Estadística descriptiva y reodificación de variables

Programación para el análisis de datos

Departamento de Ciencias Sociales, UCU - Martín Opertti

Estadística descriptiva

Medidas de tendencia central

```
glimpse(data_gapminder)
```

```
## Rows: 1,704
## Columns: 6
## $ country   <fct> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", ~
## $ continent <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, ~
## $ year      <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1997, ~
## $ lifeExp   <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.854, 40.8~
## $ pop       <int> 8425333, 9240934, 10267083, 11537966, 13079460, 14880372, 12~
## $ gdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 786.1134, ~
```

```
mean(data_gapminder$lifeExp) # Media
```

```
## [1] 59.47444
```

```
median(data_gapminder$lifeExp) # Mediana
```

```
## [1] 60.7125
```

```
sd(data_gapminder$lifeExp) # Desvío estandar
```

```
## [1] 12.91711
```

Rangos

```
range(data_gapminder$lifeExp) # Rango
```

```
## [1] 23.599 82.603
```

```
max(data_gapminder$lifeExp)
```

```
## [1] 82.603
```

```
min(data_gapminder$lifeExp)
```

```
## [1] 23.599
```

Histogramas

También podemos graficar los datos rápidamente. Por ejemplo, un histograma:

```
hist(data_gapminder$lifeExp,
     main = "Distribución de expectativa de vida (Gapminder)")
```

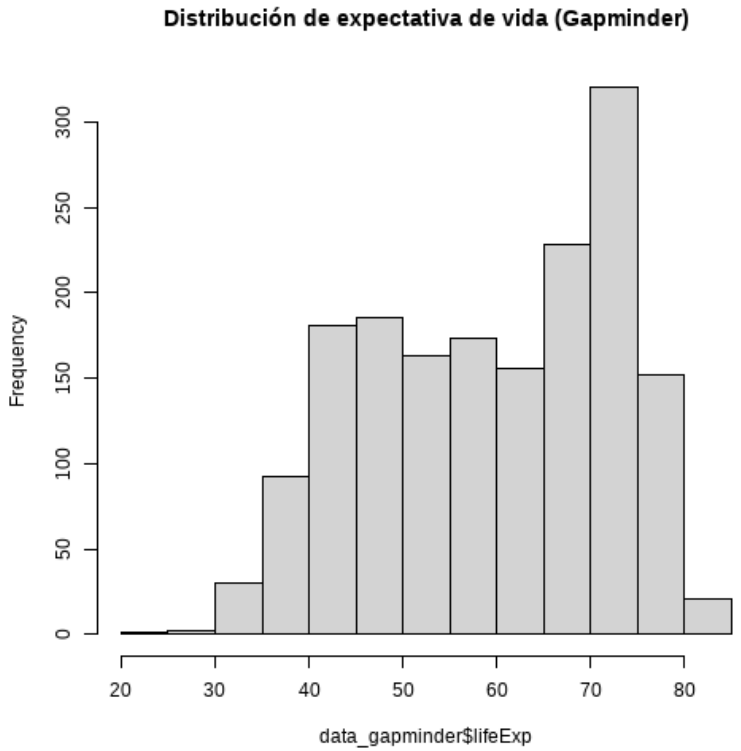
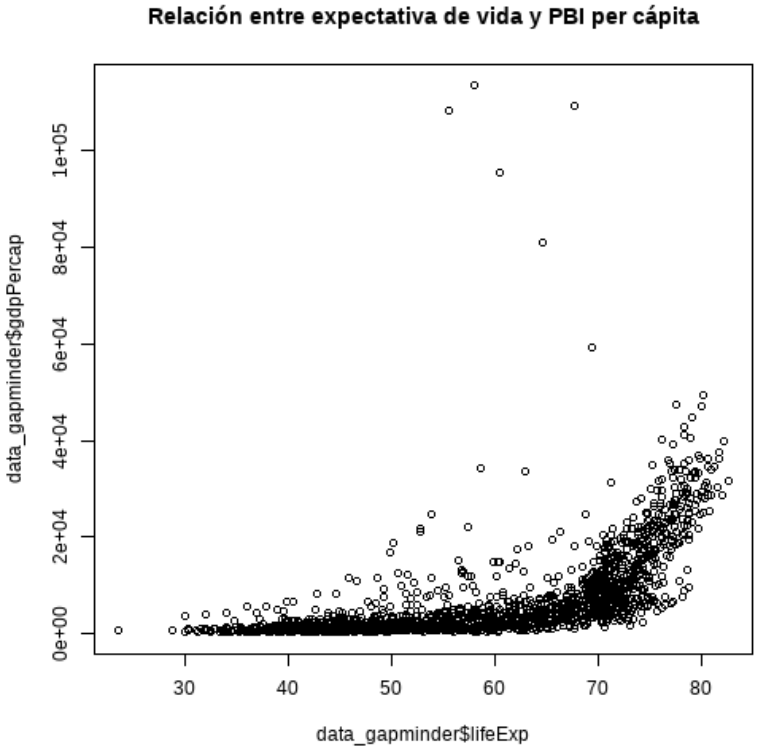


Gráfico de dispersión (scatterplot)

```
plot(data_gapminder$lifeExp, data_gapminder$gdpPercap,
main = "Relación entre expectativa de vida y PBI per cápita")
```



Cuantiles

```
quantile(data_gapminder$lifeExp, probs=c(0.2, 0.4, 0.8)) # Cuantiles
```

```
##     20%     40%     80%
## 45.8992 55.7292 72.0288
```

```
quantile(data_gapminder$lifeExp, probs=seq(0, 1, 0.2)) # Cuantiles
```

```
##      0%     20%     40%     60%     80%    100%
## 23.5990 45.8992 55.7292 66.0814 72.0288 82.6030
```

```
# Con la función ntile() de dplyr podemos asignar quintiles en una variable
data_gapminder$lifeExp_quant <- ntile(data_gapminder$lifeExp, 5)

# Tabla cruzada
table(data_gapminder$continent, data_gapminder$lifeExp_quant)
```

```
##
##                1   2   3   4   5
##   Africa     261 231  99  23  10
##   Americas    15  40  86  90  69
##   Asia        64  65 125  81  61
##   Europe       1   5  31 137 186
##   Oceania      0   0   0  10  14
```

Ejercicio

(1) Importar la base wb_paises

(2) Calcular la media, mediana, desvío estandar y moda de la variable aceso_electrico

(3) Crear un gráfico de dispersión de las variables de PBI per capita y CO2 per capita

Crear y recodificar variables

Crear variables con mutate()

El paquete dplyr contiene la función `mutate()` para crear nuevas variables. `mutate()` crea variables al final del dataframe.

```
data_gapminder <- as_tibble(gapminder) # Pasamos nuevamente a tibble

# Variable de caracteres
data_gapminder <- mutate(data_gapminder, var1 = "Valor fijo")

# Variable numérica
data_gapminder <- mutate(data_gapminder, var2 = 7)

head(data_gapminder, 3)
```

```
## # A tibble: 3 x 8
##   country     continent  year lifeExp      pop gdpPercap var1        var2
##   <fct>       <fct>     <int>   <dbl>    <int>     <dbl> <chr>      <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779. Valor fijo     7
## 2 Afghanistan Asia       1957    30.3  9240934      821. Valor fijo     7
## 3 Afghanistan Asia       1962    32.0 10267083      853. Valor fijo     7
```

```
## Podemos escribir lo mismo de distinta manera:
data_gapminder <- mutate(data_gapminder, var1 = "Valor fijo",
                                         var2 = 7)
```

Crear variables en R Base

Estas transformaciones también las podríamos haber hecho en R Base

```
d_gap <- gapminder

d_gap$var1 <- "Valor fijo"

d_gap$var2 <- 7

head(d_gap, 3)
```

```
## # A tibble: 3 x 8
##   country     continent  year lifeExp      pop gdpPercap var1        var2
##   <fct>       <fct>     <int>   <dbl>    <int>     <dbl> <chr>      <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779. Valor fijo     7
## 2 Afghanistan Asia       1957    30.3  9240934      821. Valor fijo     7
## 3 Afghanistan Asia       1962    32.0 10267083      853. Valor fijo     7
```

Recodificar una misma variable

```
## También tenemos dos maneras de recodificar una misma variable
# Con dplyr
data_uru <- filter(gapminder, country == "Uruguay")
data_uru <- mutate(data_uru, country =  "ROU")
head(data_uru, 3)
```

```
## # A tibble: 3 x 6
##   country continent  year lifeExp     pop gdpPercap
##   <chr>   <fct>     <int>   <dbl>   <int>     <dbl>
## 1 ROU     Americas   1952    66.1 2252965     5717.
## 2 ROU     Americas   1957    67.0 2424959     6151.
## 3 ROU     Americas   1962    68.3 2598466     5603.
```

```
# En R Base
data_uru <- filter(gapminder, country == "Uruguay")
data_uru$country <- "ROU"
head(data_uru, 3)
```

```
## # A tibble: 3 x 6
##   country continent  year lifeExp     pop gdpPercap
##   <chr>   <fct>     <int>   <dbl>   <int>     <dbl>
## 1 ROU     Americas   1952    66.1 2252965     5717.
## 2 ROU     Americas   1957    67.0 2424959     6151.
## 3 ROU     Americas   1962    68.3 2598466     5603.
```

Recodificar variables con mutate()

Con `mutate()` también podemos realizar operaciones sobre variables ya existentes:

```
## Podemos recodificar usando variables y operadores aritméticos
# Calculemos el pbi total (pbi per capita * población)
d_gap <- mutate(gapminder, gdp = gdpPercap * pop)
head(d_gap, 3)
```

```
## # A tibble: 3 x 7
##   country     continent  year lifeExp      pop gdpPercap         gdp
##   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>       <dbl>
## 1 Afghanistan Asia       1952    28.8  8425333      779. 6567086330.
## 2 Afghanistan Asia       1957    30.3  9240934      821. 7585448670.
## 3 Afghanistan Asia       1962    32.0 10267083      853. 8758855797.
```

```
# Podemos calcular el logaritmo
d_gap <- mutate(d_gap, gdp_log = log(gdp))
head(d_gap, 2)
```

```
## # A tibble: 2 x 8
##   country     continent  year lifeExp     pop gdpPercap         gdp gdp_log
##   <fct>       <fct>     <int>   <dbl>   <int>     <dbl>       <dbl>   <dbl>
## 1 Afghanistan Asia       1952    28.8 8425333      779. 6567086330.    22.6
## 2 Afghanistan Asia       1957    30.3 9240934      821. 7585448670.    22.7
```

Adelantar y retrasar variables

```
## Podemos retrasar -lag()- o adelantar -lead()- variables
# Primero nos quedamos con los datos de Uruguay
# Atrasamos un período el pbi per capita
data_uru <- filter(gapminder, country == "Uruguay")
data_uru <- mutate(data_uru, gdpPercap_lag = lag(gdpPercap, n=1))
head(data_uru, 4)
```

```
## # A tibble: 4 x 7
##   country continent  year lifeExp     pop gdpPercap gdpPercap_lag
##   <fct>   <fct>     <int>   <dbl>   <int>     <dbl>          <dbl>
## 1 Uruguay Americas   1952    66.1 2252965     5717.           NA
## 2 Uruguay Americas   1957    67.0 2424959     6151.         5717.
## 3 Uruguay Americas   1962    68.3 2598466     5603.         6151.
## 4 Uruguay Americas   1967    68.5 2748579     5445.         5603.
```

```
# Adelantamos dos períodos el pbi per cápita
data_uru <- mutate(data_uru, gdpPercap_lead2 = lead(gdpPercap, n=2))
head(data_uru, 4)
```

```
## # A tibble: 4 x 8
##   country continent  year lifeExp    pop gdpPercap gdpPercap_lag gdpPercap_lead2
##   <fct>   <fct>     <int>   <dbl>  <int>     <dbl>          <dbl>           <dbl>
## 1 Uruguay Americas   1952    66.1 2.25e6     5717.           NA             5603.
## 2 Uruguay Americas   1957    67.0 2.42e6     6151.         5717.            5445.
## 3 Uruguay Americas   1962    68.3 2.60e6     5603.         6151.            5703.
## 4 Uruguay Americas   1967    68.5 2.75e6     5445.         5603.            6504.
```

Rankings e identificadores

```
 # Identificador (números consecutivos)
d_gap <- mutate(gapminder, id = row_number())
head(d_gap, 4)
```

```
## # A tibble: 4 x 7
##   country     continent  year lifeExp      pop gdpPercap    id
##   <fct>       <fct>     <int>   <dbl>    <int>     <dbl> <int>
## 1 Afghanistan Asia       1952    28.8  8425333      779.     1
## 2 Afghanistan Asia       1957    30.3  9240934      821.     2
## 3 Afghanistan Asia       1962    32.0 10267083      853.     3
## 4 Afghanistan Asia       1967    34.0 11537966      836.     4
```

```
# Ranking según variable
d_gap <- mutate(d_gap, gdp_rank = row_number(gdpPercap))

# Ordeno los datos según el ranking
d_gap <- arrange(d_gap, desc(gdp_rank))
head(d_gap, 4)
```

```
## # A tibble: 4 x 8
##   country continent  year lifeExp    pop gdpPercap    id gdp_rank
##   <fct>   <fct>     <int>   <dbl>  <int>     <dbl> <int>    <int>
## 1 Kuwait  Asia       1957    58.0 212846   113523.   854     1704
## 2 Kuwait  Asia       1972    67.7 841934   109348.   857     1703
## 3 Kuwait  Asia       1952    55.6 160000   108382.   853     1702
## 4 Kuwait  Asia       1962    60.5 358266    95458.   855     1701
```

Recodificaciones condicionales

Recodificaciones condicionales

- Muchas veces transformar los datos implica recodificar una variable de forma condicional, esto es, asignar distintos valores en función de los valores de una o más variables.
- Para esto se pueden utilizar las funciones mutate(), recode(), case_when() (Tidyverse) y ifelse() (R Base)

Recodificación condicional con case_when() y mutate()

Cuando trabajmos con dataframes `case_when()` debemos trabajar también con `mutate()` (la primera se usa dentro de la segunda). `case_when()` testea condiciones en orden (esto es importante cuando pasamos condiciones no excluyentes). `case_when()` lista condiciones para las que asigna un valor en caso de que sean verdaderas, y permite pasar múltiples condiciones. `TRUE` refiere a las condiciones no listadas. La estructura de `case_when()` es:

```
mutate(data,
       var_nueva = case_when(var_original == "Valor 1" ~ "Valor A",
                             var_original == "Valor 2" ~ "Valor B",
                             TRUE ~ "Otros"))
```

Recodificación condicional con case_when() y mutate()

```
# Creemos una variable que indique si el país es Uruguay o no
d_gap <- mutate(d_gap, uruono = case_when(
  country == "Uruguay" ~ "Si",
  TRUE ~ "No")
)

table(d_gap$uruono)
```

```
##
##   No   Si
## 1692   12
```

Recodificación condicional con case_when() y mutate()

Podemos establecer varias condiciones fácilmente:

```
d_gap <- gapminder

d_gap <- mutate(d_gap, mercosur = case_when(country == "Uruguay" ~ 1,
                                            country == "Argentina" ~ 1,
                                            country == "Paraguay" ~ 1,
                                            country == "Brazil" ~ 1,
                                            TRUE ~ 0))

table(d_gap$mercosur)
```

```
##
##    0    1
## 1656   48
```

Recodificación condicional con case_when() y mutate()

También podríamos usar operadores para simplificar esto:

```
d_gap <- mutate(d_gap, mercosur = case_when(
  country %in% c("Argentina", "Paraguay", "Brazil", "Uruguay") ~ 1,
  TRUE ~ 0)
  )

d_gap <- mutate(d_gap, mercosur2 = case_when(
  country == "Argentina" | country == "Paraguay" |
    country == "Brazil" | country == "Uruguay" ~ 1,
  TRUE ~ 0)
  )

identical(d_gap$mercosur, d_gap$mercosur2)
```

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

Recodificación condicional con case_when() y mutate()

`case_when()` sirve también para recodificar una variable con condiciones basadas en múltiples variables.

Supongamos que queremos una variable que indique los países-año con expectativa de vida mayor a 75 o pbi per cápita mayor a 20.000

```
d_gap <- mutate(d_gap,
                var1 = case_when(gdpPercap > 20000 ~ 1,
                                 lifeExp > 75 ~ 1,
                                 TRUE ~ 0))
table(d_gap$var1)
```

```
##
##    0    1
## 1493  211
```

Recodificación condicional con case_when() y mutate()

Podemos también recodificar algunos valores de una variable pero manteniendo el resto. Esto lo logramos especificando las condiciones que queremos cambiar y especificando la propia variable para el resto de los valores en el `TRUE`. Por ejemplo, supongamos que hay un error en la variable años, los datos de 2007 son en realidad de 2008. Entonces hacemos:

```
table(d_gap$year)
```

```
##
## 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 2002 2007
##  142  142  142  142  142  142  142  142  142  142  142  142
```

```
d_gap <- mutate(d_gap,
                year = case_when(year == 2007 ~ 2008,
                                 TRUE ~ year))
table(d_gap$year)
```

```
##
## 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 2002 2008
##  142  142  142  142  142  142  142  142  142  142  142  142
```

Recodificación condicional con case_when() y mutate()

Aplicando la misma lógica cambiemos solamente en la variable continente "Americas" por "Las Américas" y "Europe" por "Europa". ¿Por qué nos da error?

```
table(d_gap$continent)
```

```
##
##   Africa Americas     Asia   Europe  Oceania
##      624      300      396      360       24
```

```
d_gap <- mutate(d_gap,
                continent = case_when(continent == "Americas" ~ "Las Américas",
                                      lifeExp == "Europe" ~ "Europa",
                                      TRUE ~ continent))
```

```
## Error in `mutate()`:
## ! Problem while computing `continent = case_when(...)`.
## Caused by error in `` names(message) <- `*vtmp*` ``:
## ! 'names' attribute [1] must be the same length as the vector [0]
```

Recodificación condicional con case_when() y mutate()

El tipo de datos de las condiciones especificadas debe coincidir con la variable que usamos en el argumento del `TRUE`.

```
table(d_gap$continent)
```

```
##
##   Africa Americas     Asia   Europe  Oceania
##      624      300      396      360       24
```

```
d_gap <- mutate(d_gap,
                continent = case_when(continent == "Americas" ~ "Las Américas",
                                      lifeExp == "Europe" ~ "Europa",
                                      TRUE ~ as.character(continent)))
table(d_gap$continent)
```

```
##
##       Africa         Asia       Europe Las Américas      Oceania
##          624          396          360          300           24
```

Recodificación condicional con ifelse()

Para recodificar condicionalmente una variable en R base podemos usar la función `ifelse()`. Esta función tiene tres argumentos:

- `test`: primero establece una condición a probar
- `x`: el valor para los valores en donde `test = TRUE`
- `y`: el valor para los valores en donde `test = FALSE`

Supongamos que queremos crear una variable en la data de gapminder queremos crear una nueva variable `poburu` donde 1 represente los países con más de 3 millones de habitantes y 0 represente a los países con menos de 3 millones de habitantes

```
# Usualmente no explicitamos los argumentos, los definimos por su orden,
# condición, valor si condición es verdadera y valor si condicion es falsa
d_gap$poburu <- ifelse(d_gap$pop > 3000000, 1, 0)

table(d_gap$poburu)
```

```
##
##    0    1
##  458 1246
```

Recodificación condicional con ifelse()

Para chequear varias condiciones al mismo tiempo podemos usar `ifelse()` de forma anidada.

```
d_gap$mercosur <- ifelse(d_gap$country == "Uruguay", 1,
                         ifelse(d_gap$country == "Argentina", 1,
                                ifelse(d_gap$country == "Paraguay", 1,
                                       ifelse(d_gap$country == "Brazil", 1,
                                              0))))
table(d_gap$continent, d_gap$mercosur)
```

```
##
##                    0   1
##   Africa         624   0
##   Asia           396   0
##   Europe         360   0
##   Las Américas   252  48
##   Oceania         24   0
```

Recodificación condicional con ifelse() y mutate()

`ifelse()` y `case_when()` pueden cumplir la misma función, aunque para esta última no es necesario anidar.

```
d_gap <- gapminder

## Con ifelse()
d_gap$mercosur <- ifelse(d_gap$country == "Uruguay", 1,
                         ifelse(d_gap$country == "Argentina", 1,
                                ifelse(d_gap$country == "Paraguay", 1,
                                       ifelse(d_gap$country == "Brazil", 1,
                                              0))))

## Con case_when()
d_gap <- mutate(d_gap,
                mercosur_2 = case_when(country == "Uruguay" ~ 1,
                                       country == "Argentina" ~ 1,
                                       country == "Paraguay" ~ 1,
                                       country == "Brazil" ~ 1,
                                       TRUE ~ 0))

identical(d_gap$mercosur, d_gap$mercosur_2)
```

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

Ejercicio

(1) Importar la base wb_paises_2 (de la carpeta data)

(2) Buscar si alguna variable sea de un tipo incorrecto. En caso de que así sea, transformarla al tipo correcto

(3) Crear una nueva variable que tenga 3 valores: (1) Bajo si el índice de gini es menor a 30; (2) Medio si el índice de gini está entre 30 y 40 y (3) Alto si el índice de gini es mayor a 40

Datos perdidos

Datos perdidos en R

- Muchas veces cuando trabajamos con datos no contamos con todas las observaciones.
- Es importante identificar cuando tenemos datos perdidos porque puede afectar el funcionamiento de las funciones que le aplicamos a los datos.
- En R los observaciones perdidas se representan con el termino `NA` (sin comillas) que significa Not available
- Si importamos datos de otros programas tenemos que tener cuidado de cómo están codificados los datos perdidos porque no necesariamente siempre R puede identificar los datos perdidos.

Datos perdidos en R

```
vector_n <- c(1, 2, 3, 4, NA, 5)
mean(vector_n)
```

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

```
mean(vector_n, na.rm = TRUE)
```

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

```
vector_n2 <- c(1, 2, 3, 4, 5)
mean(vector_n2)
```

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

```
# Para chequear si cada observación es un dato perdido o no
is.na(vector_n)
```

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

Datos perdidos en R

Muchas veces funciones que utilizamos generan datos perdidos. Miremos algunos ejemplos:

```
# Operaciones con vectores y datos perdidos
vector1 <- c(1, 2, 3, 4)
vector2 <- c(1, 0, 1, NA)
vector_final <- vector1 / vector2
vector_final
```

```
## [1]   1 Inf   3  NA
```

Datos perdidos en R

Podemos combinar `is.na()` con otras funciones

```
# Por ejemplo, usando any() podemos ver si hay al menos un valor perdido
any(is.na(vector2))
```

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

```
# Con which() podemos ver cuáles valores son perdidos
which(is.na(vector2))
```

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

```
# Con mean() podemos calcular el procentaje de datos perdidos
mean(is.na(vector2))
```

```
## [1] 0.25
```

```
# Con sum() podemos calcular cuántos valores son perdidos
sum(is.na(vector2))
```

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

Datos perdidos en R

Volvamos a la encuesta. Supongamos que tenemos más datos y los queremos anexar al dataframe original

```
# Datos originales
print(encuesta)
```

```
##   edad ideologia      voto
## 1   18 Izquierda Partido A
## 2   24 Izquierda Partido A
## 3   80   Derecha Partido C
```

```
# Datos para anexar
print(encuesta_2)
```

```
##   edad ideologia      voto genero
## 1   40   Derecha Partido B  Mujer
## 2   44 Izquierda Partido A Hombre
## 3   NA   Derecha Partido C  Mujer
```

Datos perdidos en R

```
# Para anexar datos podemos utilizar la función rbind.fill() de plyr
encuesta_anexada <- plyr::rbind.fill(encuesta, encuesta_2)

# Miremos la encuesta anexada
print(encuesta_anexada)
```

```
##   edad ideologia      voto genero
## 1   18 Izquierda Partido A   <NA>
## 2   24 Izquierda Partido A   <NA>
## 3   80   Derecha Partido C   <NA>
## 4   40   Derecha Partido B  Mujer
## 5   44 Izquierda Partido A Hombre
## 6   NA   Derecha Partido C  Mujer
```

Datos perdidos en R

```
# Con complete.cases vemos que filas están completas
complete.cases(encuesta_anexada)
```

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

```
# Con na.omit nos eliminamos las observaciones con datos perdidos
na.omit(encuesta_anexada)
```

```
##   edad ideologia      voto genero
## 4   40   Derecha Partido B  Mujer
## 5   44 Izquierda Partido A Hombre
```