f", "f"),
           columna3 = c(2013, 2014, 2015, 2016, 2017))
```

Crear un data.frame

```
data.frame(columna1 = c(3, 3, 2, 3, 3),
           columna2 = c("m", "f", "f", "f", "f"),
           columna3 = c(2013, 2014, 2015, 2016, 2017))
```

```
base1 <- data.frame(columna1 = c(3, 3, 2, 3, 3),
                    columna2 = c("m", "f", "f", "f", "f"),
                    columna3 = c(2013, 2014, 2015, 2016, 2017))
```

Crear un data.frame

```
data.frame(columna1 = c(3, 3, 2, 3, 3),
           columna2 = c("m", "f", "f", "f", "f"),
           columna3 = c(2013, 2014, 2015, 2016, 2017),
           stringsAsFactors = FALSE)
```

Crear un data.frame

```
> base1
  columna1 columna2 columna3
1        3        m     2013
2        3        f     2014
3        2        f     2015
4        3        f     2016
5        3        f     2017
```

Acceder a elementos de un tabla [,]

```
> tabla[filas, columnas]
```

Acceder a elementos de un tabla [filas, columnas]

```
> base1
  columna1 columna2 columna3
1        3        m     2013
2        3        f     2014
3        2        f     2015
4        3        f     2016
5        3        f     2017
```

```
> base1[2, ]
  columna1 columna2 columna3
2        3        f     2014

> base1[2:4, ]
  columna1 columna2 columna3
2        3        f     2014
3        2        f     2015
4        3        f     2016
```

Acceder a elementos de un tabla [filas, columnas]

```
> base1
  columna1 columna2 columna3
1        3        m     2013
2        3        f     2014
3        2        f     2015
4        3        f     2016
5        3        f     2017
```

```
> base1[2, 3]
[1] 2014

> base1[base1$columna3 > 2014, ]
  columna1 columna2 columna3
3        2        f     2015
4        3        f     2016
5        3        f     2017
```

Tablas: cargar una base de datos

Tipos de archivo

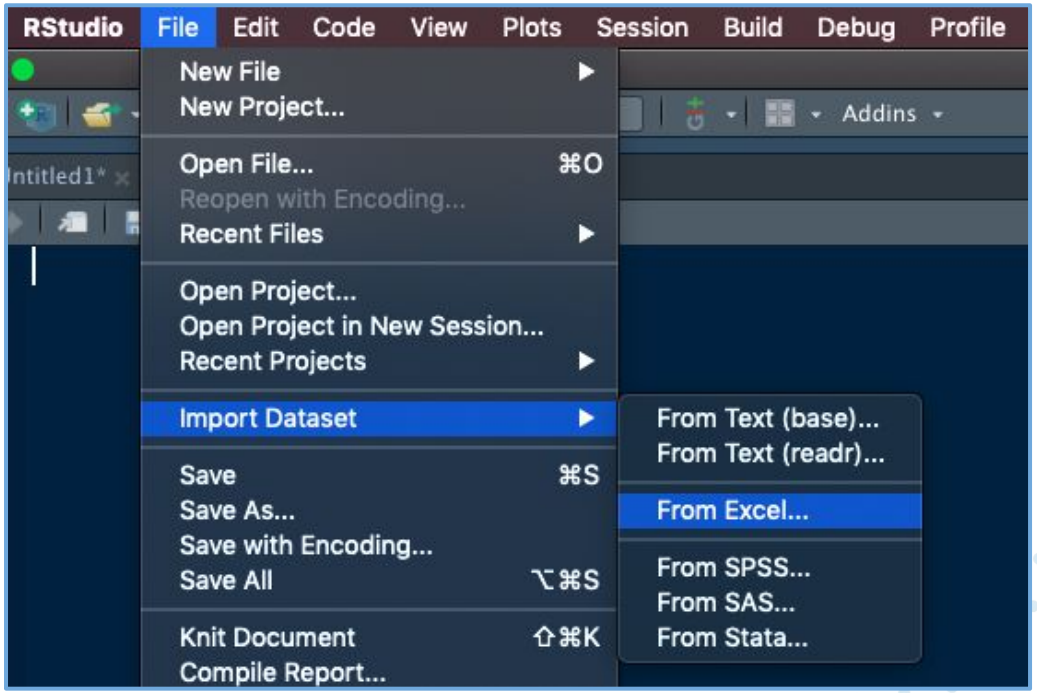
...

Tablas: cargar una base de datos

Usando R Studio:

File >

Import Dataset



Trabajar con tablas: `tidyverse`

Conjunto de paquetes para data science

```
install.packages("tidyverse")
library(dplyr)
library(ggplot2)
```

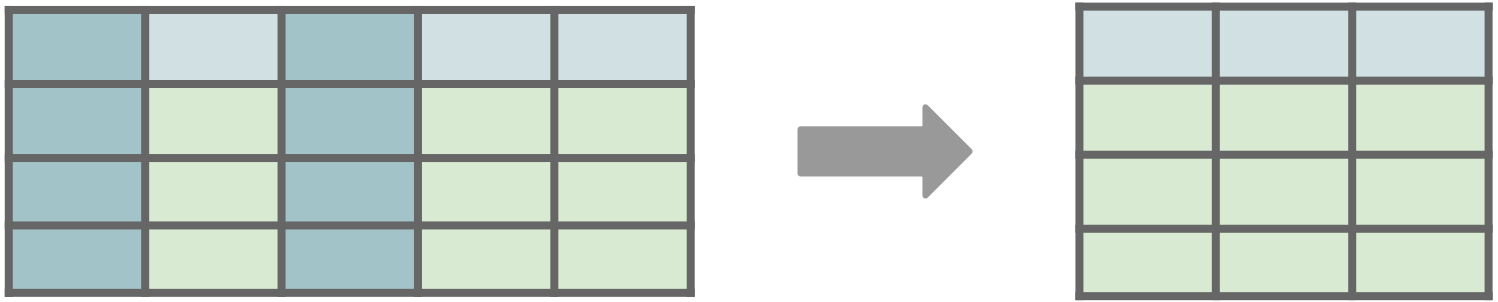
dplyr pipes %>%

Funciones sucesivas:

```
fx3(fx2(fx1(objeto)))     # R base

objeto %>% fx1() %>% fx2() %>% fx3()  # dplyr
```

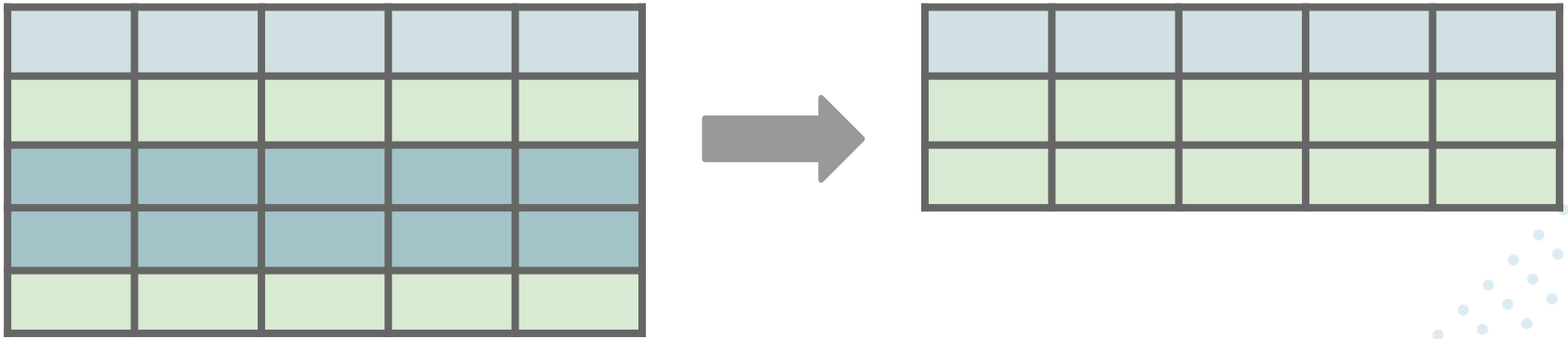
Seleccionar columnas: dplyr::select

```
tabla %>% select(2,4,5)
```



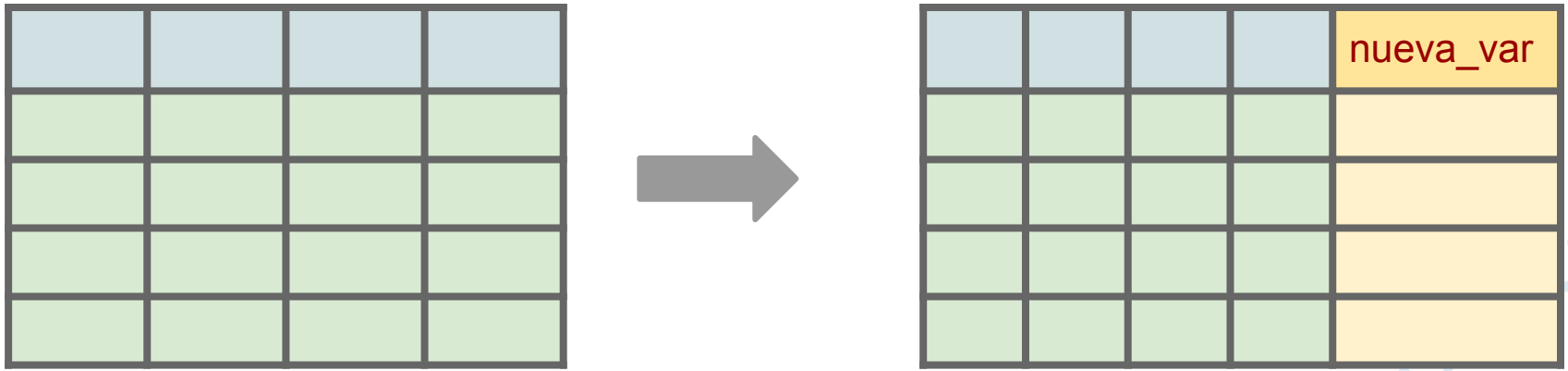
Filtrar filas: `dplyr::filter`

```
tabla %>% filter(condición)
```



Crear nueva variable: `dplyr::mutate`

```
tabla %>% mutate(nueva_var = ...)
```



XIV Jornadas de Informática en Salud

HOSPITAL ITALIANO de Buenos Aires
Departamento de Informática en Salud

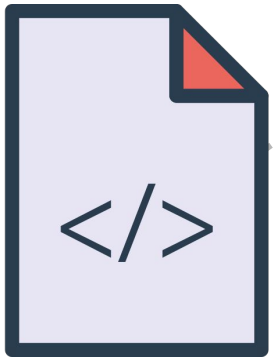
Taller con set de datos: material

https://github.com/f-binder/r_jis_2019

Taller con set de datos

```
# Instalar paquetes necesarios
install.packages(c("tidyverse", "lubridate",
"forcats", "ggplot2"))

# Cargar paquetes
library(dplyr)
library(ggplot2)
library(lubridate)
library(forcats)
```

Taller con set de datos: Código



```
# Cargar datos ------
      read.csv()
# Wrangling ------
       %>%
# Creación de variables -----
       %>% mutate()
# Análisis -----


# etc ------
```

Taller con set de datos

consultorios.csv


```
getwd()

consult <- read.csv("consultorios.csv",
                    stringsAsFactors = FALSE)
```

Cargar set de datos

Usando R Studio:

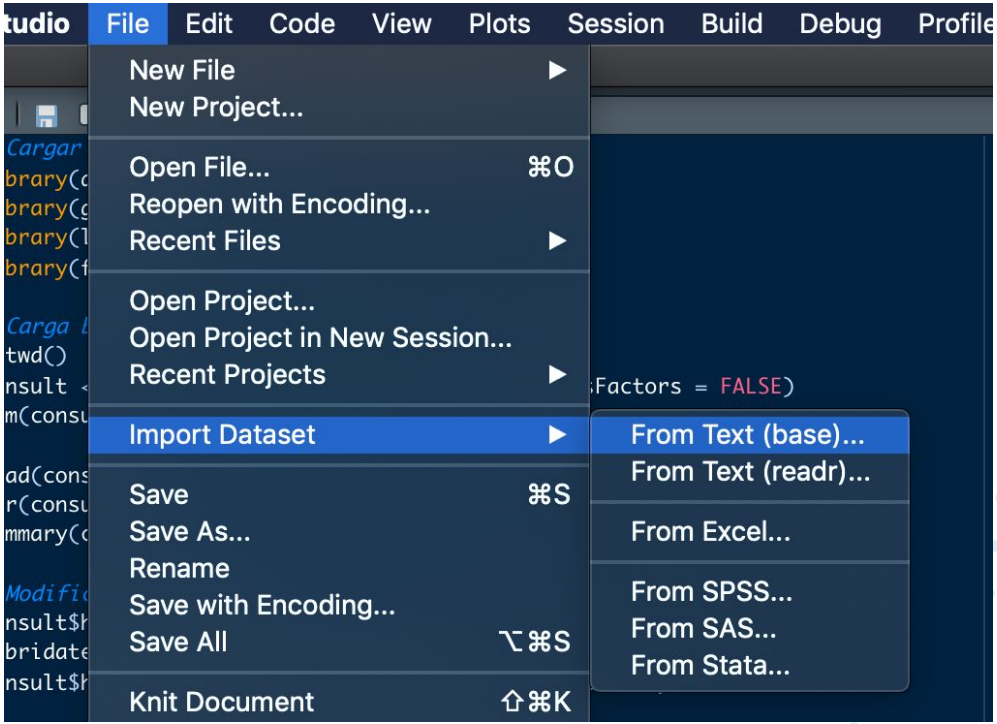
File

v

Import Dataset

v

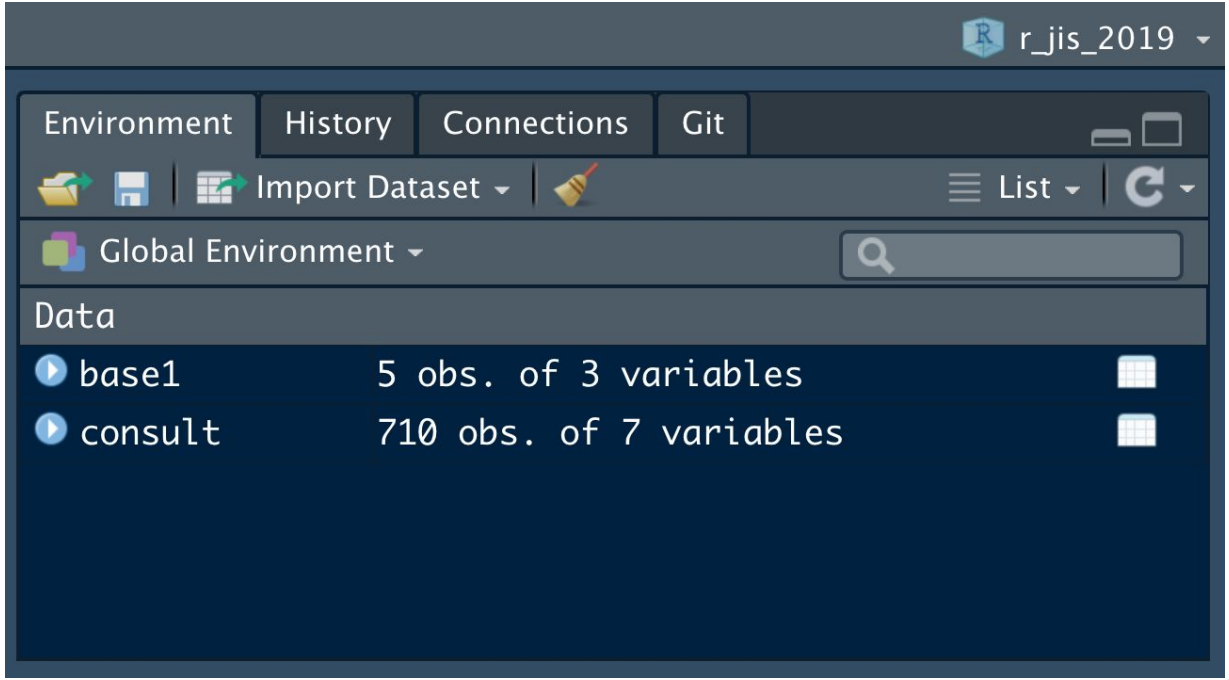
From Text (base)



Cargar set de datos: código

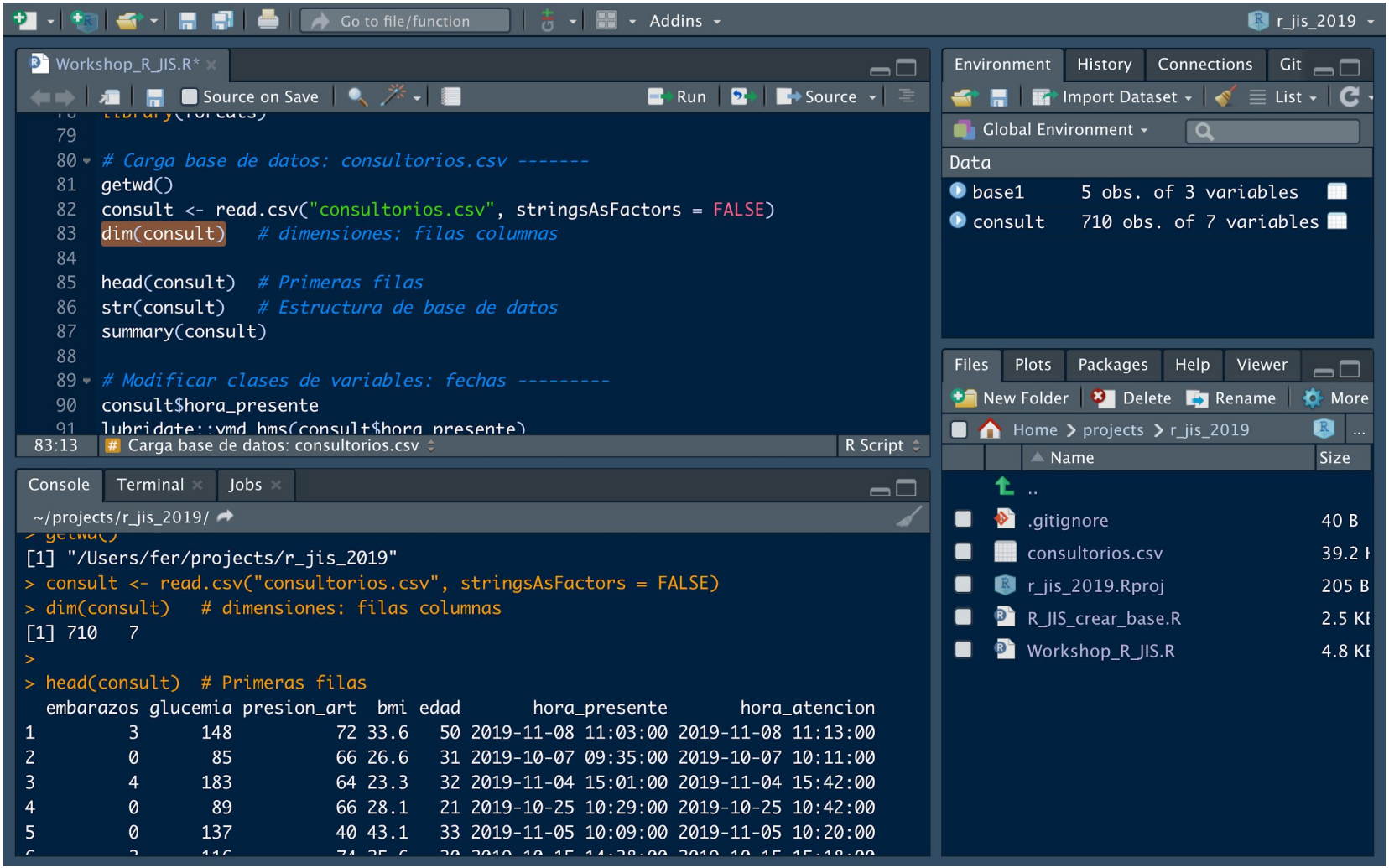
```
getwd()    # get working directory

[1] "/Users/R_User/projects/r_jis_2019"


consult <- read.csv("consultorios.csv",
                    stringsAsFactors = FALSE)
```

Cargar set de datos



Explorar data.frame

```
dim(consult)   # dimensiones: filas columnas

head(consult)  # Primeras filas
str(consult)   # Estructura de base de datos

summary(consult)
```



dim() head() str() summary()

```
> dim(consult)
[1] 710   7

> head(consult)
  embarazos glucemia presion_art  bmi edad       hora_presente       hora_atencion
1         3      148          72 33.6   50 2019-11-08 11:03:00 2019-11-08 11:13:00
2         0       85          66 26.6   31 2019-10-07 09:35:00 2019-10-07 10:11:00
3         4      183          64 23.3   32 2019-11-04 15:01:00 2019-11-04 15:42:00
4         0       89          66 28.1   21 2019-10-25 10:29:00 2019-10-25 10:42:00
5         0      137          40 43.1   33 2019-11-05 10:09:00 2019-11-05 10:20:00
6         2      116          74 25.6   30 2019-10-15 14:38:00 2019-10-15 15:18:00
```

dim() head() str() summary()

Console | Terminal | Jobs

~/projects/r_jis_2019/

```
00
> str(consult)   # Estructura de base de datos
'data.frame':   710 obs. of  7 variables:
 $ embarazos    : int  3 0 4 0 0 2 2 1 4 2 ...
 $ glucemia     : int  148 85 183 89 137 116 78 197 125 110 ...
 $ presion_art  : int  72 66 64 66 40 74 50 70 96 92 ...
 $ bmi          : num  33.6 26.6 23.3 28.1 43.1 25.6 31 30.5 0 37.6 ...
 $ edad         : int  50 31 32 21 33 30 26 53 54 30 ...
 $ hora_presente: chr  "2019-11-08 11:03:00" "2019-10-07 09:35:00" "2019-11-04 1
5:01:00" "2019-10-25 10:29:00" ...
 $ hora_atencion: chr  "2019-11-08 11:13:00" "2019-10-07 10:11:00" "2019-11-04 1
5:42:00" "2019-10-25 10:42:00" ...
```

Modificar y crear variables

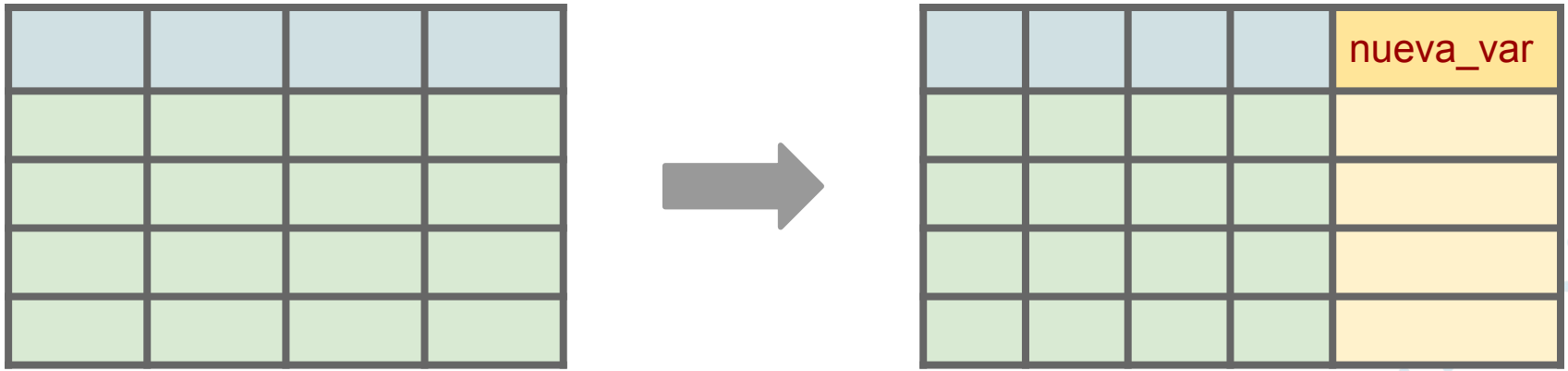
```
# Modificar clases de variables: fechas ---------
consult$hora_presente
   [1] "2019-11-08 11:03:00" "2019-10-07 09:35:00" ...

lubridate::ymd_hms(consult$hora_presente)
```

Modificar y crear variables

```
# Modificar clases de variables: fechas ---------
consult$hora_presente
   [1] "2019-11-08 11:03:00" "2019-10-07 09:35:00" ...

lubridate::ymd_hms(consult$hora_presente)

consult$hora_presente <- ymd_hms(consult$hora_presente)

consult$hora_atencion
consult$hora_atencion <- ymd_hms(consult$hora_atencion)
```

Crear nueva variable: `dplyr::mutate`

```
tabla %>% mutate(nueva_var = ...)
```



Crear variables

```
# Usando dplyr (pipes), función mutate -------
# Variable nuliparidad:
consult %>% mutate(nulipar = ifelse(embarazos > 0, 0, 1))

> consult %>% mutate(nulipar = ifelse(embarazos > 0, 0, 1))
%>% select(embarazos, nulipar)
    embarazos nulipar
1           3       0
2           0       1
3           4       0
```

Crear variables

```
# Usando dplyr (pipes), función mutate -------
# Variable nuliparidad:

consult %>% mutate(nulipar = ifelse(embarazos > 0, 0, 1)) %>%
   select(embarazos, nulipar) %>% table()

consult <- consult %>% mutate(nulipar = ifelse(embarazos > 0,
0, 1))
```

Crear variables

```
# Usando dplyr (pipes), función mutate -------
# Variable nuliparidad:

consult %>% mutate(nulipar = ifelse(embarazos > 0, 0, 1)) %>%
   select(embarazos, nulipar) %>% table()

consult <- consult %>% mutate(nulipar = ifelse(embarazos > 0,
0, 1))
```

Crear variables

```
# Variable "minutos de espera":
consult %>% mutate(espera = difftime(hora_atencion,
                                     hora_presente,
                                     units = "mins")) %>% head()
```

Crear variables

```
# Variable "minutos de espera":
consult %>% mutate(espera = difftime(hora_atencion,
                                     hora_presente,
                                     units = "mins")) %>% head()
```

```
> # Variable "minutos de espera":
> consult %>% mutate(espera = difftime(hora_atencion,
+                                      hora_presente,
+                                      units = "mins")) %>% head()
  embarazos glucemia presion_art  bmi edad       hora_presente       hora_atencion nulipar  espera
1         3      148          72 33.6   50 2019-11-08 11:03:00 2019-11-08 11:13:00       0 10 mins
2         0       85          66 26.6   31 2019-10-07 09:35:00 2019-10-07 10:11:00       1 36 mins
3         4      183          64 23.3   32 2019-11-04 15:01:00 2019-11-04 15:42:00       0 41 mins
4         0       89          66 28.1   21 2019-10-25 10:29:00 2019-10-25 10:42:00       1 13 mins
5         0      137          40 43.1   33 2019-11-05 10:09:00 2019-11-05 10:20:00       1 11 mins
6         2      116          74 25.6   30 2019-10-15 14:38:00 2019-10-15 15:18:00       0 40 mins
>
```

Crear variables

```
# Variable "minutos de espera":
consult %>% mutate(espera = difftime(hora_atencion,
                                     hora_presente,
                                     units = "mins")) %>% head()

consult <- consult %>% mutate(espera = difftime(hora_atencion,
                                                hora_presente,
                                                units = "mins"))
```

Crear variables

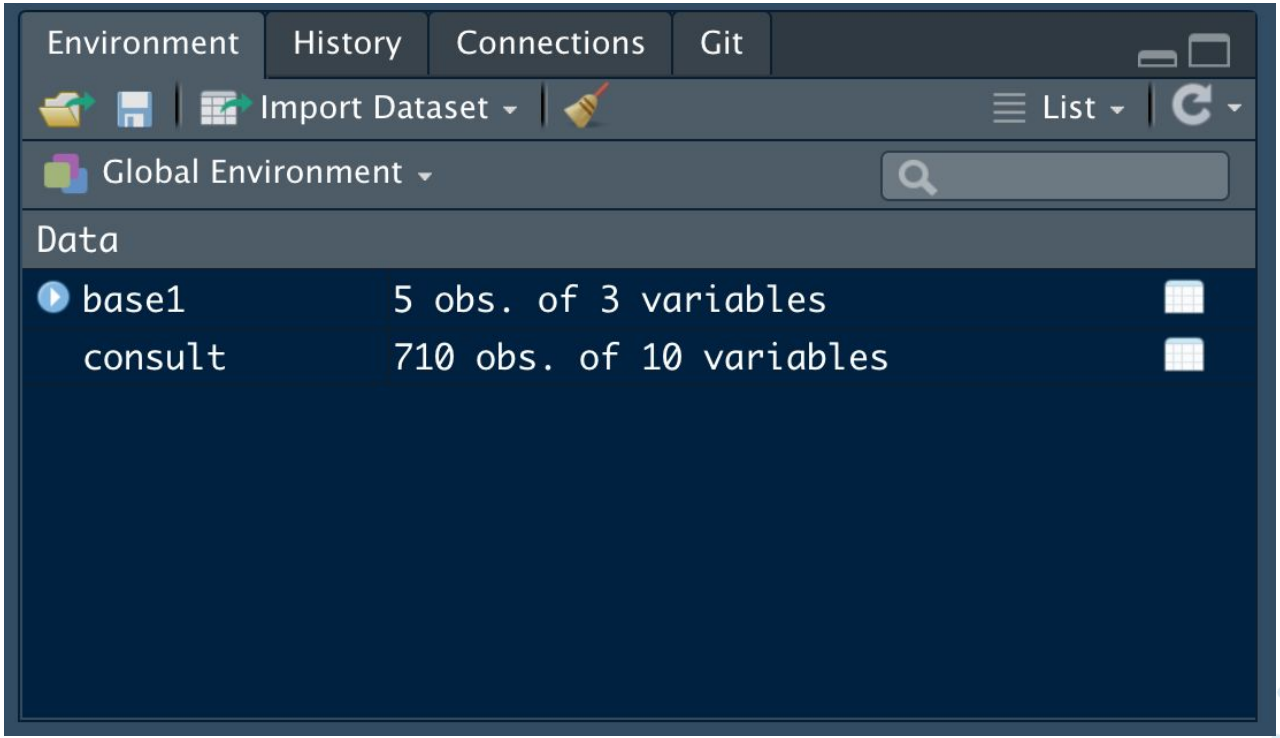
```
# Variable "semana del año"
consult %>% mutate(semana = week(hora_presente)) %>% head()
```

```
> consult %>% mutate(semana = week(hora_presente)) %>% head()
  embarazos glucemia presion_art  bmi edad       hora_presente       hora_atencion nulipar semana
1         3      148          72 33.6   50 2019-11-08 11:03:00 2019-11-08 11:13:00       0     45
2         0       85          66 26.6   31 2019-10-07 09:35:00 2019-10-07 10:11:00       1     40
3         4      183          64 23.3   32 2019-11-04 15:01:00 2019-11-04 15:42:00       0     44
4         0       89          66 28.1   21 2019-10-25 10:29:00 2019-10-25 10:42:00       1     43
5         0      137          40 43.1   33 2019-11-05 10:09:00 2019-11-05 10:20:00       1     45
6         2      116          74 25.6   30 2019-10-15 14:38:00 2019-10-15 15:18:00       0     42
```

Crear variables

```
# Variable "semana del año"
consult %>% mutate(semana = week(hora_presente)) %>% head()

consult <- consult %>% mutate(semana = week(hora_presente))
```

Crear variables



Análisis y gráficos

```
# Análisis Descriptivo ============
str(consult); summary(consult)
```

Análisis y gráficos

```
# Análisis Descriptivo ============
str(consult); summary(consult)
```

```
> str(consult)
'data.frame':   710 obs. of  10 variables:
 $ embarazos    : int  3 0 4 0 0 2 2 1 4 2 ...
 $ glucemia     : int  148 85 183 89 137 116 78 197 125 110 ...
 $ presion_art  : int  72 66 64 66 40 74 50 70 96 92 ...
 $ bmi          : num  33.6 26.6 23.3 28.1 43.1 25.6 31 30.5 0 37.6 ...
 $ edad         : int  50 31 32 21 33 30 26 53 54 30 ...
 $ hora_presente: POSIXct, format: "2019-11-08 11:03:00" "2019-10-07 09:35:00" ...
 $ hora_atencion: POSIXct, format: "2019-11-08 11:13:00" "2019-10-07 10:11:00" ...
 $ nulipar      : num  0 1 0 1 1 0 0 0 0 0 ...
 $ espera       : 'difftime' num  10 36 41 13 ...
  ..- attr(*, "units")= chr "mins"
 $ semana       : num  45 40 44 43 45 42 42 41 45 42 ...
```

Análisis y gráficos

```
# Análisis Descriptivo ============
str(consult); summary(consult)

  # Variable embarazos ---------
table(consult$embarazos) %>% addmargins()

consult %>% group_by(embarazos) %>%
    summarise(n = n(), prop = n()/nrow(consult))

fct_count(factor(consult$embarazos), prop = TRUE) %>% arrange(-n)
```

Análisis y gráficos

```
> table(consult$embarazos) %>% addmargins()

  0   1   2   3   4 Sum
246 103 200  50 111 710
```

Análisis y gráficos

```
>   # Variable tiempos de espera --------
> summary(consult$espera)
   Length     Class      Mode
      710 difftime   numeric
```

Análisis y gráficos

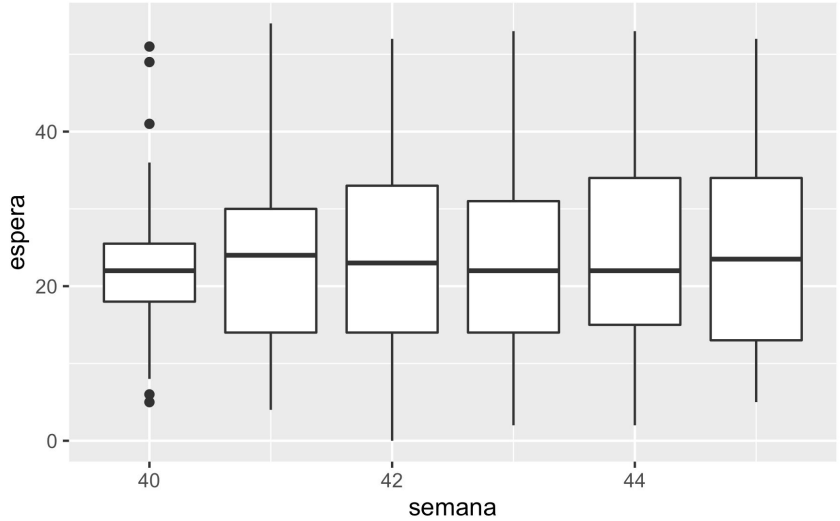
```
>   # Variable tiempos de espera --------
> summary(consult$espera)
  Length    Class     Mode
     710 difftime  numeric

> summary(as.numeric(consult$espera))
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00   15.00   23.00   23.98   32.75   54.00
```

Gráficos: paquete ggplot2

```
library(ggplot2)
consult %>% ggplot(aes(x = semana, y = espera))
```

Gráficos: paquete ggplot2

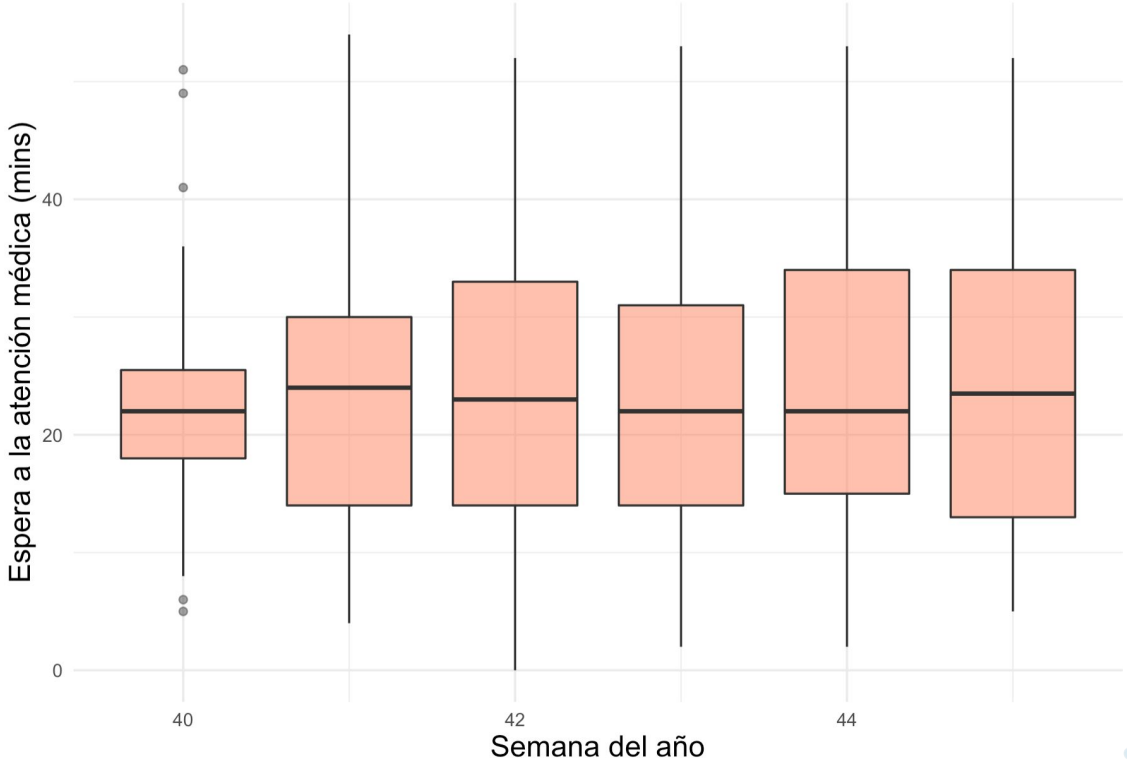
```
library(ggplot2)
consult %>% ggplot(aes(x = semana, y = espera)) +
   geom_boxplot(aes(group = semana))
```



Gráficos: paquete ggplot2

```
consult %>% ggplot(aes(x = semana, y = espera)) +
   geom_boxplot(aes(group = semana), fill = "coral", alpha = 0.5) +
   labs(y = "Espera a la atención médica (mins)",
        x = "Semana del año") +
   theme_minimal() +
   theme(axis.title = element_text(size = 14))
```

Gráficos: paquete `ggplot2`

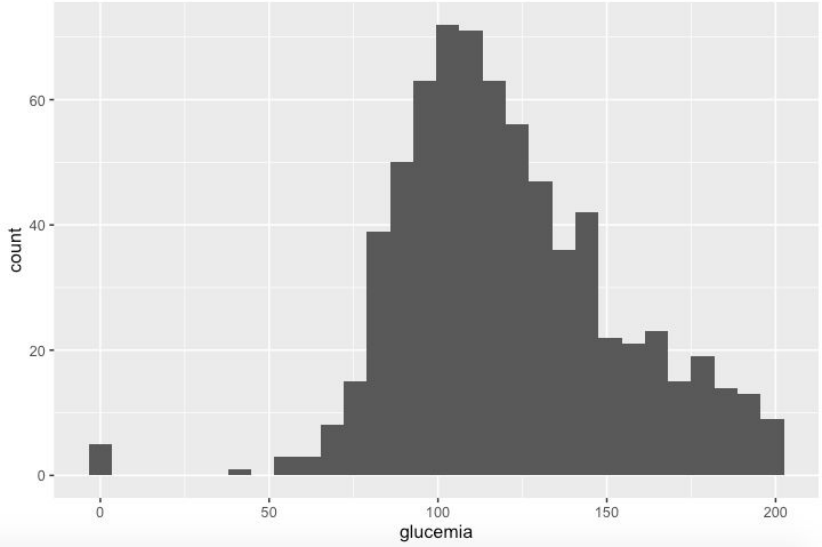


Gráficos: paquete ggplot2

```
# Variable glucemia --------
summary(consult$glucemia)

# Gráficos con paquete ggplot2
  # ggplot(aes(x = ..., y = ...))  define los ejes

consult %>% ggplot(aes(x = glucemia)) +
   geom_histogram()
```

Gráficos: paquete ggplot2

```
consult %>% ggplot(aes(x = glucemia)) +
   geom_histogram()
```

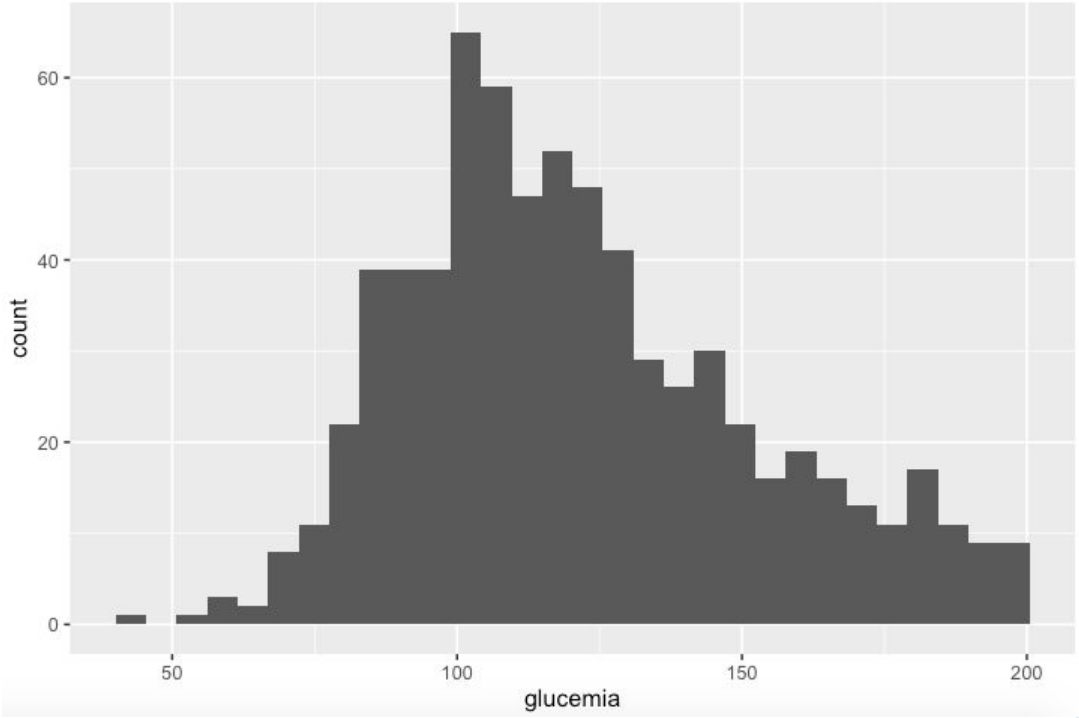


Gráficos: paquete ggplot2

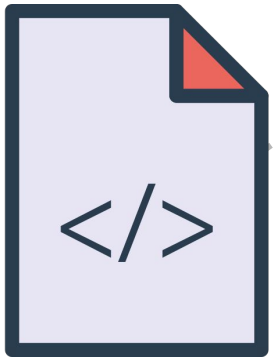
```
consult %>% filter(glucemia != 0) %>%
   ggplot(aes(x = glucemia)) +
   geom_histogram()
```

```
consult[consult$glucemia != 0, ] %>%
   ggplot(aes(x = glucemia)) +
   geom_histogram()
```

Histograma excluyendo glucemias == 0



Taller con set de datos: Código



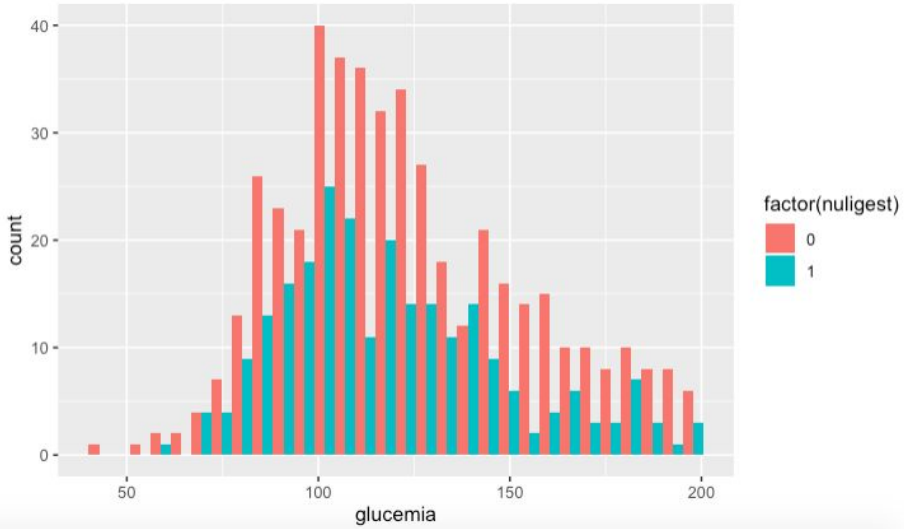
```
# Cargar datos ------
      read.csv()
# Wrangling ------
       %>%
# Creación de variables -----
       %>% mutate()
# Análisis -----


# etc ------
```

Gráficos: paquete ggplot2

```
consult %>% filter(glucemia != 0) %>%
   ggplot(aes(x = glucemia, fill = factor(nuligest))) +
   geom_histogram(position = "dodge")
```

Gráficos: paquete ggplot2

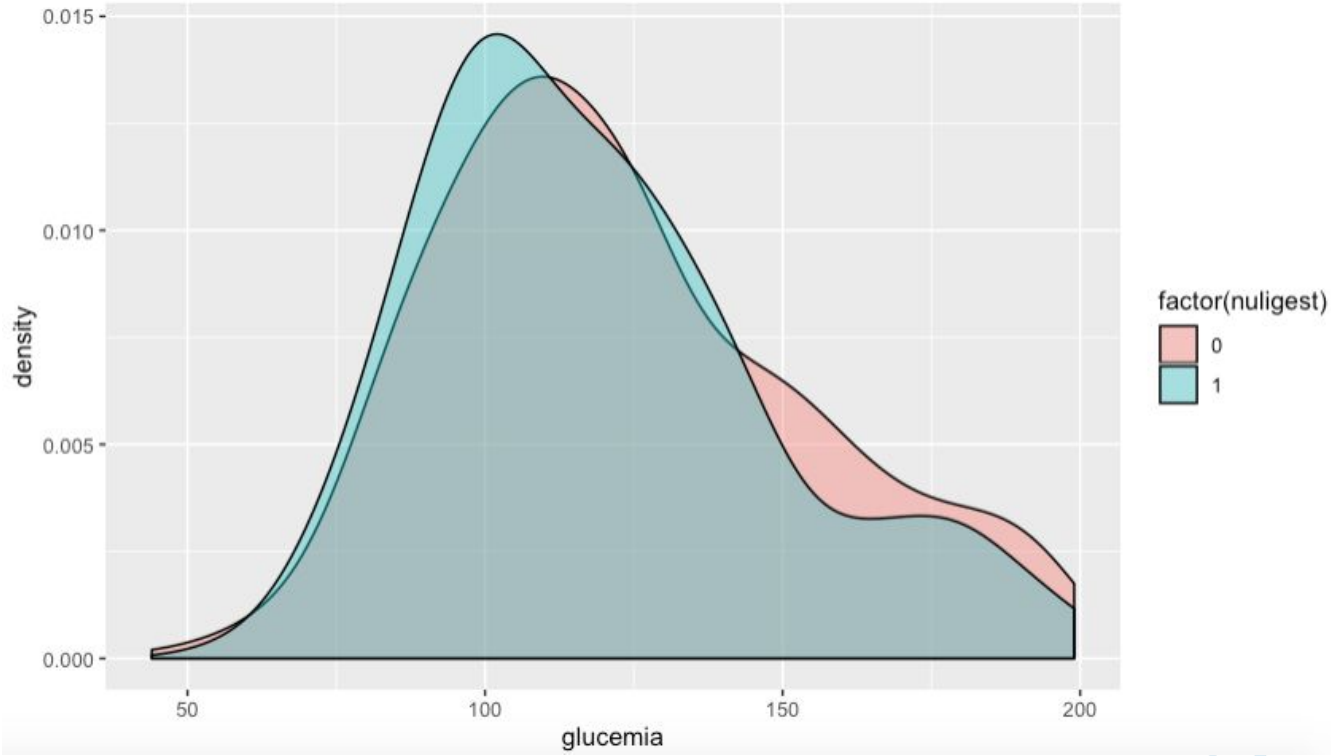
```
consult %>% filter(glucemia != 0) %>%
   ggplot(aes(x = glucemia, fill = factor(nuligest))) +
   geom_histogram(position = "dodge")
```



Gráficos: paquete ggplot2

```
# Variable glucemia --------
consult %>% filter(glucemia != 0) %>%
   ggplot(aes(x = glucemia)) +
   geom_density(aes(fill = factor(nuligest)),
                position = "dodge", alpha = 0.4)
```

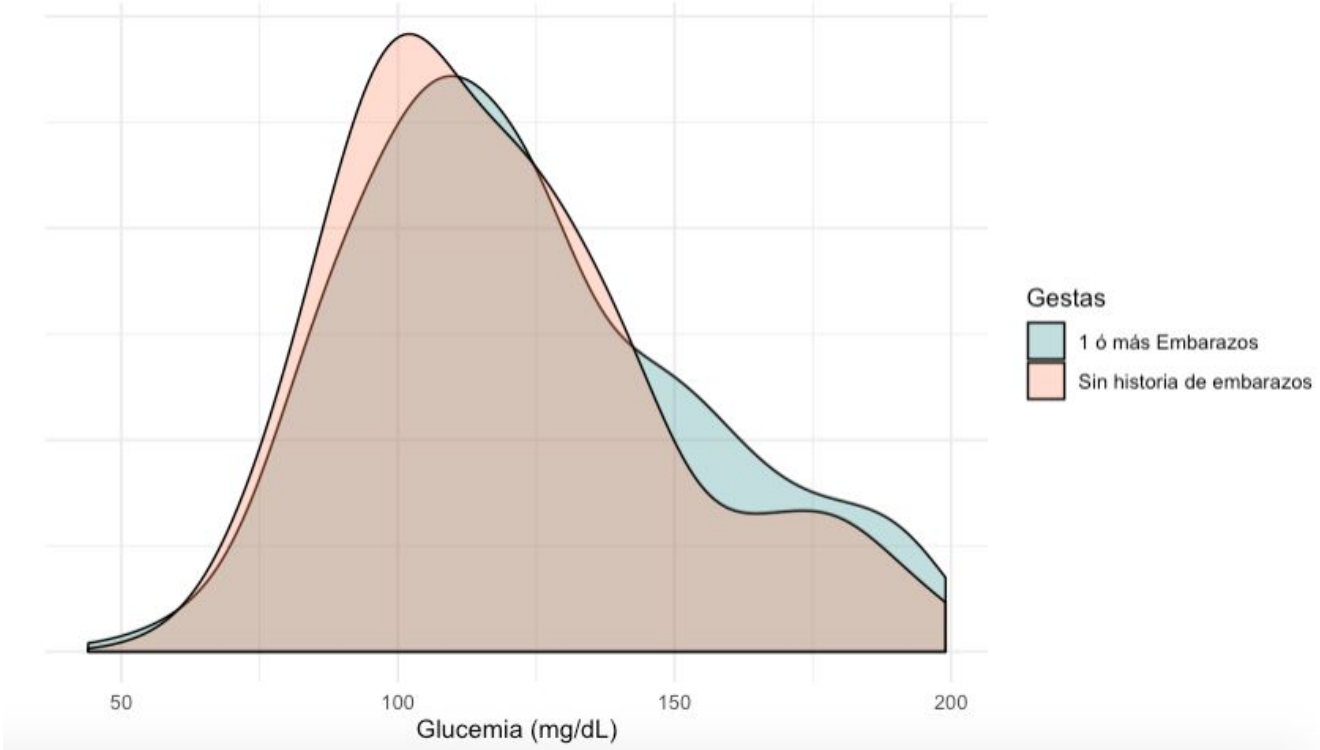
Gráficos: paquete `ggplot2`



Gráficos: paquete ggplot2

```
consult %>% filter(glucemia != 0) %>%
   ggplot(aes(x = glucemia)) +
   geom_density(aes(fill = factor(nuligest)),
                      position = "dodge", alpha = 0.3) +
   scale_fill_manual(name = "Gestas",
                     values = c("darkcyan", "coral"),
                     labels = c("1 ó más Embarazos",
                                "Sin historia de embarazos")) +
   theme_minimal() +
   labs(x = "Glucemia (mg/dL)", y = "") +
   theme(axis.text.y = element_blank())
```

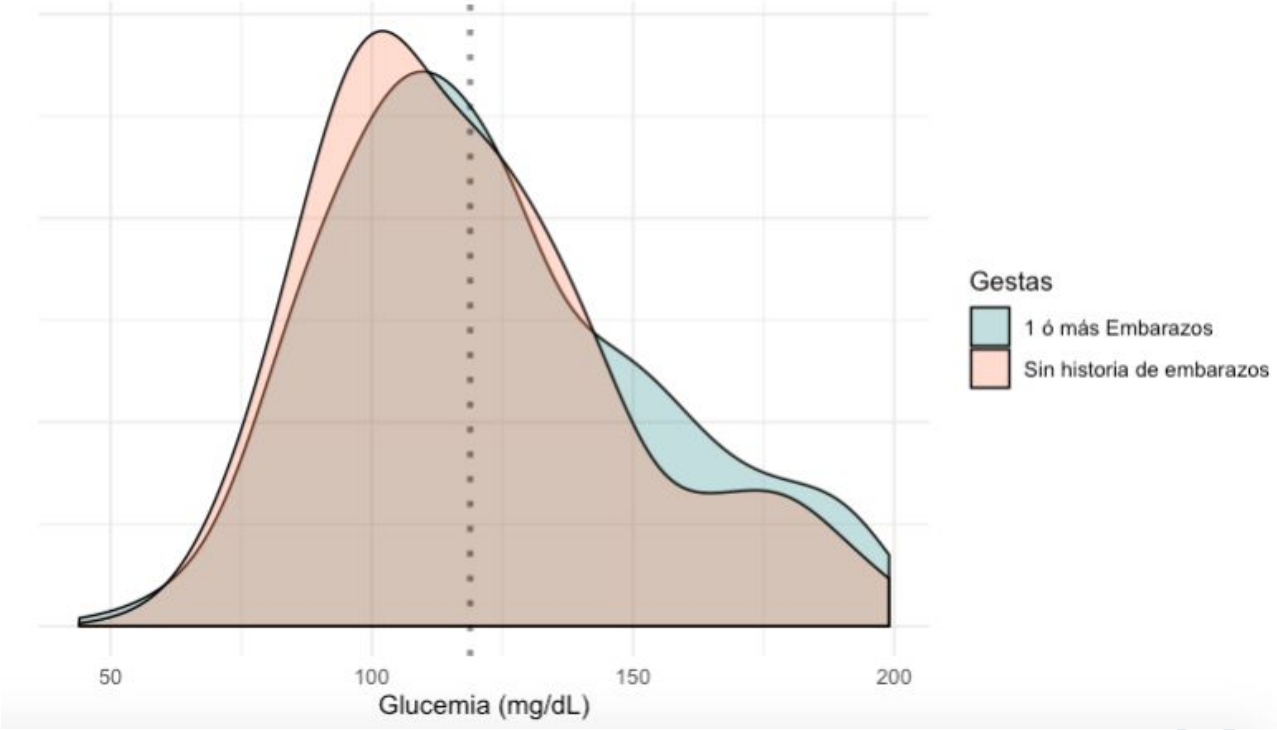
Gráficos: paquete `ggplot2`



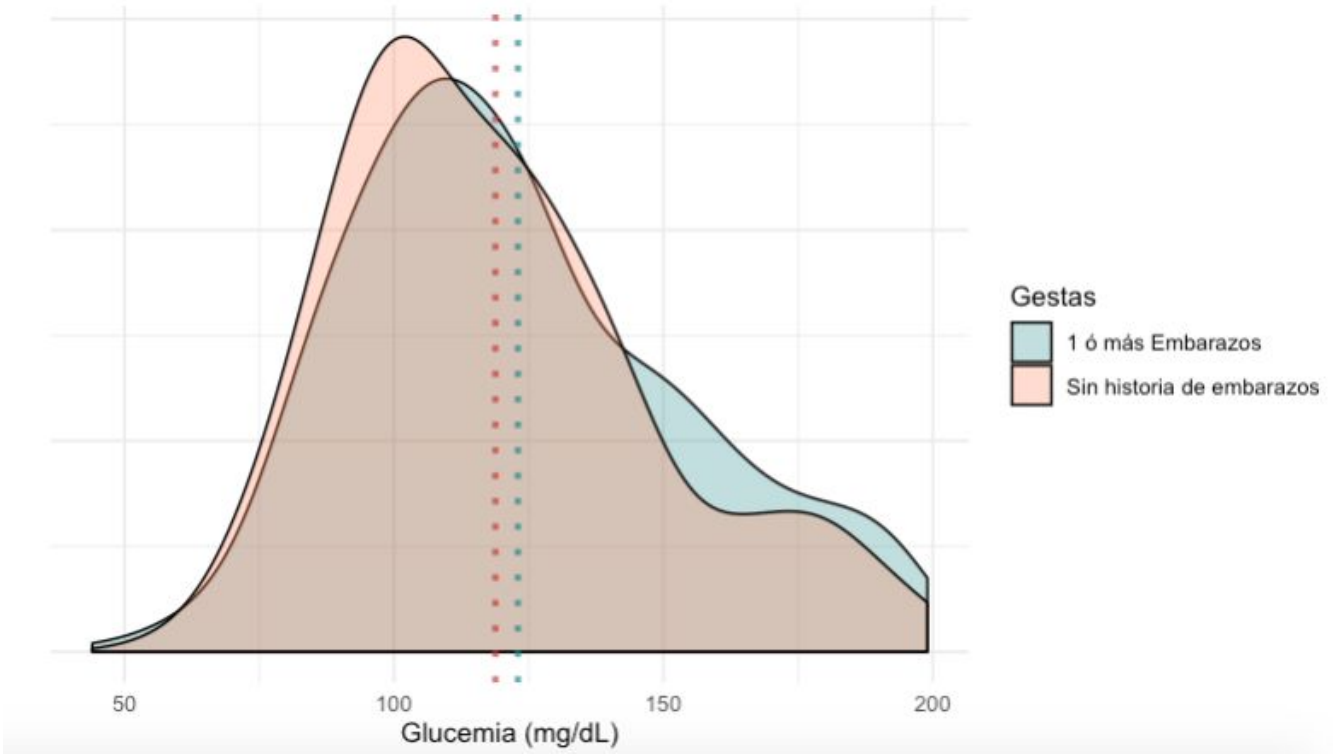
Gráficos: paquete ggplot2

```
consult %>% filter(glucemia != 0) %>%
   ggplot(aes(x = glucemia)) +
   geom_density(aes(fill = factor(nuligest)), position = "dodge", alpha = 0.3) +
   scale_fill_manual(name = "Gestas",
                     values = c("darkcyan", "coral"),
                     labels = c("1 ó más Embarazos", "Sin historia de embarazos"))
+
   theme_minimal() +
   labs(x = "Glucemia (mg/dL)", y = "") +
   theme(axis.text.y = element_blank()) +
   geom_vline(xintercept = mean(consult$glucemia[consult$glucemia !=0 &
consult$nuligest == 1]), linetype = "dotted", size = 1.2, alpha = 0.5)
```

Gráficos: paquete `ggplot2`



Gráficos: paquete `ggplot2`



Análisis estadístico

```
>   # Comparación de glucemias --------
consult %>% group_by(nuligest) %>%
   summarise("Glucemia Promedio" = mean(glucemia),
             "Glucemia Máxima" = max(glucemia))
```

Análisis estadístico

```
>   # Comparación de glucemias --------
consult %>% group_by(nuligest) %>%
   summarise("Glucemia Promedio" = mean(glucemia),
             "Glucemia Máxima" = max(glucemia))

  nuligest `Glucemia Promedio` `Glucemia Máxima`
     <dbl>               <dbl>             <int>
1        0                122.               197
2        1                117.               199
```

Análisis estadístico

```
>   # Comparación de glucemias: T-Test --------
t.test(glucemia ~ nuligest, data = consult)
```

Análisis estadístico

```
>   # Comparación de glucemias: T-Test --------
t.test(glucemia ~ nuligest, data = consult)
    Welch Two Sample t-test

data:  glucemia by nuligest
t = 2.0072, df = 502.62, p-value = 0.04526
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  0.1080275 10.0965913
sample estimates:
mean in group 0 mean in group 1
       122.4763        117.3740
```

Análisis estadístico

```
>   # Comparación de glucemias: T-Test --------
wilcox.test(glucemia ~ nuligest, data = consult)
```

Análisis estadístico

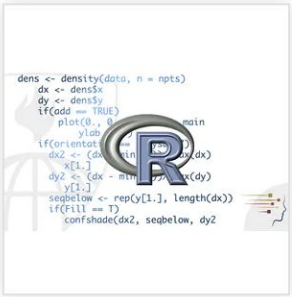
```
>   # Comparación de glucemias: T-Test --------
wilcox.test(glucemia ~ nuligest, data = consult)

Wilcoxon rank sum test with continuity correction

data:  glucemia by nuligest
W = 62026, p-value = 0.05682
alternative hypothesis: true location shift is not equal to 0
```

Principios de Codificación prolija

- Código leíble: acordar al iniciar para trabajos en grupo
- nombrar objetos:
    - minúsculas
    - preferible evitar caracteres especiales (excepto _)
    - variables: sustantivos
    - funciones: verbos
- Espacios, indentaciones, asignar con <- (y no con =)
- # Comentarios

Recursos online

- **Cursos**: EdX, Coursera, Udemy...
- R Cheat Sheets
- Libros / ebooks
- Comunidad de usuarios:
  - Stack Overflow
  - RDocumentation
  - r-bloggers
  - Github



R Programming

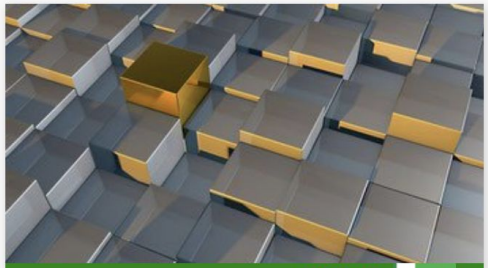
Johns Hopkins University

COURSE

4.6 (15,446) | 430K students

Intermediate

edX® HarvardX



VERIFIED

HarvardX

Data Science: R Basics

Current
Self-Paced

VERIFIED

HarvardX

Data Science: Wrangling

Current
Self-Paced

VERIFIED

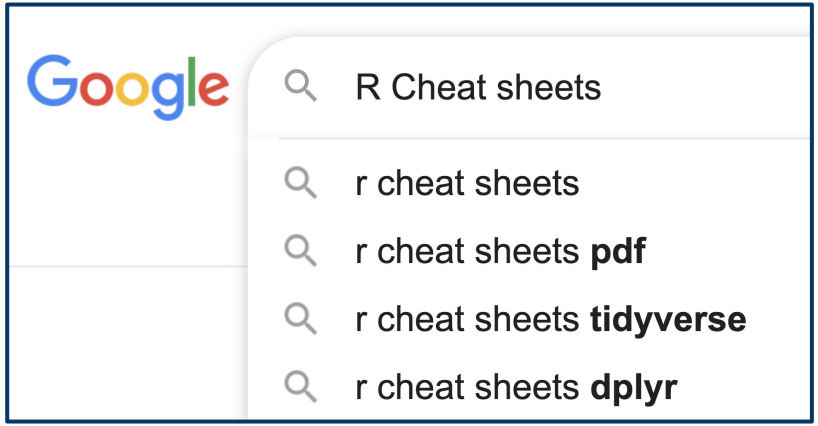
HarvardX

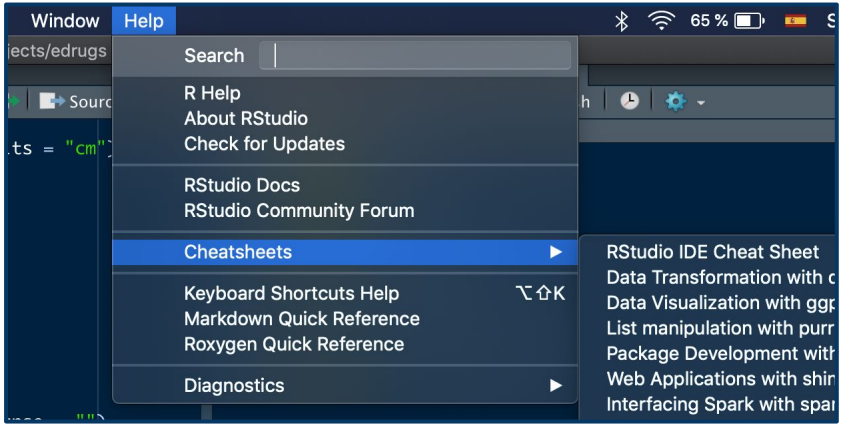
Data Science: Visualization

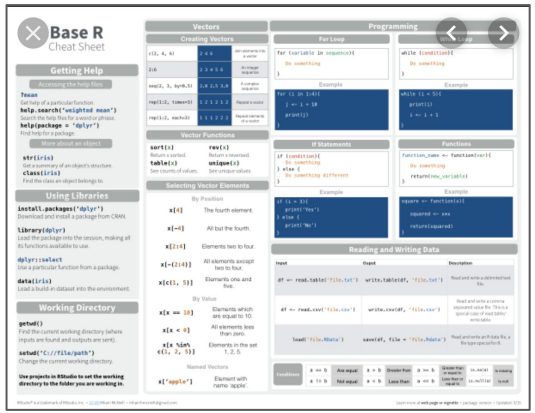
Current
Self-Paced

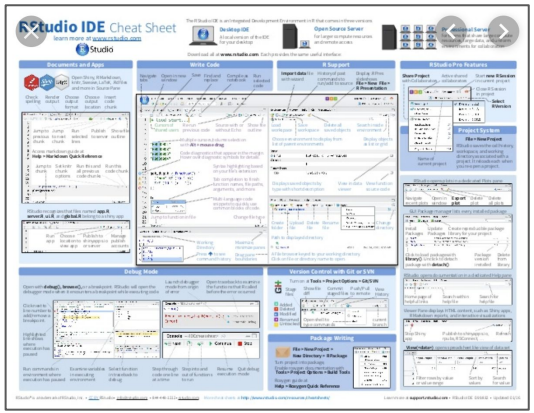
Recursos online

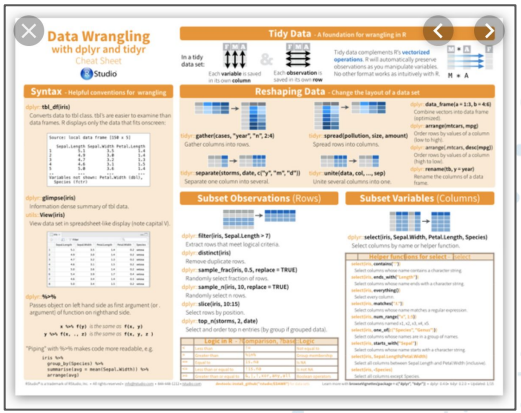
- Cursos online: EdX, Coursera, Udemy
- **R Cheat Sheets**
- Libros / ebooks
- Comunidad de usuarios:
  - Stack Overflow
  - RDocumentation
  - r-bloggers
  - Github











F Binder 2019

Data Wrangling with dplyr and tidyr Cheat Sheet

RStudio

Tidy Data - A foundation for wrangling in R

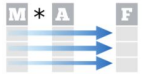
In a tidy data set:

Each **variable** is saved in its own **column**

&

Each **observation** is saved in its own **row**

Tidy data complements R's **vectorized operations**. R will automatically preserve observations as you manipulate variables. No other format works as intuitively with R.



M * A

Syntax - Helpful conventions for wrangling

dplyr::**tbl_df(iris)**

Converts data to tbl class. tbl's are easier to examine than data frames. R displays only the data that fits onscreen:

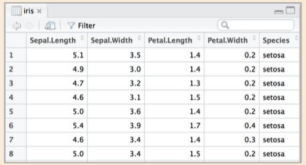
```
Source: local data frame [150 x 5]

   Sepal.Length Sepal.Width Petal.Length
1           5.1         3.5          1.4
2           4.9         3.0          1.4
3           4.7         3.2          1.3
4           4.6         3.1          1.5
5           5.0         3.6          1.4
..          ...         ...          ...
Variables not shown: Petal.Width (dbl),
  Species (fctr)
```

dplyr::**glimpse(iris)**

Information dense summary of tbl data.

utils::**View(iris)**

View data set in spreadsheet-like display (note capital V).



	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

dplyr::**%>%**

Passes object on left hand side as first argument (or . argument) of function on righthand side.

`x %>% f(y)` *is the same as* `f(x, y)`

`y %>% f(x, ., z)` *is the same as* `f(x, y, z )`

"Piping" with %>% makes code more readable, e.g.

```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Width)) %>%
  arrange(avg)
```

Reshaping Data - Change the layout of a data set

tidyr::**gather(cases, "year", "n", 2:4)**

Gather columns into rows.

tidyr::**spread(pollution, size, amount)**

Spread rows into columns.

tidyr::**separate(storms, date, c("y", "m", "d"))**

Separate one column into several.

tidyr::**unite(data, col, ..., sep)**

Unite several columns into one.

dplyr::**data_frame(a = 1:3, b = 4:6)**

Combine vectors into data frame (optimized).

dplyr::**arrange(mtcars, mpg)**

Order rows by values of a column (low to high).

dplyr::**arrange(mtcars, desc(mpg))**

Order rows by values of a column (high to low).

dplyr::**rename(tb, y = year)**

Rename the columns of a data frame.

Subset Observations (Rows)

dplyr::**filter(iris, Sepal.Length > 7)**

Extract rows that meet logical criteria.

dplyr::**distinct(iris)**

Remove duplicate rows.

dplyr::**sample_frac(iris, 0.5, replace = TRUE)**

Randomly select fraction of rows.

dplyr::**sample_n(iris, 10, replace = TRUE)**

Randomly select n rows.

dplyr::**slice(iris, 10:15)**

Select rows by position.

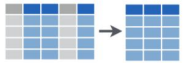
dplyr::**top_n(storms, 2, date)**

Select and order top n entries (by group if grouped data).

Logic in R - ?Comparison, ?base::Logic

<	Less than	!=	Not equal to
>	Greater than	%in%	Group membership
==	Equal to	is.na	Is NA
<=	Less than or equal to	!is.na	Is not NA
>=	Greater than or equal to	&, |, !, xor, any, all	Boolean operators

Subset Variables (Columns)



dplyr::**select(iris, Sepal.Width, Petal.Length, Species)**

Select columns by name or helper function.

Helper functions for select - ?select

select(iris, contains("."))
Select columns whose name contains a character string.

select(iris, ends_with("Length"))
Select columns whose name ends with a character string.

select(iris, everything())
Select every column.

select(iris, matches(".t."))
Select columns whose name matches a regular expression.

select(iris, num_range("x", 1:5))
Select columns named x1, x2, x3, x4, x5.

select(iris, one_of(c("Species", "Genus")))
Select columns whose names are in a group of names.

select(iris, starts_with("Sepal"))
Select columns whose name starts with a character string.

select(iris, Sepal.Length:Petal.Width)
Select all columns between Sepal.Length and Petal.Width (inclusive).

select(iris, -Species)
Select all columns except Species.

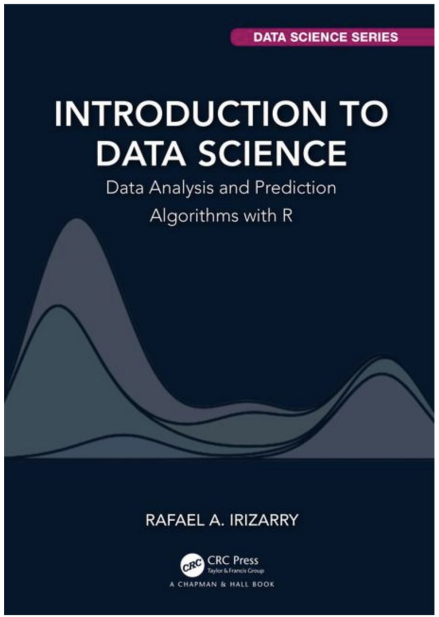
devtools::install_github("rstudio/EDAWR") for data sets

Learn more with browseVignettes(package = c("dplyr", "tidyr")) • dplyr 0.4.0• tidyr 0.2.0 • Updated: 1/15

Recursos online

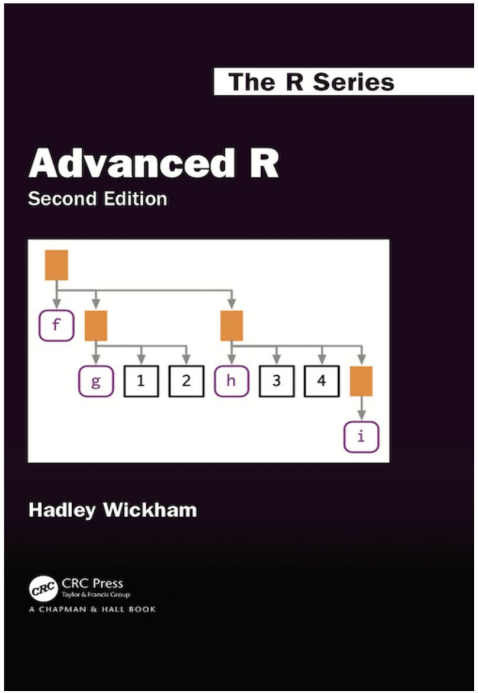
- Cursos online: EdX, Coursera, Udemy
- R Cheat Sheets
- **Libros / ebooks**
- Comunidad de usuarios:
  - Stack Overflow
  - RDocumentation
  - r-bloggers
  - Github

https://rafalab.github.io/dsbook/



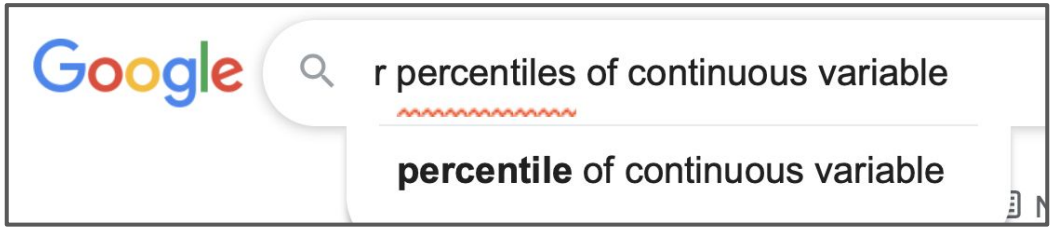
Recursos online

- Cursos online: EdX, Coursera, Udemy
- R Cheat Sheets
- **Libros / ebooks**
- Comunidad de usuarios:
  - Stack Overflow
  - RDocumentation
  - r-bloggers
  - Github



Recursos online

- Cursos online: EdX, Coursera, Udemy
- R Cheat Sheets
- Libros / ebooks
- **Comunidad de usuarios:**
  - **Stack Overflow**
  - **RDocumentation**
  - **r-bloggers**
  - **Github**



Calculating percentile of dataset column - Stack Overflow

https://stackoverflow.com › questions › calculating-per... ▾ Traducir esta página

50

If you order a vector `x`, and find the values that is half way through the vector, you just found a median, or 50th percentile. Same logic applies for any percentage. Here are two examples.

```
x <- rnorm(100)
quantile(x, probs = c(0, 0.25, 0.5, 0.75, 1)) # quartile
quantile(x, probs = seq(0, 1, by= 0.1)) # decile
```

share improve this answer

answered Jan 19 '14 at 17:05

Recursos online

- Cursos online: EdX, Coursera, Udemy
- R Cheat Sheets
- Libros / ebooks
- Comunidad de usuarios:
  - Stack Overflow
  - RDocumentation
  - r-bloggers
  - Github

¡Gracias!

fernando.binder@hospitalitaliano.org.ar

fbinder

HOSPITAL ITALIANO
de Buenos Aires

Departamento de Informática
en Salud