HarvardX PH125.6x

Mauricio Matoma

Data Science: Wrangling

Section 1: Data import

1.1. Data Import

Paths and the Working Directory

```
library(dslabs)
```

Para conocer el directorio de trabajo, se utiliza la función getwd() (funciona de manera similar a pwd de Unix).

```
getwd()
```

```
[1] "D:/Documentos/R/HarvardX_PH125/PH125.6x/HarvardX_PH125.6x"
```

Se cambia la dirección de trabajo con setwd(). Para este caso, se utilizarán datos dummy del paquete dslabs

```
path  <-  system.file("extdata", package = "dslabs")
list.files(path)
```

```
 [1] "2010_bigfive_regents.xls"
 [2] "calificaciones.csv"
 [3] "carbon_emissions.csv"
 [4] "fertility-two-countries-example.csv"
 [5] "HRlist2.txt"
 [6] "life-expectancy-and-fertility-two-countries-example.csv"
```

```
 [7] "murders.csv"
 [8] "olive.csv"
 [9] "RD-Mortality-Report_2015-18-180531.pdf"
[10] "ssa-death-probability.csv"
```

Extraer la ruta completa de un archivo a través de file.path()

```
filename <- "murders.csv"
fullpath <- file.path(path,filename)
fullpath
```

```
[1] "C:/Users/matom/AppData/Local/R/win-library/4.3/dslabs/extdata/murders.csv"
```

Copiar el archivo dentro del proyecto

```
file.copy(fullpath, getwd())
```

```
[1] FALSE
```

```
file.exists(filename)
```

```
[1] TRUE
```

The readr and readxl Packages

La función read_lines() permite revisar los encabezados de un archivo, por ejemplo:

```
library(readr)
read_lines("murders.csv", n_max=3)
```

```
[1] "state,abb,region,population,total" "Alabama,AL,South,4779736,135"
[3] "Alaska,AK,West,710231,19"
```

Leer el archivo:

```
dat <- read.csv("murders.csv")
head(dat)
```

```
       state abb region population total
1    Alabama  AL  South    4779736   135
2     Alaska  AK   West     710231    19
3    Arizona  AZ   West    6392017   232
4   Arkansas  AR  South    2915918    93
5 California  CA   West   37253956  1257
6   Colorado  CO   West    5029196    65
```

Leer datos con función de base R

```
dat2 <- read.csv(filename)
head(dat2)
```

```
       state abb region population total
1    Alabama  AL  South    4779736   135
2     Alaska  AK   West     710231    19
3    Arizona  AZ   West    6392017   232
4   Arkansas  AR  South    2915918    93
5 California  CA   West   37253956  1257
6   Colorado  CO   West    5029196    65
```

```
class(dat2)
```

```
[1] "data.frame"
```

Downloading Files from the Internet

La función read_csv() importa directamente los archivos a partir de una url

```
url <- "https://raw.githubusercontent.com/rafalab/dslabs/master/inst/extdata/murders.csv"
dat <- read.csv(url)
```

Para tener un archivo local, se descarga el archivo

```
download.file(url, "murders.csv")
```

Assessment Part 2: Data Import

```
url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdb
data <- read_csv(url, col_names = FALSE)
head(data)
```

```
# A tibble: 6 x 32
       X1 X2       X3    X4    X5    X6     X7     X8     X9    X10   X11    X12
    <dbl> <chr> <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl> <dbl>  <dbl>
1  8.42e5 M      18.0  10.4 123.  1001  0.118  0.278  0.300  0.147  0.242 0.0787
2  8.43e5 M      20.6  17.8 133.  1326  0.0847 0.0786 0.0869 0.0702 0.181 0.0567
3  8.43e7 M      19.7  21.2 130   1203  0.110  0.160  0.197  0.128  0.207 0.0600
4  8.43e7 M      11.4  20.4  77.6  386. 0.142  0.284  0.241  0.105  0.260 0.0974
5  8.44e7 M      20.3  14.3 135.  1297  0.100  0.133  0.198  0.104  0.181 0.0588
6  8.44e5 M      12.4  15.7  82.6  477. 0.128  0.17   0.158  0.0809 0.209 0.0761
# i 20 more variables: X13 <dbl>, X14 <dbl>, X15 <dbl>, X16 <dbl>, X17 <dbl>,
#   X18 <dbl>, X19 <dbl>, X20 <dbl>, X21 <dbl>, X22 <dbl>, X23 <dbl>,
#   X24 <dbl>, X25 <dbl>, X26 <dbl>, X27 <dbl>, X28 <dbl>, X29 <dbl>,
#   X30 <dbl>, X31 <dbl>, X32 <dbl>
```

```
nrow(data)
```

```
[1] 569
```

```
ncol(data)
```

```
[1] 32
```

Section 2: Tidy Data

2.1. Reshaping Data

Tidy data

```
library(dslabs)
library(tidyverse)
path <- system.file("extdata", package = "dslabs")
filename <- file.path(path,"fertility-two-countries-example.csv")
wide_data <- read_csv(filename)
#View(wide_data)
```

```r
select(wide_data, country, "1960":"1967")
```

```
# A tibble: 2 x 9
  country     `1960` `1961` `1962` `1963` `1964` `1965` `1966` `1967`
  <chr>        <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
1 Germany       2.41   2.44   2.47   2.49   2.49   2.48   2.44   2.37
2 South Korea   6.16   5.99   5.79   5.57   5.36   5.16   4.99   4.85
```

Reshaping data: pivot_longer

pivot longer utiliza dos argumentos. Primero, el data frame que será modificado y el segundo argumento contiene las columnas de los valores a mover.

Por ejemplo, se modificará de esta forma:

```r
select(wide_data, country, "1960":"1967")
```

```
# A tibble: 2 x 9
  country     `1960` `1961` `1962` `1963` `1964` `1965` `1966` `1967`
  <chr>        <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
1 Germany       2.41   2.44   2.47   2.49   2.49   2.48   2.44   2.37
2 South Korea   6.16   5.99   5.79   5.57   5.36   5.16   4.99   4.85
```

A esta forma (nombre de las columnas a niveles de una variable en una sola columna):

```r
wide_data %>% pivot_longer("1960":"2015")
```

```
# A tibble: 112 x 3
   country name  value
   <chr>   <chr> <dbl>
 1 Germany 1960   2.41
 2 Germany 1961   2.44
 3 Germany 1962   2.47
 4 Germany 1963   2.49
 5 Germany 1964   2.49
 6 Germany 1965   2.48
 7 Germany 1966   2.44
 8 Germany 1967   2.37
 9 Germany 1968   2.28
10 Germany 1969   2.17
# i 102 more rows
```

Nótese que la información se modifica a nombres (years) y values (fertility). Los podemos modificar de la siguiente forma:

```
new_tidy_data <- wide_data %>%
  pivot_longer("1960":"2015", names_to = "year", values_to = "fertility")
head(new_tidy_data)
```

```
# A tibble: 6 x 3
  country year  fertility
  <chr>   <chr>     <dbl>
1 Germany 1960       2.41
2 Germany 1961       2.44
3 Germany 1962       2.47
4 Germany 1963       2.49
5 Germany 1964       2.49
6 Germany 1965       2.48
```

Transformar la variable que aparece como caracter de la forma

```
new_tidy_data <- wide_data %>%
  pivot_longer("1960":"2015", names_to = "year", values_to = "fertility") %>% mutate(year=as
head(new_tidy_data)
```

```
# A tibble: 6 x 3
  country  year fertility
  <chr>   <dbl>     <dbl>
1 Germany  1960      2.41
2 Germany  1961      2.44
3 Germany  1962      2.47
4 Germany  1963      2.49
5 Germany  1964      2.49
6 Germany  1965      2.48
```

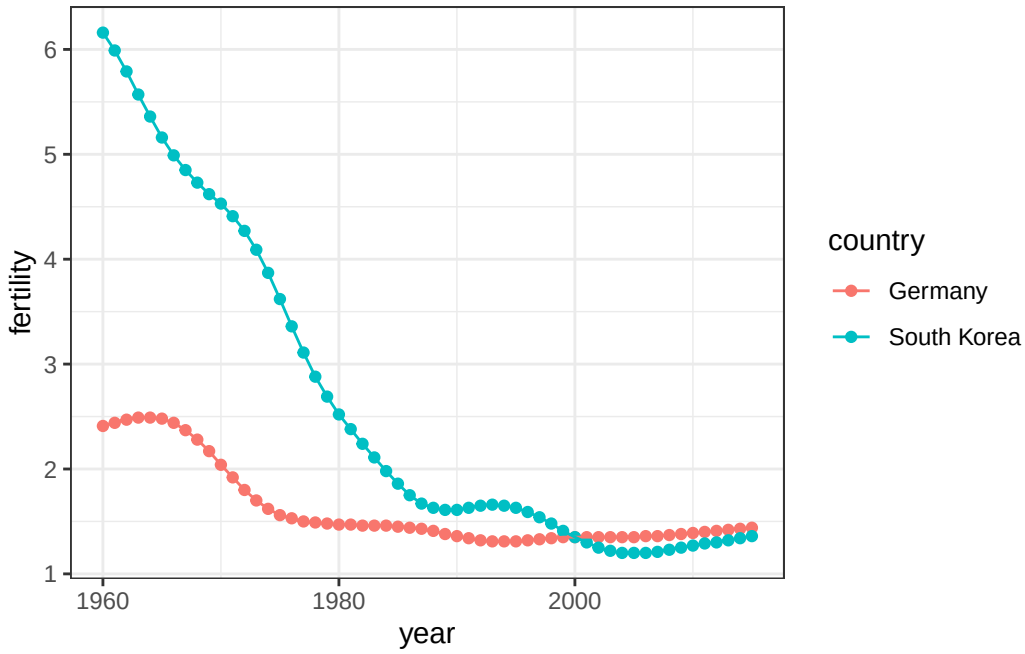
O de la forma:

```
new_tidy_data <- wide_data %>%
  pivot_longer("1960":"2015", names_to = "year", values_to = "fertility",
               names_transform = list(year=as.numeric))
head(new_tidy_data)
```

```
# A tibble: 6 x 3
  country  year fertility
  <chr>   <dbl>     <dbl>
1 Germany  1960      2.41
2 Germany  1961      2.44
3 Germany  1962      2.47
4 Germany  1963      2.49
5 Germany  1964      2.49
6 Germany  1965      2.48
```

Finalmente, plotear:

```
new_tidy_data %>% ggplot(aes(year,fertility, color=country)) +
  geom_point() + geom_line() + theme_bw()
```




```
ggsave("images/plot.png")
```

Reshaping data: pivot wider

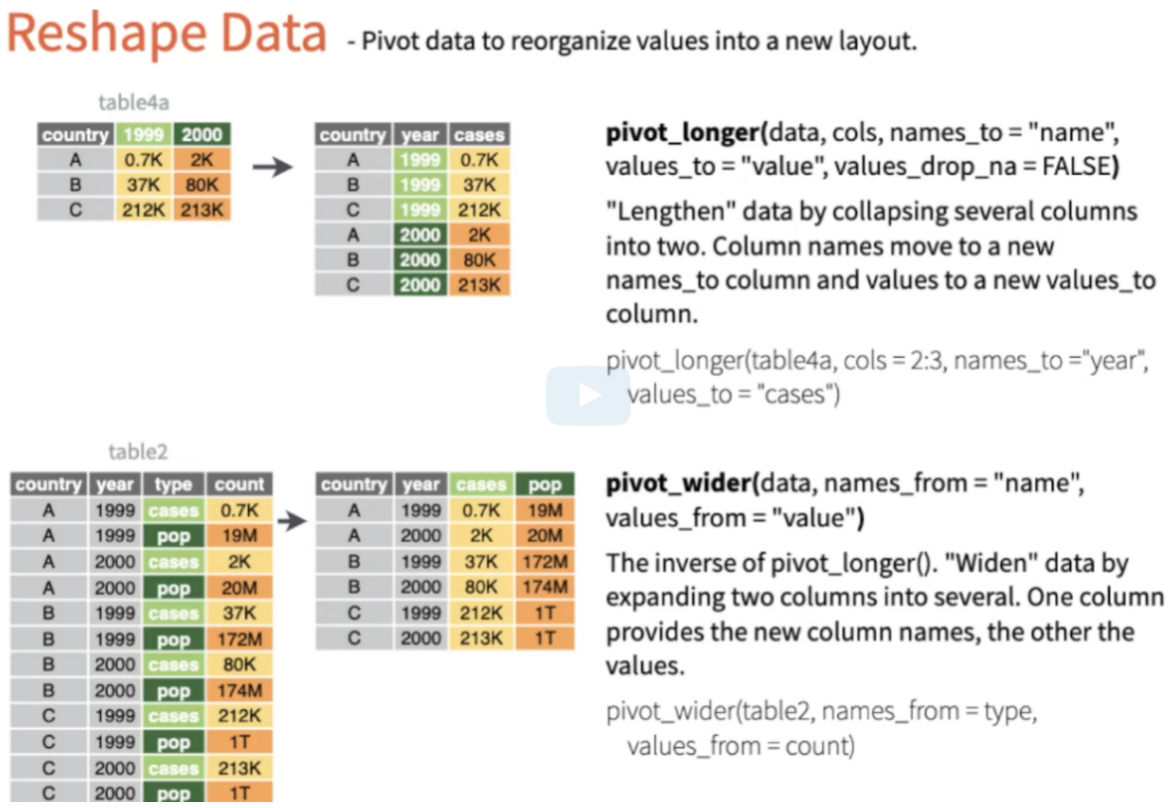
Función inversa de pivot_longer

names_from : Contiene los niveles de variable o filas que serán transformadas en columnas

values_from: Contiene los valores deseados

```
new_wide_data <- new_tidy_data %>%
  pivot_wider(names_from = year, values_from = fertility)
select(new_wide_data, country, "1960":"1967")
```

```
# A tibble: 2 x 9
  country     `1960` `1961` `1962` `1963` `1964` `1965` `1966` `1967`
  <chr>        <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
1 Germany       2.41   2.44   2.47   2.49   2.49   2.48   2.44   2.37
2 South Korea   6.16   5.99   5.79   5.57   5.36   5.16   4.99   4.85
```



Separate

```
path <- system.file("extdata", package="dslabs")
fname <- "life-expectancy-and-fertility-two-countries-example.csv"
filename <- file.path(path,fname)
```

```r
raw_dat <- read_csv(filename)
#View(raw_dat)
select(raw_dat, 1:4)
```

```
# A tibble: 2 x 4
  country     `1960_fertility` `1960_life_expectancy` `1961_fertility`
  <chr>                  <dbl>                  <dbl>            <dbl>
1 Germany                 2.41                   69.3             2.44
2 South Korea             6.16                   53.0             5.99
```

```r
dat <- raw_dat %>% pivot_longer(-country)
head(dat)
```

```
# A tibble: 6 x 3
  country name                value
  <chr>   <chr>               <dbl>
1 Germany 1960_fertility       2.41
2 Germany 1960_life_expectancy 69.3
3 Germany 1961_fertility       2.44
4 Germany 1961_life_expectancy 69.8
5 Germany 1962_fertility       2.47
6 Germany 1962_life_expectancy 70.0
```

La función separate() toma tres argumentos. El primero es el nombre de la columna que será separada. El segundo el nombre para las nuevas columnas y el tercero será el caracter que separa las variables

```r
dat %>% separate(name, c("year", "name"), sep = "_")
```

```
Warning: Expected 2 pieces. Additional pieces discarded in 112 rows [2, 4, 6, 8, 10, 12,
14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, ...].
```

```
# A tibble: 224 x 4
   country year  name      value
   <chr>   <chr> <chr>     <dbl>
 1 Germany 1960  fertility  2.41
 2 Germany 1960  life      69.3
 3 Germany 1961  fertility  2.44
 4 Germany 1961  life      69.8
 5 Germany 1962  fertility  2.47
```

```
 6 Germany 1962  life      70.0
 7 Germany 1963  fertility  2.49
 8 Germany 1963  life      70.1
 9 Germany 1964  fertility  2.49
10 Germany 1964  life      70.7
# i 214 more rows
```

Con el argumento convert = TRUE, se transforma automáticamente el año en valor numérico

```
dat %>% separate(name, c("year", "name"), convert = TRUE)
```

```
Warning: Expected 2 pieces. Additional pieces discarded in 112 rows [2, 4, 6, 8, 10, 12,
14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, ...].
```

```
# A tibble: 224 x 4
   country  year name      value
   <chr>   <int> <chr>     <dbl>
 1 Germany  1960 fertility  2.41
 2 Germany  1960 life      69.3
 3 Germany  1961 fertility  2.44
 4 Germany  1961 life      69.8
 5 Germany  1962 fertility  2.47
 6 Germany  1962 life      70.0
 7 Germany  1963 fertility  2.49
 8 Germany  1963 life      70.1
 9 Germany  1964 fertility  2.49
10 Germany  1964 life      70.7
# i 214 more rows
```

Se puede agregar otra columna para el "expectancy. Se llenarán algunos valores nulos automáticamente (espacios fertility)

```
dat %>% separate(name, c("year", "name1", "name2"),
                 fill = "right" ,
                 convert = TRUE)
```

```
# A tibble: 224 x 5
   country  year name1     name2      value
   <chr>   <int> <chr>     <chr>      <dbl>
 1 Germany  1960 fertility <NA>        2.41
```

```
 2 Germany  1960 life      expectancy 69.3
 3 Germany  1961 fertility <NA>        2.44
 4 Germany  1961 life      expectancy 69.8
 5 Germany  1962 fertility <NA>        2.47
 6 Germany  1962 life      expectancy 70.0
 7 Germany  1963 fertility <NA>        2.49
 8 Germany  1963 life      expectancy 70.1
 9 Germany  1964 fertility <NA>        2.49
10 Germany  1964 life      expectancy 70.7
# i 214 more rows
```

Sin embargo, es un mejor enfoque unir los nombres extra que se generen a partir del argumento extra= "merge"

```
dat %>%  separate(name, c("year", "name"), sep="_",
                  extra = "merge", convert = TRUE)
```

```
# A tibble: 224 x 4
   country  year name            value
   <chr>   <int> <chr>           <dbl>
 1 Germany  1960 fertility        2.41
 2 Germany  1960 life_expectancy 69.3
 3 Germany  1961 fertility        2.44
 4 Germany  1961 life_expectancy 69.8
 5 Germany  1962 fertility        2.47
 6 Germany  1962 life_expectancy 70.0
 7 Germany  1963 fertility        2.49
 8 Germany  1963 life_expectancy 70.1
 9 Germany  1964 fertility        2.49
10 Germany  1964 life_expectancy 70.7
# i 214 more rows
```

Finalmente, se separan los datos de fertility y life_expectancy en columnas

```
dat %>% separate(name, c("year", "name"), sep = "_",
                 extra="merge", convert = TRUE) %>%
  pivot_wider()
```

```
# A tibble: 112 x 4
   country  year fertility life_expectancy
   <chr>   <int>     <dbl>           <dbl>
```

```
 1 Germany  1960      2.41            69.3
 2 Germany  1961      2.44            69.8
 3 Germany  1962      2.47            70.0
 4 Germany  1963      2.49            70.1
 5 Germany  1964      2.49            70.7
 6 Germany  1965      2.48            70.6
 7 Germany  1966      2.44            70.8
 8 Germany  1967      2.37            71.0
 9 Germany  1968      2.28            70.6
10 Germany  1969      2.17            70.5
# i 102 more rows
```

Unite

Función inversa de separate()

```
dat %>%
  separate(name, c("year", "name1", "name2"),
           fill="right", convert = TRUE) %>%
  unite(variable_name, name1, name2)
```

```
# A tibble: 224 x 4
   country  year variable_name   value
   <chr>   <int> <chr>           <dbl>
 1 Germany  1960 fertility_NA     2.41
 2 Germany  1960 life_expectancy 69.3
 3 Germany  1961 fertility_NA     2.44
 4 Germany  1961 life_expectancy 69.8
 5 Germany  1962 fertility_NA     2.47
 6 Germany  1962 life_expectancy 70.0
 7 Germany  1963 fertility_NA     2.49
 8 Germany  1963 life_expectancy 70.1
 9 Germany  1964 fertility_NA     2.49
10 Germany  1964 life_expectancy 70.7
# i 214 more rows
```

Y finalmente, dispersar las columnas

```
dat %>%
  separate(name, c("year", "name1", "name2"),
           fill="right", convert = TRUE) %>% #Separar
```

```
  unite(name, name1, name2) %>% #Unir como un solo nombre
  spread(name, value) %>% #Spread() sirve como pivot_wide()
  rename(fertility = fertility_NA)
```

```
# A tibble: 112 x 4
   country  year fertility life_expectancy
   <chr>   <int>     <dbl>           <dbl>
 1 Germany  1960      2.41            69.3
 2 Germany  1961      2.44            69.8
 3 Germany  1962      2.47            70.0
 4 Germany  1963      2.49            70.1
 5 Germany  1964      2.49            70.7
 6 Germany  1965      2.48            70.6
 7 Germany  1966      2.44            70.8
 8 Germany  1967      2.37            71.0
 9 Germany  1968      2.28            70.6
10 Germany  1969      2.17            70.5
# i 102 more rows
```

Guardar como RDS

```
data_final <- dat %>%
  separate(name, c("year", "name1", "name2"),
           fill="right", convert = TRUE) %>% #Separar
  unite(name, name1, name2) %>% #Unir como un solo nombre
  spread(name, value) %>% #Spread() sirve como pivot_wide()
  rename(fertility = fertility_NA)
saveRDS(data_final, "data/data_final.rds")
```

Assessment Part 2: Reshaping Data

```
library(tidyverse)
library(dslabs)
```

```
co2
```

```
        Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct
1959 315.42 316.31 316.50 317.56 318.13 318.00 316.39 314.65 313.68 313.18
1960 316.27 316.81 317.42 318.87 319.87 319.43 318.01 315.74 314.00 313.68
```

```
1961 316.73 317.54 318.38 319.31 320.42 319.61 318.42 316.63 314.83 315.16
1962 317.78 318.40 319.53 320.42 320.85 320.45 319.45 317.25 316.11 315.27
1963 318.58 318.92 319.70 321.22 322.08 321.31 319.58 317.61 316.05 315.83
1964 319.41 320.07 320.74 321.40 322.06 321.73 320.27 318.54 316.54 316.71
1965 319.27 320.28 320.73 321.97 322.00 321.71 321.05 318.71 317.66 317.14
1966 320.46 321.43 322.23 323.54 323.91 323.59 322.24 320.20 318.48 317.94
1967 322.17 322.34 322.88 324.25 324.83 323.93 322.38 320.76 319.10 319.24
1968 322.40 322.99 323.73 324.86 325.40 325.20 323.98 321.95 320.18 320.09
1969 323.83 324.26 325.47 326.50 327.21 326.54 325.72 323.50 322.22 321.62
1970 324.89 325.82 326.77 327.97 327.91 327.50 326.18 324.53 322.93 322.90
1971 326.01 326.51 327.01 327.62 328.76 328.40 327.20 325.27 323.20 323.40
1972 326.60 327.47 327.58 329.56 329.90 328.92 327.88 326.16 324.68 325.04
1973 328.37 329.40 330.14 331.33 332.31 331.90 330.70 329.15 327.35 327.02
1974 329.18 330.55 331.32 332.48 332.92 332.08 331.01 329.23 327.27 327.21
1975 330.23 331.25 331.87 333.14 333.80 333.43 331.73 329.90 328.40 328.17
1976 331.58 332.39 333.33 334.41 334.71 334.17 332.89 330.77 329.14 328.78
1977 332.75 333.24 334.53 335.90 336.57 336.10 334.76 332.59 331.42 330.98
1978 334.80 335.22 336.47 337.59 337.84 337.72 336.37 334.51 332.60 332.38
1979 336.05 336.59 337.79 338.71 339.30 339.12 337.56 335.92 333.75 333.70
1980 337.84 338.19 339.91 340.60 341.29 341.00 339.39 337.43 335.72 335.84
1981 339.06 340.30 341.21 342.33 342.74 342.08 340.32 338.26 336.52 336.68
1982 340.57 341.44 342.53 343.39 343.96 343.18 341.88 339.65 337.81 337.69
1983 341.20 342.35 342.93 344.77 345.58 345.14 343.81 342.21 339.69 339.82
1984 343.52 344.33 345.11 346.88 347.25 346.62 345.22 343.11 340.90 341.18
1985 344.79 345.82 347.25 348.17 348.74 348.07 346.38 344.51 342.92 342.62
1986 346.11 346.78 347.68 349.37 350.03 349.37 347.76 345.73 344.68 343.99
1987 347.84 348.29 349.23 350.80 351.66 351.07 349.33 347.92 346.27 346.18
1988 350.25 351.54 352.05 353.41 354.04 353.62 352.22 350.27 348.55 348.72
1989 352.60 352.92 353.53 355.26 355.52 354.97 353.75 351.52 349.64 349.83
1990 353.50 354.55 355.23 356.04 357.00 356.07 354.67 352.76 350.82 351.04
1991 354.59 355.63 357.03 358.48 359.22 358.12 356.06 353.92 352.05 352.11
1992 355.88 356.63 357.72 359.07 359.58 359.17 356.94 354.92 352.94 353.23
1993 356.63 357.10 358.32 359.41 360.23 359.55 357.53 355.48 353.67 353.95
1994 358.34 358.89 359.95 361.25 361.67 360.94 359.55 357.49 355.84 356.00
1995 359.98 361.03 361.66 363.48 363.82 363.30 361.94 359.50 358.11 357.80
1996 362.09 363.29 364.06 364.76 365.45 365.01 363.70 361.54 359.51 359.65
1997 363.23 364.06 364.61 366.40 366.84 365.68 364.52 362.57 360.24 360.83
        Nov    Dec
1959 314.66 315.43
1960 314.84 316.03
1961 315.94 316.85
1962 316.53 317.53
1963 316.91 318.20
```

```
1964 317.53 318.55
1965 318.70 319.25
1966 319.63 320.87
1967 320.56 321.80
1968 321.16 322.74
1969 322.69 323.95
1970 323.85 324.96
1971 324.63 325.85
1972 326.34 327.39
1973 327.99 328.48
1974 328.29 329.41
1975 329.32 330.59
1976 330.14 331.52
1977 332.24 333.68
1978 333.75 334.78
1979 335.12 336.56
1980 336.93 338.04
1981 338.19 339.44
1982 339.09 340.32
1983 340.98 342.82
1984 342.80 344.04
1985 344.06 345.38
1986 345.48 346.72
1987 347.64 348.78
1988 349.91 351.18
1989 351.14 352.37
1990 352.69 354.07
1991 353.64 354.89
1992 354.09 355.33
1993 355.30 356.78
1994 357.59 359.05
1995 359.61 360.74
1996 360.80 362.38
1997 362.49 364.34
```

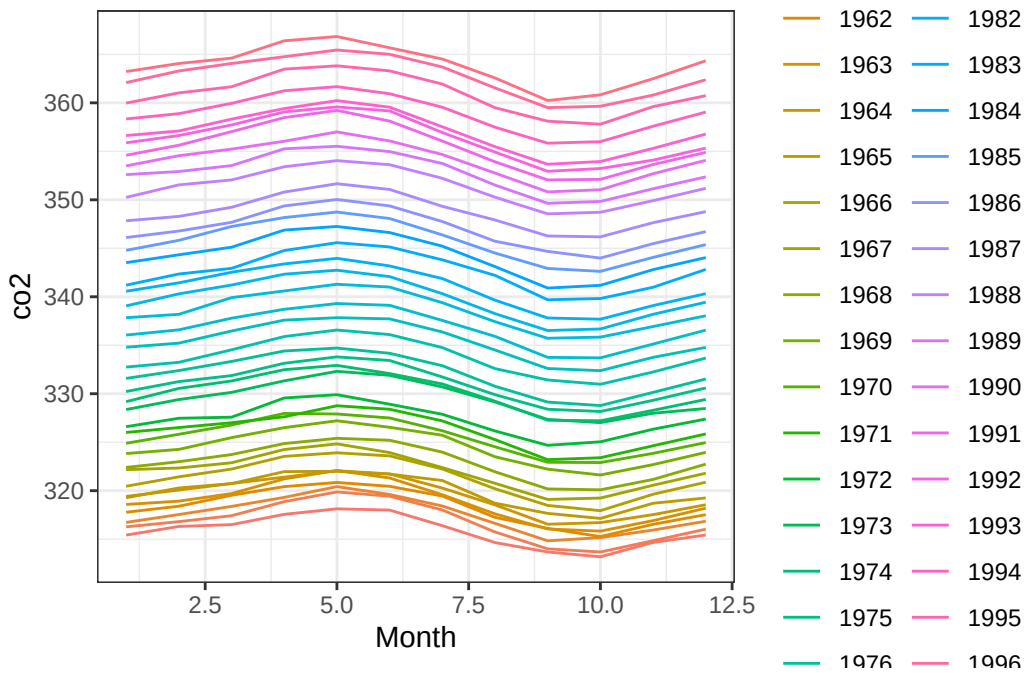
```
#View(co2)
head(co2)
```

```
[1] 315.42 316.31 316.50 317.56 318.13 318.00
```

```
co2_wide <- data.frame(matrix(co2, ncol = 12, byrow = TRUE)) %>%
    setNames(1:12) %>%
    mutate(year = as.character(1959:1997))
#View(co2_wide)
```

```
co2_tidy <- pivot_longer(co2_wide, -year, names_to = "month", values_to = "co2")
#View(co2_tidy)
```

```
co2_tidy %>% ggplot(aes(as.numeric(month), co2, color = year)) + geom_line() + theme_bw() + 
```




```
ggsave("images/graf2.png")
```

```
library(dslabs)
data("admissions")
dat <- admissions %>% select(-applicants)
#View(dat)
```

```
dat_tidy <- pivot_wider(dat, names_from = gender, values_from = admitted)
dat_tidy
```

```
# A tibble: 6 x 3
```

```
  major   men women
  <chr> <dbl> <dbl>
1 A        62    82
2 B        63    68
3 C        37    34
4 D        33    35
5 E        28    24
6 F         6     7
```

```
tmp <- admissions %>%
  pivot_longer(cols = c(admitted, applicants), names_to = "key", values_to = "value")
tmp
```

```
# A tibble: 24 x 4
   major gender key        value
   <chr> <chr>  <chr>      <dbl>
 1 A     men    admitted      62
 2 A     men    applicants   825
 3 B     men    admitted      63
 4 B     men    applicants   560
 5 C     men    admitted      37
 6 C     men    applicants   325
 7 D     men    admitted      33
 8 D     men    applicants   417
 9 E     men    admitted      28
10 E     men    applicants   191
# i 14 more rows
```

```
tmp2 <- unite(tmp, column_name, c(key, gender))
tmp2
```

```
# A tibble: 24 x 3
   major column_name    value
   <chr> <chr>          <dbl>
 1 A     admitted_men      62
 2 A     applicants_men   825
 3 B     admitted_men      63
 4 B     applicants_men   560
 5 C     admitted_men      37
 6 C     applicants_men   325
 7 D     admitted_men      33
```

```
 8 D     applicants_men   417
 9 E     admitted_men      28
10 E     applicants_men   191
# i 14 more rows
```

```
tmp2 <- unite(tmp, column_name, c(key, gender))
tmp2
```

```
# A tibble: 24 x 3
   major column_name    value
   <chr> <chr>          <dbl>
 1 A     admitted_men      62
 2 A     applicants_men   825
 3 B     admitted_men      63
 4 B     applicants_men   560
 5 C     admitted_men      37
 6 C     applicants_men   325
 7 D     admitted_men      33
 8 D     applicants_men   417
 9 E     admitted_men      28
10 E     applicants_men   191
# i 14 more rows
```

Section 2: Tidy Data

2.2. Combining Tables

Combining Tables

```
library(dslabs)
library(tidyverse)
library(ggrepel)
data(murders)
#View(murders)
data(results_us_election_2016)
#View(results_us_election_2016)
#Ordenar datos por orden alfabético de state a través de arrange
results_us_election_2016 <- results_us_election_2016 %>% arrange(state)
```

Por medio de la función left_join, es posible unir tablas por una variable en común. En este caso, la variable state:

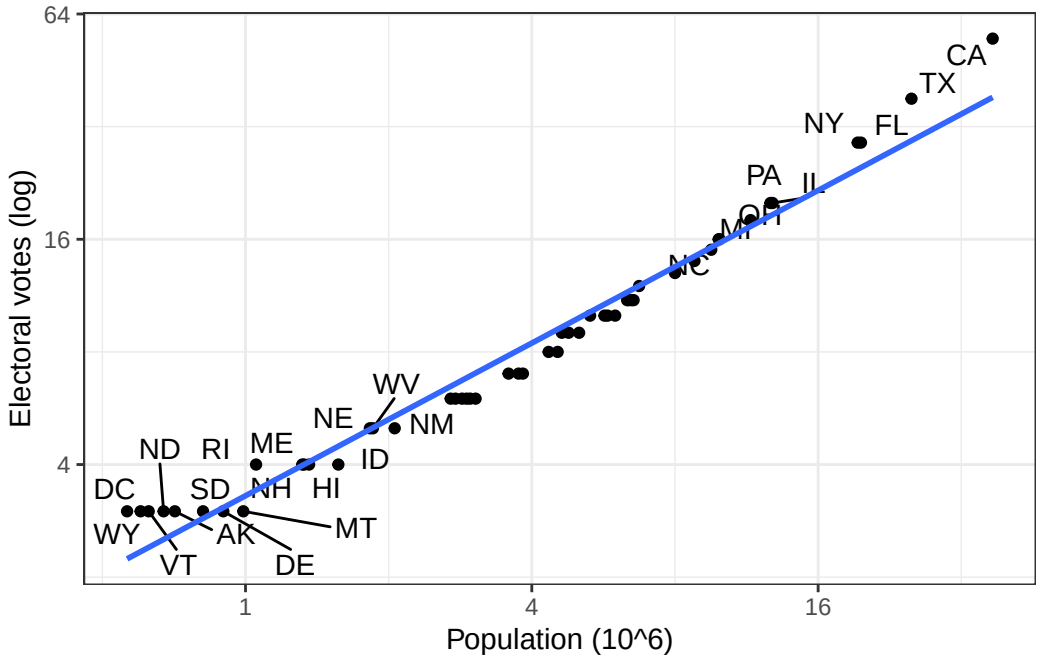
```r
tab <- left_join(murders, results_us_election_2016, by="state")
head(tab)
```

```
       state abb region population total electoral_votes clinton trump others
1    Alabama  AL  South    4779736   135               9    34.4  62.1    3.6
2     Alaska  AK   West     710231    19               3    36.6  51.3   12.2
3    Arizona  AZ   West    6392017   232              11    45.1  48.7    6.2
4   Arkansas  AR  South    2915918    93               6    33.7  60.6    5.8
5 California  CA   West   37253956  1257              55    61.7  31.6    6.7
6   Colorado  CO   West    5029196    65               9    48.2  43.3    8.6
```

```r
#View(tab)
```

Graficar

```r
tab %>% ggplot(aes(population/10^6, electoral_votes, label=abb)) +
  geom_point()+
  geom_text_repel()+
  scale_x_continuous(trans = "log2") +
  scale_y_continuous(trans= "log2") +
  geom_smooth(method = "lm", se=FALSE) +
  xlab("Population (10^6)")+
  ylab("Electoral votes (log)")+
  theme_bw()
```




```
ggsave("images/plot2.png")
```

Otro ejemplo

```
tab1 <- slice(murders, 1:6) %>% select(state,population)
tab1
```

```
       state population
1    Alabama    4779736
2     Alaska     710231
3    Arizona    6392017
4   Arkansas    2915918
5 California   37253956
6   Colorado    5029196
```

```
tab2 <- slice(results_us_election_2016, c(1:3, 5, 7:8)) %>%
  select(state,electoral_votes)
tab2
```

```
        state electoral_votes
1     Alabama               9
```

```
2      Alaska               3
3     Arizona              11
4  California              55
5 Connecticut               7
6    Delaware               3
```

```
left_join(tab1, tab2)
```

```
       state population electoral_votes
1    Alabama    4779736               9
2     Alaska     710231               3
3    Arizona    6392017              11
4   Arkansas    2915918              NA
5 California   37253956              55
6   Colorado    5029196              NA
```

Otras formas de unir:

```
tab1 %>% left_join(tab2)
```

```
       state population electoral_votes
1    Alabama    4779736               9
2     Alaska     710231               3
3    Arizona    6392017              11
4   Arkansas    2915918              NA
5 California   37253956              55
6   Colorado    5029196              NA
```

```
tab1 %>% right_join(tab2)
```

```
        state population electoral_votes
1     Alabama    4779736               9
2      Alaska     710231               3
3     Arizona    6392017              11
4  California   37253956              55
5 Connecticut         NA               7
6    Delaware         NA               3
```

Si quiero conservar sólo las filas que tienen información en ambas tablas, utilizo inner_join()

```
inner_join(tab1, tab2)
```

```
       state population electoral_votes
1    Alabama    4779736              9
2     Alaska     710231              3
3    Arizona    6392017             11
4 California   37253956             55
```

Si quiero conservar todas las filas, aunque tengan valores nulos, utilizo full_join()

```
full_join(tab1,tab2)
```

```
        state population electoral_votes
1     Alabama    4779736              9
2      Alaska     710231              3
3     Arizona    6392017             11
4    Arkansas    2915918             NA
5  California   37253956             55
6    Colorado    5029196             NA
7 Connecticut         NA              7
8    Delaware         NA              3
```

La función semi_join() conserva la información de la tabla 1 que también está en la tabla 2

```
semi_join(tab1, tab2)
```

```
       state population
1    Alabama    4779736
2     Alaska     710231
3    Arizona    6392017
4 California   37253956
```

La función anti_join() conserva la información de la tabla 1 que no está en la tabla 2. Es una función inversa de semi_join()

```
anti_join(tab1,tab2)
```

```
     state population
1 Arkansas    2915918
2 Colorado    5029196
```

Binding

La función bind_cols() combina tablas por columnas sin importar las variables en común

```
bind_cols(a=1:3,b=4:6)
```

```
# A tibble: 3 x 2
      a     b
  <int> <int>
1     1     4
2     2     5
3     3     6
```

También sirve para concatenar data frames

```
tab1 <- tab[,1:3]
tab2 <- tab[,4:6]
tab3 <- tab[,7:9]
new_tab <- bind_cols(tab1,tab2,tab3)
head(new_tab)
```

```
       state abb region population total electoral_votes clinton trump others
1    Alabama  AL  South    4779736   135               9    34.4  62.1    3.6
2     Alaska  AK   West     710231    19               3    36.6  51.3   12.2
3    Arizona  AZ   West    6392017   232              11    45.1  48.7    6.2
4   Arkansas  AR  South    2915918    93               6    33.7  60.6    5.8
5 California  CA   West   37253956  1257              55    61.7  31.6    6.7
6   Colorado  CO   West    5029196    65               9    48.2  43.3    8.6
```

bind_rows() funciona de manera similar, pero concatenando filas

```
tab1 <- tab[1:2,]
tab2 <- tab[3:4,]
bind_rows(tab1, tab2)
```

```
     state abb region population total electoral_votes clinton trump others
1  Alabama  AL  South    4779736   135               9    34.4  62.1    3.6
2   Alaska  AK   West     710231    19               3    36.6  51.3   12.2
3  Arizona  AZ   West    6392017   232              11    45.1  48.7    6.2
4 Arkansas  AR  South    2915918    93               6    33.7  60.6    5.8
```

Set operators

Intersect() aplica para vectores y para data frames

```
intersect(1:10, 6:15)
```

```
[1]  6  7  8  9 10
```

```
head(tab,7)
```

```
        state abb    region population total electoral_votes clinton trump
1     Alabama  AL     South    4779736   135               9    34.4  62.1
2      Alaska  AK      West     710231    19               3    36.6  51.3
3     Arizona  AZ      West    6392017   232              11    45.1  48.7
4    Arkansas  AR     South    2915918    93               6    33.7  60.6
5  California  CA      West   37253956  1257              55    61.7  31.6
6    Colorado  CO      West    5029196    65               9    48.2  43.3
7 Connecticut  CT Northeast    3574097    97               7    54.6  40.9
  others
1    3.6
2   12.2
3    6.2
4    5.8
5    6.7
6    8.6
7    4.5
```

```
tab1 <- tab[1:5,]
tab1
```

```
       state abb region population total electoral_votes clinton trump others
1    Alabama  AL  South    4779736   135               9    34.4  62.1    3.6
2     Alaska  AK   West     710231    19               3    36.6  51.3   12.2
3    Arizona  AZ   West    6392017   232              11    45.1  48.7    6.2
4   Arkansas  AR  South    2915918    93               6    33.7  60.6    5.8
5 California  CA   West   37253956  1257              55    61.7  31.6    6.7
```

```
tab2 <- tab[3:7,]
tab2
```

```
        state abb    region population total electoral_votes clinton trump
3     Arizona  AZ      West    6392017   232              11    45.1  48.7
4    Arkansas  AR     South    2915918    93               6    33.7  60.6
5  California  CA      West   37253956  1257              55    61.7  31.6
6    Colorado  CO      West    5029196    65               9    48.2  43.3
7 Connecticut  CT Northeast    3574097    97               7    54.6  40.9
  others
3    6.2
4    5.8
5    6.7
6    8.6
7    4.5
```

```
intersect(tab1, tab2)
```

```
       state abb region population total electoral_votes clinton trump others
1    Arizona  AZ   West    6392017   232              11    45.1  48.7    6.2
2   Arkansas  AR  South    2915918    93               6    33.7  60.6    5.8
3 California  CA   West   37253956  1257              55    61.7  31.6    6.7
```

Unión funciona similar para vectores y data frames

```
union(c("a","b","c"), c("b","c","d"))
```

```
[1] "a" "b" "c" "d"
```

```
head(tab,7)
```

```
        state abb    region population total electoral_votes clinton trump
1     Alabama  AL     South    4779736   135               9    34.4  62.1
2      Alaska  AK      West     710231    19               3    36.6  51.3
3     Arizona  AZ      West    6392017   232              11    45.1  48.7
4    Arkansas  AR     South    2915918    93               6    33.7  60.6
5  California  CA      West   37253956  1257              55    61.7  31.6
6    Colorado  CO      West    5029196    65               9    48.2  43.3
7 Connecticut  CT Northeast    3574097    97               7    54.6  40.9
  others
1    3.6
2   12.2
3    6.2
```

```
4    5.8
5    6.7
6    8.6
7    4.5
```

```
tab1 <- tab[1:5,]
tab2 <- tab[3:7,]
union(tab1,tab2)
```

```
        state abb    region population total electoral_votes clinton trump
1     Alabama  AL     South    4779736   135               9    34.4  62.1
2      Alaska  AK      West     710231    19               3    36.6  51.3
3     Arizona  AZ      West    6392017   232              11    45.1  48.7
4    Arkansas  AR     South    2915918    93               6    33.7  60.6
5  California  CA      West   37253956  1257              55    61.7  31.6
6    Colorado  CO      West    5029196    65               9    48.2  43.3
7 Connecticut  CT Northeast    3574097    97               7    54.6  40.9
  others
1    3.6
2   12.2
3    6.2
4    5.8
5    6.7
6    8.6
7    4.5
```

La función setdiff() permite encontrar todas las filas de x que no están en y. Por lo tanto, el orden de los argumentos importa.

```
setdiff(1:10,6:15)
```

```
[1] 1 2 3 4 5
```

```
setdiff(6:15,1:10)
```

```
[1] 11 12 13 14 15
```

```
head(tab,7)
```

```
        state abb    region population total electoral_votes clinton trump
1     Alabama  AL     South    4779736   135               9    34.4  62.1
2      Alaska  AK      West     710231    19               3    36.6  51.3
3     Arizona  AZ      West    6392017   232              11    45.1  48.7
4    Arkansas  AR     South    2915918    93               6    33.7  60.6
5  California  CA      West   37253956  1257              55    61.7  31.6
6    Colorado  CO      West    5029196    65               9    48.2  43.3
7 Connecticut  CT Northeast    3574097    97               7    54.6  40.9
  others
1    3.6
2   12.2
3    6.2
4    5.8
5    6.7
6    8.6
7    4.5
```

```
tab1 <- tab[1:5,]
tab2 <- tab[3:7,]
setdiff(tab1,tab2)
```

```
    state abb region population total electoral_votes clinton trump others
1 Alabama  AL  South    4779736   135               9    34.4  62.1    3.6
2  Alaska  AK   West     710231    19               3    36.6  51.3   12.2
```

Por último, la función setequal() returna TRUE si x y y contienen las mismas filas sin importar el orden:

```
setequal(1:5, 1:6)
```

```
[1] FALSE
```

```
setequal(1:5,5:1)
```

```
[1] TRUE
```

```
setequal(tab1, tab2)
```

```
[1] FALSE
```

Assessment: Combining Tables

```
library(Lahman)
library(writexl)
#View(Batting)
top <- Batting %>%
  filter(yearID==2016) %>%
  arrange(desc(HR)) %>% #arrange by descending HR count
  slice(1:10)  #Take entries 1:10
top %>% as_tibble()
```

```
# A tibble: 10 x 22
   playerID  yearID stint teamID lgID      G    AB     R     H   X2B   X3B    HR
   <chr>      <int> <int> <fct>  <fct> <int> <int> <int> <int> <int> <int> <int>
 1 trumbma01   2016     1 BAL    AL      159   613    94   157    27     1    47
 2 cruzne02    2016     1 SEA    AL      155   589    96   169    27     1    43
 3 daviskh01   2016     1 OAK    AL      150   555    85   137    24     2    42
 4 doziebr01   2016     1 MIN    AL      155   615   104   165    35     5    42
 5 encared01   2016     1 TOR    AL      160   601    99   158    34     0    42
 6 arenano01   2016     1 COL    NL      160   618   116   182    35     6    41
 7 cartech02   2016     1 MIL    NL      160   549    84   122    27     1    41
 8 frazito01   2016     1 CHA    AL      158   590    89   133    21     0    40
 9 bryankr01   2016     1 CHN    NL      155   603   121   176    35     3    39
10 canoro01    2016     1 SEA    AL      161   655   107   195    33     2    39
# i 10 more variables: RBI <int>, SB <int>, CS <int>, BB <int>, SO <int>,
#   IBB <int>, HBP <int>, SH <int>, SF <int>, GIDP <int>
```

Revisar el data frame People, que tiene información demográfica de todos los jugadores

```
People %>% as_tibble()
```

```
# A tibble: 21,010 x 26
   playerID  birthYear birthMonth birthDay birthCity    birthCountry birthState
   <chr>         <int>      <int>    <int> <chr>        <chr>        <chr>
 1 aardsda01      1981         12       27 Denver       USA          CO
 2 aaronha01      1934          2        5 Mobile       USA          AL
 3 aaronto01      1939          8        5 Mobile       USA          AL
 4 aasedo01       1954          9        8 Orange       USA          CA
 5 abadan01       1972          8       25 Palm Beach   USA          FL
 6 abadfe01       1985         12       17 La Romana    D.R.         La Romana
 7 abadijo01      1850         11        4 Philadelphia USA          PA
```

```
 8 abbated01      1877          4       15 Latrobe      USA           PA
 9 abbeybe01      1869         11       11 Essex        USA           VT
10 abbeych01      1866         10       14 Falls City   USA           NE
# i 21,000 more rows
# i 19 more variables: deathYear <int>, deathMonth <int>, deathDay <int>,
#   deathCountry <chr>, deathState <chr>, deathCity <chr>, nameFirst <chr>,
#   nameLast <chr>, nameGiven <chr>, weight <int>, height <int>, bats <fct>,
#   throws <fct>, debut <chr>, bbrefID <chr>, finalGame <chr>, retroID <chr>,
#   deathDate <date>, birthDate <date>
```

```
#View(People)
```

Vincular información al top 10:

```
top_names <- top %>%
  left_join(People) %>%
  select(playerID,nameFirst, nameLast, HR)
top_names
```

```
    playerID nameFirst    nameLast HR
1  trumbma01      Mark      Trumbo 47
2   cruzne02    Nelson        Cruz 43
3  daviskh01     Khris       Davis 42
4  doziebr01     Brian      Dozier 42
5  encared01     Edwin Encarnacion 42
6  arenano01     Nolan     Arenado 41
7  cartech02     Chris      Carter 41
8  frazito01      Todd     Frazier 40
9  bryankr01      Kris      Bryant 39
10  canoro01  Robinson        Cano 39
```

```
#View(top_names)
saveRDS(top_names, "data/top_names.rds")
write_csv(top_names, "data/top_names.csv")
write_rds(top_names, "data/top_names2.rds")
```

Question 6

Inspect the `Salaries` data frame. Filter this data frame to the 2016 salaries, then use the correct bind join function to add a `salary` column to the `top_names` data frame from the previous question. Name the new data frame `top_salary`. Use this code framework:

```r
#View(Salaries)
top_salaries <- Salaries %>%
  filter(yearID==2016) %>%
  right_join(top_names)
top_salaries
```

```
   yearID teamID lgID  playerID   salary nameFirst    nameLast HR
1    2016    BAL   AL trumbma01  9150000      Mark      Trumbo 47
2    2016    CHA   AL frazito01  8250000      Todd     Frazier 40
3    2016    CHN   NL bryankr01   652000      Kris      Bryant 39
4    2016    COL   NL arenano01  5000000     Nolan     Arenado 41
5    2016    MIL   NL cartech02  2500000     Chris      Carter 41
6    2016    MIN   AL doziebr01  3000000     Brian      Dozier 42
7    2016    OAK   AL daviskh01   524500     Khris       Davis 42
8    2016    SEA   AL  canoro01 24000000  Robinson        Cano 39
9    2016    SEA   AL  cruzne02 14250000    Nelson        Cruz 43
10   2016    TOR   AL encared01 10000000     Edwin Encarnacion 42
```

```r
write_xlsx(top_salaries, "data/top_salaries.xlsx")
```

Question 7

Inspect the `AwardsPlayers` table. Filter awards to include only the year 2016.

How many players from the top 10 home run hitters won at least one award in 2016?

```r
#View(AwardsPlayers)
AwardsPlayers_2016 <- AwardsPlayers %>%
  filter(yearID==2016)
top_names %>% left_join(AwardsPlayers_2016)
```

```
    playerID nameFirst    nameLast HR              awardID yearID lgID  tie
1  trumbma01      Mark      Trumbo 47       Silver Slugger   2016   AL <NA>
2   cruzne02    Nelson        Cruz 43                 <NA>     NA <NA> <NA>
3  daviskh01     Khris       Davis 42                 <NA>     NA <NA> <NA>
4  doziebr01     Brian      Dozier 42                 <NA>     NA <NA> <NA>
5  encared01     Edwin Encarnacion 42                 <NA>     NA <NA> <NA>
6  arenano01     Nolan     Arenado 41       Silver Slugger   2016   NL <NA>
7  arenano01     Nolan     Arenado 41           Gold Glove   2016   NL <NA>
8  cartech02     Chris      Carter 41                 <NA>     NA <NA> <NA>
9  frazito01      Todd     Frazier 40                 <NA>     NA <NA> <NA>
```

```
10 bryankr01      Kris      Bryant 39 Most Valuable Player   2016   NL <NA>
11 bryankr01      Kris      Bryant 39     Hank Aaron Award   2016   NL <NA>
12 bryankr01      Kris      Bryant 39         TSN All-Star   2016   NL <NA>
13  canoro01  Robinson        Cano 39                 <NA>     NA <NA> <NA>
   notes
1     OF
2   <NA>
3   <NA>
4   <NA>
5   <NA>
6     3B
7     3B
8   <NA>
9   <NA>
10  <NA>
11    3B
12    3B
13  <NA>
```

How many players won an award in 2016 but were not one of the top 10 home run hitters in 2016?

```
dat_awa <- AwardsPlayers_2016 %>%
  anti_join(top_names) %>%
  select(playerID) %>%
  unique()
dat_awa
```

```
    playerID
1  altuvjo01
2  rizzoan01
3  lindofr01
4  hosmeer01
5  lestejo01
6  mcgowdu01
7  perezsa02
8  cabremi01
10 donaljo02
11 bogaexa01
12 bettsmo01
13 troutmi01
14 ortizda01
```

```
15 ramoswi01
17 murphda08
18 seageco01
19 blackch02
20 cespeyo01
21 yelicch01
22 arrieja01
24 morelmi01
25 kinslia01
26 beltrad01
28 gardnbr01
29 kiermke01
31 keuchda01
32 poseybu01
34 panikjo01
35 crawfbr01
36 martest01
37 inciaen01
38 heywaja01
39 greinza01
41 porceri01
42 scherma01
43 fulmemi01
45 grandcu01
47 zobribe01
48 millean01
49  baezja01
52 rendoan01
54 brittza01
55 janseke01
58 klubeco01
62 freemfr01
77 bradlja02
```

Otras formas de resolver los ejercicios:

```
Awards_2016 <- AwardsPlayers %>% filter(yearID == 2016)
length(intersect(Awards_2016$playerID, top_names$playerID))
```

```
[1] 3
```

```
length(setdiff(Awards_2016$playerID, top_names$playerID))
```

```
[1] 46
```

2.3. Web Scraping

El paquete rvest (incluído dentro de tidyverse), permite extraer información de páginas web (formato html y xml). Esto es posible a través de la función html_table()

```
library(rvest)
```

```
Warning: package 'rvest' was built under R version 4.3.3
```

```
url <- "https://en.wikipedia.org/wiki/Murder_in_the_United_States_by_state"
h <- read_html(url)
class(h)
```

```
[1] "xml_document" "xml_node"
```

```
h
```

```
{html_document}
<html class="client-nojs vector-feature-language-in-header-enabled vector-feature-language-i
[1] <head>\n<meta http-equiv="Content-Type" content="text/html; charset=UTF-8 ...
[2] <body class="skin--responsive skin-vector skin-vector-search-vue mediawik ...
```

Se utiliza la función html_nodes() para extraer elementos de la página, en este caso table. Revisar sus propiedades y extraer la tabla 2

```
tab <- h %>% html_nodes("table")
tab <- tab[[2]]
tab
```

```
{html_node}
<table class="wikitable sortable sticky-header static-row-numbers sort-under col1left" style=
[1] <caption>Gun deaths by intent\n</caption>
[2] <tbody>\n<tr>\n<th>State</th>\n<th>Gun <br> deaths</th>\n<th>Suicide</th> ...
```

Y se utiliza la función html_table() para convertir en data frame

```
tab <- tab %>% html_table()
head(tab)
```

```
# A tibble: 6 x 6
  State         `Gun  deaths` Suicide Homicide Accident   Law
  <chr>         <chr>         <chr>   <chr>       <int> <int>
1 United States 48,830        26,328  20,958        549   537
2 Texas         4,613         2,528   1,942          53    38
3 California    3,576         1,575   1,861          32    89
4 Florida       3,142         1,928   1,150          18    25
5 Georgia       2,200         1,115   1,021          25    22
6 Illinois      1,995         656     1,292          15    18
```

Assessment: Web Scraping

```
library(rvest)
url <- "https://web.archive.org/web/20181024132313/http://www.stevetheump.com/Payrolls.htm"
h <- read_html(url)
```

We learned that tables in html are associated with the `table` node. Use the `html_nodes()` function and the `table` node type to extract the first table. Store it in an object `nodes`:

```
nodes <- html_nodes(h, "table")
```

The `html_nodes()` function returns a list of objects of class `xml_node`. We can see the content of each one using, for example, the `html_text()` function. You can see the content for an arbitrarily picked component like this:

```
html_text(nodes[[8]])
```

```
[1] "Team\nPayroll\nAverge\nMedianNew York Yankees\n$ 197,962,289\n$ 6,186,321\n$ 1,937,500Pl
```

If the content of this object is an html table, we can use the `html_table()` function to convert it to a data frame:

```
tab1 <- html_table(nodes[[8]])
```

```
html_table(nodes[[8]])
```

```
# A tibble: 30 x 4
   Team                  Payroll       Averge      Median
   <chr>                 <chr>         <chr>       <chr>
 1 New York Yankees      $ 197,962,289 $ 6,186,321 $ 1,937,500
 2 Philadelphia Phillies $ 174,538,938 $ 5,817,964 $ 1,875,000
 3 Boston Red Sox        $ 173,186,617 $ 5,093,724 $ 1,556,250
 4 Los Angeles Angels    $ 154,485,166 $ 5,327,074 $ 3,150,000
 5 Detroit Tigers        $ 132,300,000 $ 4,562,068 $ 1,100,000
 6 Texas Rangers         $ 120,510,974 $ 4,635,037 $ 3,437,500
 7 Miami Marlins         $ 118,078,000 $ 4,373,259 $ 1,500,000
 8 San Francisco Giants  $ 117,620,683 $ 3,920,689 $ 1,275,000
 9 St. Louis Cardinals   $ 110,300,862 $ 3,939,316 $ 800,000
10 Milwaukee Brewers     $ 97,653,944  $ 3,755,920 $ 1,981,250
# i 20 more rows
```

Question 3

1 point possible (graded)

Create a table called `tab_1` using entry 10 of `nodes`. Create a table called `tab_2` using entry 19 of `nodes`.

Note that the column names should be `c("Team", "Payroll", "Average")`. You can see that these column names are actually in the first data row of each table, and that `tab_1` has an extra first column `No.` that should be removed so that the column names for both tables match.

Remove the extra column in `tab_1`, remove the first row of each dataset, and change the column names for each table to `c("Team", "Payroll", "Average")`. Use a `full_join()` by the `Team` to combine these two tables.

How many rows are in the joined data table?

```
library(tidyverse)
tab1 <- html_table(nodes[[10]])
head(tab1)
```

```
# A tibble: 6 x 4
  X1    X2                    X3           X4
  <chr> <chr>                 <chr>        <chr>
1 No.   Team                  Payroll      Average
```

```
2 1.    New York Yankees      $206,333,389 $8,253,336
3 2.    Boston Red Sox        $162,747,333 $5,611,977
4 3.    Chicago Cubs          $146,859,000 $5,439,222
5 4.    Philadelphia Phillies $141,927,381 $5,068,835
6 5.    New York Mets         $132,701,445 $5,103,902
```

```
nrow(tab1)
```

```
[1] 31
```

```
tab1 <- tab1 %>% select("X2","X3","X4") %>% slice(2:31) %>% set_names(c("Team", "Payroll", "A
head(tab1)
```

```
# A tibble: 6 x 3
  Team                  Payroll       Average
  <chr>                 <chr>         <chr>
1 New York Yankees      $206,333,389 $8,253,336
2 Boston Red Sox        $162,747,333 $5,611,977
3 Chicago Cubs          $146,859,000 $5,439,222
4 Philadelphia Phillies $141,927,381 $5,068,835
5 New York Mets         $132,701,445 $5,103,902
6 Detroit Tigers        $122,864,929 $4,550,553
```

```
tab2 <- html_table(nodes[[19]])
head(tab2)
```

```
# A tibble: 6 x 3
  X1          X2           X3
  <chr>       <chr>        <chr>
1 Team        Payroll      Average
2 NY Yankees  $109,791,893 $3,541,674
3 Boston      $109,558,908 $3,423,716
4 Los Angeles $108,980,952 $3,757,964
5 NY Mets     $93,174,428  $3,327,658
6 Cleveland   $91,974,979  $3,065,833
```

```
nrow(tab2)
```

```
[1] 31
```

```
tab2 <- tab2 %>% slice(2:nrow(tab2)) %>% set_names(c("Team", "Payroll", "Average"))
head(tab2)
```

```
# A tibble: 6 x 3
  Team        Payroll      Average
  <chr>       <chr>        <chr>
1 NY Yankees  $109,791,893 $3,541,674
2 Boston      $109,558,908 $3,423,716
3 Los Angeles $108,980,952 $3,757,964
4 NY Mets     $93,174,428  $3,327,658
5 Cleveland   $91,974,979  $3,065,833
6 Atlanta     $91,851,687  $2,962,958
```

```
full_join(tab1,tab2, by="Team")
```

```
# A tibble: 58 x 5
   Team                 Payroll.x    Average.x  Payroll.y   Average.y
   <chr>                <chr>        <chr>      <chr>       <chr>
 1 New York Yankees     $206,333,389 $8,253,336 <NA>        <NA>
 2 Boston Red Sox       $162,747,333 $5,611,977 <NA>        <NA>
 3 Chicago Cubs         $146,859,000 $5,439,222 $64,015,833 $2,462,147
 4 Philadelphia Phillies $141,927,381 $5,068,835 <NA>       <NA>
 5 New York Mets        $132,701,445 $5,103,902 <NA>        <NA>
 6 Detroit Tigers       $122,864,929 $4,550,553 <NA>        <NA>
 7 Chicago White Sox    $108,273,197 $4,164,354 $62,363,000 $2,309,741
 8 Los Angeles Angels   $105,013,667 $3,621,161 <NA>        <NA>
 9 Seattle Mariners     $98,376,667  $3,513,452 <NA>        <NA>
10 San Francisco Giants $97,828,833  $3,493,887 <NA>        <NA>
# i 48 more rows
```

Section 3: String Processing

3.1. String Processing Part 1

String Parsing

```
s <- '10"'
cat(s)
```

```
10"
```

```
ss <- "10'"
cat(ss)
```

```
10'
```

```
#Scape the quote a través de /
sss <- '5\'10"'
cat(sss)
```

```
5'10"
```

```
ssss <- "5'10\""
cat(ssss)
```

```
5'10"
```

```
# read in raw murders data from Wikipedia
library(rvest)
url <- "https://en.wikipedia.org/w/index.php?title=Gun_violence_in_the_United_States_by_state
murders_raw <- read_html(url) %>%
  html_nodes("table") %>%
  html_table() %>%
  .[[1]] %>%
  setNames(c("state", "population", "total", "murder_rate"))

# inspect data and column classes
head(murders_raw)
```

```
# A tibble: 6 x 4
  state      population total murder_rate
  <chr>      <chr>      <chr>       <dbl>
1 Alabama    4,853,875  348           7.2
2 Alaska     737,709    59            8
3 Arizona    6,817,565  309           4.5
4 Arkansas   2,977,853  181           6.1
5 California 38,993,940 1,861         4.8
6 Colorado   5,448,819  176           3.2
```

```
class(murders_raw$population)
```

```
[1] "character"
```

```
class(murders_raw$total)
```

```
[1] "character"
```

```
library(tidyverse)
murders_raw$population[1:3]
```

```
[1] "4,853,875" "737,709"   "6,817,565"
```

```
as.numeric(murders_raw$population[1:3]) #La coerción as.numeric() no funciona acá
```

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

Por lo tanto debe utilizarse stringr y sus funciones str_ para modificar datos. En este caso, se utilizará str_detect() dentro de una función, para detectar si las columnas tienen comas.

```
commas <- function(x) any(str_detect(x,","))
murders_raw %>% summarize_all(funs(commas))
```

```
# A tibble: 1 x 4
  state population total murder_rate
  <lgl> <lgl>      <lgl> <lgl>
1 FALSE TRUE       TRUE  FALSE
```

Utilizar la función str_replace_all() para reemplazar todos los valores que tengan coma por un espacio nulo y luego convertir el string a numeric.

```
test_1 <- str_replace_all(murders_raw$population, ",", "")
test_1 <- as.numeric(test_1)
head(test_1)
```

```
[1]  4853875   737709  6817565  2977853 38993940  5448819
```

También, es posible utilizar la función parse_number() que realiza los dos procedimiento de manera simultánea

```
test_2 <- parse_number(murders_raw$population)
head(test_2)
```

```
[1]  4853875   737709  6817565  2977853 38993940  5448819
```

```
identical(test_1,test_2)
```

```
[1] TRUE
```

La tabla final se obtendría a través del siguiente código:

```
murders_new <- murders_raw %>% mutate_at(2:3, parse_number) #La función mutate_at funciona co
head(murders_new)
```

```
# A tibble: 6 x 4
  state      population total murder_rate
  <chr>           <dbl> <dbl>       <dbl>
1 Alabama       4853875   348         7.2
2 Alaska         737709    59         8
3 Arizona       6817565   309         4.5
4 Arkansas      2977853   181         6.1
5 California   38993940  1861         4.8
6 Colorado      5448819   176         3.2
```

Key points

- Use the `str_detect()` function to determine whether a string contains a certain pattern.
- Use the `str_replace_all()` function to replace all instances of one pattern with another pattern. To remove a pattern, replace with the empty string (`""`).
- The `parse_number()` function removes punctuation from strings and converts them to numeric.
- `mutate_at()` performs the same transformation on the specified column numbers.

```
library(dslabs)
library(tidyverse)
library(writexl)
data("reported_heights")
#View(reported_heights)

class(reported_heights$height)
```

```
[1] "character"
```

```
write_xlsx(reported_heights, "data/reported_heights.xlsx")
```

```
# convert to numeric, inspect, count NAs
x <- as.numeric(reported_heights$height)
```

```
Warning: NAs introduced by coercion
```

```
head(x)
```

```
[1] 75 70 68 74 61 65
```

```
sum(is.na(x))
```

```
[1] 81
```

```
# keep only entries that result in NAs
reported_heights %>% mutate(new_height = as.numeric(height)) %>%
  filter(is.na(new_height)) %>%
  head(n=10)
```

```
Warning: There was 1 warning in `mutate()`.
i In argument: `new_height = as.numeric(height)`.
Caused by warning:
! NAs introduced by coercion
```

	time_stamp	sex	height	new_height
1	2014-09-02 15:16:28	Male	5' 4"	NA
2	2014-09-02 15:16:37	Female	165cm	NA
3	2014-09-02 15:16:52	Male	5'7	NA
4	2014-09-02 15:16:56	Male	>9000	NA
5	2014-09-02 15:16:56	Male	5'7"	NA
6	2014-09-02 15:17:09	Female	5'3"	NA
7	2014-09-02 15:18:00	Male	5 feet and 8.11 inches	NA
8	2014-09-02 15:19:48	Male	5'11	NA
9	2014-09-04 00:46:45	Male	5'9''	NA
10	2014-09-04 10:29:44	Male	5'10''	NA

```
# calculate cutoffs that cover 99.999% of human population
alpha <- 1/10^6
qnorm(1-alpha/2, 69.1, 2.9)
```

```
[1] 83.28575
```

```
qnorm(alpha/2, 63.7, 2.7)
```

```
[1] 50.49258
```

```
# keep only entries that either result in NAs or are outside the plausible range of heights
not_inches <- function(x, smallest = 50, tallest = 84){
  inches <- suppressWarnings(as.numeric(x))
  ind <- is.na(inches) | inches < smallest | inches > tallest
  ind
}

# number of problematic entries
problems <- reported_heights %>%
  filter(not_inches(height)) %>%
  .$height
length(problems)
```

```
[1] 292
```

```
# 10 examples of x'y or x'y" or x'y\"
pattern <- "^\\d\\s*'\\s*\\d{1,2}\\.*\\d*'*\"*$"
str_subset(problems, pattern) %>% head(n=10) %>% cat
```

```
5' 4" 5'7 5'7" 5'3" 5'11 5'9'' 5'10'' 5' 10 5'5" 5'2"
```

```
# 10 examples of x.y or x,y
pattern <- "^[4-6]\\s*[\\.|,]\\s*([0-9]|10|11)$"
str_subset(problems, pattern) %>% head(n=10) %>% cat
```

```
5.3 5.5 6.5 5.8 5.6 5,3 5.9 6,8 5.5 6.2
```

```
# 10 examples of entries in cm rather than inches
ind <- which(between(suppressWarnings(as.numeric(problems))/2.54, 54, 81) )
ind <- ind[!is.na(ind)]
problems[ind] %>% head(n=10) %>% cat
```

```
150 175 177 178 163 175 178 165 165 180
```

Regex

Para detectar una coma

```
pattern <- ","
str_detect(murders_raw$total,pattern )
```

```
 [1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
[49] FALSE FALSE FALSE
```

Para detectar valores que tienen cm, u otro patrón, se utiliza la función str_subset()-

```
str_subset(reported_heights$height, "cm")
```

```
[1] "165cm"  "170 cm"
```

```
yes <- c("180 cm", "70 inches")
no <- c("180", "70''")
s <- c(yes, no)
s
```

```
[1] "180 cm"    "70 inches" "180"       "70''"
```

Detectar los valores con cm o inches

```
str_detect(s, "cm") | str_detect(s, "inches")
```

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

Otra forma de detectar cm o inches, a través de la expresión “cm|inches”

```
str_detect(s, "cm|inches")
```

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

Para detectar patrones que incliyan dígitos , se utiliza la regex “\\d”. Por ejemplo:

```
yes <- c("5","6","5'10","5 feet", "4'11")
no <- c("",".","Five","six")
s <- c(yes,no)
pattern <- "\\d"
str_detect(s,pattern)
```

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

La función str_view() resalta el primer patrón detectado

```
str_view(s,pattern)
```

```
[1] | <5>
[2] | <6>
[3] | <5>'<1><0>
[4] | <5> feet
[5] | <4>'<1><1>
```

Mientras que la función str_view_all() muestra todo el patrón

```
str_view_all(s,pattern)
```

```
Warning: `str_view_all()` was deprecated in stringr 1.5.0.
i Please use `str_view()` instead.
```

```
[1] | <5>
[2] | <6>
[3] | <5>'<1><0>
[4] | <5> feet
[5] | <4>'<1><1>
[6] |
[7] | .
[8] | Five
[9] | six
```

Key points

- A regular expression (regex) is a way to describe a specific pattern of characters of text. A set of rules has been designed to do this specifically and efficiently.
- **stringr** functions can take a regex as a pattern.
- `str_detect()` indicates whether a pattern is present in a string.
- The main difference between a regex and a regular string is that a regex can include special characters.
- The | symbol inside a regex means "or".
- Use `'\\d'` to represent digits. The backlash is used to distinguish it from the character `'d'`. In R, you must use two backslashes for digits in regular expressions; in some other languages, you will only use one backslash for regex special characters.
- `str_view()` highlights the first occurrence of a pattern, and the `str_view_all()` function highlights all occurrences of the pattern.

Character Classes, Anchors and Quantifiers

Si quiero resaltar carácteres específicos, debo expresarlos dentro de llaves

```
str_view(s,"[56]")
```

```
[1] | <5>
[2] | <6>
[3] | <5>'10
[4] | <5> feet
```

Si quiero detectar rangos, utilizo -. Por ejemplo, para las expresiones entre 4 y 7, la expresión regular sería "[4-7]"

```
yes <- as.character(4:7)
no <- as.character(1:3)
s <- c(yes,no)
s
```

```
[1] "4" "5" "6" "7" "1" "2" "3"
```

```
str_detect(s,"[4-7]")
```

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

```
str_view(s,"[4-7]")
```

```
[1] | <4>
[2] | <5>
[3] | <6>
[4] | <7>
```

Para detectar todas las letras del alfabeto en minúsculas se utiliza la expresión "[a-z]" y en mayúscula "[A-Z]". Para detectar todas las letras en minúscula o mayúscula la expresión sería "[a-zA-Z]"

^ representa el principio de un string

$ representa el final de un string

Para obtener los string que tienen un sólo dígito, se utilizaría la siguiente expresión:

```
pattern <- "^\\d$"
yes <- c("1","5","9")
no <- c("12","123", " 1", "a4", "b")
s <- c(yes, no)
str_view(s, pattern)
```

```
[1] | <1>
[2] | <5>
[3] | <9>
```

Para obtener los string con uno o dos dígitos, se utiliza la expresión \\d{1,2}

```
pattern <- "^\\d{1,2}$"
yes <- c("1","5","9","12")
no <- c("123","a4","b")
s <- c(yes, no)
s
```

```
[1] "1"   "5"   "9"   "12"  "123" "a4"  "b"
```

```
str_view(s,pattern)
```

```
[1] | <1>
[2] | <5>
[3] | <9>
[4] | <12>
```

Para detectar el patrón número, pie, número, pulgada (ej. 5'7") se utiliza la expresión "^[4-7]'\\d{1,2}\"$"

```
pattern <- "^[4-7]'\\d{1,2}\"$"
yes <- c("5'7\"", "6'2\"",  "5'12\"")
no <- c("6,2\"", "6.2\"","I am 5'11\"", "3'2\"", "64")

str_detect(yes,pattern)
```

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

```
str_detect(no, pattern)
```

```
[1] FALSE FALSE FALSE FALSE FALSE
```

Key points

- Define strings to test your regular expressions, including some elements that match and some that do not. This allows you to check for the two types of errors: failing to match and matching incorrectly.
- Square brackets define character classes: groups of characters that count as matching the pattern. You can use ranges to define character classes, such as `[0-9]` for digits and `[a-zA-Z]` for all letters.
- Anchors define patterns that must start or end at specific places. `^` and `$` represent the beginning and end of the string respectively.
- Curly braces are quantifiers that state how many times a certain character can be repeated in the pattern. `\\d{1,2}` matches exactly 1 or 2 consecutive digits.

Search and Replace with Regex

Sólo 14 string cumplen con el patrón deseado

```
pattern <- "^[4-7]'\\d{1,2}\"$"
sum(str_detect(problems,pattern))
```

```
[1] 14
```

Por ejemplo:

```
problems[c(2,10,11,12,15)] %>% str_view_all(pattern)
```

```
[1] | 5' 4"
[2] | <5'7">
[3] | <5'3">
[4] | 5 feet and 8.11 inches
[5] | 5.5
```

El problema es que escribieron feet and inches en lugar de ' y "

```
str_subset(problems, "inches")
```

```
[1] "5 feet and 8.11 inches" "Five foot eight inches" "5 feet 7inches"
[4] "5ft 9 inches"           "5 ft 9 inches"          "5 feet 6 inches"
```

O ' ' en ligar de "

```
str_subset(problems, "''")
```

```
 [1] "5'9''"   "5'10''"  "5'10''"  "5'3''"   "5'7''"   "5'6''"   "5'7.5''"
 [8] "5'7.5''" "5'10''"  "5'11''"  "5'10''"  "5'5''"
```

Si no se usa el símbolo para pulgadas (") al final, la expresión sería"^[4-7]'\\d{1,2}$"

```
pattern <- "^[4-7]'\\d{1,2}$"
problems %>% str_replace("feet|ft|foot", "'") %>%
  str_replace("inches|in|''|\"", "") %>%
  str_detect(pattern) %>%
  sum
```

```
[1] 48
```

En regex es posible representar espacio con la expresión \\s

```
pattern2 <- "^[4-7]'\\s\\d{1,2}\"$"
str_subset(problems,pattern2)
```

```
[1] "5' 4\""  "5' 11\"" "5' 7\""
```

También, es posible reemplazar una cantidad de caracteres similares de manera posterior a * . Es decir, cero o más ejemplos del caracter previo. Esto también se puede aplicar después de espacio (\\s)

```
yes <- c("AB", "A1B", "A11B", "A111B", "A1111B")
no <- c("A2B", "A21B")
str_detect(yes, "A1*B") #Cero o más ejemplos del caracter previo a *
```

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

```
str_detect(no, "A1*B")
```

```
[1] FALSE FALSE
```

```
# test how *, ? and + differ
data.frame(string = c("AB", "A1B", "A11B", "A111B", "A1111B"),
           none_or_more = str_detect(yes, "A1*B"),
           nore_or_once = str_detect(yes, "A1?B"),
           once_or_more = str_detect(yes, "A1+B"))
```

```
  string none_or_more nore_or_once once_or_more
1     AB         TRUE         TRUE        FALSE
2    A1B         TRUE         TRUE         TRUE
3   A11B         TRUE        FALSE         TRUE
4  A111B         TRUE        FALSE         TRUE
5 A1111B         TRUE        FALSE         TRUE
```

```
pattern <- "^[4-7]\\s*'\\s*\\d{1,2}$"
problems %>%
  str_replace("feet|ft|foot" ,"'") %>% #reemplazar con '
  str_replace("inches|in|''|\"", "") %>% #remover todos los símbolos de inches
  str_detect(pattern) %>%
  sum
```

```
[1] 53
```

- `str_replace()` replaces the first instance of the detected pattern with a specified string.
- Spaces are characters and R does not ignore them. Spaces are specified by the special character `\\s`.
- Additional quantifiers include `*, + and ?`. `*` means 0 or more instances of the previous character. `?` means 0 or 1 instances. `+` means 1 or more instances.
- Before removing characters from strings with functions like`str_replace()` and`str_replace_all()`, consider whether that replacement would have unintended effects.

Groups with Regex

```
pattern_without_groups <- "[4-7],\\d*$"
```

```
pattern_with_groups <- "^([4-7]),(\\d*)$"
```

```
yes <- c("5,9","5,11","6,","6,1")
no <- c("5'9", ",", "2,8", "6.1.1")
s <- c(yes,no)
str_detect(s,pattern_without_groups)
```

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

```
str_detect(s,pattern_with_groups)
```

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

A través de str_match() , es posible obtener los grupos si el patrón a detectar está en grupos y str_extract() los extrae

```
str_match(s,pattern_with_groups)
```

```
     [,1]   [,2] [,3]
[1,] "5,9"  "5"  "9"
[2,] "5,11" "5"  "11"
[3,] "6,"   "6"  ""
[4,] "6,1"  "6"  "1"
[5,] NA     NA   NA
[6,] NA     NA   NA
[7,] NA     NA   NA
[8,] NA     NA   NA
```

La segunda y tercera columna representa los grupos, mientras que la primera columna representa el valor original. Una vez agrupadas las expresiones, es posible señalar cómo reemplazar el primero o segundo grupo a través de las expresiones \\1 y \\2 o la cantidad correspondiente. En este caso, reemplazar "," por ""'

```
pattern_with_groups <- "^([4-7]),(\\d*)$"
yes <- c("5,9","5,11","6,","6,1")
no <- c("5'9", ",", "2,8", "6.1.1")
s <- c(yes,no)
str_replace(s,pattern_with_groups, "\\1'\\2")
```

```
[1] "5'9"   "5'11"  "6'"    "6'1"   "5'9"   ","     "2,8"   "6.1.1"
```

Identificar strings que cumplen las condiciones

```
pattern_with_groups <- "^([4-7])\\s*[,\\.\\s+]\\s*(\\d*)$"
str_subset(problems,pattern_with_groups) %>% head
```

```
[1] "5.3"  "5.25" "5.5"  "6.5"  "5.8"  "5.6"
```

Luego , reemplazar

```
str_subset(problems, pattern_with_groups) %>%
  str_replace(pattern_with_groups, "\\1'\\2") %>% head
```

```
[1] "5'3"  "5'25" "5'5"  "6'5"  "5'8"  "5'6"
```

```
reemp1 <- str_subset(problems, pattern_with_groups) %>%
  str_replace(pattern_with_groups, "\\1'\\2")
write_xlsx(as.data.frame(reemp1), "data/reemp_1.xlsx")
```

Key Points

- Groups are defined using parentheses.
- Once we define groups, we can use the function `str_match()` to extract the values these groups define. `str_extract()` extracts only strings that match a pattern, not the values defined by groups.
- You can refer to the `i`th group with `\\i`. For example, refer to the value in the second group with `\\2`.

Testing and Improving

```
# function to detect entries with problems
not_inches_or_cm <- function(x, smallest = 50, tallest = 84){
    inches <- suppressWarnings(as.numeric(x))
    ind <- !is.na(inches) &
        ((inches >= smallest & inches <= tallest) |
             (inches/2.54 >= smallest & inches/2.54 <= tallest))
    !ind
}

# identify entries with problems
problems <- reported_heights %>%
  filter(not_inches_or_cm(height)) %>%
  .$height
length(problems)
```

```
[1] 200
```

```
#View(problems)
```

Para convertir algunos string

```
converted <- problems %>%
  str_replace("feet|foot|ft", "'") %>%
  str_replace("inches|in|''\"","") %>%
  str_replace("^([4-7])\\s*[,\\.\\s+]\\s*(\\d*)$", "\\1'\\2")
#head(converted)
```

```
pattern <- "^[4-7]\\s*'\\s*\\d{1,2}$"
index <- str_detect(converted, pattern)
mean(index)
```

```
[1] 0.48
```

Todos los string que no hacen parte del index, se obtienen a través de:

```
converted[!index]
```

```
  [1] "6"             "5' 4\""        "165cm"         "511"
  [5] "6"             "2"             ">9000"         "5'7\""
  [9] "5'3\""         "5 ' and 8.11 " "11111"         "5'9''"
 [13] "6"             "5'10''"        "103.2"         "19"
 [17] "5"             "300"           "6'"            "6"
 [21] "Five ' eight " "5'5\""         "5'2\""         "7"
 [25] "214"           "6"             "5'10''"        "5'3''"
 [29] "0.7"           "5'7''"         "6"             "2'33"
 [33] "5'3\""         "5'6''"         "612"           "1,70"
 [37] "87"            "5'7.5''"       "5'7.5''"       "111"
 [41] "5'2\""         "5' 7.78\""     "12"            "6"
 [45] "yyy"           "89"            "34"            "25"
 [49] "6"             "6"             "22"            "684"
 [53] "6"             "1"             "1"             "6*12"
 [57] "87"            "6"             "1.6"           "120"
 [61] "120"           "23"            "1.7"           "6"
 [65] "5'8\""         "5'11\""        "5'7\""         "5' 11\""
 [69] "5"             "6'1\""         "69\""          "5' 7\""
 [73] "5'10''"        "5' 9 "         "5 ' 9 "        "6"
 [77] "5'11''"        "5'8\""         "6"             "86"
 [81] "708,661"       "5 ' 6 "        "5'10''"        "6"
 [85] "6'3\""         "649,606"       "10000"         "1"
 [89] "728,346"       "0"             "5'5''"         "5'7\""
 [93] "6"             "6"             "6"             "100"
 [97] "6'4\""         "88"            "6"             "170 cm"
[101] "7,283,465"     "5"             "5"             "34"
```

String Splitting

```
# read raw murders data line by line
library(tidyverse)
filename <- system.file("extdata/murders.csv", package = "dslabs")
lines <- readLines(filename)
lines %>% head()
```

```
[1] "state,abb,region,population,total" "Alabama,AL,South,4779736,135"
[3] "Alaska,AK,West,710231,19"          "Arizona,AZ,West,6392017,232"
[5] "Arkansas,AR,South,2915918,93"      "California,CA,West,37253956,1257"
```

Para separar archivos separados por coma, se utiliza la función str_split()

```
x <- str_split(lines, ",")
x %>% head
```

```
[[1]]
[1] "state"      "abb"        "region"     "population" "total"

[[2]]
[1] "Alabama" "AL"      "South"   "4779736" "135"

[[3]]
[1] "Alaska" "AK"     "West"   "710231" "19"

[[4]]
[1] "Arizona" "AZ"      "West"    "6392017" "232"

[[5]]
[1] "Arkansas" "AR"       "South"    "2915918"  "93"

[[6]]
[1] "California" "CA"         "West"       "37253956"   "1257"
```

```
col_names <- x[[1]]
col_names #Extraer los nombres de los estados
```

```
[1] "state"      "abb"        "region"     "population" "total"
```

```
x <- x[-1] #Extraer exclusivamente los datos, sin encabezado
```

La función map, del paquete purrr permite aplicar una función simultáneamente

```
library(purrr)
map(x, function(y)y[1]) %>% head()
```

```
[[1]]
[1] "Alabama"

[[2]]
[1] "Alaska"

[[3]]
[1] "Arizona"

[[4]]
[1] "Arkansas"

[[5]]
[1] "California"

[[6]]
[1] "Colorado"
```

Otra forma

```
map(x, 1) %>% head()
```

```
[[1]]
[1] "Alabama"

[[2]]
[1] "Alaska"

[[3]]
[1] "Arizona"

[[4]]
[1] "Arkansas"
```

```
[[5]]
[1] "California"

[[6]]
[1] "Colorado"
```

Las funciones map_chr() retorna un vector de caracteres y map_int() retorna vector de enteros

```
dat <- data.frame(map_chr(x,1), #Convierte vectores en caracteres
                  map_chr(x,2),
                  map_chr(x,3),
                  map_chr(x,4),
                  map_chr(x,5)) %>%
  mutate_all(parse_guess) %>% #Transforma todos los vectores al tipo de datos que el programa
  set_names(col_names) #Asigna nombre a los vectores
dat %>% head
```

```
       state abb region population total
1    Alabama  AL  South    4779736   135
2     Alaska  AK   West     710231    19
3    Arizona  AZ   West    6392017   232
4   Arkansas  AR  South    2915918    93
5 California  CA   West   37253956  1257
6   Colorado  CO   West    5029196    65
```

Se obtiene código más sencillo a través de stringr y el argumento simpligy = TRUE esto obliga a la función a retornar una matriz en lugar de una lista

```
x <- str_split(lines, ",", simplify = TRUE)
col_names <- x[1,]
x <- x[-1,]
x %>% as_data_frame() %>%
  setNames(col_names) %>%
  mutate_all(parse_guess)
```

```
Warning: `as_data_frame()` was deprecated in tibble 2.0.0.
i Please use `as_tibble()` (with slightly different semantics) to convert to a
  tibble, or `as.data.frame()` to convert to a data frame.
```

```
Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if
`.name_repair` is omitted as of tibble 2.0.0.
i Using compatibility `.name_repair`.
i The deprecated feature was likely used in the tibble package.
  Please report the issue at <https://github.com/tidyverse/tibble/issues>.
```

```
# A tibble: 51 x 5
   state                abb   region    population total
   <chr>                <chr> <chr>          <dbl> <dbl>
 1 Alabama              AL    South        4779736   135
 2 Alaska               AK    West          710231    19
 3 Arizona              AZ    West         6392017   232
 4 Arkansas             AR    South        2915918    93
 5 California           CA    West        37253956  1257
 6 Colorado             CO    West         5029196    65
 7 Connecticut          CT    Northeast    3574097    97
 8 Delaware             DE    South         897934    38
 9 District of Columbia DC    South         601723    99
10 Florida              FL    South       19687653   669
# i 41 more rows
```

Separate with Regex

Ejemplo

```
s <- c("5'10", "6'1")
tab <- data.frame(x = s)
tab
```

```
     x
1 5'10
2  6'1
```

Separar inches y feet a través de separate()

```
tab %>% separate(x, into=c("feet","inches"), sep="'" ) #Se debe específicar el nombre de las
```

```
  feet inches
1    5     10
2    6      1
```

También es posible realizar esto a través de expresiones regulares

```
tab %>% extract(x, c("feet", "inches"), regex = "([\\d])'(\\d{1,2})")
```

```
  feet inches
1    5     10
2    6      1
```

```
s <- c("5'10", "6'1\"", "5'8inches")
tab <- data.frame(x=s)
tab
```

```
          x
1      5'10
2      6'1"
3 5'8inches
```

```
tab %>% separate(x, c("feet", "inches"), sep="'", fill="right")
```

```
  feet  inches
1    5      10
2    6      1"
3    5 8inches
```

Pero si utilizamos extract()

```
tab %>% extract(x, into = c("feet", "inches"), regex="(\\d)'(\\d{1,2})")
```

```
  feet inches
1    5     10
2    6      1
3    5      8
```

Using Groups and Quantifiers

Case 1

For case 1, if we add a `'0` to, for example, convert all `6` to `6'0`, then our pattern will match. This can be done using groups using the following code:

```
yes <- c("5", "6", "5")
no <- c("5'", "5''", "5'4")
s <- c(yes, no)
str_replace(s, "^([4-7])$", "\\1'0") #^$ poner este rango es otra manera de expresar = "mante
```

```
[1] "5'0" "6'0" "5'0" "5'"  "5''" "5'4"
```

The pattern says it has to start (`^`), be followed with a digit between 4 and 7, and then end there (`$`). The parenthesis defines the group that we pass as `\\1` to the replace regex.

Cases 2 and 4

We can adapt this code slightly to handle case 2 as well which covers the entry `5'`. Note that the `5'` is left untouched by the code above. This is because the extra `'` makes the pattern not match since we have to end with a 5 or 6. To handle case 2, we want to permit the 5 or 6 to be followed by no or one symbol for feet. So we can simply add `'{0,1}` after the `'` to do this. We can also use the none or once special character `?`. As we saw previously, this is different from `*` which is none or more. We now see that this code also handles the fourth case as well:

```
str_replace(s, "^([56])'?$", "\\1'0")
```

```
[1] "5'0" "6'0" "5'0" "5'0" "5''" "5'4"
```

Note that here we only permit 5 and 6 but not 4 and 7. This is because heights of exactly 5 and exactly 6 feet tall are quite common, so we assume those that typed 5 or 6 really meant either `60` or `72` inches. However, heights of exactly 4 or exactly 7 feet tall are so rare that, although we accept `84` as a valid entry, we assume that a `7` was entered in error.

Case 3

We can use quantifiers to deal with case 3. These entries are not matched because the inches include decimals and our pattern does not permit this. We need allow the second group to include decimals and not just digits. This means we must permit zero or one period `.` followed by zero or more digits. So we will use both `?` and `*`. Also remember that for this particular case, the period needs to be escaped since it is a special character (it means any character except a line break).

So we can adapt our pattern, currently `^[4-7]\\s*'\\s*\\d{1,2}$`, to permit a decimal at the end:

```
pattern <- "^[4-7]\\s*'\\s*(\\d+\\.?\\d*)$"
```

Case 5

Case 5, meters using commas, we can approach similarly to how we converted the `x.y` to `x'y`. A difference is that we require that the first digit is 1 or 2:

```
yes <- c("1,7", "1, 8", "2, " )
no <- c("5,8", "5,3,2", "1.7")
s <- c(yes, no)
str_replace(s, "^([12])\\s*,\\s*(\\d*)$", "\\1\\.\\2")
```

```
[1] "1.7"   "1.8"   "2."    "5,8"   "5,3,2" "1.7"
```

Trimming

In general, spaces at the start or end of the string are uninformative. These can be particularly deceptive because sometimes they can be hard to see:

```
s <- "Hi "
cat(s)
```

```
Hi
```

```
identical(s, "Hi")
```

```
[1] FALSE
```

```
str_trim("5 ' 9 ")
```

```
[1] "5 ' 9"
```

To upper and to lower case

One of the entries writes out numbers as words: **`Five foot eight inches`**. Although not efficient, we could add 12 extra **`str_replace`** to convert **`zero`** to **`0`**, **`one`** to **`1`**, and so on. To avoid having to write two separate operations for **`Zero`** and **`zero`**, **`One`** and **`one`**, etc., we can use the `str_to_lower()` function to make all words lower case first:

```
s <- c("Five feet eight inches")
str_to_lower(s)
```

```
[1] "five feet eight inches"
```

Putting it into a function

We are now ready to define a procedure that handles converting all the problematic cases.

We can now put all this together into a function that takes a string vector and tries to convert as many strings as possible to a single format. Below is a function that puts together the previous code replacements:

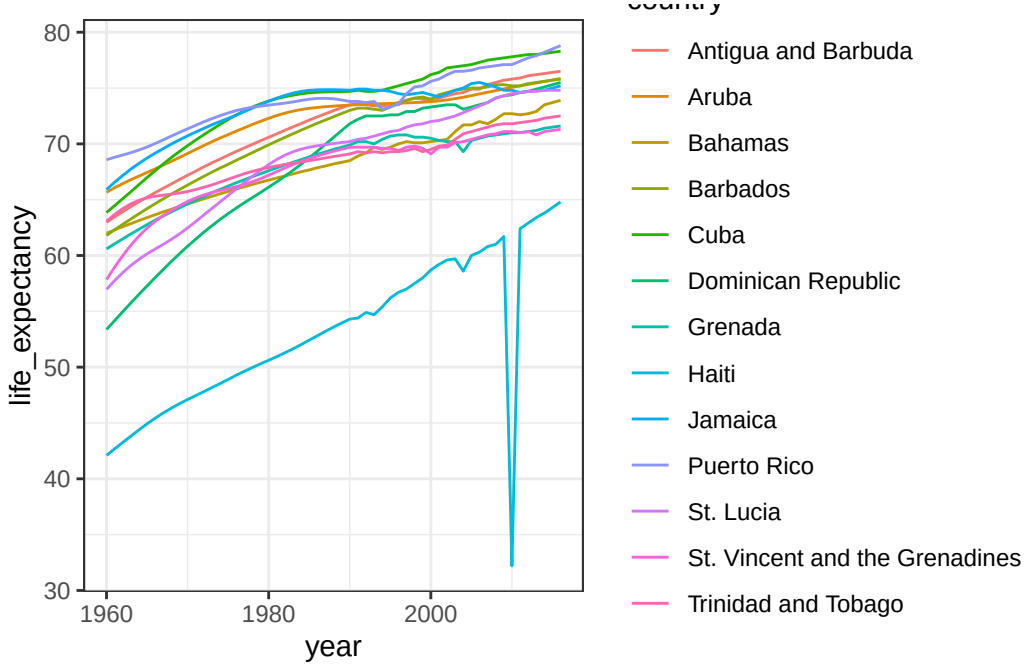
```
convert_format <- function(s){
  s %>%
    str_replace("feet|foot|ft", "'") %>% #convert feet symbols to '
    str_replace_all("inches|in|''|\"|cm|and", "") %>%  #remove inches and other symbols
    str_replace("^([4-7])\\s*[,\\.\\s+]\\s*(\\d*)$", "\\1'\\2") %>% #change x.y, x,y x y
    str_replace("^([56])'?$", "\\1'0") %>% #add 0 when to 5 or 6
    str_replace("^([12])\\s*,\\s*(\\d*)$", "\\1\\.\\2") %>% #change european decimal
    str_trim() #remove extra space
}
```

We can also write a function that converts words to numbers:

```
words_to_numbers <- function(s){
  str_to_lower(s) %>%
    str_replace_all("zero", "0") %>%
    str_replace_all("one", "1") %>%
    str_replace_all("two", "2") %>%
    str_replace_all("three", "3") %>%
    str_replace_all("four", "4") %>%
    str_replace_all("five", "5") %>%
    str_replace_all("six", "6") %>%
    str_replace_all("seven", "7") %>%
    str_replace_all("eight", "8") %>%
    str_replace_all("nine", "9") %>%
    str_replace_all("ten", "10") %>%
    str_replace_all("eleven", "11")
}
```

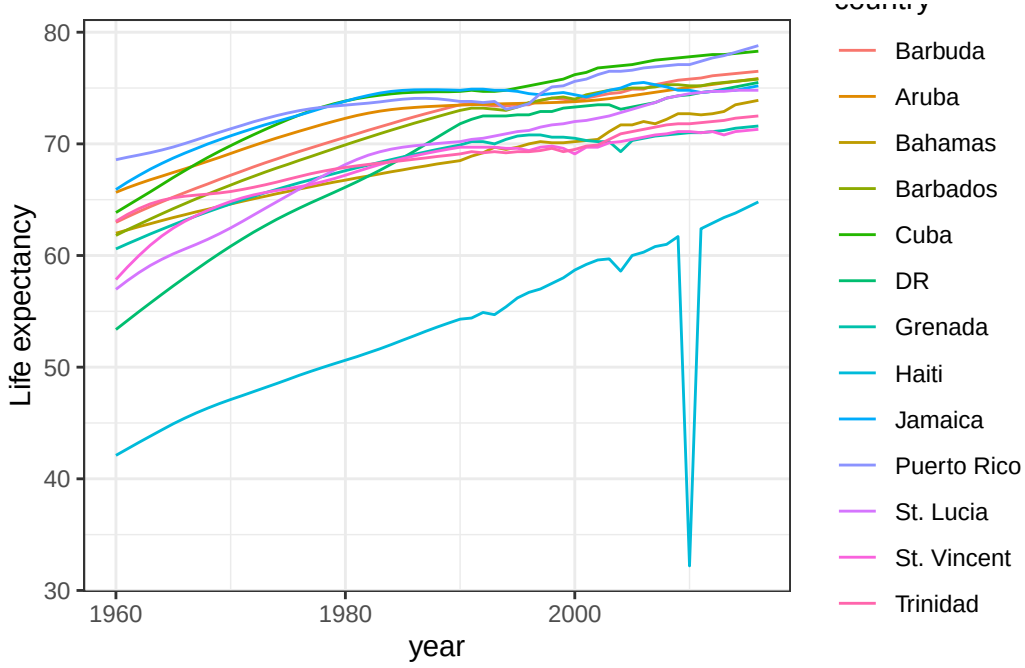
Recoding

```
library(dslabs)
data("gapminder")
gapminder %>%
  filter(region=="Caribbean") %>%
  ggplot(aes(year, life_expectancy, color=country)) +
  geom_line() + theme_bw()
```

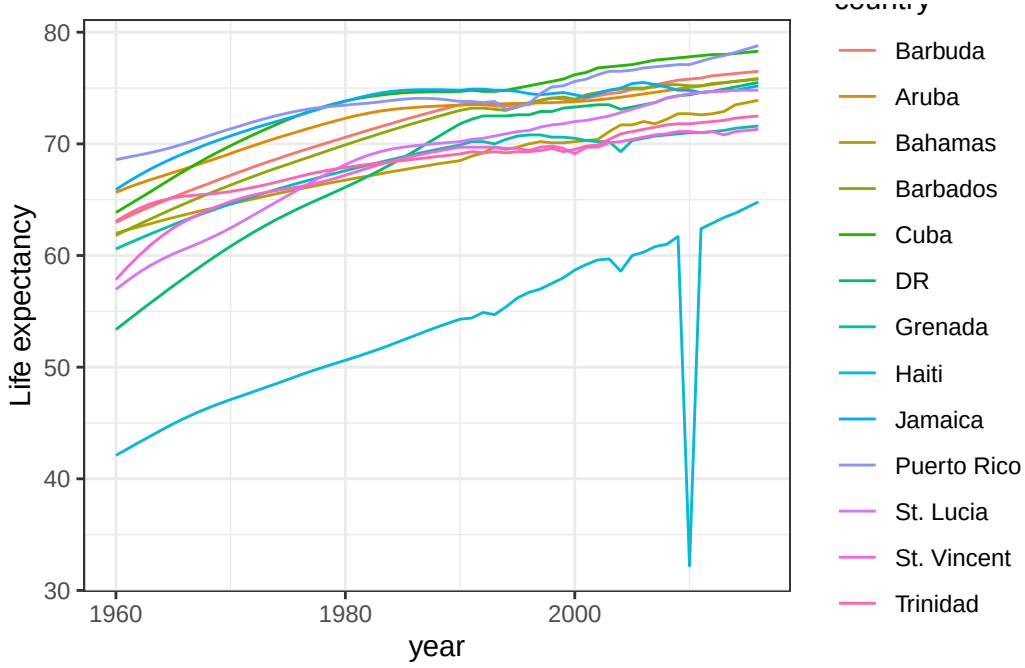


Utilizando recode() para reducir los nombres

```
# recode long country names and remake plot
gapminder %>% filter(region=="Caribbean") %>%
  mutate(country = recode(country,
                          'Antigua and Barbuda'="Barbuda",
                          'Dominican Republic' = "DR",
                          'St. Vincent and the Grenadines' = "St. Vincent",
                          'Trinidad and Tobago' = "Trinidad")) %>%
  ggplot(aes(year, life_expectancy, color = country)) +
  geom_line() + ylab("Life expectancy") + theme_bw()
```




```
graf_final <- gapminder %>% filter(region=="Caribbean") %>%
  mutate(country = recode(country,
                          'Antigua and Barbuda'="Barbuda",
                          'Dominican Republic' = "DR",
                          'St. Vincent and the Grenadines' = "St. Vincent",
                          'Trinidad and Tobago' = "Trinidad")) %>%
  ggplot(aes(year, life_expectancy, color = country)) +
  geom_line() + ylab("Life expectancy") + theme_bw()
graf_final
```




```
ggsave( "images/graf_final.png")
```

```
Saving 5.5 x 3.5 in image
```

Assessment Part 2: String Processing Part 3

Import raw Brexit referendum polling data from Wikipedia:

```
library(rvest)
library(tidyverse)
library(stringr)
url <- "https://en.wikipedia.org/w/index.php?title=Opinion_polling_for_the_United_Kingdom_Eu
tab <- read_html(url) %>% html_nodes("table")
polls <- tab[[6]] %>% html_table(fill = TRUE)
nrow(polls)
```

```
[1] 134
```

```
head(polls)
```

```
# A tibble: 6 x 9
  `Date(s) conducted` Remain  Leave   Undecided Lead  Sample     `Conducted by`
  <chr>               <chr>   <chr>   <chr>     <chr> <chr>      <chr>
1 Date(s) conducted   ""      ""      Undecided Lead  Sample     Conducted by
2 23 June 2016        "48.1%" "51.9%" N/A       3.8%  33,577,342 Results of the~
3 23 June             "52%"   "48%"   N/A       4%    4,772      YouGov
4 22 June             "55%"   "45%"   N/A       10%   4,700      Populus
5 20-22 June          "51%"   "49%"   N/A       2%    3,766      YouGov
6 20-22 June          "49%"   "46%"   1%        3%    1,592      Ipsos MORI
# i 2 more variables: `Polling type` <chr>, Notes <chr>
```

```
polls <- polls %>% slice(-1) %>% setNames(c("dates", "remain", "leave", "undecided", "lead",
#View(polls)
sum(str_detect(polls$remain, "%"))
```

```
[1] 129
```

```
str_replace(polls$undecided, "N/A", "0")
```

```
 [1] "0"
 [2] "0"
 [3] "0"
 [4] "0"
 [5] "1%"
 [6] "9%"
 [7] "0"
 [8] "11%"
 [9] "16%"
[10] "11%"
[11] "13%"
[12] "2%"
[13] "13%"
[14] "9%"
[15] "12%"
[16] "All official campaigning suspended until 19 June after the fatal shooting of Jo Cox M
[17] "9%"
[18] "13%"
[19] "16%"
[20] "11%"
[21] "3%"
[22] "15%"
```

```
[23] "5%"
[24] "7%"
[25] "9%"
[26] "13%"
[27] "3%"
[28] "0"
[29] "11%"
[30] "13%"
[31] "0"
[32] "11%"
[33] "9%"
[34] "5%"
[35] "11%"
[36] "16%"
[37] "16%"
[38] "13%"
[39] "15%"
[40] "9%"
[41] "3%"
[42] "12%"
[43] "18%"
[44] "13%"
[45] "16%"
[46] "10%"
[47] "3%"
[48] "14%"
[49] "12%"
[50] "7%"
[51] "5%"
[52] "14%"
[53] "10%"
[54] "5%"
[55] "21%"
[56] "22%"
[57] "16%"
[58] "11%"
[59] "13%"
[60] "11%"
[61] "11%"
[62] "14%"
[63] "0"
[64] "26%"
[65] "13%"
```

```
 [66] "17%"
 [67] "13%"
 [68] "10%"
 [69] "6%"
 [70] "9%"
 [71] "8%"
 [72] "11%"
 [73] "13%"
 [74] "6%"
 [75] "The EU referendum campaign officially begins.[31]"
 [76] "28%"
 [77] "16%"
 [78] "17%"
 [79] "30%"
 [80] "17%"
 [81] "12%"
 [82] "HM Government starts sending a pro-Remain pamphlet to 27 million UK households and beg
 [83] "16%"
 [84] "18%"
 [85] "13%"
 [86] "5%"
 [87] "18%"
 [88] "30%"
 [89] "14%"
 [90] "0"
 [91] "12%"
 [92] "10%"
 [93] "19%"
 [94] "11%"
 [95] "17%"
 [96] "19%"
 [97] "4%"
 [98] "16%"
 [99] "16%"
[100] "7%"
[101] "15%"
[102] "19%"
[103] "18%"
[104] "19%"
[105] "19%"
[106] "18%"
[107] "18%"
[108] "15%"
```

```
[109] "0"
[110] "25%"
[111] "David Cameron announces the date of UK's In/Out EU referendum after an EU summit in Br
[112] "25%"
[113] "17%"
[114] "10%"
[115] "23%"
[116] "19%"
[117] "10%"
[118] "25%"
[119] "18%"
[120] "10%"
[121] "17%"
[122] "19%"
[123] "19%"
[124] "20%"
[125] "9%"
[126] "14%"
[127] "10%"
[128] "18%"
[129] "0"
[130] "17%"
[131] "22%"
[132] "12%"
[133] "18%"
```