

Semana 01 - code sessions

Introducción a R, Rstudio y el paquete tidyverse

Carlos Daboín

April, 2022

Talleres de análisis de datos

Tras la culminación de cada clase teórica tendremos un taller donde revisaremos algunas maneras de aplicar los métodos presentados en clase.

Para ello nos valdremos de **R** y **Rstudio**

Si aún no lo has hecho:

1. Descarga e instala **R**.
2. Luego descarga e instala **RStudio**.



*Nota: También puedes trabajar desde tu buscador accediendo a **Rstudio Cloud**. El plan gratuito tiene límites de almacenamiento y procesamiento, pero basta para manejar las asignaturas del curso si lo usas bien.*

R y Rstudio

R es un ecosistema de software gratuito para análisis estadístico y la visualización de datos. **RStudio** es un Entorno Integrado de Desarrollo (IDE) que ayuda a los usuarios de R a programar cómodamente.

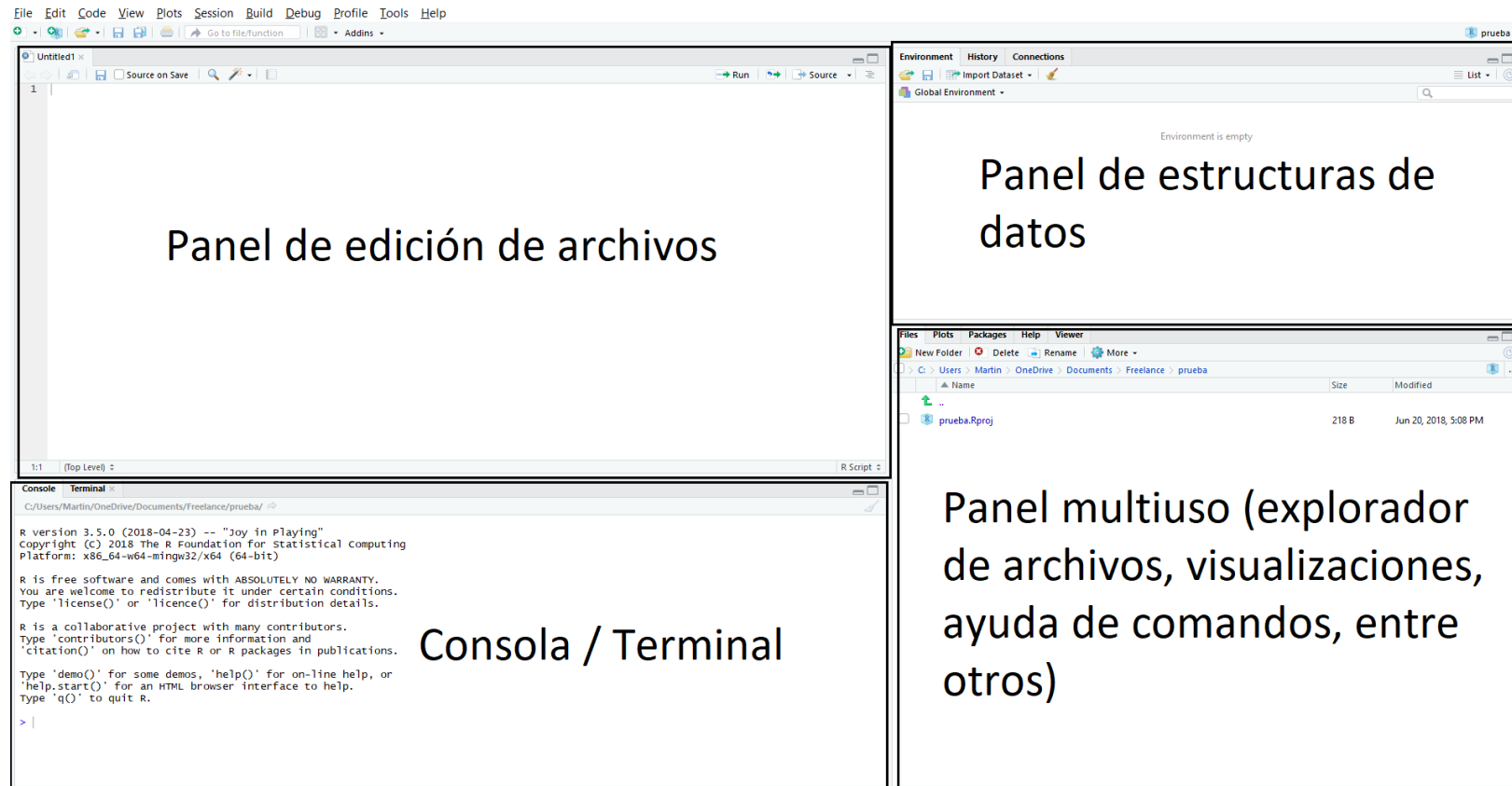
Piensa en **R como el motor** corriendo tu análisis, y en **Rstudio como la cabina de control**.



¿Por qué R y RStudio?

- Es gratis
- Comunidad activa e innovación constante (Tip: sigue a [@R4DScommunity](#) en twitter)
- Programación orientada objetos (tal cómo Python o Javascript)
- Excelentes librerías para el análisis y la visualización de datos
- Soluciones cómodas para crear [reportes](#), [presentaciones](#), libros, [páginas web](#), [APIs](#), y más
- Alta demanda en el mercado laboral

Un vistazo a tu ambiente de trabajo (R Studio IDE)



From Montané

Principios de la programación en R

1. Todo es un **objeto**
2. Cada objeto tiene un **nombre** y un **valor**
3. Puedes insertar los objetos en **funciones**
(bloques de código abreviado)
4. Las funciones vienen con **instrucciones**
5. Las funciones son empaquetadas en **librerías**
6. Las funciones emiten **alertas sobre posibles errores**

```
precio
```

```
precio<-100
```

```
log(precio, base = 10)
```

```
??log
```

```
library(ggplot2)
```

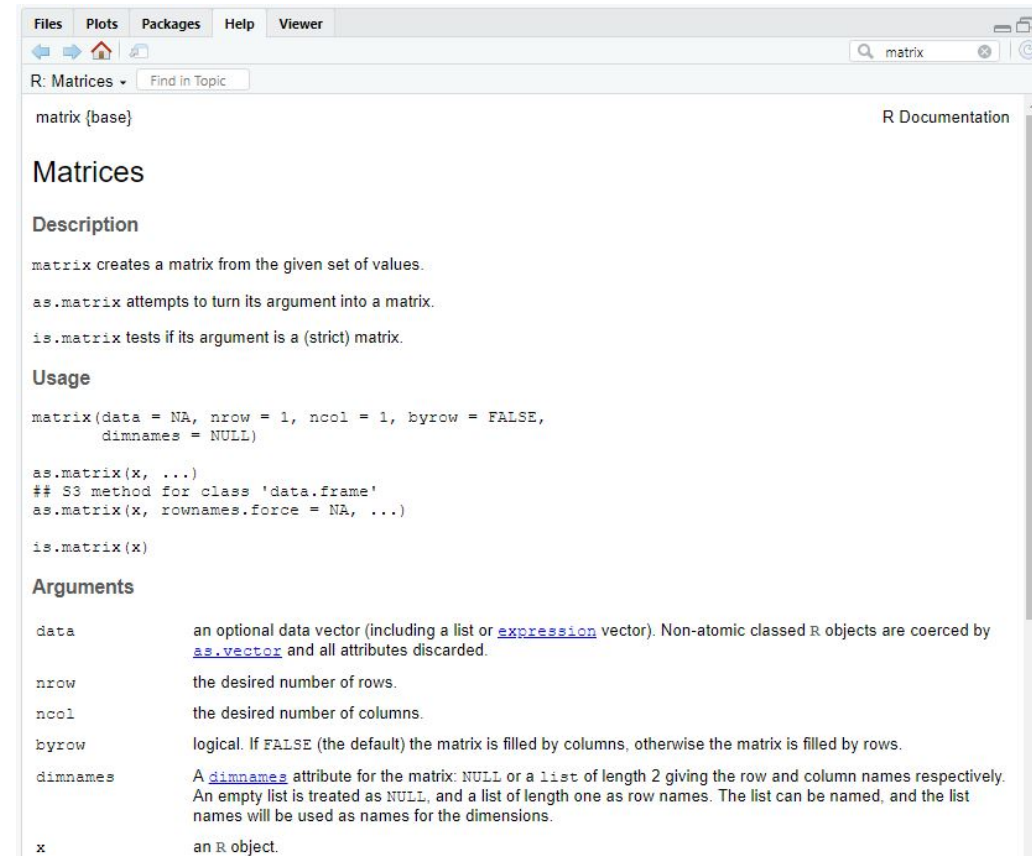
```
log(-1)
```

Objetos y funciones

Las matrices son objetos útiles en la programación y aplicación de métodos lineales.

Veamos cómo se crea una matriz en R con la función `matrix()`.

```
## revisa la documentación
??matrix()
# data: opcional, require insertar un vector
# nrow: número filas de la matriz
# ncol: número columnas de la matriz
```



The screenshot shows the R Documentation page for the `matrix` function. The page is titled "R: Matrices" and includes a search bar with "matrix" entered. The content is organized into sections: "Description", "Usage", and "Arguments".

Description

- `matrix` creates a matrix from the given set of values.
- `as.matrix` attempts to turn its argument into a matrix.
- `is.matrix` tests if its argument is a (strict) matrix.

Usage

```
matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE,
        dimnames = NULL)
as.matrix(x, ...)
## S3 method for class 'data.frame'
as.matrix(x, rownames.force = NA, ...)
```

Arguments

Argument	Description
<code>data</code>	an optional data vector (including a list or expression vector). Non-atomic classed R objects are coerced by as.vector and all attributes discarded.
<code>nrow</code>	the desired number of rows.
<code>ncol</code>	the desired number of columns.
<code>byrow</code>	logical. If FALSE (the default) the matrix is filled by columns, otherwise the matrix is filled by rows.
<code>dimnames</code>	A dimnames attribute for the matrix: NULL or a list of length 2 giving the row and column names respectively. An empty list is treated as NULL, and a list of length one as row names. The list can be named, and the list names will be used as names for the dimensions.
<code>x</code>	an R object.

From Mcdermott and Imbens

Objetos y funciones

```
# Creemos un vector con un cero
obj_1<-0

# Matriz A: 5x2 llena de ceros
A<-matrix(data = obj_1, nrow = 5, ncol = 2)

# Veamos la matriz A
A
```

```
##      [,1] [,2]
## [1,]    0    0
## [2,]    0    0
## [3,]    0    0
## [4,]    0    0
## [5,]    0    0
```

```
# Creemos un vector con numeros del 1 al 10
obj_2<-c(1:10)

# Matrix B: 5x2 con una secuencia numerica
B<-matrix( data = obj_2, nrow = 5, ncol = 2)

# Veamos la matriz B
B
```

```
##      [,1] [,2]
## [1,]    1    6
## [2,]    2    7
## [3,]    3    8
## [4,]    4    9
## [5,]    5   10
```


Funciones para análisis estadístico

Este es el código que vamos a correr.

```
x<-c(1:10) # vector x
y<-x*2+5   # vector y
# Mean
mean(x)
# Median
median(x)
# Std. dev. and variance
sd(x)
var(x)
# Min. and max.
min(x)
max(x)
# Correlation/covariance
cor(x, y)
cov(x, y)
# Quartiles and mean of x
summary(x)
```

Este es el output que veremos en la consola de Rstudio:

```
## [1] 5.5
## [1] 5.5
## [1] 3.02765
## [1] 9.166667
## [1] 1
## [1] 10
## [1] 1
## [1] 18.33333

##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      1.00   3.25   5.50    5.50   7.75   10.00
```

Funciones para análisis estadístico

Otras funciones

Este es el código que vamos a correr:

```
# Set seed (pin down random number generation)
set.seed(1)
# 4 random draws from N(3,5)
rnorm(n = 4, mean = 3, sd = sqrt(5))
# CDF for N(0,1) at z=1.96
pnorm(q = 1.96, mean = 0, sd = 1)
# Sample 5 draws from x w/ repl.
sample(
  x = x,
  size = 5,
  replace = T
)
# First and last 3 elements of x
head(x, 3)
tail(x, 3)
```

Este es el output que veremos en la consola de Rstudio:

```
## [1] 1.599207 3.410639 1.131478 6.567156

## [1] 0.9750021

## [1] 2 3 1 5 5

## [1] 1 2 3

## [1] 8 9 10
```

Funciones para análisis estadístico

Otras funciones

Este es el código que vamos a correr:

```
# Set seed (pin down random number generation)
set.seed(1)
# 4 random draws from N(3,5)
distribucion_normal<-rnorm(n = 4, mean = 3, sd
# CDF for N(0,1) at z=1.96
cdf<-pnorm(q = 1.96, mean = 0, sd = 1)
# Sample 5 draws from x w/ repl.
muestra<-sample(
  x = x,
  size = 5,
  replace = T
)
# First and last 3 elements of x
head_x<-head(x, 3)
tail_x<-tail(x, 3)
```

Este es el output que veremos en la consola de Rstudio:

Ahora no veo el código ¿Que pasó?

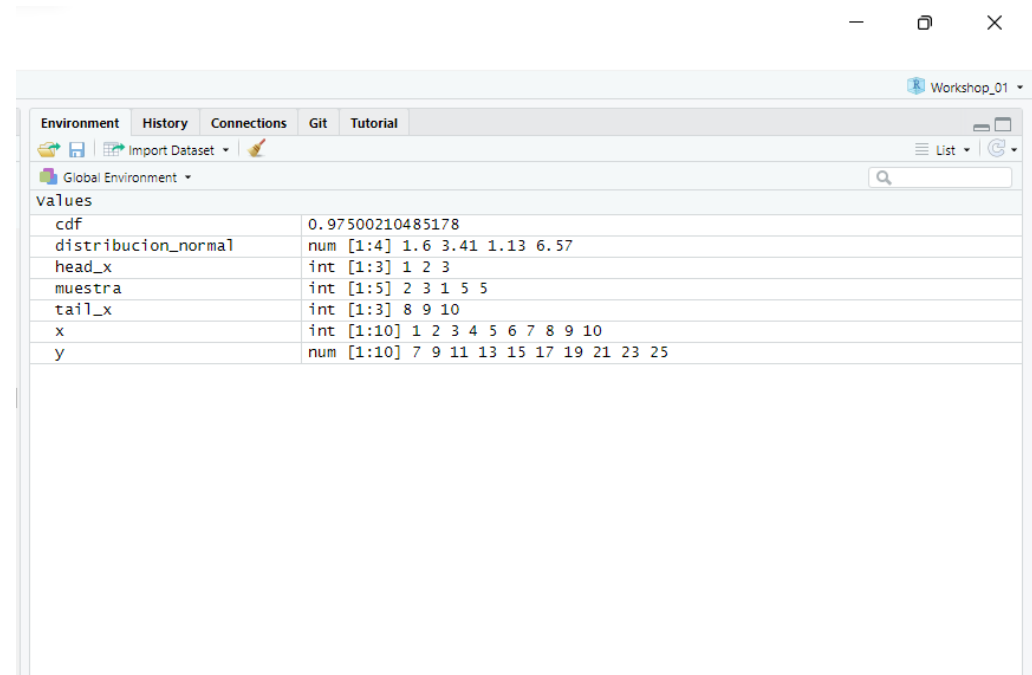
Funciones para análisis estadístico

Otras funciones

Este es el código que vamos a correr:

```
# Set seed (pin down random number generation)
set.seed(1)
# 4 random draws from N(3,5)
distribucion_normal<-rnorm(n = 4, mean = 3, sd
# CDF for N(0,1) at z=1.96
cdf<-pnorm(q = 1.96, mean = 0, sd = 1)
# Sample 5 draws from x w/ repl.
muestra<-sample(
  x = x,
  size = 5,
  replace = T
)
# First and last 3 elements of x
head_x<-head(x, 3)
tail_x<-tail(x, 3)
```

Los objetos que definimos a la derecha fueron guardados al correr el nuevo código. En el panel de estructura de datos queda constancia de ello.



The screenshot shows the R Studio 'Environment' pane for 'Workshop_01'. It lists the following objects and their values:

Object	Value
cdf	0.97500210485178
distribucion_normal	num [1:4] 1.6 3.41 1.13 6.57
head_x	int [1:3] 1 2 3
muestra	int [1:5] 2 3 1 5 5
tail_x	int [1:3] 8 9 10
x	int [1:10] 1 2 3 4 5 6 7 8 9 10
y	num [1:10] 7 9 11 13 15 17 19 21 23 25

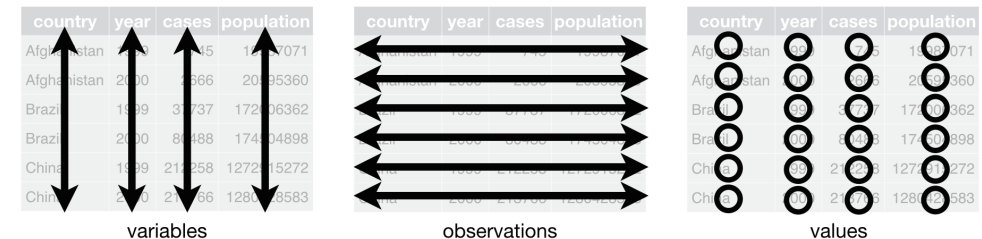
Introducción al tidyverse

(Tidy ~ Ordenado) + (verse ~ universo)

- El tidyverse es un conjunto de librerías en R basados en la misma filosofía
- Tiene su propia sintaxis y fue pensado para ser más intuitivo que las funciones "base" de R

Promueve usar datos en formato *Tidy*:

1. Cada variable tiene su propia columna
2. Cada observación tiene su propia fila
3. Cada valor tiene su propia celda

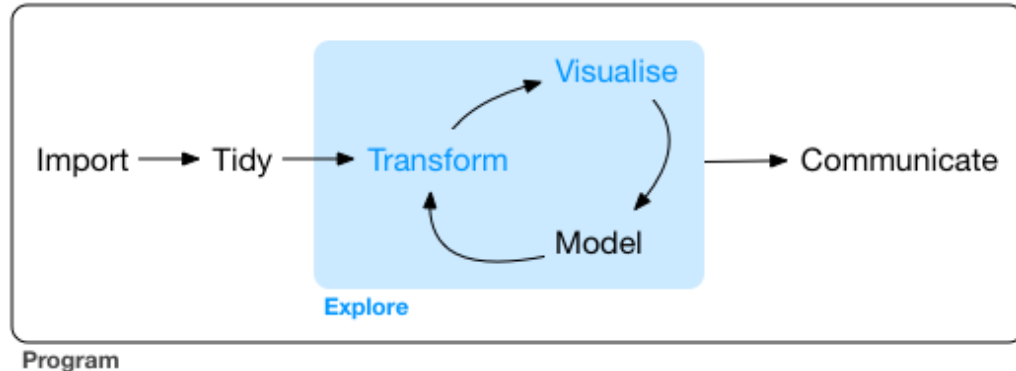


¿Qué se gana con esto?: Orden. Hay mil de maneras de tener datos desordenados, pero sólo una manera de tener datos tidy.

Introducción al tidyverse

Brinda soluciones para cada etapa del análisis de datos

Cuenta con al menos 8 librerías que usaremos a lo largo del curso.



Esta semana:

- dplyr** para manipular datos en formato tidy.
- ggplot2** para visualizarlos.

¿Tidy or not?

World development indicators (World Bank database)

Country Name	Series Name	Series Code	1960 [YR1960]	1961 [YR1961]	1962 [YR1962]
Afghanistan	GDP per capita (current US\$)	NY.GDP.PCAP.CD	59.7732337032148	59.8608999923829	58.4580086983139
Afghanistan	GDP (current US\$)	NY.GDP.MKTP.CD	537777811.111111	548888895.555556	546666677.777778
Afghanistan	Life expectancy at birth, total (years)	SP.DYN.LE00.IN	32.446	32.962	33.471
Afghanistan	Population, total	SP.POP.TOTL	8996967	9169406	9351442

Opiniones: ¿es tidy? ¿por qué?

¿Tidy or not?

World development indicators, version tidy

year	continent_name	country_name	gdp_pc	life_exp	population
1960	Asia	Afghanistan	59.77323	32.446	8996967
1961	Asia	Afghanistan	59.86090	32.962	9169406
1962	Asia	Afghanistan	58.45801	33.471	9351442
1963	Asia	Afghanistan	78.70643	33.971	9543200
1964	Asia	Afghanistan	82.09531	34.463	9744772
1965	Asia	Afghanistan	101.10833	34.948	9956318
1966	Asia	Afghanistan	137.59430	35.430	10174840
1967	Asia	Afghanistan	160.89843	35.914	10399936

Librería dplyr

Esta librería sirve para manipular de datos en formato tidy (variables en columnas, observaciones en filas, un valor por celda).

Para instalarlo en tu equipo:

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

Para cargar todas las funciones a en tu sesión:

```
library(dplyr)
```

Usa "::" para acceder a todas las funciones de la librería

```
dplyr::
```

Librería dplyr

Esta librería sirve para manipular de datos en formato tidy (variables en columnas, observaciones en filas, un valor por celda).

Funciones principales:

- `filter()`: Devuelve los **registros** que cumplen ciertas condiciones.
- `select()`: Selecciona, ordena y cambia el nombre de las **variables**.
- `arrange()`: Ordena los registros según ciertas variables.
- `mutate()`: Crea o transforma variables.
- `summarise()`: Collapsa todos los registros individuales en uno solo.

Ejemplos:

```
filter(data=datos, continent=="Europe")
```

```
select(data=datos, year,pais=country,poblacion=pop)
```

```
arrange(data=datos, country,year)
```

```
mutate(datos, gdp=pop*gdpPercap)
```

```
summarise(data=filter(data=datos, year==2007),  
           lifeExp=mean(lifeExp))
```

El operador puente (%>%)

Es una *función operacional especial* incluida en dplyr. Hace el código mas legible y eficiente. **Short-cut: Ctrl+Shift+M**

Conecta **cualquier objeto** a a la izquierda con el primer argumento de **cualquier función** a la derecha.

Ejemplos originales:

```
filter(data=datos, continent=="Europe")
```

```
select(data=datos, year, pais=country, pob=pop)
```

```
arrange(data=datos, country, year)
```

```
mutate(datos, gdp=pop*gdpPercap)
```

```
summarise(data=filter(data=datos, year==2007),  
           lifeExp=mean(lifeExp))
```

Ejemplos con %>% :

```
datos %>% filter(continent=="Europe")
```

```
datos %>% select(year, pais=country, poblacion=pop)
```

```
datos %>% arrange(country, year)
```

```
datos %>% mutate(gdp=pop*gdpPercap)
```

```
datos %>% filter(year==2007) %>% summarise(lifeExp=mean(lifeExp))
```

dplyr en acción

Usa `select()` para darle orden a tus datos

Imagina que te mandan a trabajar con este dataset

nombre_terrible.1	value	nombre.peor.2
A	-0.01	I
A	2.40	I
B	0.76	II
B	-0.80	II

Reordena las columnas

```
datos_terribles %>%
  select(nombre_terrible.1, nombre.peor.2,value)
```

```
##   nombre_terrible.1 nombre.peor.2 value
## 1                A              I -0.01
## 2                A              I  2.40
## 3                B             II  0.76
## 4                B             II -0.80
```

dplyr en acción

Usa `select()` para darle orden a tus datos

Imagina que te mandan a trabajar con este dataset

nombre_terrible.1	value	nombre.peor.2
A	-0.01	I
A	2.40	I
B	0.76	II
B	-0.80	II

Reordena, excluye y cambia los nombres:

```
datos_terribles %>%
  select(categoria=nombre_terrible.1,
         valor=value)
```

```
##   categoria valor
## 1         A -0.01
## 2         A  2.40
## 3         B  0.76
## 4         B -0.80
```

dplyr en acción

Usa **select()** para darle orden a tus datos

Imagina que te mandan a trabajar con este dataset

nombre_terrible.1	value	nombre.peor.2
A	-0.01	I
A	2.40	I
B	0.76	II
B	-0.80	II

Selecciona variables según su formato

```
datos_terribles %>%
  select(where(is.numeric))
```

```
##   value
## 1 -0.01
## 2  2.40
## 3  0.76
## 4 -0.80
```

dplyr en acción

Usa **select()** para darle orden a tus datos

Imagina que te mandan a trabajar con este dataset

nombre_terrible.1	value	nombre.peor.2
A	-0.01	I
A	2.40	I
B	0.76	II
B	-0.80	II

Selecciona variables según su nombre

```
datos_terribles %>%
  select(starts_with("nombre"))
```

```
##   nombre_terrible.1 nombre.peor.2
## 1                A              I
## 2                A              I
## 3                B             II
## 4                B             II
```

dplyr en acción

Restringe tu analisis a grupos específicos.

filter()

```
WDI_long %>%
  ## only years where data is available
  filter(year==2018) %>%
  ## only North American countries
  filter(continent_name=="South America" ) %>%
  ## only year, country, and per capita gdp
  select(year,country_name,gdp_pc)
```

```
## # A tibble: 12 x 3
##   year country_name gdp_pc
##   <dbl> <chr>      <dbl>
## 1  2018 Argentina    11633.
## 2  2018 Bolivia      3549.
## 3  2018 Brazil       9151.
## 4  2018 Chile       15888.
## 5  2018 Colombia     6730.
## 6  2018 Ecuador      6296.
## 7  2018 Guyana       6146.
## 8  2018 Paraguay     5783.
## 9  2018 Peru         6958.
## 10 2018 Suriname     6938.
## 11 2018 Uruguay     18704.
## 12 2018 Venezuela, RB      NA
```


dplyr en acción

Crea nuevas variables

mutate()

```
WDI_long %>%
  ## select variables of your interest
  select(year, country_name, gdp_pc, population)
  ## estimate total GDP by country (in billion)
  mutate(gdp_bn = round(population * gdp_pc / (10^9)))
  head()
```

```
## # A tibble: 6 x 5
##   year country_name gdp_pc population gdp_bn
##   <dbl> <chr>      <dbl>      <dbl> <dbl>
## 1  1960 Afghanistan    59.8    8996967  0.538
## 2  1961 Afghanistan    59.9    9169406  0.549
## 3  1962 Afghanistan    58.5    9351442  0.547
## 4  1963 Afghanistan    78.7    9543200  0.751
## 5  1964 Afghanistan    82.1    9744772  0.8
## 6  1965 Afghanistan   101.     9956318  1.01
```

dplyr en acción

Obten montos totales, promedios y otras medidas agregadas

summarise()

```
WDI_long %>%
  ## keep year 2018
  filter(year==2018) %>%
  ## estimate total GDP by country (in billion.
  mutate(gdp_bn=round(population*gdp_pc/(10^9)
  ## estimate world gdp and world population
  summarise(life_exp=mean(life_exp, na.rm = T)
            population=sum(population, na.rm =
            gdp_bn=sum(gdp_bn, na.rm = T))
```

```
## # A tibble: 1 x 3
##   life_exp population gdp_bn
##   <dbl>         <dbl> <dbl>
## 1      72.8 7828350268 87756.
```

dplyr en acción

Extiende tus cálculos a lo largo de diferentes grupos

group_by()

```
WDI_long %>%
  filter(year==2018) %>%
  group_by(continent_name) %>%
  summarise(count=n(),
             mean_life_exp=mean(life_exp,na.rm = T),
             sd_life_exp=sd(life_exp,na.rm = T))
```

```
## # A tibble: 6 x 4
##   continent_name count mean_life_exp sd_life_exp
##   <chr>          <int>      <dbl>      <dbl>
## 1 Africa           54        63.8        5.98
## 2 Asia             51        74.4        5.04
## 3 Europe           54        78.7        3.80
## 4 North America    34        76.0        3.86
## 5 Oceania          19        73.4        5.99
## 6 South America    12        74.9        3.13
```

dplyr en acción

Datos: Ingreso per cápita de cada país desde 1950 hasta 2018.

Desafío: Obtén un resumen de la distribución del ingreso por continente en el año más reciente ¿cuál sintaxis te parece más clara?

(a) Con funciones base de R:

```
aggregate(x = WDI_long[WDI_long$year==max(WDI_long$year)],
          list(continent = WDI_long[WDI_long$year==max(WDI_long$year)]$continent,
          FUN = function(x) c(min=min(x,na.rm = T),
                               mean=mean(x,na.rm = T),
                               max=max(x,na.rm = T))
```

(b) Con dplyr:

```
WDI_long %>%
  filter(year==max(year)) %>%
  group_by(continent_name) %>%
  summarize(min=min(gdp_pc,na.rm = T),
            mean=mean(gdp_pc,na.rm = T),
            max=max(gdp_pc,na.rm = T))
```

dplyr en acción

Tenemos el ingreso por habitante de cada país desde 1950 hasta 2018.

Desafío: Obtén un resumen de la distribución del ingreso por continente en el año mas reciente ¿cuál sintáxis te parece mas clara?

(a) Con funciones base de R:

```
## # A tibble: 6 x 2
##   continent      gdp_pc[, "min"] [, "mean"] [, "max"]
##   <chr>          <dbl>      <dbl>      <dbl>
## 1 Africa          261.      2620.     16199.
## 2 Asia            507.     15233.     86118.
## 3 Europe          3663.     33384.    190513.
## 4 North America   1272.     22471.    117098.
## 5 Oceania         1655.     13524.     55057.
## 6 South America   3552.      8520.     17688.
```

(b) Con dplyr:

```
## # A tibble: 6 x 4
##   continent_name  min    mean    max
##   <chr>          <dbl>  <dbl>  <dbl>
## 1 Africa          261.   2620.  16199.
## 2 Asia            507.  15233.  86118.
## 3 Europe          3663.  33384. 190513.
## 4 North America   1272.  22471. 117098.
## 5 Oceania         1655.  13524.  55057.
## 6 South America   3552.   8520.  17688.
```

dplyr en acción

Desafío: ¿Qué es mayor? ¿La varianza del ingreso entre continentes o la varianza lo interno de cada continente?

Replica el F-statistic del analisis de varianzas (Test ANOVA):

$$F = \frac{\frac{\sum_{j=1}^n N_j (\bar{X}_j - \bar{X})^2}{k-1}}{\frac{\sum \sum (X - \bar{X}_j)^2}{N-k}}$$

1. Calcula la variancia entre grupos ("between"):
 - 1.1 Calcula el promedio de cada grupo
 - 1.2 Calcula la variance entre las medias muestrales y la media total
2. Calcula la varianza a lo interno de cada grupo ("within")
3. Produce el F-statistic: retio entre Varbetween/Varwithin

Where:

- X: GDP per capita
- k: Number of continents or groups
- j: Continents
- N: Number of countries.

dplyr en acción

Desafío: ¿Qué es mayor? ¿La varianza del ingreso entre continentes o la varianza lo interno de cada continente?

Cálcula varianza entre continentes:

```
between_variance<-gapminder_07 %>% # Data similar
  ## Calcula la media total
  mutate(total_mean=mean(lifeExp)) %>%
  ## Calcula la media por continente con summarise
  group_by(continent) %>%
  summarise(n_countries=n(),
            group_mean=mean(lifeExp),
            total_mean=mean(total_mean)) %>%
  ungroup() %>%
  ## Calcula las diferencias between
  mutate(dif_btw=(group_mean-total_mean)) %>%
  ## calcula la varianza entre continentes
  summarise(var_btw=sum(n_countries*dif_btw^2),
```

Cálcula varianza intra-continental

```
within_variance<-gapminder_07 %>% # Data similar
  ## Calcula la media por continente con summarise
  group_by(continent) %>%
  mutate(group_mean=mean(lifeExp)) %>%
  ungroup() %>%
  ## Calcula las diferencias within continents
  mutate(dif_wtn=(lifeExp-group_mean)) %>%
  # Calcula varianzas: 1) Eleva las diferencias
  ## 2) sumalas, y divide entre tamaño de muestra
  summarise(var_wtn=sum(dif_wtn^2)/(n()-5))
```

dplyr en acción

Desafío: ¿Qué es mayor? ¿La varianza del ingreso entre continentes o la varianza lo interno de cada continente?

Veamos los resultados:

```
# Usa print() y paste() para dejar mensajes pe
message_1<-print(paste("La varianza entre cont
round(between_variance$var_btw,1))

## [1] "La varianza entre continentes es: 3265.2"
```

```
message_2<-print(paste("La varianza intra-cont
round(within_variance$var_wtn,1)))

## [1] "La varianza intra-continental es: 54.7"
```

```
message_3<-print(paste("El ratio entre ambas (l
between_variance$var_btw/within_var
```

```
## [1] "El ratio entre ambas (F-statistic) es: 59.714003730209"
```

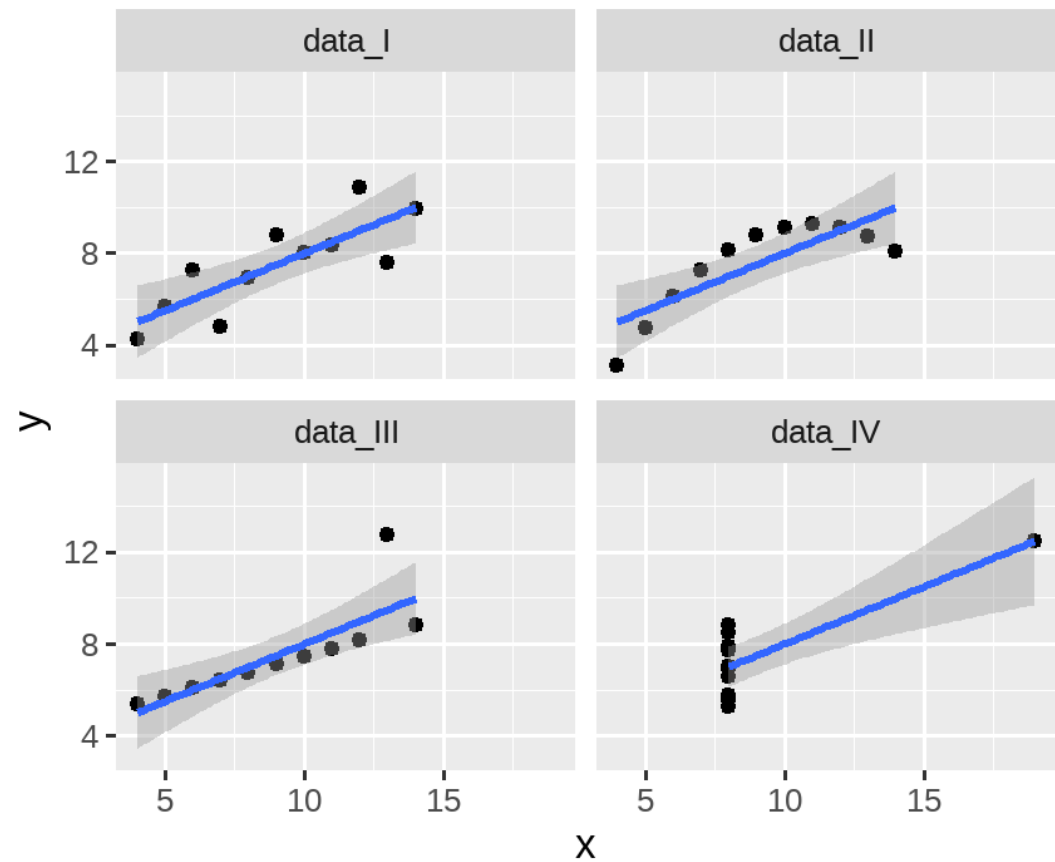
Checkeamos resultados con las funciones adecuadas

```
# Compare it with the output in the analysis o
aov.model <- aov(lifeExp ~ continent,
data = gapminder_07)

# Summary of the analysis
summary(aov.model)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## continent      4  13061    3265   59.71 <2e-16 ***
## Residuals    137   7491     55
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'
```


Una imagen vale mas que 1000 *líneas de código*



Data from: Francis Anscombe

ggplot2() and the grammar of graphics

1. Tu gráfica esta vinculada a los datos mediante coordenadas (aesthetic mappings)
2. Una vez que esas coordenadas estan definidas puedes presentar tus graficos en distintas formas (geoms), tales como puntos, lineas, barras, etc
3. Puedes agregar tantas capas como gustes a una grafica

Complete the template below to build a graph.

```

ggplot (data = <DATA>) +
<GEOM_FUNCTION> (mapping = aes(<MAPPINGS>),
stat = <STAT>, position = <POSITION>) +
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>

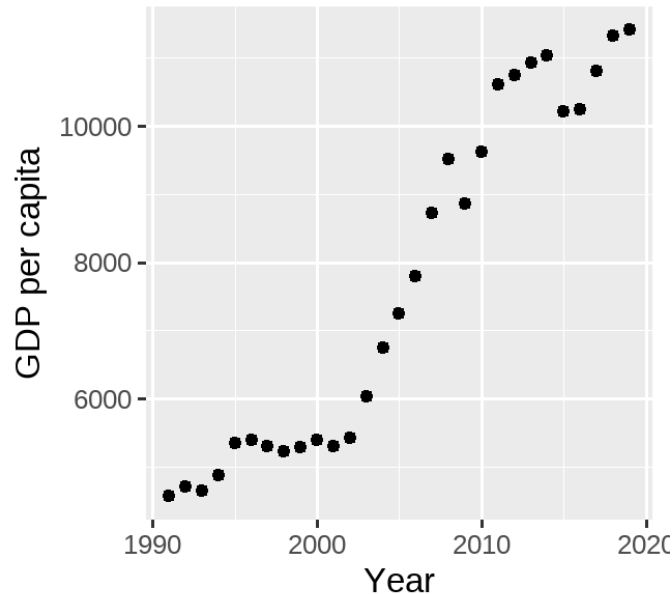
```

required

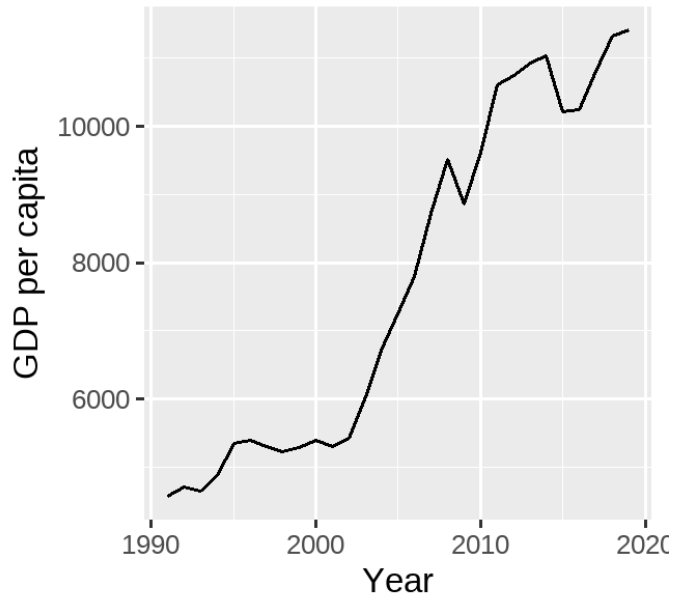
Not required, sensible defaults supplied

mapeo de coordenadas vs. y aplicacion de geoms

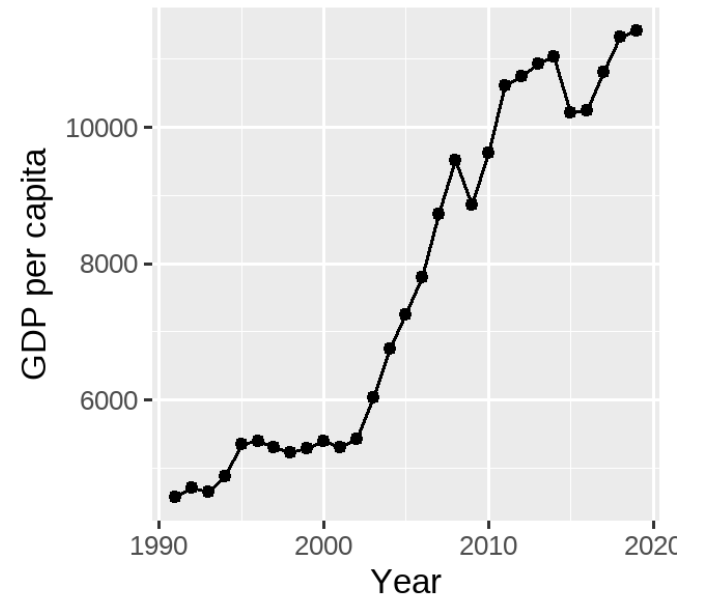
```
ggplot(data = gdp_pc_by_year,
       aes(x = year, y = gdp_pc)) +
  geom_point() +
  labs(x = "Year",
       y = "GDP per capita")
```



```
ggplot(data = gdp_pc_by_year,
       aes(x = year, y = gdp_pc)) +
  geom_line() +
  labs(x = "Year",
       y = "GDP per capita")
```

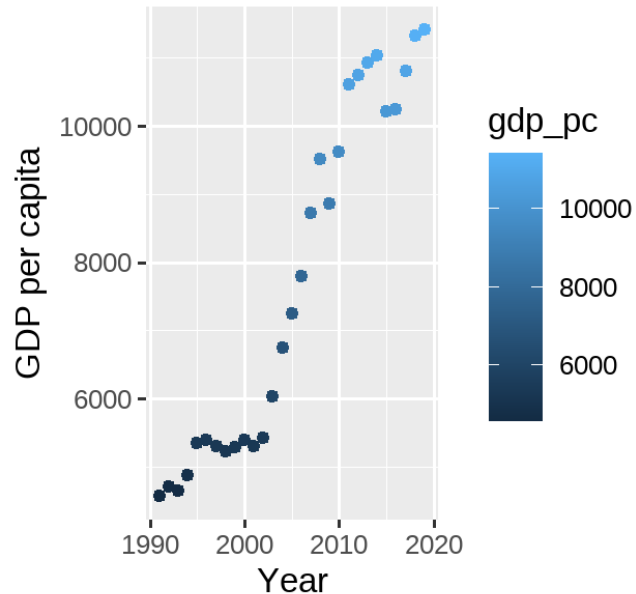


```
ggplot(data = gdp_pc_by_year,
       aes(x = year, y = gdp_pc)) +
  geom_point() +
  geom_line() +
  labs(x = "Year",
       y = "GDP per capita")
```

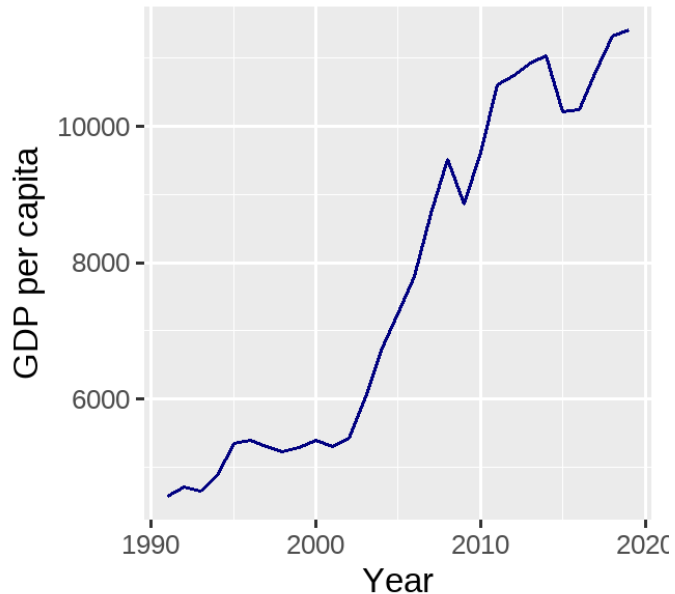


mapeo de atributos vs. hard-coded values

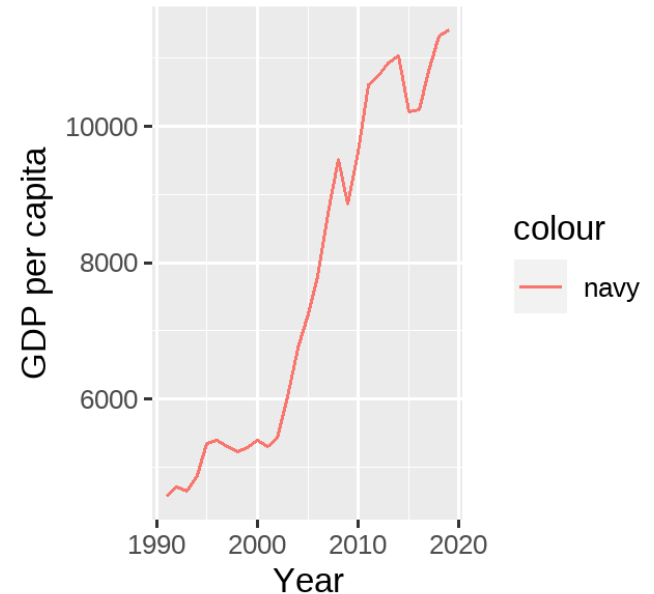
```
ggplot(data = gdp_pc_by_year,
       aes(x = year, y = gdp_pc)) +
  geom_point(aes(color=gdp_pc)) +
  labs(x = "Year",
       y = "GDP per capita")
```



```
ggplot(data = gdp_pc_by_year,
       aes(x = year, y = gdp_pc)) +
  geom_line(color="navy") +
  labs(x = "Year",
       y = "GDP per capita")
```



```
ggplot(data = gdp_pc_by_year,
       aes(x = year, y = gdp_pc)) +
  geom_line(aes(color="navy")) +
  labs(x = "Year",
       y = "GDP per capita")
```



Hablemos de la evolucion del ingreso por habitante y la esperanza de vida

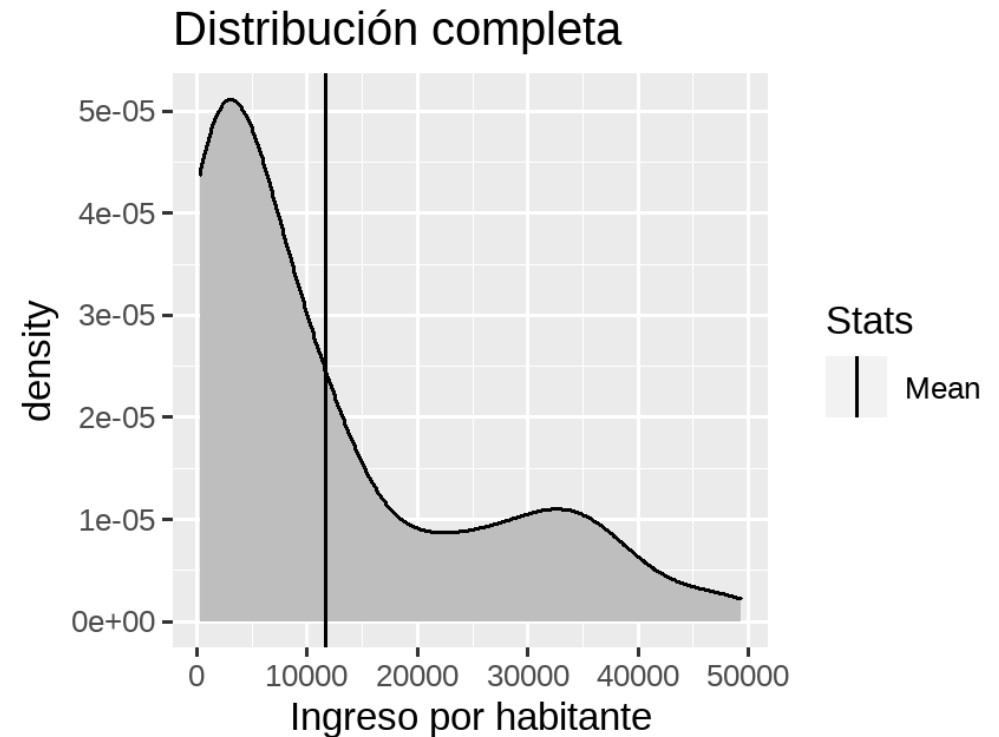
World development indicators (World Bank)

country_name	country_code	year	gdp_pc	gdp	life_exp	population	continent_name
Afghanistan	AFG	1960	59.77323	537777811	32.446	8996967	Asia
Afghanistan	AFG	1961	59.86090	548888896	32.962	9169406	Asia
Afghanistan	AFG	1962	58.45801	546666678	33.471	9351442	Asia
Afghanistan	AFG	1963	78.70643	751111191	33.971	9543200	Asia
Afghanistan	AFG	1964	82.09531	800000044	34.463	9744772	Asia
Afghanistan	AFG	1965	101.10833	1006666638	34.948	9956318	Asia

Gráficos de distribución

Distribución del ingreso por habitante

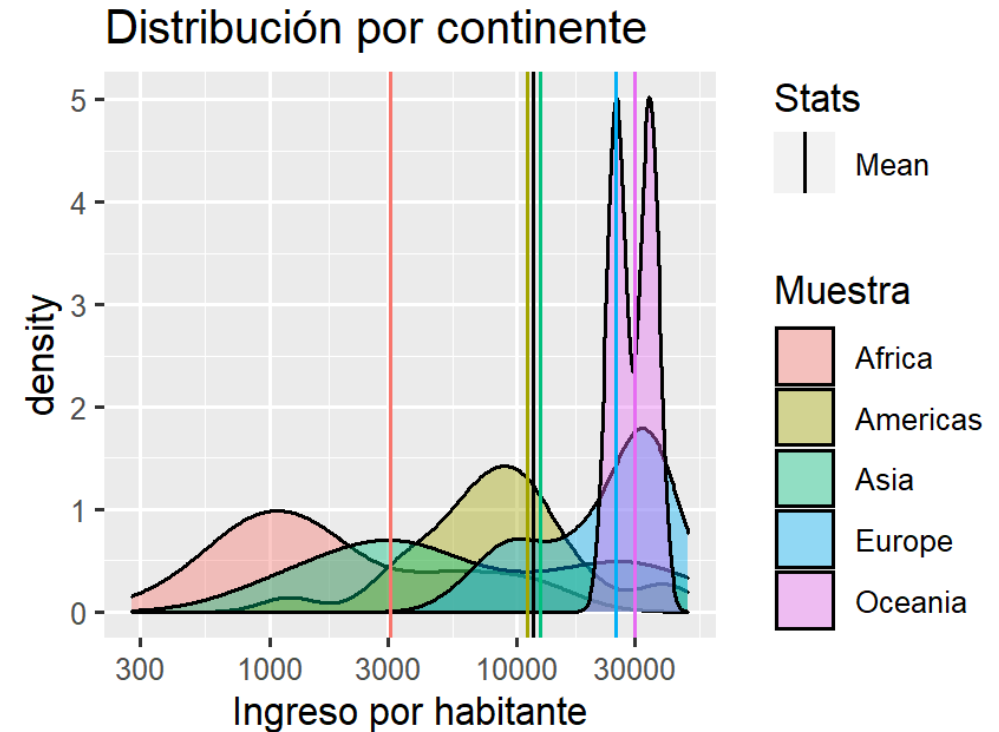
```
# Total
ggplot(data=gapminder_07, aes(x=gdpPercap))+
  # Geom de distribuciones de densidad. Ponemo.
  geom_density(fill="gray")+
  # Geom de lineas verticales. Requieren el valo
  # Usamos la palabra "Mean" en punto de corte
  # linea muestre esa palabra (not correct, bu
  geom_vline(aes(xintercept = mean(gdpPercap),
  labs(title="Distribución completa",
        linetype="Stats",
        x="Ingreso por habitante")
```



Gráficos de distribución

Distribución del ingreso por habitante, por continente

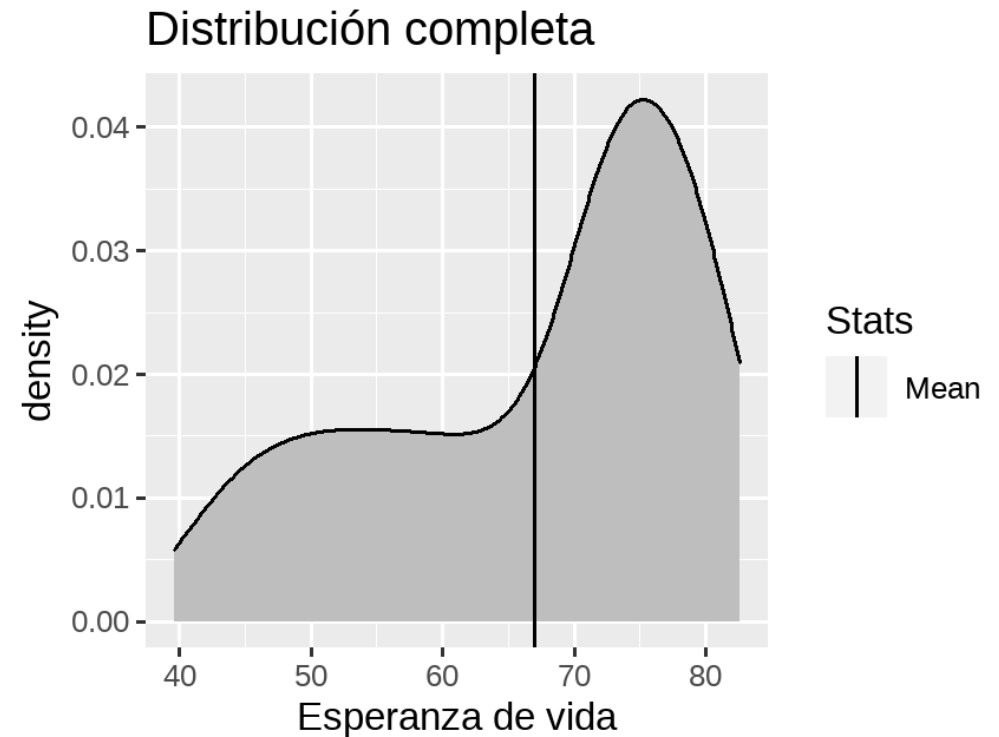
```
# Por continente
ggplot(data= gapminder_07, aes(x=gdpPercap)) +
  # Geom de distribucion de densidades, especi
  geom_density(aes(fill=continent), alpha=0.4)
  # Geom de lineas verticales
  geom_vline(aes(xintercept = mean(gdpPercap),
  # Geom de lineas verticales por continente.
  geom_vline(data= group_by(gapminder_07, contin
    summarise(gdpPercap=mean(gdpPercap)),
    aes(xintercept = gdpPercap,color=continent),
    show.legend = F)+
  scale_x_log10()+
  labs(title="Distribución por continente",
    fill="Muestra",
    linetype="Stats",
    x="Ingreso por habitante")
```



Gráficos de distribución

Distribución de la esperanza de vida en el mundo

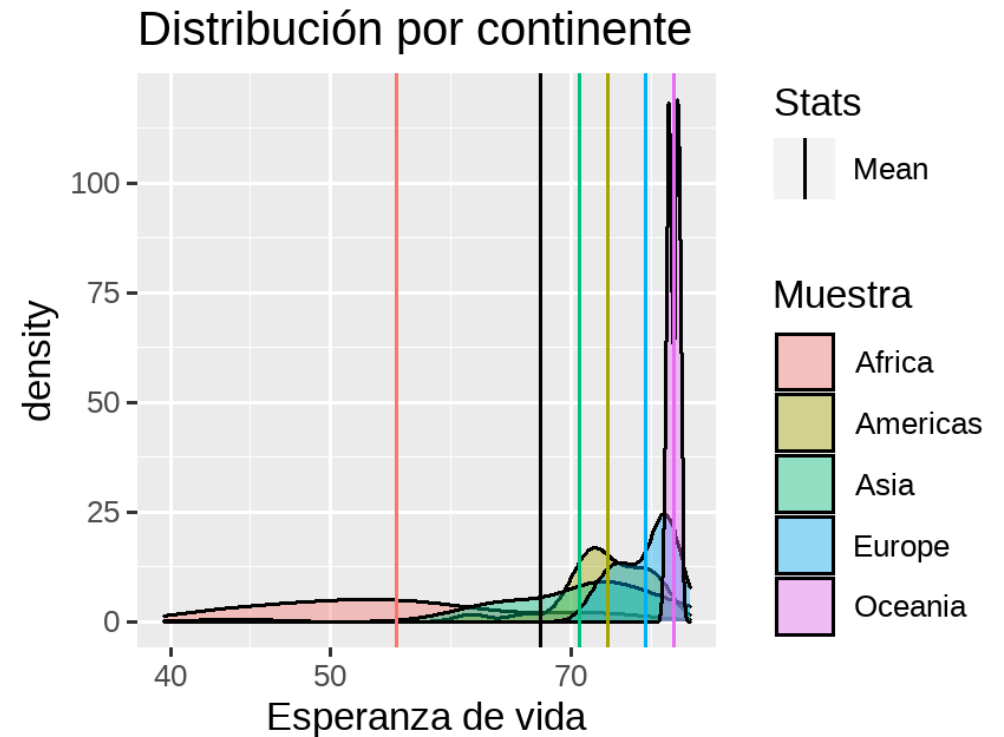
```
gapminder_07 %>%
  ggplot(aes(x=lifeExp))+
  # Geom de distribuciones de densidad. Ponemo.
  geom_density(fill="gray")+
  # Geom de lineas verticales. Requieren el valo
  # Usamos la palabra "Mean" en punto de corte
  # linea muestre esa palabra (not correct, bu
  geom_vline(aes(xintercept = mean(lifeExp), lin
  labs(title="Distribución completa",
        linetype="Stats",
        x="Esperanza de vida")
```



Gráficos de distribución

Distribución de la esperanza de vida, por continente

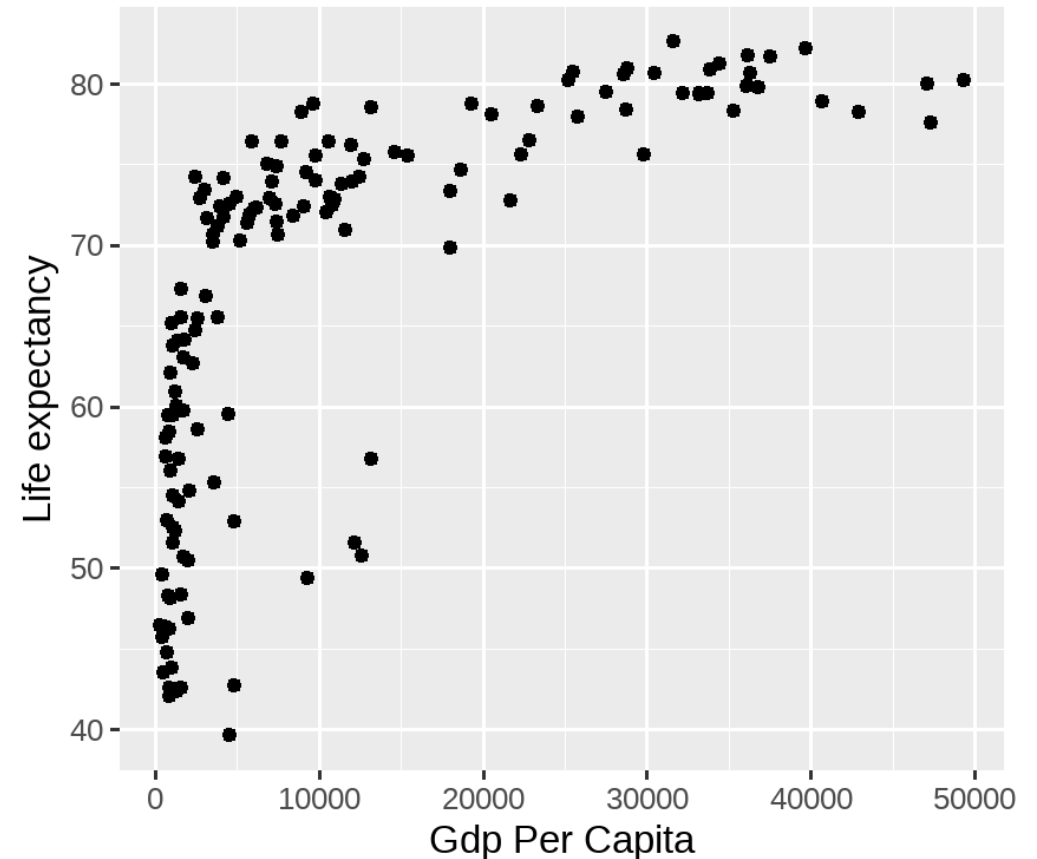
```
gapminder_07 %>%
  ggplot(aes(x=lifeExp))+
  # Geom de distribucion de densidades, especi
  geom_density(aes(fill=continent), alpha=0.4)
  # Geom de lineas verticales
  geom_vline(aes(xintercept = mean(lifeExp), lin
  # Geom de lineas verticales por contienente
  # Insertamos datos agregados a nivel contine
  # lineas
  geom_vline(data= gapminder_07 %>%
              group_by(continent) %>%
              summarise(lifeExp=mean(lifeExp))
              aes(xintercept = lifeExp,color=con
              show.legend = F))+
  scale_x_log10()+
  labs(title="Distribución por continente",
       fill="Muestra",
       linetype="Stats",
       x="Esperanza de vida")
```



Relación entre dos variables continuas

Define la data, las coordenadas, y la forma

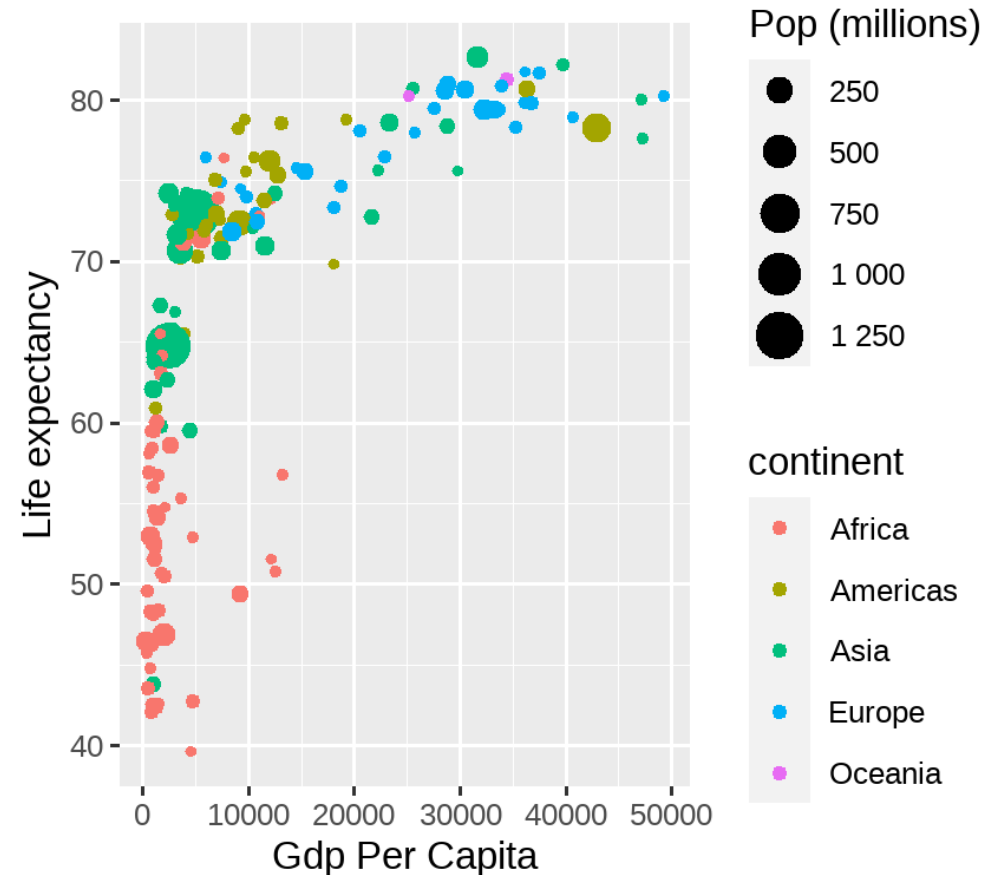
```
# Datos y coordenadas
ggplot(data=gapminder_07,
       mapping = aes(y=lifeExp,x=gdpPercap))+
# Formas o geometrias
geom_point()+
labs(x="Gdp Per Capita",
     y="Life expectancy")
```



Relación entre dos variables continuas

Añade otras formas y haz cambios en el formato

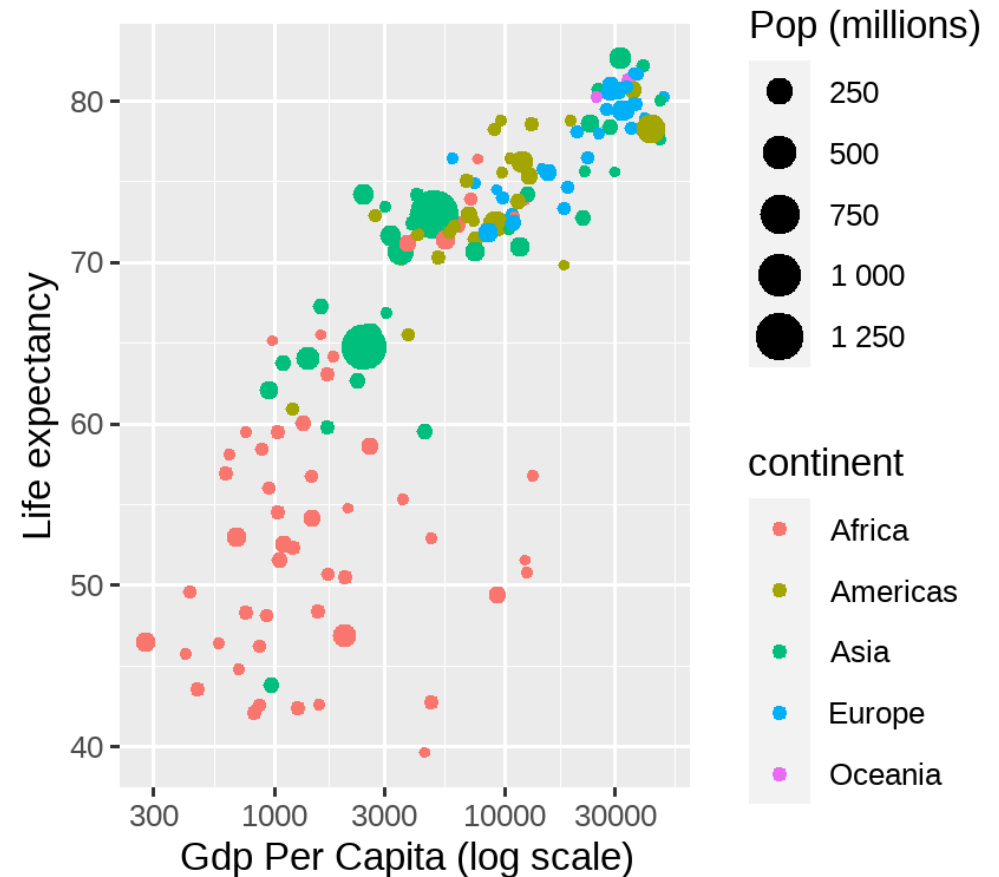
```
ggplot(data=gapminder_07,
       mapping = aes(y=lifeExp,x=gdpPercap)) +
  # coordenadas para una geometria especifica
  geom_point(aes(size=pop/1000000, color=continent)) +
  scale_size_continuous(labels=scales::number_abbrev)
labs(x="Gdp Per Capita",
     y="Life expectancy",
     size="Pop (millions)")
```



Relación entre dos variables continuas

Cambiamos la escala de gdp per capita ¿Qué ganamos con logs?

```
ggplot(data=gapminder_07,
       mapping = aes(y=lifeExp,x=gdpPercap))+
  # coordenadas para una geometria especifica
  geom_point(aes(size=pop/1000000, color=continent)) +
  scale_size_continuous(labels=scales::number_abbrev) +
  scale_x_log10()+
  labs(x="Gdp Per Capita (log scale)",
       y="Life expectancy",
       size="Pop (millions)")
```



Para finalizar: *Materiales complementarios*

Fuentes recomendadas para seguir aprendiendo:

- [R for Economists video series](#) (by Nick Huntington-Klein)
- [R for Data Science](#) (Wickham & Grolemund, 2017)
- [Statistical Inference via Data Science](#) (Ismay & Kim, 2022)
- [Top 50 ggplot2 Visualizations - The Master List](#) (by Selva Prabhakaran).
- [Statistics without the agonizing pain](#) (by John Rauser)

Próxima semana

- Repasen sus apuntes de econometría 1 y 2. Discutiremos el método de MCO.
- Descarguen R y Rstudio. La clase de código será más interactiva.
- En modulo 7 podran encontrar el código base del análisis de hoy. Juenguen con el y guardenlo de referencia para las asignaciones que vienen.
- Instalen las siguientes librerías:

```
# Librerías
librerías <- c("tidyverse", "readxl", "haven",
              "lubridate", "stringr",
              "broom", "modelr", "pml",
              "palmerpenguins",
              "knitr", "rmarkdown")

# Instalar librerías aun no instaladas
installed_packages <- librerías %in% rownames(installed.packages())
if (any(installed_packages == FALSE)) {
  install.packages(librerías[!installed_packages])
}
```

Fin de primer taller

Gracias

Datos para practicar: Restaurant inspections (causaldata::)

```
knitr::kable(head(causaldata::restaurant_inspections), format = 'html')
```

business_name	inspection_score	Year	NumberofLocations	Weekend
MCGINLEYS PUB	94	2017	9	FALSE
VILLAGE INN #1	86	2015	66	FALSE
RONNIE SUSHI 2	80	2016	79	FALSE
FRED MEYER - RETAIL FISH	96	2003	86	FALSE
PHO GRILL	83	2017	53	FALSE
TACO KING #2	95	2008	89	FALSE

Datos para practicar: Titanic (causaldta::)

```
library(haven)

read_data <- function(df)
{
  full_path <- paste("https://raw.githubusercontent.com/scunning1975/mixtape/master/",
                     df, sep = "")
  df <- read_dta(full_path)
  return(df)
}
# knitr::kable(head(read_data("titanic.dta")), format = 'html')
```

Datos para practicar: NYLS

```
library(modelr)

knitr::kable(head(modelr::heights), format = 'html')
```

income	height	weight	age	marital	sex	education	afqt
19000	60	155	53	married	female	13	6.841
35000	70	156	51	married	female	10	49.444
105000	65	195	52	married	male	16	99.393
40000	63	197	54	married	female	14	44.022
75000	66	190	49	married	male	14	59.683
102000	68	200	49	divorced	female	18	98.798

Datos para practicar: NYLS (Kennedy School)

```
library(readr)

knitr::kable(head(read_csv("data/nyls79.csv")), format = 'html')
```

id	sex	poor	education
2	2	0	12
3	2	0	12
6	1	0	16
7	1	0	12
9	1	0	14
11	1	0	16