Sesion 7. Estructura y agrupamiento de datos en R

Curso: POL304 - Estadística para el análisis político 2

Jefes de práctica: Alexander Benites y Wendy Adrianzén

Ciclo 2022-2

Hasta el momento, hemos visto cómo traer data desde repositorios virtuales, desde portales web (con técnicas de scrapping) y diferentes funciones para limpiar inconsistencias en nuestras bases de datos. Continuando con todo lo anterior, en esta sesión repasaremos las diferentes estructuras de datos en R, enfocándonos en el formateo de fechas, horas y operaciones con ese tipo de datos.

1. Trabajando con fechas

Veamos cómo especificar la conversión en el formato según la manera en la cual ha sido sistematizada nuestra data:

Conversion specification	Description	Example
%a	Abbreviated weekday	Sun, Thu
%A	Full weekday	Sunday, Thursday
%b or %h	Abbreviated month	May, Jul
%B	Full month	May, July
%d	Day of the month 01-31	27, 07
%j	Day of the year 001-366	148, 188
%m	Month 01-12	05, 07
%U	Week 01-53 with Sunday as first day of the week	22, 27
%w	Weekday 0-6 Sunday is 0	0, 4
%W	Week 00-53 with Monday as first day of the week	21, 27
%x	Date, locale-specific	
%y	Year without century 00-99	84, 05
%Y	Year with century on input: 00 to 68 prefixed by 20 69 to 99 prefixed by 19	1984, 2005
%C	Century	19, 20
%D	Date formatted %m/%d/%y	05/27/84, 07/07/05
%u	Weekday 1-7 Monday is 1	7, 4
%n	Newline on output or Arbitrary whitespace on input	
%t	Tab on output or Arbitrary whitespace on input	

Traigamos una base de datos con información sobre la fecha de declaración de independencia de varios países en el mundo:

```
library(rvest)

linkToWebIDE="https://en.wikipedia.org/wiki/List_of_national_independence_days"
xpathToTableIDE='//*[@id="mw-content-text"]/div[1]/table[2]'
data <- read_html(linkToWebIDE)%>%html_nodes(xpath = xpathToTableIDE)%>%html_table()%>% .[[1]]

head(data)
```

```
## # A tibble: 6 x 7
##   Country             ``    `Name of holiday`   `Date of holid~` `Year of event`
##   <chr>               <lgl> <chr>               <chr>            <chr>
```

```
## 1 Afghanistan         NA    Afghan Independenc~ 19 August        1919
## 2 Albania             NA    Flag Day            28 November      1912
## 3 Algeria             NA    Independence Day    5 July           1962
## 4 Angola              NA    Independence Day    11 November      1975
## 5 Antigua and Barbuda NA    Independence Day    1 November       1981
## 6 Argentina           NA    Independence Day    9 July           1816[8]
## # ... with 2 more variables: `Independence from` <chr>,
## #   `Event commemorated and notes` <chr>
```

```
data$Date = paste(data$`Date of holiday`,data$`Year of event`)
```

Veamos cómo se encuentran las celdas donde está la información de las fechas de independencia:

```
data[1,8]
```

```
## # A tibble: 1 x 1
##   Date
##   <chr>
## 1 19 August 1919
```

Esta es la manera de adecuar el formato de fechas con lenguaje base de R. Utilizamos la función *as.Date* y especificamos el vector y el formato en el cuál se sistematizó la información. Siempre realizamos una verificación antes de guardar la información:

Ojo:

A veces cuando apliquemos as.Date, podemos obtener NA en vez de lo que buscamos, debido a la configuración de la fecha. En ese caso, dependiendo si trabajamos con fechas locales (en español) o en inglés, apliquemos los siguientes códigos:

Sys.setlocale("LC_TIME") # para fechas en español Sys.setlocale("LC_TIME", "English") # para retornar a fechas en inglés

```
as.Date(data$Date, format = "%d %B %Y")
```

```
##   [1] "1919-08-19" "1912-11-28" "1962-07-05" "1975-11-11" "1981-11-01"
##   [6] "1816-07-09" "1918-05-28" "1991-09-21" "1918-05-28" "1991-10-18"
##  [11] "1973-07-10" "1971-12-16" "1971-03-26" "1971-12-16" "1966-11-30"
##  [16] "1991-07-03" "1831-07-21" "1981-09-21" "1960-08-01" "1907-12-17"
##  [21] "1825-08-06" "1992-03-01" "1966-09-30" "1822-09-07" "1984-02-23"
##  [26] "1878-03-03" "1908-09-22" "1958-12-11" "1960-08-05" "1962-07-01"
##  [31] "1953-11-09" "1960-05-20" "1867-07-01" "1975-07-05" "1958-12-01"
##  [36] "1960-08-13" "1958-11-28" "1960-08-11" "1810-09-18" "1949-10-01"
##  [41] "1810-07-20" "1975-07-06" "1960-06-30" "1958-11-28" "1960-08-15"
##  [46] "1821-09-15" NA           "1868-10-10" "1960-10-01" "1918-10-28"
##  [51] "1993-01-01" "1977-06-27" "1978-11-03" "1821-11-30" "1844-02-27"
##  [56] "1865-08-16" "1975-11-28" "1809-08-10" "1952-07-23" "1821-09-15"
##  [61] "1968-10-12" "1991-05-24" "1918-02-24" "1991-08-20" "1968-09-06"
##  [66] "1970-10-10" "1917-12-06" "1960-08-17" "1965-02-18" "1918-05-26"
##  [71] "1991-04-09" "1990-10-03" "1957-03-06" "1821-03-25" "1974-02-07"
##  [76] "1821-09-15" "1958-10-02" "1973-09-24" "1974-09-10" "1975-07-05"
##  [81] "1966-05-26" "1804-01-01" "1821-09-15" NA           "1944-06-17"
##  [86] "1947-08-15" "1945-08-17" "1979-04-01" "1932-10-03" NA
```

```
##  [91] "1960-08-07" "1962-08-06" "0660-02-11" "1946-05-25" "1991-12-16"
##  [96] "1963-12-12" "1979-07-12" "1945-08-15" "1919-03-01" "1945-08-15"
## [101] "1961-02-25" "1991-08-31" "1918-11-18" "1990-05-04" "1943-11-22"
## [106] "1966-10-04" "1847-07-26" "1951-12-24" "1886-08-15" "1918-02-16"
## [111] "1990-03-11" "1960-06-26" "1964-07-06" "1957-08-31" "1965-09-16"
## [116] "1965-07-26" "1960-09-22" "1964-09-21" "1979-05-01" "1960-11-28"
## [121] "1968-03-12" "1810-09-16" "1986-11-03" "1991-08-27" NA
## [126] "2006-05-21" "1944-01-11" "1955-11-18" "1975-06-25" "1948-01-04"
## [131] "1990-03-21" "1968-01-31" "1581-07-26" "1821-09-15" "1960-08-03"
## [136] "1960-10-01" "1991-09-08" "1814-05-17" "1905-06-07" "1650-11-18"
## [141] "1947-08-14" "1994-10-01" NA           "1821-11-28" "1903-11-03"
## [146] "1975-09-16" NA           "1821-07-28" "1898-06-12" "1946-07-04"
## [151] "1918-11-11" "1640-12-01" "1878-12-18" "1971-09-03" "1876-05-10"
## [156] "1918-12-01" "1962-07-01" "1983-09-19" "1979-02-22" "1979-10-27"
## [161] "1962-06-01" "1975-07-12" "1960-04-04" "1804-02-15" "1976-06-29"
## [166] "1961-04-27" "1965-08-09" "1992-07-17" "1918-10-28" "1993-01-01"
## [171] "1991-06-25" "1990-12-26" "1978-07-07" "1960-07-01" "1960-06-26"
## [176] "2011-07-09" "1948-02-04" "1956-01-01" "1975-11-25" "1523-06-06"
## [181] "1291-08-01" "1946-04-17" "1912-10-10" "1945-10-25" "1991-09-09"
## [186] "1961-12-09" "1960-04-27" "1970-06-04" "1962-08-31" "1956-03-20"
## [191] "1991-09-27" "1978-10-01" "1962-10-09" "1991-08-24" "1919-01-22"
## [196] "1971-12-02" "1776-07-04" "1825-08-25" "1991-09-01" "1980-07-30"
## [201] "1929-02-11" "1811-07-05" "1945-09-02" "1967-11-30" "1964-10-24"
## [206] "1980-04-18"
```

```
data$fechas=as.Date(data$Date, format = "%d %B %Y")
str(data)
```

```
## tibble [206 x 9] (S3: tbl_df/tbl/data.frame)
##  $ Country                     : chr [1:206] "Afghanistan" "Albania" "Algeria" "Angola" ...
##  $                             : logi [1:206] NA NA NA NA NA NA ...
##  $ Name of holiday             : chr [1:206] "Afghan Independence Day (Afghan Victory Day)" "Flag Da
##  $ Date of holiday             : chr [1:206] "19 August" "28 November" "5 July" "11 November" ...
##  $ Year of event               : chr [1:206] "1919" "1912" "1962" "1975" ...
##  $ Independence from           : chr [1:206] "United Kingdom" "Ottoman Empire" "France" "Portugal" .
##  $ Event commemorated and notes: chr [1:206] "Anglo-Afghan Treaty of 1919 or Treaty of Rawalpindi, a
##  $ Date                        : chr [1:206] "19 August 1919" "28 November 1912" "5 July 1962" "11 N
##  $ fechas                      : Date[1:206], format: "1919-08-19" "1912-11-28" ...
```

Una vez que hemos otorgado un formato adecuado, esto nos permite hacer operaciones con el vector, tal y como si fuera una variable con formatos a los que estamos acostumbrados. Para realizar estas operaciones, podemos utilizar la librería *lubridate*.

Vamos, además, a agregar una columna con una fecha cercana:

```
data$act_date = as.Date("2023-11-02", format = "%Y-%m-%d")
head(data)
```

```
## # A tibble: 6 x 10
##   Country             ``    `Name of holiday`   `Date of holid~` `Year of event`
##   <chr>               <lgl> <chr>               <chr>            <chr>
## 1 Afghanistan         NA    Afghan Independenc~ 19 August        1919
## 2 Albania             NA    Flag Day            28 November      1912
```

```
## 3 Algeria             NA    Independence Day    5 July           1962
## 4 Angola              NA    Independence Day    11 November      1975
## 5 Antigua and Barbuda NA    Independence Day    1 November       1981
## 6 Argentina           NA    Independence Day    9 July           1816[8]
## # ... with 5 more variables: 'Independence from' <chr>,
## #   'Event commemorated and notes' <chr>, Date <chr>, fechas <date>,
## #   act_date <date>
```

¿Cuantos tiempo ha pasado desde que se declaró la independencia de estos países? Con el paquete antes mencionado, podemos utilizar la función *interval*, que nos permite sacar las diferencias en años, meses, semanas e incluso días:

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
data$ind_years=interval(data$fechas,data$act_date) %/% years(1)
head(data)
```

```
## # A tibble: 6 x 11
##   Country             ''    'Name of holiday'   'Date of holid~' 'Year of event'
##   <chr>               <lgl> <chr>               <chr>            <chr>
## 1 Afghanistan         NA    Afghan Independenc~ 19 August        1919
## 2 Albania             NA    Flag Day            28 November      1912
## 3 Algeria             NA    Independence Day    5 July           1962
## 4 Angola              NA    Independence Day    11 November      1975
## 5 Antigua and Barbuda NA    Independence Day    1 November       1981
## 6 Argentina           NA    Independence Day    9 July           1816[8]
## # ... with 6 more variables: 'Independence from' <chr>,
## #   'Event commemorated and notes' <chr>, Date <chr>, fechas <date>,
## #   act_date <date>, ind_years <dbl>
```

Ojo: siempre debemos tener datos completos!

Si quisieramos tener más detalle:

```
data$ind_months=interval(data$fechas,data$act_date) %/% months(1)
data$ind_weeks=interval(data$fechas,data$act_date) %/% weeks(1)
data$ind_days=interval(data$fechas,data$act_date) %/% days(1)
head(data[8:14])
```

```
## # A tibble: 6 x 7
##   Date             fechas     act_date   ind_years ind_months ind_weeks ind_days
##   <chr>            <date>     <date>         <dbl>      <dbl>     <dbl>    <dbl>
## 1 19 August 1919   1919-08-19 2023-11-02       104       1250      5437    38061
## 2 28 November 1912 1912-11-28 2023-11-02       110       1331      5788    40516
## 3 5 July 1962      1962-07-05 2023-11-02        61        735      3200    22400
## 4 11 November 1975 1975-11-11 2023-11-02        47        575      2503    17523
## 5 1 November 1981  1981-11-01 2023-11-02        42        504      2191    15341
## 6 9 July 1816[8]   1816-07-09 2023-11-02       207       2487     10817    75721
```

¿Qué país tiene menos tiempo como República independiente y qué país tiene más tiempo siendo independiente? Al tener un formato adecuado, podemos extraer esa información.

```
data[which.min(data$ind_years),]
```

```
## # A tibble: 1 x 14
##   Country     ``    `Name of holiday` `Date of holiday` `Year of event`
##   <chr>       <lgl> <chr>             <chr>             <chr>
## 1 South Sudan NA    Independence Day  9 July            2011
## # ... with 9 more variables: `Independence from` <chr>,
## #   `Event commemorated and notes` <chr>, Date <chr>, fechas <date>,
## #   act_date <date>, ind_years <dbl>, ind_months <dbl>, ind_weeks <dbl>,
## #   ind_days <dbl>
```

```
data[which.max(data$ind_years),]
```

```
## # A tibble: 1 x 14
##   Country ``    `Name of holiday`       `Date of holiday` `Year of event`
##   <chr>   <lgl> <chr>                   <chr>             <chr>
## 1 Japan   NA    National Foundation Day 11 February       660 BC
## # ... with 9 more variables: `Independence from` <chr>,
## #   `Event commemorated and notes` <chr>, Date <chr>, fechas <date>,
## #   act_date <date>, ind_years <dbl>, ind_months <dbl>, ind_weeks <dbl>,
## #   ind_days <dbl>
```

Parece que hay una inconsistencia con Japón. Para efectos prácticos del ejercicio, eliminemos el caso:

```
data = data[-c(93),]
```

Ahora vemos que es Suiza:

```
data[which.max(data$ind_years),]
```

```
## # A tibble: 1 x 14
##   Country     ``    `Name of holiday`  `Date of holiday` `Year of event`
##   <chr>       <lgl> <chr>              <chr>             <chr>
## 1 Switzerland NA    Swiss National Day 1 August          1291
## # ... with 9 more variables: `Independence from` <chr>,
## #   `Event commemorated and notes` <chr>, Date <chr>, fechas <date>,
## #   act_date <date>, ind_years <dbl>, ind_months <dbl>, ind_weeks <dbl>,
## #   ind_days <dbl>
```

Veamos otro ejemplo. ¿Qué deberíamos hacer si la información la tenemos en columnas separadas? Llamemos a otra base de datos para simular un evento así:

```
dat1 <- read.csv("http://mgimond.github.io/ES218/Data/CO2.csv")
head(dat1)
```

```
##   Year Month Average Interpolated  Trend Daily_mean
## 1 1959     1  315.62       315.62 315.70         -1
```

```
## 2 1959     2  316.38       316.38 315.88         -1
## 3 1959     3  316.71       316.71 315.62         -1
## 4 1959     4  317.72       317.72 315.56         -1
## 5 1959     5  318.29       318.29 315.50         -1
## 6 1959     6  318.15       318.15 315.92         -1
```

Primero, construyamos un vector con toda la información que necesitamos. Para eso podemos utilizar la función *paste* que ya conocemos. En este caso demás, vamos a utilizar el argumento separados, para diferenciar año, mes y día con un "-":

```
#paste(dat1$Year,dat1$Month, sep="-")
#paste(dat1$Year, dat1$Month, "15", sep="-")
```

Elaborado el vector, podemos darle formato. El paquete *lubridate*, adicionalmente, contiene funciones para detectar rápidamente cuando tenemos un vector que contiene años, meses y días. Esto lo hacemos con la función *ymd* si es que el orden de la información es año - mes - día. El orden de las letras cambiará según se presente el orden de las fechas.

Functions	Date Format
dmy()	day/month/year
ymd()	year/month/day
ydm()	year/day/month

Asignemos el día 15 como una fecha arbitraria para otorgar el formato con la función *ymd*:

```
dat1$nd = ymd(paste(dat1$Year, dat1$Month, "15", sep = "-"))
str(dat1$nd)
```

```
##  Date[1:721], format: "1959-01-15" "1959-02-15" "1959-03-15" "1959-04-15" "1959-05-15" ...
```

2. Trabajando con horas, minutos y segundos:

El trabajo con horas, minutos y segundos es bastante similar al caso de las fechas. En esta tabla tenemos los códigos según la estructura de nuestros datos:

Code	Meaning	Code	Meaning
%a	Abbreviated weekday	%A	Full weekday
%b	Abbreviated month	%B	Full month
%c	Locale-specific date and time	%d	Decimal date
%H	Decimal hours (24 hour)	%I	Decimal hours (12 hour)
%j	Decimal day of the year	%m	Decimal month
%M	Decimal minute	%p	Locale-specific AM/PM
%S	Decimal second	%U	Decimal week of the year (starting on Sunday)
%w	Decimal Weekday (0=Sunday)	%W	Decimal week of the year (starting on Monday)
%x	Locale-specific Date	%X	Locale-specific Time
%y	2-digit year	%Y	4-digit year
%z	Offset from GMT	%Z	Time zone (character)

Llamemos a la data *ventanillas*, es una base de datos con los tiempos de atención que pasan en ventanilla las y los ciudadanos de una Municipalidad distrital cuando acuden a acceder a ciertos servicios. Cargamos la data:

```r
library(rio)
ventanillas = import("https://github.com/Alexanderbenit7/EleccionesGenerales2021/blob/master/Santa%20An
head(ventanillas)
```

```
##     VENTANILLA           OPERADOR TICKET OPERACION SUB-OPERACION H. GENERACION
## 1 VENTANILLA 1 JUAN QUISPE MENDEZ   MP 1        NA            NA      08:04:11
## 2 VENTANILLA 1 JUAN QUISPE MENDEZ   MP 2        NA            NA      08:09:32
## 3 VENTANILLA 1 JUAN QUISPE MENDEZ   MP 4        NA            NA      08:44:09
## 4 VENTANILLA 1 JUAN QUISPE MENDEZ   MP 6        NA            NA      09:09:56
## 5 VENTANILLA 1 JUAN QUISPE MENDEZ  EC 37        NA            NA      10:44:02
## 6 VENTANILLA 1 JUAN QUISPE MENDEZ  EC 42        NA            NA      10:57:12
##   H. LLAMADA H. ATENCION H. FIN DE ATENCION         FECHA
## 1   08:24:26    08:24:29           08:25:07 2/01/18 08:04
## 2   08:25:10    08:25:11           08:45:54 2/01/18 08:09
## 3   08:46:01    08:46:03           08:57:57 2/01/18 08:44
## 4   09:21:49    09:21:50           09:27:31 2/01/18 09:09
## 5   10:45:25    10:46:02           10:57:36 2/01/18 10:44
## 6   10:57:40    10:57:45           11:46:57 2/01/18 10:57
##   HORA INICIO DE SUB-OPERACION HORA FIN DE SUB-OPERACION
## 1                     00:00:00                 00:00:00
## 2                     00:00:00                 00:00:00
## 3                     00:00:00                 00:00:00
## 4                     00:00:00                 00:00:00
## 5                     00:00:00                 00:00:00
## 6                     00:00:00                 00:00:00
```

Veamos algo de su estructura:

```r
ventanillas = ventanillas[,c(1,3,6:9)]
str(ventanillas)
```

```
## 'data.frame':    53972 obs. of  6 variables:
##  $ VENTANILLA        : chr  "VENTANILLA 1" "VENTANILLA 1" "VENTANILLA 1" "VENTANILLA 1" ...
##  $ TICKET            : chr  "MP 1" "MP 2" "MP 4" "MP 6" ...
##  $ H. GENERACION     : chr  "08:04:11" "08:09:32" "08:44:09" "09:09:56" ...
##  $ H. LLAMADA        : chr  "08:24:26" "08:25:10" "08:46:01" "09:21:49" ...
##  $ H. ATENCION       : chr  "08:24:29" "08:25:11" "08:46:03" "09:21:50" ...
##  $ H. FIN DE ATENCION: chr  "08:25:07" "08:45:54" "08:57:57" "09:27:31" ...
```

Cambiemos algunos nombres de las columnas para trabajar con mayor facilidad:

```r
names(ventanillas)
```

```
## [1] "VENTANILLA"         "TICKET"             "H. GENERACION"
## [4] "H. LLAMADA"         "H. ATENCION"        "H. FIN DE ATENCION"
```

```r
colnames(ventanillas) = c("VENTANILLA","TICKET","GENERACION","LLAMADA","ATENCION","FIN")
```

Otorguemos formato a uno de los vectores:

```r
#as.POSIXct(ventanillas$GENERACION, format = "%H:%M:%S") #Pone la fecha de hoy por defecto
ventanillas$GENER_TEST = as.POSIXct(ventanillas$GENERACION, format = "%H:%M:%S")
str(ventanillas$GENER_TEST)
```

```
##  POSIXct[1:53972], format: "2023-11-02 08:04:11" "2023-11-02 08:09:32" "2023-11-02 08:44:09" ...
```

Puede presentarse el caso eventual en el que toda la información esté concentrada en una celda; es decir, que una columna tenga no solo la hora, los minutos y segundos, sino también la fecha en la que se produjo la atención. ¿Cómo daríamos formato en casos así? Simulemos un escenario como el descrito asignando fechas aleatorias a cada ventanilla. De paso, practicamos *ifelse*:

```r
ventanillas$day = ifelse(ventanillas$VENTANILLA == "VENTANILLA 1", "July 5 2021",
                         ifelse(ventanillas$VENTANILLA == "VENTANILLA 2", "May 21 2021",
                         ifelse(ventanillas$VENTANILLA == "VENTANILLA 3", "June 13 2021",
                         ifelse(ventanillas$VENTANILLA == "VENTANILLA 4", "June 25 2021",
                         ifelse(ventanillas$VENTANILLA == "VENTANILLA 5", "June 2 2021",
                         ifelse(ventanillas$VENTANILLA == "VENTANILLA 6", "July 3 2021",
                         ifelse(ventanillas$VENTANILLA == "VENTANILLA 7", "May 5 2021",
                         ifelse(ventanillas$VENTANILLA == "VENTANILLA 8", "June 26 2021",
                         ifelse(ventanillas$VENTANILLA == "VENTANILLA 9", "May 13 2021",
                         ifelse(ventanillas$VENTANILLA == "VENTANILLA 10", "November 15 2021",
                         ifelse(ventanillas$VENTANILLA == "VENTANILLA 11", "June 8 2021",
                         ifelse(ventanillas$VENTANILLA == "VENTANILLA 12", "June 13 2021",
                         ifelse(ventanillas$VENTANILLA == "VENTANILLA 13", "June 13 2021",
                         ifelse(ventanillas$VENTANILLA == "VENTANILLA 14", "June 29 2021",
                         ifelse(ventanillas$VENTANILLA == "VENTANILLA 15", "November 13 2021",
                         ifelse(ventanillas$VENTANILLA == "VENTANILLA 16", "June 13 2021",
                         ifelse(ventanillas$VENTANILLA == "VENTANILLA 17", "May 22 2021",0)))))))))))))))))
```

Imaginando que tuvieramos una celda con las fechas y las horas, esta sería la manera de darle formato a todo completo:

```r
ventanillas$gen_fecha = paste(ventanillas$day, ventanillas$GENERACION)
ventanillas$gen_fecha=as.POSIXct(ventanillas$gen_fecha,format="%B %d %Y %H:%M:%S")
str(ventanillas$gen_fecha)
```

```
##  POSIXct[1:53972], format: "2021-07-05 08:04:11" "2021-07-05 08:09:32" "2021-07-05 08:44:09" ...
```

Otorguemos formato a las columnas que nos interesan para el análisis: observemos que va a agregar por defecto la fecha de hoy. Notemos los **:** como argumento separador entre horas, minutos y segundos. Eso varía dependendiendo de la presentación de los datos. Puede ser un / o un -.

```r
ventanillas$GENERACION = as.POSIXct(ventanillas$GENERACION, format ="%H:%M:%S")
ventanillas$LLAMADA = as.POSIXct(ventanillas$LLAMADA, format ="%H:%M:%S")
ventanillas$ATENCION = as.POSIXct(ventanillas$ATENCION, format ="%H:%M:%S")
ventanillas$FIN = as.POSIXct(ventanillas$FIN, format ="%H:%M:%S")
str(ventanillas)
```

```
## 'data.frame':    53972 obs. of  9 variables:
##  $ VENTANILLA: chr  "VENTANILLA 1" "VENTANILLA 1" "VENTANILLA 1" "VENTANILLA 1" ...
##  $ TICKET    : chr  "MP 1" "MP 2" "MP 4" "MP 6" ...
```

```
##  $ GENERACION: POSIXct, format: "2023-11-02 08:04:11" "2023-11-02 08:09:32" ...
##  $ LLAMADA   : POSIXct, format: "2023-11-02 08:24:26" "2023-11-02 08:25:10" ...
##  $ ATENCION  : POSIXct, format: "2023-11-02 08:24:29" "2023-11-02 08:25:11" ...
##  $ FIN       : POSIXct, format: "2023-11-02 08:25:07" "2023-11-02 08:45:54" ...
##  $ GENER_TEST: POSIXct, format: "2023-11-02 08:04:11" "2023-11-02 08:09:32" ...
##  $ day       : chr  "July 5 2021" "July 5 2021" "July 5 2021" "July 5 2021" ...
##  $ gen_fecha : POSIXct, format: "2021-07-05 08:04:11" "2021-07-05 08:09:32" ...
```

Veamos la duración total de la atención en cada ventanilla. Para eso podemos utilizar la función **diff**:

```
ventanillas$diff = difftime(ventanillas$FIN,ventanillas$GENERACION, units="min") #secs si fueran segund
ventanillas$secs = difftime(ventanillas$FIN,ventanillas$GENERACION, units="secs")
```

Al tener ya esa información, podemos realizar algunos análisis. Por ejemplo, ¿cuál fue el tiempo record de atención? ¿Qué ventanilla tiene el record?

```
min(ventanillas$diff)
```

```
## Time difference of 0.1833333 mins
```

```
ventanillas[which.min(ventanillas$diff),]
```

```
##         VENTANILLA TICKET          GENERACION             LLAMADA
## 37920 VENTANILLA 4  PF 11 2023-11-02 10:30:48 2023-11-02 10:30:49
##                   ATENCION                 FIN          GENER_TEST          day
## 37920 2023-11-02 10:30:58 2023-11-02 10:30:59 2023-11-02 10:30:48 June 25 2021
##                 gen_fecha           diff    secs
## 37920 2021-06-25 10:30:48 0.1833333 mins 11 secs
```

¿Cuáles fueron los tardones?

```
max(ventanillas$diff)
```

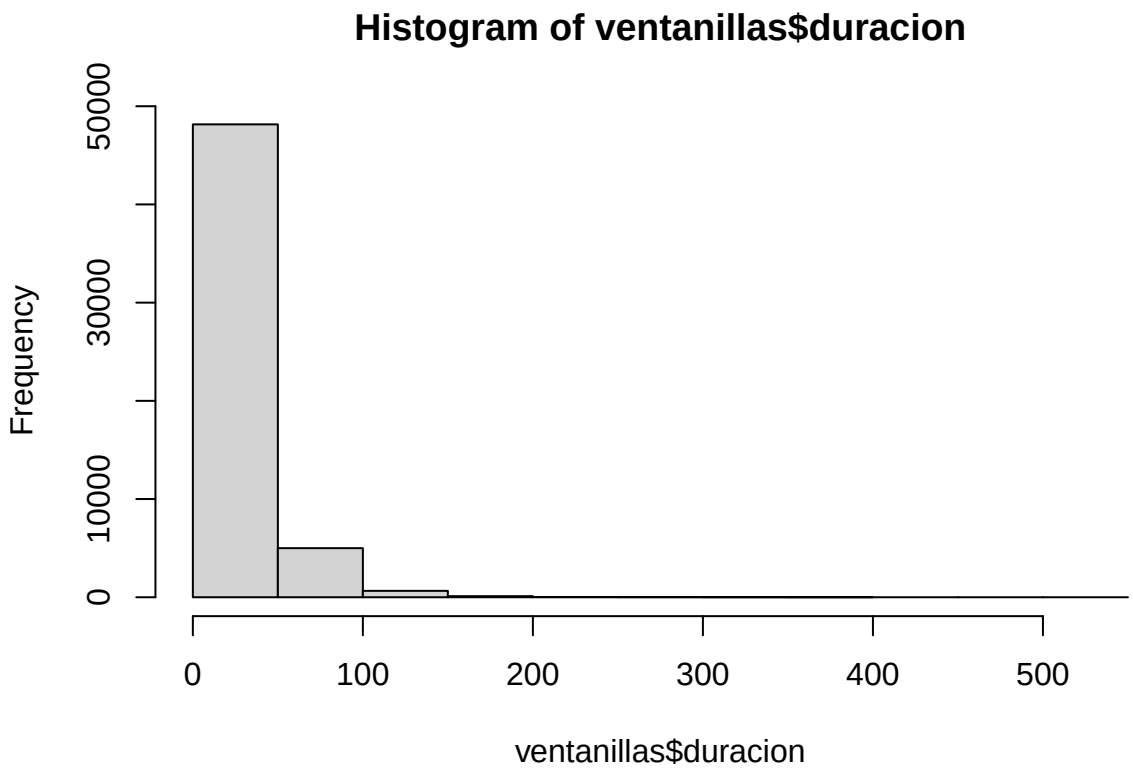
```
## Time difference of 529.5333 mins
```

```
ventanillas[which.max(ventanillas$diff),]
```

```
##         VENTANILLA TICKET          GENERACION             LLAMADA
## 42846 VENTANILLA 5   MP 1 2023-11-02 08:03:12 2023-11-02 08:11:13
##                   ATENCION                 FIN          GENER_TEST         day
## 42846 2023-11-02 08:11:15 2023-11-02 16:52:44 2023-11-02 08:03:12 June 2 2021
##                 gen_fecha          diff       secs
## 42846 2021-06-02 08:03:12 529.5333 mins 31772 secs
```

En tanto este es un vector numérico, ya podemos hacer algunas operaciones. Veamos la distribución del tiempo de atención:

```
ventanillas$duracion=as.numeric(ventanillas$diff)
hist(ventanillas$duracion)
```



Hay tickets que duran más en promedio? Veamos:

```
table(ventanillas$TICKET)
```

```
##
##   CA 1  CA 10 CA 100 CA 101 CA 102 CA 103 CA 104 CA 105 CA 106 CA 107 CA 108
##     89     65      5      5      5      5      5      5      4      5      3
## CA 109  CA 11 CA 110 CA 111 CA 112 CA 113 CA 114 CA 115 CA 116 CA 117 CA 118
##      3     66      4      3      3      3      3      3      3      3      3
## CA 119  CA 12 CA 120 CA 121 CA 122 CA 123 CA 124 CA 125 CA 126 CA 127 CA 128
##      3     65      3      3      3      3      3      3      2      3      3
## CA 129  CA 13 CA 130 CA 131 CA 132 CA 133 CA 134 CA 135 CA 136 CA 137 CA 138
##      3     64      3      3      3      3      3      3      3      3      3
## CA 139  CA 14 CA 140 CA 141 CA 142 CA 143 CA 144 CA 145 CA 146 CA 147 CA 148
##      3     65      3      3      3      3      3      3      2      2      2
## CA 149  CA 15 CA 150 CA 151 CA 152 CA 153 CA 154 CA 155 CA 156 CA 157 CA 158
##      2     64      2      2      2      2      2      2      2      2      2
## CA 159  CA 16 CA 161 CA 162 CA 163  CA 17  CA 18  CA 19   CA 2  CA 20  CA 21
##      1     64      1      1      1     64     62     63     79     62     63
##  CA 22  CA 23  CA 24  CA 25  CA 26  CA 27  CA 28  CA 29   CA 3  CA 30  CA 31
##     63     60     60     60     58     59     58     56     72     55     56
##  CA 32  CA 33  CA 34  CA 35  CA 36  CA 37  CA 38  CA 39   CA 4  CA 40  CA 41
##     54     53     53     50     50     51     50     50     73     49     48
##  CA 42  CA 43  CA 44  CA 45  CA 46  CA 47  CA 48  CA 49   CA 5  CA 50  CA 51
##     49     47     45     45     46     45     46     44     72     42     42
##  CA 52  CA 53  CA 54  CA 55  CA 56  CA 57  CA 58  CA 59   CA 6  CA 60  CA 61
```

```
##     40     41     40     37     35     35     34     30     71     30     30
##  CA 62  CA 63  CA 64  CA 65  CA 66  CA 67  CA 68  CA 69   CA 7  CA 70  CA 71
##     28     28     28     27     25     24     26     26     69     24     25
##  CA 72  CA 73  CA 74  CA 75  CA 76  CA 77  CA 78  CA 79   CA 8  CA 80  CA 81
##     22     23     24     22     22     20     22     21     67     21     17
##  CA 82  CA 83  CA 84  CA 85  CA 86  CA 87  CA 88  CA 89   CA 9  CA 90  CA 91
##     17     19     19     19     18     14     13     15     67     12     13
##  CA 92  CA 93  CA 94  CA 95  CA 96  CA 97  CA 98  CA 99  CON 1 CON 10 CON 11
##     11     11      9      8      8      8      7      5      9      1      1
## CON 12 CON 13 CON 14 CON 15 CON 16 CON 17 CON 18 CON 19  CON 2 CON 20 CON 21
##      1      1      1      1      1      1      1      1      1      1      1
## CON 22 CON 23 CON 24 CON 25 CON 26 CON 27 CON 28 CON 29  CON 3 CON 30 CON 31
##      1      1      1      1      1      1      1      1      1      1      1
## CON 32 CON 33 CON 34 CON 35 CON 36 CON 37 CON 38 CON 39  CON 4 CON 40 CON 41
##      1      1      1      1      1      1      1      1      1      1      1
## CON 42 CON 43 CON 44 CON 45 CON 46 CON 47 CON 48 CON 49  CON 5  CON 6  CON 7
##      1      1      1      1      1      1      1      1      1      1      1
##  CON 8  CON 9   DC 1   DC 2   DC 3   DC 4   DJ 1  DJ 10  DJ 11  DJ 12  DJ 13
##      1      1     10      3      2      2    118     92     93     91     90
##  DJ 14  DJ 15  DJ 16  DJ 17  DJ 18  DJ 19   DJ 2  DJ 20  DJ 21  DJ 22  DJ 23
##     85     81     75     69     59     53    111     46     39     36     32
##  DJ 24  DJ 25  DJ 26  DJ 27  DJ 28  DJ 29   DJ 3  DJ 30  DJ 31  DJ 32  DJ 33
##     21     16     13     12     11     11    108      8      8      6      5
##  DJ 34  DJ 35  DJ 36  DJ 37   DJ 4   DJ 5   DJ 6   DJ 7   DJ 8   DJ 9   EC 1
##      4      3      1      1    109    105    105    101    100     97    149
##  EC 10 EC 100 EC 101 EC 102 EC 103 EC 104 EC 105 EC 106 EC 107 EC 108 EC 109
##    150    123    123    123    123    122    121    121    117    119    115
##  EC 11 EC 110 EC 111 EC 112 EC 113 EC 114 EC 115 EC 116 EC 117 EC 118 EC 119
##    150    118    119    116    119    119    120    114    115    114    113
##  EC 12 EC 120 EC 121 EC 122 EC 123 EC 124 EC 125 EC 126 EC 127 EC 128 EC 129
##    148    112    112    111    113    109    110    108    111    108    105
##  EC 13 EC 130 EC 131 EC 132 EC 133 EC 134 EC 135 EC 136 EC 137 EC 138 EC 139
##    147    105    104    103    102    103     99     96    101     96     94
##  EC 14 EC 140 EC 141 EC 142 EC 143 EC 144 EC 145 EC 146 EC 147 EC 148 EC 149
##    148     94     93     93     90     90     89     89     91     92     90
##  EC 15 EC 150 EC 151 EC 152 EC 153 EC 154 EC 155 EC 156 EC 157 EC 158 EC 159
##    145     87     88     86     86     89     89     79     79     81     80
##  EC 16 EC 160 EC 161 EC 162 EC 163 EC 164 EC 165 EC 166 EC 167 EC 168 EC 169
##    144     81     82     81     78     79     79     78     76     79     74
##  EC 17 EC 170 EC 171 EC 172 EC 173 EC 174 EC 175 EC 176 EC 177 EC 178 EC 179
##    146     76     73     74     73     73     73     74     72     72     72
##  EC 18 EC 180 EC 181 EC 182 EC 183 EC 184 EC 185 EC 186 EC 187 EC 188 EC 189
##    146     73     74     71     69     71     72     72     68     68     71
##  EC 19 EC 190 EC 191 EC 192 EC 193 EC 194 EC 195 EC 196 EC 197 EC 198 EC 199
##    144     67     68     66     65     67     63     67     66     64     65
##   EC 2  EC 20 EC 200 EC 201 EC 202 EC 203 EC 204 EC 205 EC 206 EC 207 EC 208
##    149    146     62     62     64     63     62     61     58     57     54
## EC 209  EC 21 EC 210 EC 211 EC 212 EC 213 EC 214 EC 215 EC 216 EC 217 EC 218
##     53    146     54     54     54     54     54     53     52     50     49
## EC 219  EC 22 EC 220 EC 221 EC 222 EC 223 EC 224 EC 225 EC 226 EC 227 EC 228
##     51    145     50     47     47     46     46     44     44     44     41
## EC 229  EC 23 EC 230 EC 231 EC 232 EC 233 EC 234 EC 235 EC 236 EC 237 EC 238
##     43    146     43     40     38     40     38     37     38     38     38
## EC 239  EC 24 EC 240 EC 241 EC 242 EC 243 EC 244 EC 245 EC 246 EC 247 EC 248
```

```
##     34    146     35     36     34     35     35     34     32     32     29
## EC 249  EC 25 EC 250 EC 251 EC 252 EC 253 EC 254 EC 255 EC 256 EC 257 EC 258
##     31    145     23     25     22     23     22     23     22     20     21
## EC 259  EC 26 EC 260 EC 261 EC 262 EC 263 EC 264 EC 265 EC 266 EC 267 EC 268
##     21    145     20     21     20     19     18     19     17     18     15
## EC 269  EC 27 EC 270 EC 271 EC 272 EC 273 EC 274 EC 275 EC 276 EC 277 EC 278
##     15    144     16     17     15     15     13     14     11     11     11
## EC 279  EC 28 EC 280 EC 281 EC 282 EC 283 EC 284 EC 285 EC 286 EC 287 EC 288
##     12    143      9     13     11     11     11     11     10     10      9
## EC 289  EC 29 EC 290 EC 291 EC 292 EC 293 EC 294 EC 295 EC 296 EC 297 EC 298
##      8    143      9      8      7      8      8      7      8      7      4
## EC 299   EC 3  EC 30 EC 300 EC 301 EC 302 EC 303 EC 304 EC 305 EC 306 EC 307
##      4    151    143      6      7      7      7      5      6      5      4
## EC 308 EC 309  EC 31 EC 310 EC 311 EC 312 EC 313 EC 314 EC 315 EC 316 EC 317
##      5      5    145      6      5      4      4      4      4      5      4
## EC 318 EC 319  EC 32 EC 320 EC 321 EC 322 EC 323 EC 324 EC 325 EC 326 EC 327
##      4      4    145      4      4      4      4      4      3      4      3
## EC 328 EC 329  EC 33 EC 330 EC 331 EC 332 EC 333 EC 334 EC 335 EC 336 EC 337
##      3      2    145      3      3      2      2      2      3      3      3
## EC 338 EC 339  EC 34 EC 340 EC 341 EC 342 EC 343 EC 344 EC 345 EC 346 EC 347
##      2      3    142      2      3      2      3      2      2      1      3
## EC 348 EC 349  EC 35 EC 350 EC 351 EC 352 EC 353 EC 354 EC 355 EC 356 EC 357
##      3      2    145      3      3      3      3      3      3      3      3
## EC 358 EC 359  EC 36 EC 360 EC 361 EC 362 EC 363 EC 364 EC 365 EC 366 EC 367
##      3      3    144      3      2      3      3      3      2      2      2
## EC 368 EC 369  EC 37 EC 370 EC 371 EC 372 EC 373 EC 374 EC 375 EC 376 EC 377
##      2      2    143      2      2      2      2      2      2      2      2
## EC 378 EC 379  EC 38 EC 380 EC 381 EC 382 EC 383 EC 384 EC 385 EC 386 EC 387
##      1      2    141      2      2      2      1      1      2      2      2
## EC 388 EC 389  EC 39 EC 390 EC 391 EC 392 EC 393 EC 394 EC 395 EC 396 EC 397
##      2      2    141      2      2      1      1      1      2      2      2
## EC 398 EC 399   EC 4  EC 40 EC 400 EC 401 EC 402 EC 403 EC 404 EC 405 EC 406
##      2      2    147    141      1      2      2      2      2      2      2
## EC 407 EC 408 EC 409  EC 41 EC 410 EC 411 EC 412 EC 413 EC 414 EC 415 EC 416
##      2      2      1    144      2      1      2      1      1      1      1
## EC 417 EC 418 EC 419  EC 42 EC 420 EC 421 EC 422 EC 423 EC 424 EC 425 EC 426
##      1      1      1    143      1      1      1      1      1      1      1
## EC 427 EC 429  EC 43 EC 430 EC 431 EC 432 EC 433 EC 434 EC 436 EC 437 EC 438
##      1      1    140      1      1      1      1      1      1      1      1
##  EC 44 EC 441 EC 442 EC 443 EC 444 EC 445 EC 446 EC 447 EC 448 EC 449  EC 45
##    144      1      1      1      1      1      1      1      1      1    142
## EC 450 EC 452 EC 453 EC 454 EC 455 EC 456 EC 457 EC 458 EC 459  EC 46 EC 460
##      1      1      1      1      1      1      1      1      1    141      1
## EC 461 EC 462 EC 463 EC 464 EC 465 EC 466 EC 467 EC 469  EC 47 EC 470 EC 473
##      1      1      1      1      1      1      1      1    143      1      1
## EC 474 EC 475 EC 476 EC 477 EC 478 EC 479  EC 48 EC 480 EC 482 EC 484 EC 486
##      1      1      1      1      1      1    141      1      1      1      1
## EC 488  EC 49   EC 5  EC 50  EC 51  EC 52  EC 53  EC 54  EC 55  EC 56  EC 57
##      1    139    148    141    141    143    140    140    143    141    140
##  EC 58  EC 59   EC 6  EC 60  EC 61  EC 62  EC 63  EC 64  EC 65  EC 66  EC 67
##    142    139    148    140    140    139    140    137    140    138    140
##  EC 68  EC 69   EC 7  EC 70  EC 71  EC 72  EC 73  EC 74  EC 75  EC 76  EC 77
##    138    138    147    140    138    136    134    137    136    134    135
##  EC 78  EC 79   EC 8  EC 80  EC 81  EC 82  EC 83  EC 84  EC 85  EC 86  EC 87
```

```
##    135    132    149    133    136    133    134    132    133    131    133
##  EC 88  EC 89   EC 9  EC 90  EC 91  EC 92  EC 93  EC 94  EC 95  EC 96  EC 97
##    133    131    147    129    126    130    131    126    127    124    122
##  EC 98  EC 99   MP 1  MP 10 MP 100 MP 101 MP 102 MP 103 MP 104 MP 105 MP 106
##    125    125    146    145     45     40     40     39     33     33     30
## MP 107 MP 108 MP 109  MP 11 MP 110 MP 111 MP 112 MP 113 MP 114 MP 115 MP 116
##     25     22     19    144     19     18     19     20     17     12     12
## MP 117 MP 118 MP 119  MP 12 MP 120 MP 121 MP 122 MP 123 MP 124 MP 125 MP 126
##     11     12     10    143     10      8      7      7      4      5      5
## MP 127 MP 128 MP 129  MP 13 MP 130 MP 131 MP 132 MP 133 MP 134 MP 135 MP 136
##      3      4      2    146      2      2      1      1      1      1      1
## MP 137 MP 138 MP 139  MP 14 MP 140 MP 141 MP 142 MP 143  MP 15  MP 16  MP 17
##      1      1      1    141      1      1      1      1    139    137    137
##  MP 18  MP 19   MP 2  MP 20  MP 21  MP 22  MP 23  MP 24  MP 25  MP 26  MP 27
##    138    138    141    138    142    132    132    135    135    132    129
##  MP 28  MP 29   MP 3  MP 30  MP 31  MP 32  MP 33  MP 34  MP 35  MP 36  MP 37
##    132    129    144    129    124    129    126    126    120    119    121
##  MP 38  MP 39   MP 4  MP 40  MP 41  MP 42  MP 43  MP 44  MP 45  MP 46  MP 47
##    119    120    144    117    118    119    118    117    115    120    116
##  MP 48  MP 49   MP 5  MP 50  MP 51  MP 52  MP 53  MP 54  MP 55  MP 56  MP 57
##    118    117    144    121    117    115    116    116    120    118    115
##  MP 58  MP 59   MP 6  MP 60  MP 61  MP 62  MP 63  MP 64  MP 65  MP 66  MP 67
##    117    114    147    118    115    111    113    114    113    115    116
##  MP 68  MP 69   MP 7  MP 70  MP 71  MP 72  MP 73  MP 74  MP 75  MP 76  MP 77
##    113    116    145    114    110    108    104    105    106    107    104
##  MP 78  MP 79   MP 8  MP 80  MP 81  MP 82  MP 83  MP 84  MP 85  MP 86  MP 87
##    103    103    145     96     95     94     92     88     87     90     86
##  MP 88  MP 89   MP 9  MP 90  MP 91  MP 92  MP 93  MP 94  MP 95  MP 96  MP 97
##     86     83    144     80     76     73     67     63     63     54     59
##  MP 98  MP 99   OP 1   PF 1  PF 10 PF 100 PF 101 PF 102 PF 103 PF 104 PF 105
##     50     49      5    135    105     14     13     12     14     12     13
## PF 106 PF 107 PF 108 PF 109  PF 11 PF 110 PF 111 PF 112 PF 113 PF 114 PF 115
##     12     14     14     13    103     10     11     10      9      8      8
## PF 116 PF 117 PF 118 PF 119  PF 12 PF 120 PF 121 PF 122 PF 123 PF 124 PF 125
##      6      7      7      7     99      7      7      6      7      6      6
## PF 126 PF 127 PF 128 PF 129  PF 13 PF 130 PF 131 PF 132 PF 133 PF 134 PF 135
##      6      7      6      6     96      6      5      6      5      5      5
## PF 136 PF 137 PF 138 PF 139  PF 14 PF 140 PF 141 PF 142 PF 143 PF 144 PF 145
##      5      4      5      5     95      5      4      4      4      3      3
## PF 146 PF 147 PF 148 PF 149  PF 15 PF 150 PF 151 PF 152 PF 153 PF 154 PF 155
##      2      2      3      2     91      2      1      2      2      2      2
## PF 156 PF 157 PF 158 PF 159  PF 16 PF 160 PF 161 PF 162 PF 163 PF 164 PF 165
##      2      1      1      1     90      1      1      1      1      1      1
## PF 167 PF 168 PF 169  PF 17 PF 170 PF 172 PF 174  PF 18  PF 19   PF 2  PF 20
##      1      1      1     85      1      1      1     83     77    134     77
##  PF 21  PF 22  PF 23  PF 24  PF 25  PF 26  PF 27  PF 28  PF 29   PF 3  PF 30
##     75     74     68     65     63     60     60     55     53    127     53
##  PF 31  PF 32  PF 33  PF 34  PF 35  PF 36  PF 37  PF 38  PF 39   PF 4  PF 40
##     50     49     49     46     48     43     42     44     41    121     40
##  PF 41  PF 42  PF 43  PF 44  PF 45  PF 46  PF 47  PF 48  PF 49   PF 5  PF 50
##     40     40     39     37     40     40     37     34     33    116     37
##  PF 51  PF 52  PF 53  PF 54  PF 55  PF 56  PF 57  PF 58  PF 59   PF 6  PF 60
##     37     33     32     35     34     31     31     29     29    119     30
##  PF 61  PF 62  PF 63  PF 64  PF 65  PF 66  PF 67  PF 68  PF 69   PF 7  PF 70
```

```
##     27     29     27     27     27     27     25     24     26    115     22
##  PF 71  PF 72  PF 73  PF 74  PF 75  PF 76  PF 77  PF 78  PF 79   PF 8  PF 80
##     22     22     23     22     22     22     20     19     20    117     19
##  PF 81  PF 82  PF 83  PF 84  PF 85  PF 86  PF 87  PF 88  PF 89   PF 9  PF 90
##     20     18     19     18     18     19     15     17     15    109     17
##  PF 91  PF 92  PF 93  PF 94  PF 95  PF 96  PF 97  PF 98  PF 99   RC 1  RC 10
##     16     16     14     16     15     16     16     16     14     11      5
##  RC 11  RC 12  RC 13  RC 14  RC 15  RC 16  RC 17  RC 18  RC 19   RC 2  RC 20
##      5      5      5      5      5      4      4      4      4      8      4
##  RC 21  RC 22  RC 23  RC 24  RC 25  RC 26  RC 27  RC 28  RC 29   RC 3  RC 30
##      3      3      3      2      2      2      2      2      2      6      2
##  RC 31  RC 32  RC 33  RC 34  RC 35  RC 36   RC 4   RC 5   RC 6   RC 7   RC 8
##      2      1      1      1      1      1      6      6      5      5      5
##   RC 9   SF 1   SF 2   SF 3   SF 4   VA 1  VA 10 VA 100 VA 101 VA 102 VA 103
##      5     13      4      1      1     43     26     10     10     10     10
## VA 104 VA 105 VA 106 VA 107 VA 108 VA 109  VA 11 VA 110 VA 111 VA 112 VA 113
##     10     10     10      9      8     10     25      9     11     11      9
## VA 114 VA 115 VA 116 VA 117 VA 118 VA 119  VA 12 VA 120 VA 121 VA 122 VA 123
##     10     10     10      9     10      9     23     10     10     10      9
## VA 124 VA 125 VA 126 VA 127 VA 128 VA 129  VA 13 VA 130 VA 131 VA 132 VA 133
##      9     10     10      9     10     10     25     10     10     10     10
## VA 134 VA 135 VA 136 VA 137 VA 138 VA 139  VA 14 VA 140 VA 141 VA 142 VA 143
##     10     10     10     10      9     10     24     10      9      9     10
## VA 144 VA 145 VA 146 VA 147 VA 148 VA 149  VA 15 VA 150 VA 151 VA 152 VA 153
##     10     10     10      6      8      9     24      9      8      8      6
## VA 154 VA 155 VA 156 VA 157 VA 158 VA 159  VA 16 VA 160 VA 161 VA 162 VA 163
##      8      7      7      8      8      7     22      7      7      7      9
## VA 164 VA 165 VA 166 VA 167 VA 168 VA 169  VA 17 VA 170 VA 171 VA 172 VA 173
##      8      9      8      9      7      8     22      8     10      9      9
## VA 174 VA 175 VA 176 VA 177 VA 178 VA 179  VA 18 VA 180 VA 181 VA 182 VA 183
##     10     10     10      9      8      9     23      6      6      6      7
## VA 184 VA 185 VA 186 VA 187 VA 188 VA 189  VA 19 VA 190 VA 191 VA 192 VA 193
##      6      7      7      5      6      8     21      7      8      7      8
## VA 194 VA 195 VA 196 VA 197 VA 198 VA 199   VA 2  VA 20 VA 200 VA 201 VA 202
##      6      7      6      7      8      7     33     21      6      7      7
## VA 203 VA 204 VA 205 VA 206 VA 207 VA 208 VA 209  VA 21 VA 210 VA 211 VA 212
##      8      8      7      5      7      6      7     22      6      8      7
## VA 213 VA 214 VA 215 VA 216 VA 217 VA 218 VA 219  VA 22 VA 220 VA 221 VA 222
##      7      8      8      7      7      6      4     21      7      5      6
## VA 223 VA 224 VA 225 VA 226 VA 227 VA 228 VA 229  VA 23 VA 230 VA 231 VA 232
##      6      7      6      7      6      7      6     21      5      5      5
## VA 233 VA 234 VA 235 VA 236 VA 237 VA 238 VA 239  VA 24 VA 240 VA 241 VA 242
##      4      6      6      6      6      5      7     19      5      6      6
## VA 243 VA 244 VA 245 VA 246 VA 247 VA 248 VA 249  VA 25 VA 250 VA 251 VA 252
##      7      6      7      6      7      5      5     19      6      6      6
## VA 253 VA 254 VA 255 VA 256 VA 257 VA 258 VA 259  VA 26 VA 260 VA 261 VA 262
##      5      6      5      4      3      4      5     20      6      5      5
## VA 263 VA 264 VA 265 VA 266 VA 267 VA 268 VA 269  VA 27 VA 270 VA 271 VA 272
##      5      4      4      5      5      4      6     19      5      5      6
## VA 273 VA 274 VA 275 VA 276 VA 277 VA 278 VA 279  VA 28 VA 280 VA 281 VA 282
##      6      6      5      5      5      5      5     18      5      4      5
## VA 283 VA 284 VA 285 VA 286 VA 287 VA 288 VA 289  VA 29 VA 290 VA 291 VA 292
##      6      5      6      6      6      5      6     18      5      5      5
## VA 293 VA 294 VA 295 VA 296 VA 297 VA 298 VA 299   VA 3  VA 30 VA 300 VA 301
```

```
##      4      4      4      6      3      5      4     30     19      5      6
## VA 302 VA 303 VA 304 VA 305 VA 306 VA 307 VA 308 VA 309  VA 31 VA 310 VA 311
##      5      4      3      5      5      6      5      4     19      6      5
## VA 312 VA 313 VA 314 VA 315 VA 316 VA 317 VA 318 VA 319  VA 32 VA 320 VA 321
##      5      6      6      6      6      6      6      6     18      6      6
## VA 322 VA 323 VA 324 VA 325 VA 326 VA 327 VA 328 VA 329  VA 33 VA 330 VA 331
##      5      5      4      5      5      4      5      6     17      6      6
## VA 332 VA 333 VA 334 VA 335 VA 336 VA 337 VA 338 VA 339  VA 34 VA 340 VA 341
##      6      6      5      5      5      5      6      5     17      3      5
## VA 342 VA 343 VA 344 VA 345 VA 346 VA 347 VA 348 VA 349  VA 35 VA 350 VA 351
##      6      4      5      6      5      6      5      5     17      5      6
## VA 352 VA 353 VA 354 VA 355 VA 356 VA 357 VA 358 VA 359  VA 36 VA 360 VA 361
##      5      5      5      4      5      5      3      5     16      5      4
## VA 362 VA 363 VA 364 VA 365 VA 366 VA 367 VA 368 VA 369  VA 37 VA 370 VA 371
##      6      5      6      5      5      6      5      6     16      5      6
## VA 372 VA 373 VA 374 VA 375 VA 376 VA 377 VA 378 VA 379  VA 38 VA 380 VA 381
##      6      5      6      5      5      4      5      5     16      5      6
## VA 382 VA 383 VA 384 VA 385 VA 386 VA 387 VA 388 VA 389  VA 39 VA 390 VA 391
##      5      6      6      6      6      5      6      5     16      4      5
## VA 392 VA 393 VA 394 VA 395 VA 396 VA 397 VA 398 VA 399   VA 4  VA 40 VA 400
##      4      2      4      4      4      6      6      5     27     15      5
## VA 401 VA 402 VA 403 VA 404 VA 405 VA 406 VA 407 VA 408 VA 409  VA 41 VA 410
##      5      6      6      5      5      5      5      6      4     14      4
## VA 411 VA 412 VA 413 VA 414 VA 415 VA 416 VA 417 VA 418 VA 419  VA 42 VA 420
##      5      5      5      6      6      6      6      6      6     14      6
## VA 421 VA 422 VA 423 VA 424 VA 425 VA 426 VA 427 VA 428 VA 429  VA 43 VA 430
##      6      6      5      5      5      5      6      4      6     15      4
## VA 431 VA 432 VA 433 VA 434 VA 435 VA 436 VA 437 VA 438 VA 439  VA 44 VA 440
##      5      4      5      5      3      3      4      4      5     15      5
## VA 441 VA 442 VA 443 VA 444 VA 445 VA 446 VA 447 VA 448 VA 449  VA 45 VA 450
##      5      5      4      5      4      5      5      5      5     13      4
## VA 451 VA 452 VA 453 VA 454 VA 455 VA 456 VA 457 VA 458 VA 459  VA 46 VA 460
##      5      5      5      5      3      5      3      5      4     12      5
## VA 461 VA 462 VA 463 VA 464 VA 465 VA 466 VA 467 VA 468 VA 469  VA 47 VA 470
##      5      5      5      4      4      4      5      3      4     11      4
## VA 471 VA 472 VA 473 VA 474 VA 475 VA 476 VA 477 VA 478 VA 479  VA 48 VA 480
##      4      4      4      4      4      3      4      4      4     13      4
## VA 481 VA 482 VA 483 VA 484 VA 485 VA 486 VA 487 VA 488 VA 489  VA 49 VA 490
##      4      3      4      4      4      2      3      4      4     13      2
## VA 491 VA 492 VA 493 VA 494 VA 495 VA 496 VA 497 VA 498 VA 499   VA 5  VA 50
##      2      4      3      4      4      3      4      2      3     27     13
## VA 500 VA 501 VA 502 VA 503 VA 504 VA 505 VA 506 VA 507 VA 508 VA 509  VA 51
##      4      3      3      3      2      2      2      2      3      3     13
## VA 510 VA 511 VA 512 VA 513 VA 514 VA 515 VA 516 VA 517 VA 518 VA 519  VA 52
##      3      2      3      3      3      2      3      2      3      3     13
## VA 520 VA 521 VA 522 VA 523 VA 524 VA 525 VA 526 VA 527 VA 528 VA 529  VA 53
##      3      1      3      1      2      3      2      1      2      2     13
## VA 530 VA 531 VA 532 VA 533 VA 534 VA 535 VA 537 VA 539  VA 54 VA 540 VA 541
##      2      3      3      2      1      3      2      2     13      2      2
## VA 542 VA 543 VA 544 VA 545 VA 546 VA 547 VA 548 VA 549  VA 55 VA 550 VA 551
##      3      2      2      2      1      2      1      2     12      2      2
## VA 552 VA 553 VA 554 VA 555 VA 556 VA 557 VA 558 VA 559  VA 56 VA 560 VA 561
##      2      2      2      2      2      2      2      1     12      2      1
## VA 562 VA 563 VA 564 VA 565 VA 566 VA 567 VA 568 VA 569  VA 57 VA 570 VA 571
```

```
##      2      2      2      1      1      2      2      2     13      2      1
## VA 572 VA 573 VA 574 VA 575 VA 576 VA 577 VA 578 VA 579  VA 58 VA 580 VA 581
##      1      2      2      2      2      2      1      1     12      2      2
## VA 583 VA 584 VA 588 VA 589  VA 59 VA 590 VA 592 VA 593 VA 595 VA 596 VA 597
##      1      1      2      1     12      1      1      1      1      1      1
## VA 598 VA 599   VA 6  VA 60 VA 600 VA 601 VA 602 VA 603 VA 604 VA 605 VA 606
##      1      1     26     13      1      1      1      1      1      1      1
## VA 607 VA 608  VA 61 VA 611 VA 612 VA 613 VA 614 VA 617 VA 618 VA 619  VA 62
##      1      1     12      1      1      1      1      1      1      1     13
## VA 620 VA 621 VA 622 VA 624 VA 625 VA 626 VA 627 VA 628 VA 629  VA 63 VA 630
##      1      1      1      1      1      1      1      1      1     10      1
## VA 632 VA 633 VA 634 VA 635 VA 636 VA 637  VA 64 VA 643 VA 644 VA 645 VA 648
##      1      1      1      1      1      1     11      1      1      1      1
## VA 649  VA 65 VA 650 VA 651 VA 652 VA 653 VA 654 VA 655 VA 656 VA 657 VA 658
##      1     11      1      1      1      1      1      1      1      1      1
## VA 659  VA 66 VA 660 VA 661 VA 662 VA 663 VA 664 VA 665 VA 666 VA 667 VA 669
##      1     10      1      1      1      1      1      1      1      1      1
##  VA 67 VA 670 VA 671 VA 672 VA 674 VA 675 VA 676 VA 678 VA 679  VA 68 VA 681
##     11      1      1      1      1      1      1      1      1     11      1
## VA 683 VA 686 VA 688 VA 689  VA 69 VA 690 VA 691 VA 692 VA 693 VA 695 VA 696
##      1      1      1      1     11      1      1      1      1      1      1
##   VA 7  VA 70  VA 71  VA 72  VA 73  VA 74  VA 75  VA 76  VA 77  VA 78  VA 79
##     26     11     10     10     11     10     11     11     11     10     11
##   VA 8  VA 80  VA 81  VA 82  VA 83  VA 84  VA 85  VA 86  VA 87  VA 88  VA 89
##     29     11     11     11     11     10     11     11     10     11     11
##   VA 9  VA 90  VA 91  VA 92  VA 93  VA 94  VA 95  VA 96  VA 97  VA 98  VA 99
##     27     11     11      9     11     11     11     11     11      9     10
```

Son varios. Saquemos el promedio de atención por ticket:

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
TICKETS = ventanillas %>% group_by(TICKET) %>%
  summarize(DURACION = mean(duracion, na.rm = T))
```

¿Cuál tarda más? ¿Cuál dura menos?

```
TICKETS[which.max(TICKETS$DURACION),]
```

```
## # A tibble: 1 x 2
##   TICKET DURACION
##   <chr>     <dbl>
## 1 CON 30     197.
```

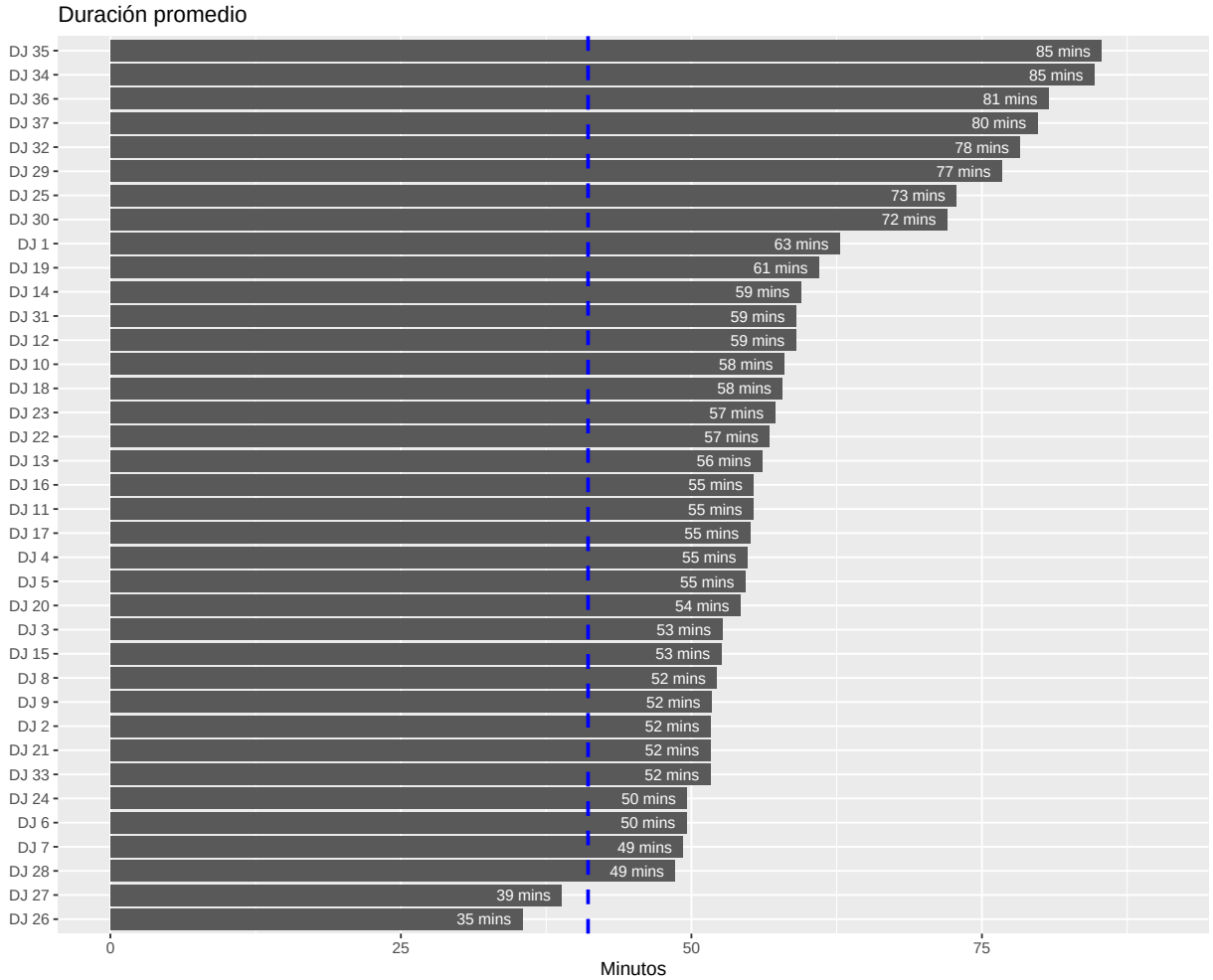
```
TICKETS[which.min(TICKETS$DURACION),]
```

```
## # A tibble: 1 x 2
##   TICKET DURACION
##   <chr>     <dbl>
## 1 EC 488     2.28
```

Grafiquemos solo los que son trámites administrativos (DJ):

```
ADMIN = filter(TICKETS, grepl("DJ", TICKET))
ADMIN = ADMIN[complete.cases(ADMIN$DURACION),]
```

Utilicemos los códigos que ya conocemos para graficar:



¿Y si la Municipalidad los contratara para hacer un estudio de los días con mayor y menor actividad? Podríamos hacer algo así:

Carguemos la original:

```
ventanillas = import("https://github.com/Alexanderbenit7/EleccionesGenerales2021/blob/master/Santa%20An
ventanillas = ventanillas[,c(1,3,6:9,10)]
colnames(ventanillas) = c("VENTANILLA","TICKET","GENERACION","LLAMADA","ATENCION","FIN","FECHA")
```

```
ventanillas$FECHA = substr(ventanillas$FECHA,1,8) #está la fecha de la atención
```

```
ventanillas$ATEN = paste(ventanillas$FECHA, ventanillas$GENERACION)
ventanillas$ATEN= as.POSIXct(ventanillas$ATEN,format="%d/%m/%y %H:%M:%S")
```

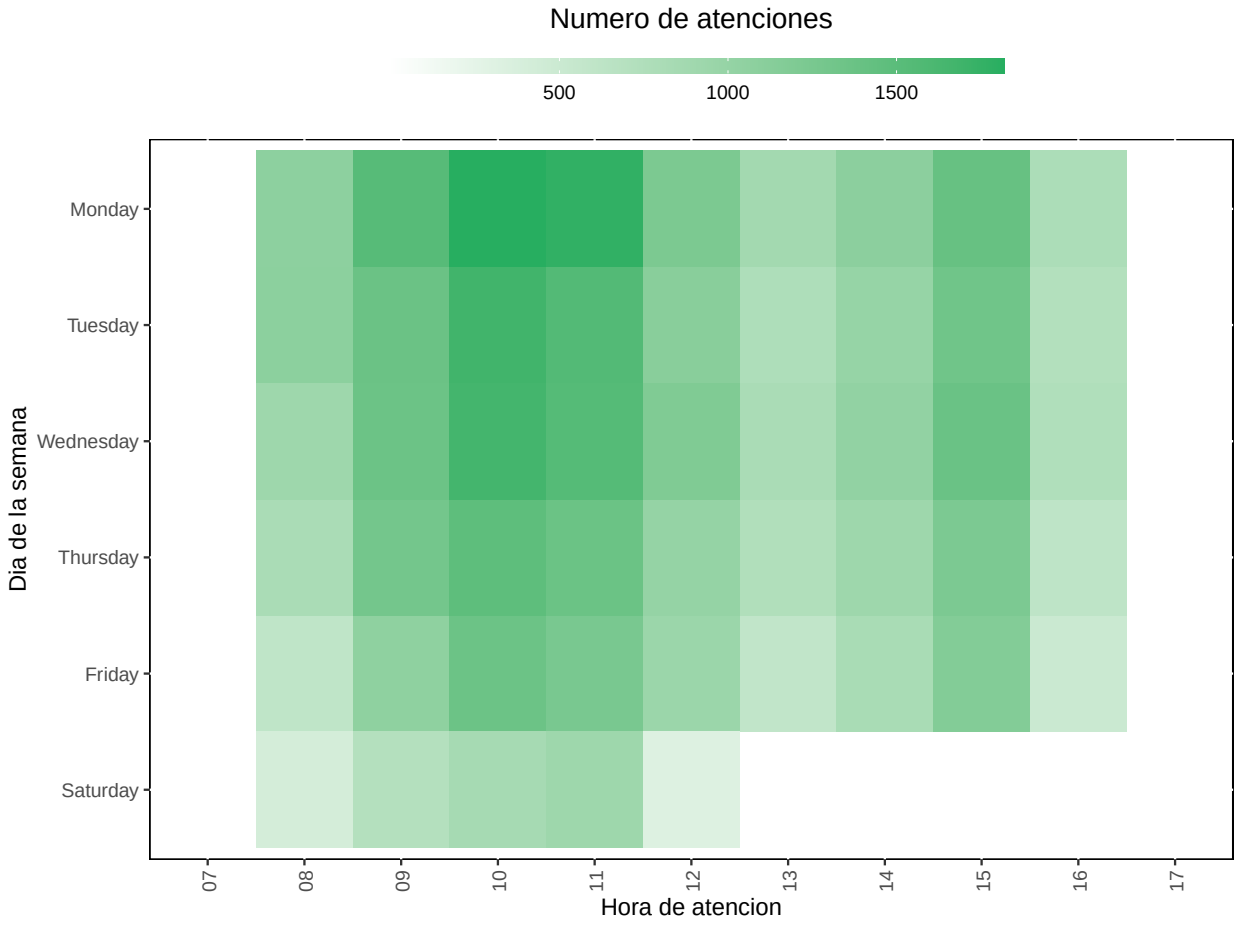
```
ventanillas$day=weekdays(ventanillas$ATEN, abbreviate = F) #nos quedamos con el día
ventanillas$hour = substr(ventanillas$ATEN,12,13) #nos quedamos con la hora
```

Número de atenciones por día y hora:

```
tiempos_Fecha = ventanillas %>%
  group_by(day, hour) %>%
  summarise(ATENCIONES = n ())
```

```
## `summarise()` has grouped output by 'day'. You can override using the `.groups`
## argument.
```

```
dow_format <- c("Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")
tiempos_Fecha$day <- factor(tiempos_Fecha$day, level = rev(dow_format))
```



3. Agrupación y construcción de intervalos:

Agrupar datos y construir intervalos desde los datos son dos conceptos que se encuntran relacionados. En lo que respecta al primero, lo que hacemos es transformar una variable numérica en una categórica, asignando cortes a los datos. Es un intercambio, en el sentido que perdemos precisión y ganamos "claridad" al momento de presentar la información.

A lo largo del curso ya hemos visto formas de agrupar nuestros datos, principalmente utilizando la función *ifelse*. Utilicemos la base de datos del artículo "Pavimentando con Votos" para ejemplificar diferentes formas de agrupar nuestros datos:

```
data = import("https://github.com/PoliticayGobiernoPUCP/estadistica_anapol2/blob/master/DATA/pavimentan
```

Hagamos algunos cortes a la variable con el porcentaje de votos de la oposición (pctopo) de la base de datos:

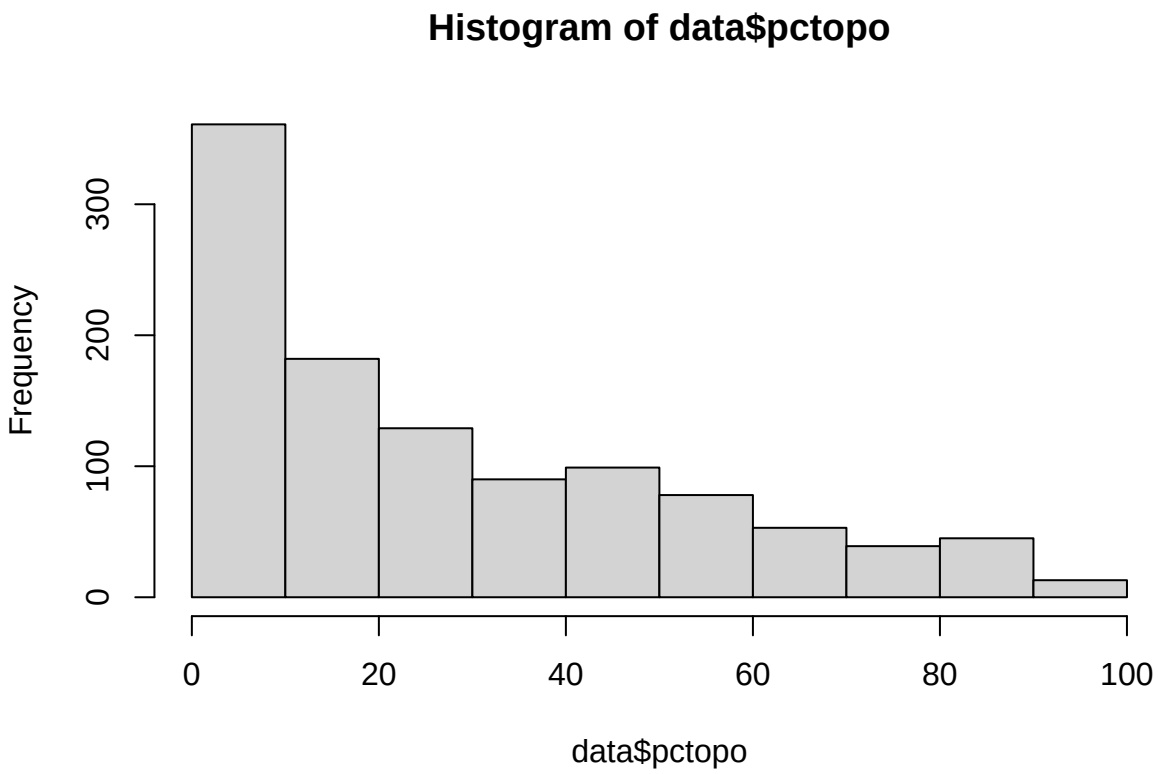
```
#Veamos cómo está formateada:
str(data$pctopo)
```

```
##  num [1:1096] 14.82 14.51 15.08 6.15 47.31 ...
##  - attr(*, "label")= chr "Porcentaje de votos de la Oposición"
##  - attr(*, "format.spss")= chr "F5.2"
```

```
summary(data$pctopo)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.000   5.922  20.308  27.874  45.711  99.419       7
```

```
hist(data$pctopo)
```



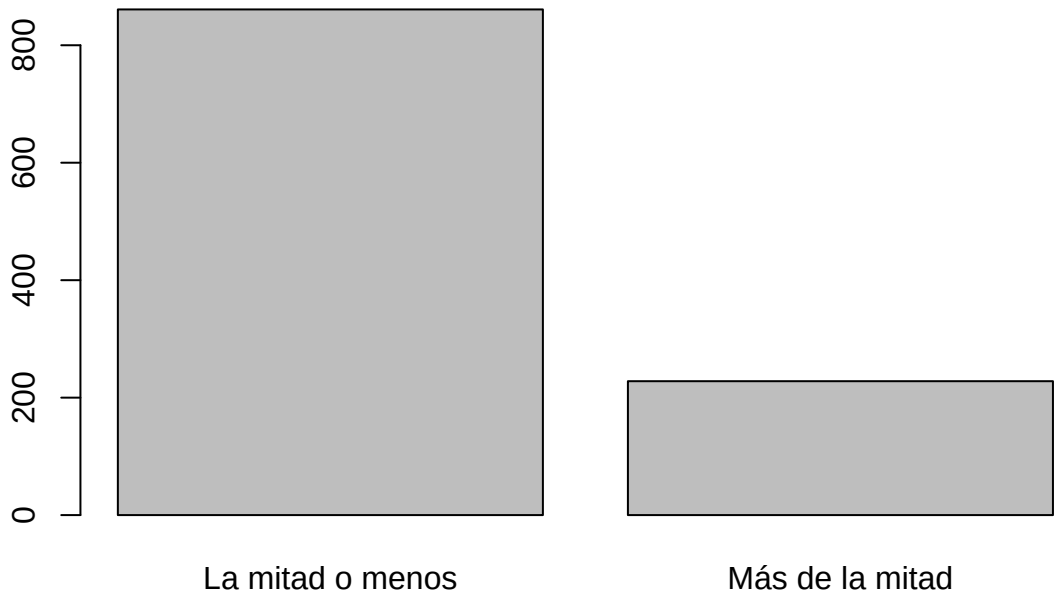
Empecemos diferenciando aquellos Municipios en los que la oposición obtuvo más de la mitad del % total de los votos. La forma clásica de hacerlo es utilizando *ifelse* con lenguaje base:

```
data$cortes_pctopo = factor(ifelse(data$pctopo>50,1,0))
data$cortes_pctopo = factor(data$cortes_pctopo, levels = c(0:1), labels = c("La mitad o menos","Más de l
```

```
table(data$cortes_pctopo)
```

```
##
## La mitad o menos  Más de la mitad
##              861              228
```

```
barplot(table(data$cortes_pctopo))
```



También hemos visto maneras de implementar la función *ifelse* a la librería *dplyr* y la función *mutate* para crear una variable nueva:

```
library(dplyr)
data = mutate(data, cortes_pctopo_2 = ifelse(pctopo>50,1,0))
data$cortes_pctopo_2 = factor(data$cortes_pctopo_2, levels = c(0:1), labels = c("La mitad o menos","Más
```

```
table(data$cortes_pctopo_2)
```

```
##
## La mitad o menos  Más de la mitad
##              861              228
```

Ahora, al agrupar nuestra data, podemos crear diferentes condicionales para realizar el agrupamiento. Por ejemplo, qué sucede si queremos crear grupos para dividir la data entre los Municipios tomando en cuenta el % de votos de la oposición y si es que había o no prioridad técnica (priorizado):

0 –> Municipio no priorizado 1 –> Municipio priorizado

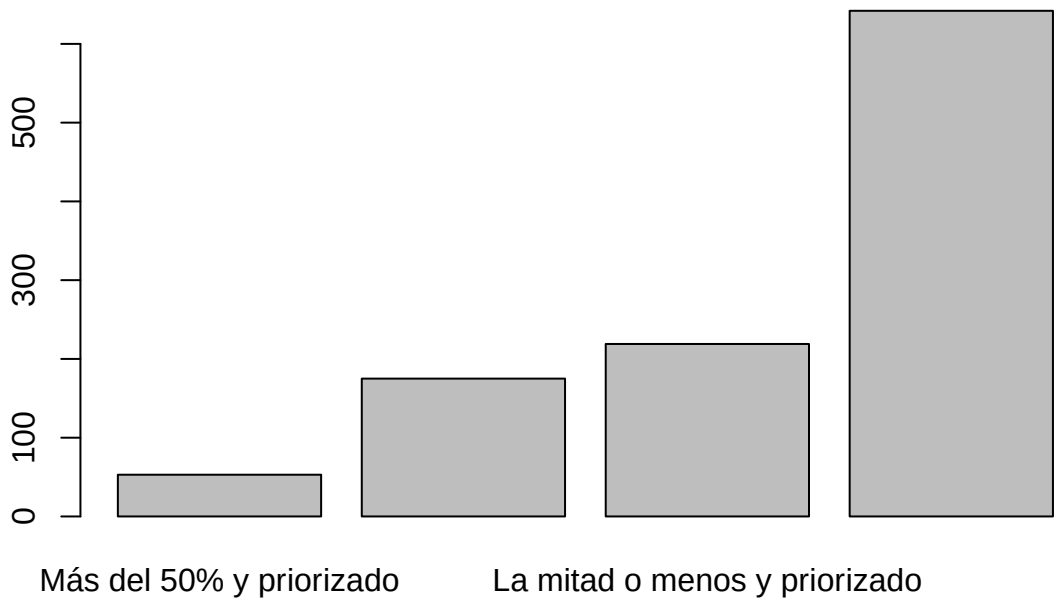
```
data$cat_dob = ifelse(data$pctopo > 50 & data$priorizado == 1, 1, #Ojo que son dos condiciones, por eso
                      ifelse(data$pctopo > 50 & data$priorizado == 0,2,
                             ifelse(data$pctopo <= 50 & data$priorizado == 1,3,
                                    ifelse(data$pctopo <= 50 & data$priorizado == 0,4,0))))
```

```r
data$cat_dob = factor(data$cat_dob, levels = c(1:4), labels = c("Más del 50% y priorizado",
                                                                "Más del 50% y no priorizado",
                                                                "La mitad o menos y priorizado",
                                                                "La mitad o menos y no priorizado"))
```

```r
table(data$cat_dob)
```

```
##
##         Más del 50% y priorizado      Más del 50% y no priorizado
##                               53                              175
##    La mitad o menos y priorizado La mitad o menos y no priorizado
##                              219                              642
```

```r
barplot(table(data$cat_dob))
```



Podemos seguir agregando más y más condiciones si ya entendimos la lógica del código. Es bueno, además, darle una mirada a los operadores en R, que se utilizan mucho en los condicionales: https://www.datamentor.io/r-programming/operator/

Si bien hacer esto es relativamente "sencillo", se puede volver más complejo cuando se asignan cortes más arbitrarios. ¿Qué es alto, medio, bajo y muy bajo? ¿Qué es muy bueno, bueno, malo y muy malo? Esto siempre puede ser contestado, ya que, si lo trasladamos al ámbito de las ciencias sociales, el aumento de una décima podría implicar el traslado de un país de un régimen híbrido a uno democrático. En suma, las consecuencias de esa decisión no son menores. Construir las categorías debe realizarse a partir de un buen balance entre la observación de los datos y la teoría.

En lo relacionado a la construcción de intervalos, estos se elaboran a partir de los datos. Para elaborarlos, necesitamos conocer el valor mínimo y máximo teórico, y decidir dónde cortar el rango. En R, podemos utilizar la función *cut*, y asignarle un vector de cortes que incluya al menor y al mayor valor. Ahora trabajemos con la variable de necesidades básicas insatisfechas (nbi):

```
str(data$nbi)
```

```
##  num [1:1096] 12.2 33.8 28.5 33.1 27.1 ...
##  - attr(*, "label")= chr "Necesidades Básicas Insatisfechas"
##  - attr(*, "format.spss")= chr "F5.2"
##  - attr(*, "display_width")= int 7
```

```
summary(data$nbi)
```

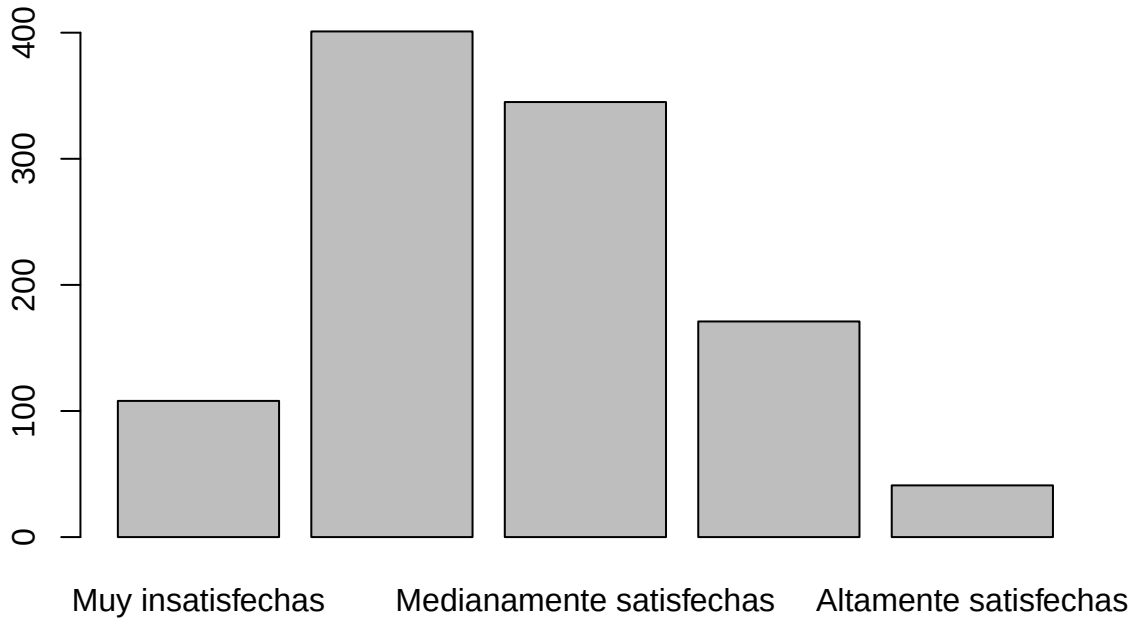
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    5.36   28.35   41.30   42.96   55.48   98.81      30
```

```
breaks_nbi = c(0,20,40,60,80,100)
labels_nbi = c("Muy insatisfechas","Insatisfechas","Medianamente satisfechas","Satisfechas","Altamente s
data$int_nbi = cut(data$nbi,
                   breaks = breaks_nbi,
                   labels = labels_nbi,
                   oredered_result = T)
```

```
table(data$int_nbi)
```

```
##
##        Muy insatisfechas            Insatisfechas Medianamente satisfechas
##                      108                      401                      345
##              Satisfechas    Altamente satisfechas
##                      171                       41
```

```
barplot(table(data$int_nbi))
```



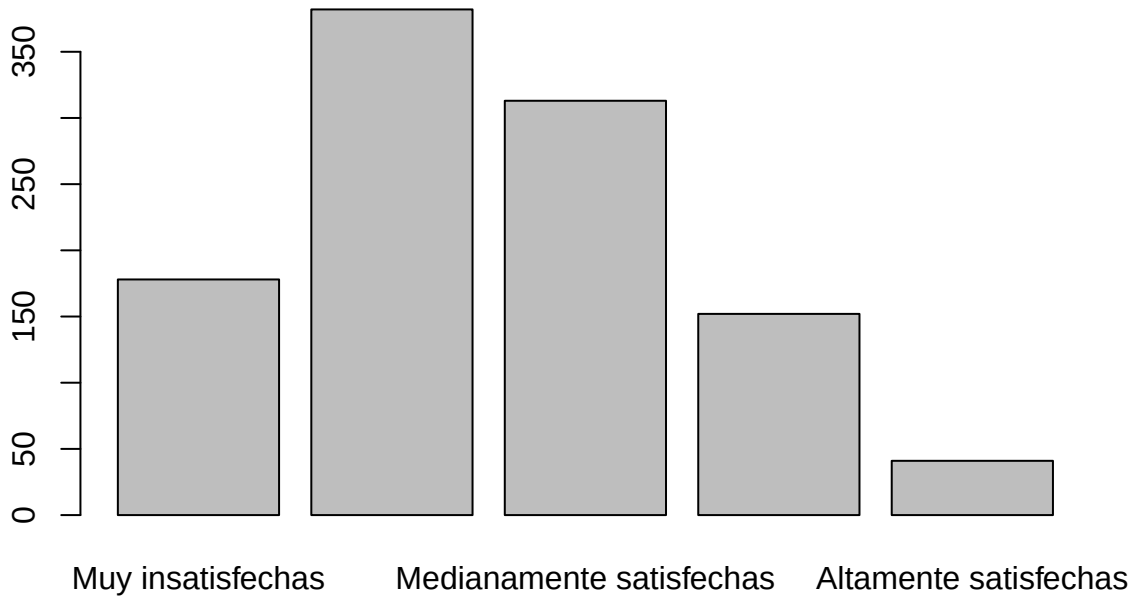
Pero podríamos simplemente especificar los cortes y dejar que se asignen los casos en función a la distribución de los datos:

```
breaks_nbi = 5
labels_nbi = c("Muy insatisfechas","Insatisfechas","Medianamente satisfechas","Satisfechas","Altamente s
data$int_nbi = cut(data$nbi,
                   breaks = breaks_nbi,
                   labels = labels_nbi,
                   oredered_result = T)
```

```
table(data$int_nbi)
```

```
##
##        Muy insatisfechas            Insatisfechas Medianamente satisfechas
##                      178                      382                      313
##              Satisfechas    Altamente satisfechas
##                      152                       41
```

```
barplot(table(data$int_nbi))
```



4. Agregando datos:

Muchas veces, la data con la que nos encontramos se encuentra bastante más desagregada de lo que en realidad necesitamos analizar. En esos casos, lo que nos interesa llevar a cabo es *agregar* los datos, de tal forma que trabajemos con un menor número de unidades de análisis. Veamos un ejemplo:

Carguemos una base de datos con los resultados de las EG 2016 en segunda vuelta a nivel distrital. La base, además, cuenta con variables como el índice de desarrollo humano, la población, el PBI per capita, entre otras:

```
data_idh = import("https://github.com/PoliticayGobiernoPUCP/estadistica_anapol2/raw/master/DATA/idh_ele
```

Hay diferentes formas de agregar datos, y va a depender mucho de los objetivos de nuestro análisis y la naturaleza de nuestras variables. Por ejemplo, ¿cómo podríamos proceder si quisieramos el IDH a nivel de provincias y no de distritos? Una forma de hacerlo es con la función *tapply*:

```
prov_idh = tapply(data_idh$idh, data_idh$prov, mean, na.rm = T)
prov_idh = as.data.frame(prov_idh)
prov_idh
```

```
##                          prov_idh
## ABANCAY                 0.3108333
## ACOBAMBA                0.2498375
## ACOMAYO                 0.2277000
```

```
## AIJA                        0.2756200
## ALTO AMAZONAS               0.2279167
## AMBO                        0.2811750
## ANDAHUAYLAS                 0.2515737
## ANGARAES                    0.2486833
## ANTA                        0.3122778
## ANTABAMBA                   0.2411000
## ANTONIO RAYMONDI            0.2279000
## AREQUIPA                    0.5481103
## ASCOPE                      0.4865750
## ASUNCION                    0.2606000
## ATALAYA                     0.2304500
## AYABACA                     0.2078900
## AYMARAES                    0.2547235
## AZANGARO                    0.2438267
## BAGUA                       0.3813500
## BARRANCA                    0.5291800
## BELLAVISTA                  0.3440500
## BOLIVAR                     0.2069500
## BOLOGNESI                   0.3493733
## BONGARA                     0.2938833
## CAJABAMBA                   0.2439000
## CAJAMARCA                   0.2654000
## CAJATAMBO                   0.3205800
## CALCA                       0.2856250
## CALLAO                      0.6461833
## CAMANA                      0.5341250
## CANAS                       0.2321125
## CANCHIS                     0.3203000
## CANDARAVE                   0.3567000
## CANETE                      0.5262625
## CANGALLO                    0.2233667
## CANTA                       0.3855286
## CARABAYA                    0.2577700
## CARAVELI                    0.4900231
## CARHUAZ                     0.2939091
## CARLOS FERMIN FITZCARRALD 0.2236667
## CASMA                       0.4291250
## CASTILLA                    0.3950857
## CASTROVIRREYNA              0.3104615
## CAYLLOMA                    0.3699050
## CELENDIN                    0.2087000
## CHACHAPOYAS                 0.2882571
## CHANCHAMAYO                 0.4362000
## CHEPEN                      0.4237000
## CHICLAYO                    0.4584300
## CHINCHA                     0.4611455
## CHINCHEROS                  0.2488375
## CHOTA                       0.2509789
## CHUCUITO                    0.3015857
## CHUMBIVILCAS                0.1966625
## CHUPACA                     0.3893444
## CHURCAMPA                   0.2712200
## CONCEPCION                  0.3348600
```

```
## CONDESUYOS                0.4246000
## CONDORCANQUI              0.1733000
## CONTRALMIRANTE VILLAR     0.4891333
## CONTUMAZA                 0.3113750
## CORONEL PORTILLO          0.3730714
## CORONGO                   0.3082286
## COTABAMBAS                0.2151833
## CUSCO                     0.5158750
## CUTERVO                   0.2571933
## DANIEL A. CARRION         0.3189125
## DATEM DEL MARANON         0.2116000
## DOS DE MAYO               0.2587222
## EL COLLAO                 0.2890800
## EL DORADO                 0.2722400
## ESPINAR                   0.3113625
## FERRENAFE                 0.3141667
## GENERAL SANCHEZ CERRO     0.4660000
## GRAN CHIMU                0.2392000
## GRAU                      0.2158929
## HUACAYBAMBA               0.2426000
## HUALGAYOC                 0.2767333
## HUALLAGA                  0.2954500
## HUAMALIES                 0.2445909
## HUAMANGA                  0.2812200
## HUANCA SANCOS             0.2671250
## HUANCABAMBA               0.2004625
## HUANCANE                  0.2451875
## HUANCAVELICA              0.2655263
## HUANCAYO                  0.3846929
## HUANTA                    0.2519750
## HUANUCO                   0.3139273
## HUARAL                    0.4192000
## HUARAZ                    0.3258667
## HUARI                     0.2842375
## HUARMEY                   0.3919600
## HUAROCHIRI                0.3483281
## HUAURA                    0.4132333
## HUAYLAS                   0.2739100
## HUAYTARA                  0.3205188
## ICA                       0.5255000
## ILO                       0.7197667
## ISLAY                     0.5402333
## JAEN                      0.3069083
## JAUJA                     0.3789912
## JORGE BASADRE             0.6211000
## JULCAN                    0.1649000
## JUNIN                     0.3685500
## LA CONVENCION             0.3474500
## LA MAR                    0.2196125
## LA UNION                  0.2443364
## LAMAS                     0.2826727
## LAMBAYEQUE                0.3536583
## LAMPA                     0.3197900
## LAURICOCHA                0.3379286
```

```
## LEONCIO PRADO            0.3215167
## LIMA                     0.6592256
## LORETO                   0.2727200
## LUCANAS                  0.3190667
## LUYA                     0.2862304
## MANU                     0.4801500
## MARANON                  0.2265333
## MARISCAL CACERES         0.3372400
## MARISCAL LUZURIAGA       0.2469875
## MARISCAL NIETO           0.5918500
## MARISCAL RAMON CASTILLA  0.2765750
## MAYNAS                   0.3308154
## MELGAR                   0.3098556
## MOHO                     0.2391250
## MORROPON                 0.3175700
## MOYOBAMBA                0.3580333
## NAZCA                    0.5012600
## OCROS                    0.3578500
## OTUZCO                   0.1998400
## OXAPAMPA                 0.3399286
## OYON                     0.4184833
## PACASMAYO                0.4745600
## PACHITEA                 0.2103750
## PADRE ABAD               0.3494333
## PAITA                    0.4223000
## PALLASCA                 0.2967364
## PALPA                    0.4468800
## PARINACOCHAS             0.2670125
## PARURO                   0.2081556
## PASCO                    0.4207231
## PATAZ                    0.2019846
## PAUCAR DEL SARA SARA     0.3096800
## PAUCARTAMBO              0.1838833
## PICOTA                   0.3760900
## PISCO                    0.4652625
## PIURA                    0.3936556
## POMABAMBA                0.2281500
## PUERTO INCA              0.3034400
## PUNO                     0.2905333
## PURUS                    0.2862000
## QUISPICANCHI             0.2864167
## RECUAY                   0.3170200
## REQUENA                  0.2799909
## RIOJA                    0.3459111
## RODRIGUEZ DE MENDOZA     0.2995083
## SAN ANTONIO DE PUTINA    0.3133800
## SAN IGNACIO              0.2842000
## SAN MARCOS               0.2304429
## SAN MARTIN               0.4095786
## SAN MIGUEL               0.2803923
## SAN PABLO                0.2254750
## SAN ROMAN                0.3611000
## SANCHEZ CARRION          0.1407625
## SANDIA                   0.3057100
```

```
## SANTA                     0.4530444
## SANTA CRUZ                0.2916091
## SANTIAGO DE CHUCO         0.2613375
## SATIPO                    0.3127625
## SECHURA                   0.4009833
## SIHUAS                    0.2438800
## SUCRE                     0.2866909
## SULLANA                   0.4271875
## TACNA                     0.5051900
## TAHUAMANU                 0.5871000
## TALARA                    0.4938333
## TAMBOPATA                 0.5159750
## TARATA                    0.3166000
## TARMA                     0.3491778
## TAYACAJA                  0.2617500
## TOCACHE                   0.3971200
## TRUJILLO                  0.4977818
## TUMBES                    0.5041167
## UCAYALI                   0.3150000
## URUBAMBA                  0.4361429
## UTCUBAMBA                 0.3395429
## VICTOR FAJARDO            0.2567167
## VILCAS HUAMAN             0.2181125
## VIRU                      0.3770333
## YAROWILCA                 0.2433250
## YAULI                     0.5160400
## YAUYOS                    0.3372727
## YUNGAY                    0.2510375
## YUNGUYO                   0.3044286
## ZARUMILLA                 0.4245750
```

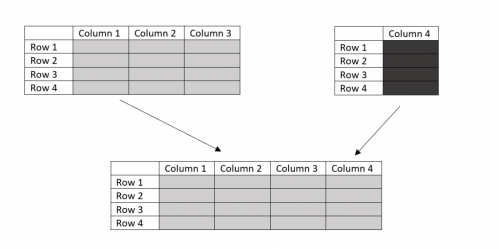
Y si lo queremos a nivel de regiones:

```
reg_idh = tapply(data_idh$idh, data_idh$depa, mean, na.rm = T)
reg_idh = as.data.frame(reg_idh)
reg_idh
```

```
##                 reg_idh
## AMAZONAS      0.2969310
## ANCASH        0.3064355
## APURIMAC      0.2487463
## AREQUIPA      0.4476486
## AYACUCHO      0.2730820
## CAJAMARCA     0.2649811
## CALLAO        0.6461833
## CUSCO         0.2985287
## HUANCAVELICA  0.2775787
## HUANUCO       0.2759408
## ICA           0.4858698
## JUNIN         0.3827691
## LA LIBERTAD   0.2993048
## LAMBAYEQUE    0.4025658
## LIMA          0.4590117
```

```
## LORETO         0.2819118
## MADRE DE DIOS 0.5223455
## MOQUEGUA       0.5418200
## PASCO          0.3714357
## PIURA          0.3459953
## PUNO           0.2867303
## SAN MARTIN     0.3472182
## TACNA          0.4291926
## TUMBES         0.4761846
## UCAYALI        0.3245200
```

Podemos hacerlo con dos variables a la vez: veamos promedios agregados de IDH y PBI per capita:

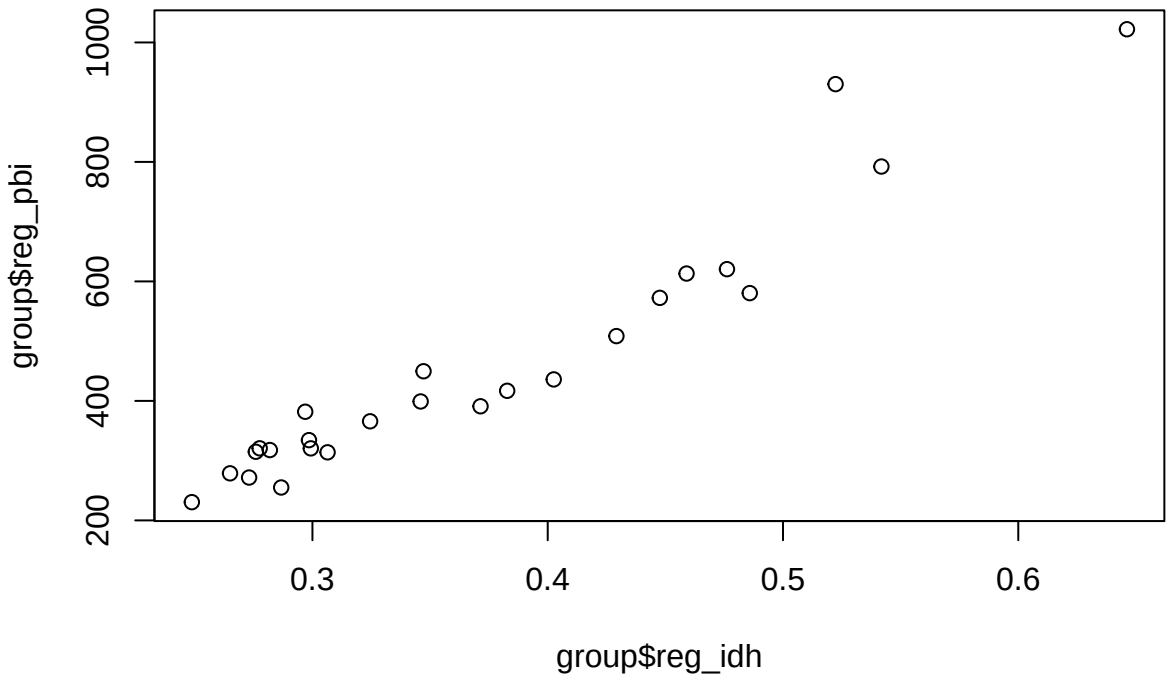



```r
reg_idh = tapply(data_idh$idh, data_idh$depa, mean, na.rm = T)
reg_pbi = tapply(data_idh$percapitaf, data_idh$depa, mean, na.rm = T)
group = as.data.frame(cbind(reg_idh, reg_pbi)) #Cuidado cuando hay casos perdidos y usan cbind!!
group
```

```
##                 reg_idh   reg_pbi
## AMAZONAS      0.2969310  381.8274
## ANCASH        0.3064355  313.9512
## APURIMAC      0.2487463  230.5375
## AREQUIPA      0.4476486  572.4587
## AYACUCHO      0.2730820  271.7333
## CAJAMARCA     0.2649811  278.7559
## CALLAO        0.6461833 1022.0667
## CUSCO         0.2985287  334.3417
## HUANCAVELICA  0.2775787  320.6266
## HUANUCO       0.2759408  314.8711
## ICA           0.4858698  580.4093
## JUNIN         0.3827691  416.9398
## LA LIBERTAD   0.2993048  320.5506
## LAMBAYEQUE    0.4025658  435.9605
## LIMA          0.4590117  613.1936
## LORETO        0.2819118  317.7608
## MADRE DE DIOS 0.5223455  930.2273
## MOQUEGUA      0.5418200  792.2850
## PASCO         0.3714357  391.0036
## PIURA         0.3459953  399.0453
## PUNO          0.2867303  255.0872
## SAN MARTIN    0.3472182  449.6195
## TACNA         0.4291926  508.4704
## TUMBES        0.4761846  620.5000
## UCAYALI       0.3245200  365.8733
```

Y con eso, ya podemos mirar nuestros datos a nivel regional. Repasemos Estadística 1:

```
plot(group$reg_idh, group$reg_pbi)
```



Hay, sin embargo, varias formas de agregar nuestros datos; más aún cuando lo hacemos sobre la media. Veamos la función *aggregate*:

```
aggregate(data_idh[,c(8)], by = list(data_idh[,4]), mean) #idh
```

```
##          Group.1         x
## 1       AMAZONAS 0.2969310
## 2         ANCASH 0.3064355
## 3       APURIMAC 0.2487463
## 4       AREQUIPA 0.4476486
## 5       AYACUCHO 0.2730820
## 6      CAJAMARCA 0.2649811
## 7         CALLAO 0.6461833
## 8          CUSCO 0.2985287
## 9   HUANCAVELICA 0.2775787
## 10       HUANUCO 0.2759408
## 11           ICA 0.4858698
## 12         JUNIN 0.3827691
## 13   LA LIBERTAD 0.2993048
## 14    LAMBAYEQUE 0.4025658
## 15          LIMA 0.4590117
```

```
## 16        LORETO 0.2819118
## 17 MADRE DE DIOS 0.5223455
## 18      MOQUEGUA 0.5418200
## 19         PASCO 0.3714357
## 20         PIURA 0.3459953
## 21          PUNO 0.2867303
## 22    SAN MARTIN 0.3472182
## 23         TACNA 0.4291926
## 24        TUMBES 0.4761846
## 25       UCAYALI 0.3245200
```

```
aggregate(data_idh[,c(8,12)], by = list(data_idh[,4]), mean) #idh y pbi per capita
```

```
##          Group.1       idh percapitaf
## 1       AMAZONAS 0.2969310   381.8274
## 2         ANCASH 0.3064355   313.9512
## 3       APURIMAC 0.2487463   230.5375
## 4       AREQUIPA 0.4476486   572.4587
## 5       AYACUCHO 0.2730820   271.7333
## 6      CAJAMARCA 0.2649811   278.7559
## 7         CALLAO 0.6461833  1022.0667
## 8          CUSCO 0.2985287   334.3417
## 9   HUANCAVELICA 0.2775787   320.6266
## 10       HUANUCO 0.2759408   314.8711
## 11           ICA 0.4858698   580.4093
## 12         JUNIN 0.3827691   416.9398
## 13   LA LIBERTAD 0.2993048   320.5506
## 14    LAMBAYEQUE 0.4025658   435.9605
## 15          LIMA 0.4590117   613.1936
## 16        LORETO 0.2819118   317.7608
## 17 MADRE DE DIOS 0.5223455   930.2273
## 18      MOQUEGUA 0.5418200   792.2850
## 19         PASCO 0.3714357   391.0036
## 20         PIURA 0.3459953   399.0453
## 21          PUNO 0.2867303   255.0872
## 22    SAN MARTIN 0.3472182   449.6195
## 23         TACNA 0.4291926   508.4704
## 24        TUMBES 0.4761846   620.5000
## 25       UCAYALI 0.3245200   365.8733
```

Lo mismo podemos hacer utilizando las funciones de *dplyr*:

```
data_idh %>%
  group_by(depa) %>%
  summarize(mean_idh = mean(idh, na.rm = TRUE))
```

```
## # A tibble: 25 x 2
##    depa         mean_idh
##    <chr>           <dbl>
##  1 AMAZONAS        0.297
##  2 ANCASH          0.306
##  3 APURIMAC        0.249
```

```
##  4 AREQUIPA        0.448
##  5 AYACUCHO        0.273
##  6 CAJAMARCA       0.265
##  7 CALLAO          0.646
##  8 CUSCO           0.299
##  9 HUANCAVELICA    0.278
## 10 HUANUCO         0.276
## # ... with 15 more rows
```

Y aquí podemos solicitar más medidas para agregar nuestra data:

```
data_idh %>%
  group_by(depa) %>%
  summarize(mean_idh = mean(idh, na.rm = TRUE), #media
            min_idh_reg = min(idh, na.rm = TRUE), #idh mínimo de la región
            max_idh_reg = max(idh, na.rm = TRUE), #idh máximo de la región
            median_idh = median(idh, na.rm = TRUE)) #mediana
```

```
## # A tibble: 25 x 5
##    depa         mean_idh min_idh_reg max_idh_reg median_idh
##    <chr>           <dbl>       <dbl>       <dbl>      <dbl>
##  1 AMAZONAS        0.297       0.149       0.547      0.295
##  2 ANCASH          0.306       0.115       0.607      0.300
##  3 APURIMAC        0.249       0.101       0.546      0.232
##  4 AREQUIPA        0.448       0.147       0.736      0.476
##  5 AYACUCHO        0.273       0.112       0.498      0.257
##  6 CAJAMARCA       0.265       0.124       0.534      0.259
##  7 CALLAO          0.646       0.532       0.764      0.644
##  8 CUSCO           0.299       0.135       0.682      0.28
##  9 HUANCAVELICA    0.278       0.139       0.585      0.258
## 10 HUANUCO         0.276       0.164       0.544      0.258
## # ... with 15 more rows
```

Ahora, esta base nos da información sobre la cantidad de votos por PPK y KF en cada distrito. Si nos solicitaran explorar visualmente la asociación entre el PBI per capita y el porcentaje de votos por PPK a nivel de departamentos, ¿cuál sería la forma de agregar los datos? Aquí el promedio no es útil. Necesitamos sumar los valores de cada caso. *Dplyr* es muy útil para ello:

```
pob_reg=data_idh %>%
  group_by(depa) %>%
  summarise(pob_reg = sum(pobla, na.rm = T))
pob_reg
```

```
## # A tibble: 25 x 2
##    depa         pob_reg
##    <chr>          <int>
##  1 AMAZONAS      417508
##  2 ANCASH       1129391
##  3 APURIMAC      451881
##  4 AREQUIPA     1245251
##  5 AYACUCHO      666029
##  6 CAJAMARCA    1513892
```

```
##  7 CALLAO        969170
##  8 CUSCO        1292175
##  9 HUANCAVELICA  483580
## 10 HUANUCO       840984
## # ... with 15 more rows
```

```
ppk_reg=data_idh %>%
  group_by(depa) %>%
  summarise(ppk_reg = sum(PPK, na.rm = T)) #para los conteos
ppk_reg
```

```
## # A tibble: 25 x 2
##    depa         ppk_reg
##    <chr>          <int>
##  1 AMAZONAS       78899
##  2 ANCASH        290878
##  3 APURIMAC       94183
##  4 AREQUIPA      566908
##  5 AYACUCHO      129186
##  6 CAJAMARCA     334586
##  7 CALLAO        296562
##  8 CUSCO         416788
##  9 HUANCAVELICA   95167
## 10 HUANUCO       164708
## # ... with 15 more rows
```

Ahora necesitamos juntar la información. El código que hemos venido utilizando es *cbind* para unir ambos objetos. Sin embargo, hay una función denominada *merge*, que permite juntar dos objetos en base a un vector que sirve como variable de identificación de casos. Veremos más detalles de la función en las siguientes clases pero, por ahora, veamos su forma más básica:

```
total = merge(pob_reg, ppk_reg, by = "depa")
```

Sacamos la proporción de voto por PPK:

```
total$prop_ppk = (total$ppk_reg/total$pob_reg)*100
```

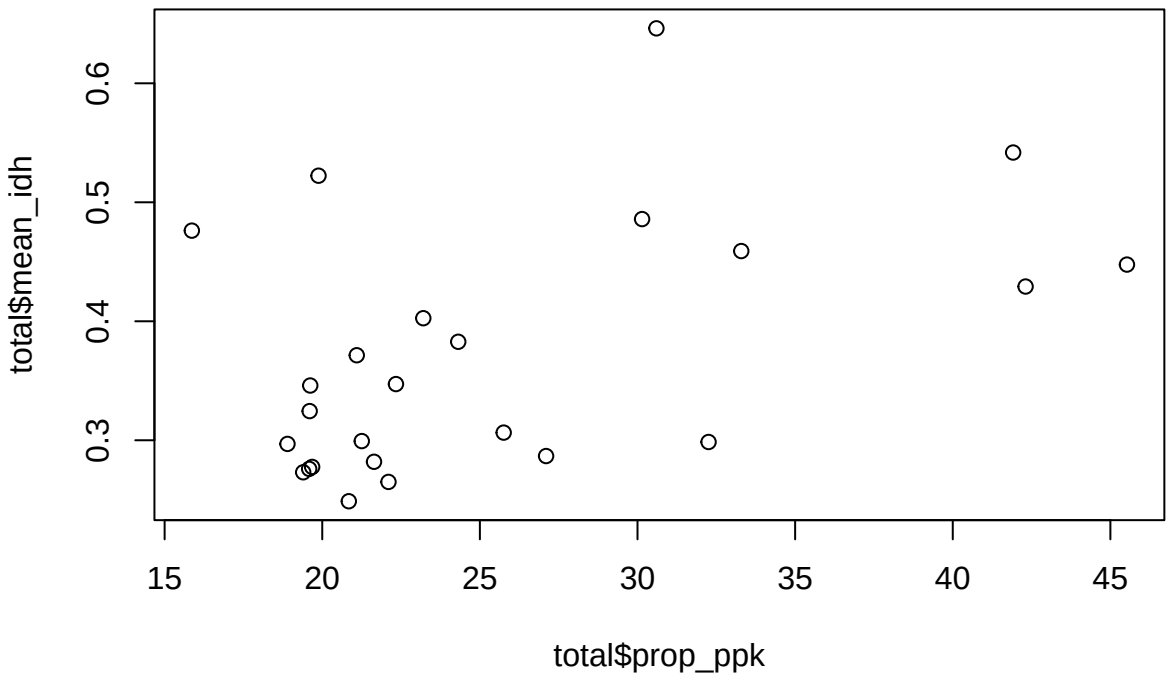
Ahora agregamos el IDH por departamento:

```
idh_reg = data_idh %>%
  group_by(depa) %>%
  summarize(mean_idh = mean(idh, na.rm = TRUE))
```

```
total = merge(total,idh_reg, by = "depa")
```

Exploremos visualmente la asociación entre IDH y proporción de voto por PPK: ¿Qué opinan?

```
plot(total$prop_ppk, total$mean_idh)
```



Finalmente, otra forma súper útil de agregar datos, es cuando **cada caso de nuestra base de datos es de interés**. Es decir, cuando queremos agruparlos en función al número de ocurrencias. Por ejemplo, si queremos mostrar el número de distritos de cada región, podría utilizar esto:

```
dis_reg = data_idh %>%
  group_by(depa) %>%
  summarise(count = n())

dis_reg
```

```
## # A tibble: 25 x 2
##    depa         count
##    <chr>        <int>
##  1 AMAZONAS        84
##  2 ANCASH         166
##  3 APURIMAC        80
##  4 AREQUIPA       109
##  5 AYACUCHO       111
##  6 CAJAMARCA      127
##  7 CALLAO           6
##  8 CUSCO          108
##  9 HUANCAVELICA    94
## 10 HUANUCO         76
## # ... with 15 more rows
```

```r
sum(dis_reg$count) #verificamos
```

```
## [1] 1834
```

Exporte la data de este link: https://github.com/appliedepi/epirhandbook/blob/master/inst/extdata/linelist_cleaned.xlsx

1. Otorgue un formato adecuado para las variables que tienen fechas. Encuentre la diferencia en días entre la fecha de infección del paciente (date_infection) y la fecha de hospitalización (date_hospitalisation). Describa cuáles fueron los síntomas del paciente o pacientes con menor y mayor diferencia de tiempo entre ambas fechas.
2. Cree un vector que marque las 18:00 y encuentre la diferencia en minutos y segundos con respecto a la variable *time_admission*.

```r
data=import("https://github.com/appliedepi/epirhandbook/blob/master/inst/extdata/linelist_cleaned.xlsx?
```