R4Finanace

Data Mining

J. Gibrán Peniche

Versión 0.0.1

2020/06/19

jgpeniche

PenicheGibran

jgpeniche@gmail.com

Recap

- R y RStudio no son lo mismo
- Sintáxis básica
    1. `for(i in x){ #DoStuff }`
    2. `if( condition ){ #DoStuff }else{ #DoMoreStuff }`
    3. `foo <- function( x, y, z, ...){ #DoStuffWithXYZ return(something) }`

Recap

- Sintáxis Avanzada

    1. Sustituir `functionA(functionB(functionC(functionD(functionD(object)))))`

    2. Por

```
library(magritter)

object %>%
  functionA() %>%
  functionB() %>%
  functionC() %>%
  functionD() ...
```

Recap

- Tidy Data:
    1. Cada columna es una variable
    2. Cada renglón es una observación
    3. Cada tabla es una unidad observacional

Agenda

1. ¿Qué es data manipulation?
2. Verbos `dplyr`
3. Long format

Antes...

Solución tarea

```
stocks <- EuStockMarkets %>%
  as_tibble() %>%
  janitor::clean_names()

returns <- function(precios, type = c('log','arithmetic')){
  returns <- c()
  returns[1] <- 0
  if(type == 'log'){
      for(i in 2:length(precios))
      returns[i] <- log(precios[i]/precios[i-1])
    }else{
      if(type == 'arithmetic'){
        for(i in 2:length(precios)){
        returns[i] <- precios[i]/precios[i-1] - 1
       }
      }else{
       return(print('Not a valid return type'))
      }
    }
  returns %>%
    as_tibble() %>%
    return()
}
```

Solución tarea

```
stocks %$%
  returns(precios = dax, type = 'log')
```

```
## # A tibble: 1,860 x 1
##       value
##       <dbl>
##  1  0
##  2 -0.00933
##  3 -0.00442
##  4  0.00900
##  5 -0.00178
##  6 -0.00468
##  7  0.0124
##  8  0.00576
##  9 -0.00287
## 10  0.00635
## # ... with 1,850 more rows
```

Solución tarea

```
stocks %$%
  returns(precios = dax, type = 'arithmetic')
```

```
## # A tibble: 1,860 x 1
##       value
##       <dbl>
##  1  0
##  2 -0.00928
##  3 -0.00441
##  4  0.00904
##  5 -0.00178
##  6 -0.00467
##  7  0.0125
##  8  0.00578
##  9 -0.00287
## 10  0.00637
## # ... with 1,850 more rows
```

Solución tarea

```
stocks %$%
  returns(precios = dax, type = 'otro')
```

```
## [1] "Not a valid return type"
```

Algo que no hemos explicado

R como proyecto open-source

Como ya mencionamos durante la primera sesión, R es un proyecto **open-source**, entre otras cosas, esto significa que tiene la cualidad de que cualquiera puede **participar**

Generalmente la participación se manifiesta en la forma de *issues*, *pull requests* y *librerias*

Desde la primera sesión utilizamos la línea `library(#Somelibrary)` pero, ¿Qué es una libreria?

Una libreria es un **conjunto** de funciones con un fin específico

RStudio por default solo carga cierto número de librerias (por cuestiones de tiempo de inicio), por eso si requerimos funciones con un fin más específico es necesario primero **instalar** la libreria y luego llamarla con `library()` para tener disponibles las funciones

`library(#somelibrary)` vs. `somelibrary::`

El hecho de llamar una biblioteca usando `library()` implica que **todas** las funciones van a a ser llamadas a la memoria.

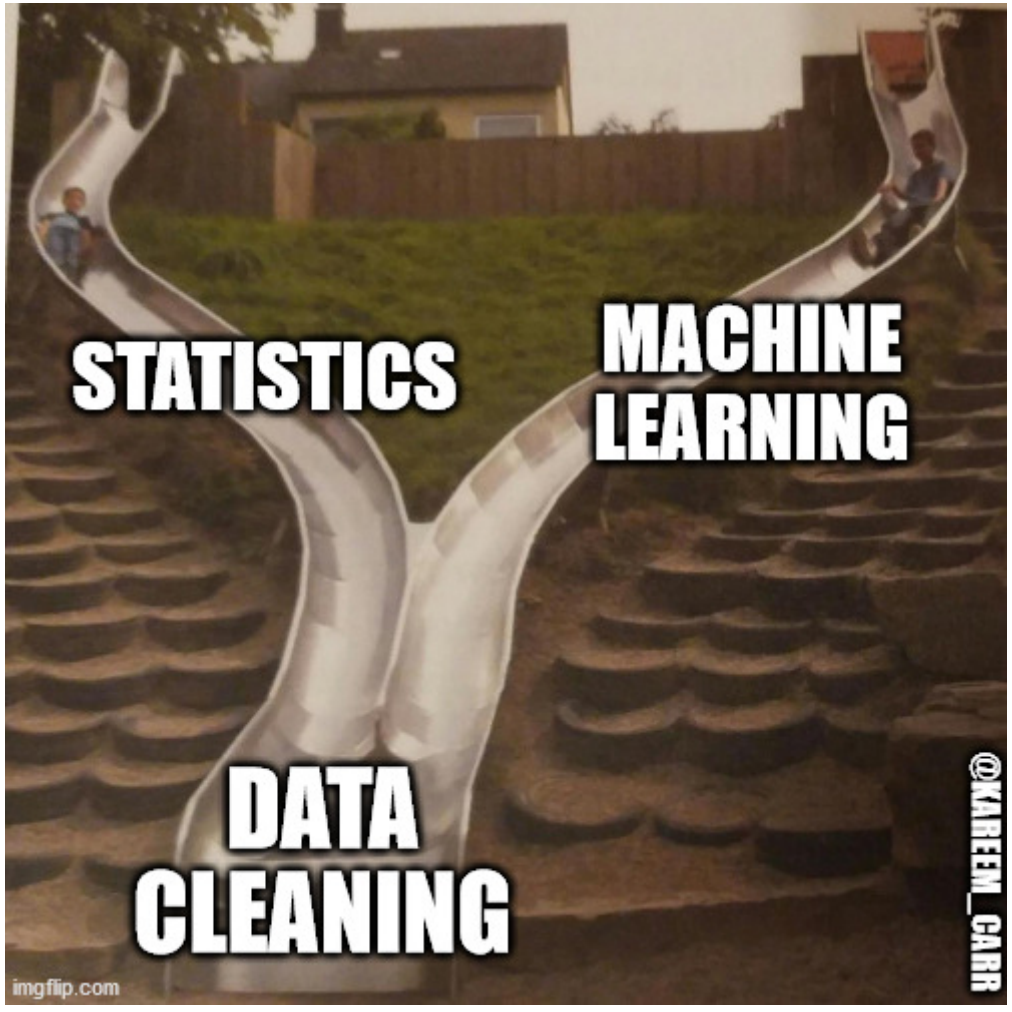
Esto significa que estas funciones van a ocupar espacio de manera permanente en la memoria

Si solo se va a utilizar la función una vez, es buena práctica no llamar toda la libreria a la memoria y utlizar `libreria::función`

Esto optimiza el espacio disponible y se vuelve reelavante cuando se trabajan bases de datos grandes o se requiere espacio para procedimientos computacionalmente exhaustivos

Data Mining

¿Qué es Data Mining?



¿Qué Data Mining?

1. Es el proceso de **limpieza** de los datos para llevarlos al formato de **TIDY DATA**

2. Es la manipulación para obtener resúmenes **minimales**, **suficientes** y en particular desde el punto de vista del **análisis exploratorio de datos**

¿Cómo se realiza esto en R?

En la comunidad de R existen dos librerias para realizar este proceso y una gran polémica sobre cual es mejor

`dplyr` que es cercano a la sintáxis de SQL y `data.table` que es más cercano a la sintáxis de `pandas` en python

Con data.table

```
library(data.table)
diamondsDT <- data.table(diamonds)
diamondsDT[
  cut != "Fair",
  .(AvgPrice = mean(price),
    MedianPrice = as.numeric(median(price)),
    Count = .N
  ),
  by = cut
][
  order(-Count)
]
```

Con dplyr

```
library(dplyr)
diamonds %>%
  filter(cut != "Fair") %>%
  group_by(cut) %>%
  summarize(
    AvgPrice = mean(price),
    MedianPrice = as.numeric(median(price)),
    Count = n()
  ) %>%
  arrange(desc(Count))
```

dplyr vs. data.table

Personalente encuentro `dplyr` mucho más intuitivo, incluso si no estás familiriaƶdo con el paquete

Además soy fan del *Hadleyverse*

Es la herramienta que vamos a utiliƶar de aquí en adelante

Los verbos de `dplyr`

Partiendo del objeto *data frame*

Todos los verbos en `dplyr` aceptan algo llamado *tidyselect* esto quiere decir que no hace falta llamar el nombre de la solumna usando commillas

1. **select** reduce el objeto a ciertas columnas
2. **filter** filtra los datos del *data frame* de acuerdo a cierta condición lógica
3. **group_by** realiza un paso intermedio que agrupa las observaciones de acuerdo a cierto valor para después hacer un resúmen
4. **summarise** permite hacer resumenes de las columunas del *data frame**

Un ejemplo práctico

```
baby_names <- babynames::babynames %>%
  as_tibble()

baby_names %>%
  glimpse()
```

```
## Rows: 1,924,665
## Columns: 5
## $ year <dbl> 1880, 1880, 1880, 1880, 1880, 1880, 1880, 1880, 1880, 1880, 18...
## $ sex  <chr> "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F...
## $ name <chr> "Mary", "Anna", "Emma", "Elizabeth", "Minnie", "Margaret", "Id...
## $ n    <int> 7065, 2604, 2003, 1939, 1746, 1578, 1472, 1414, 1320, 1288, 12...
## $ prop <dbl> 0.07238359, 0.02667896, 0.02052149, 0.01986579, 0.01788843, 0....
```

Un ejemplo práctico

```
baby_names %>%
  select(year,name,n)
```

```
## # A tibble: 1,924,665 x 3
##     year name           n
##    <dbl> <chr>      <int>
##  1  1880 Mary        7065
##  2  1880 Anna        2604
##  3  1880 Emma        2003
##  4  1880 Elizabeth   1939
##  5  1880 Minnie      1746
##  6  1880 Margaret    1578
##  7  1880 Ida         1472
##  8  1880 Alice       1414
##  9  1880 Bertha      1320
## 10  1880 Sarah       1288
## # ... with 1,924,655 more rows
```

Un ejemplo práctico

```
baby_names %>%
  select(-prop)
```

```
## # A tibble: 1,924,665 x 4
##     year sex   name          n
##    <dbl> <chr> <chr>     <int>
##  1  1880 F     Mary       7065
##  2  1880 F     Anna       2604
##  3  1880 F     Emma       2003
##  4  1880 F     Elizabeth  1939
##  5  1880 F     Minnie     1746
##  6  1880 F     Margaret   1578
##  7  1880 F     Ida        1472
##  8  1880 F     Alice      1414
##  9  1880 F     Bertha     1320
## 10  1880 F     Sarah      1288
## # ... with 1,924,655 more rows
```

Un ejemplo práctico

```
baby_names %>%
  filter(sex == 'F')
```

```
## # A tibble: 1,138,293 x 5
##     year sex   name           n   prop
##    <dbl> <chr> <chr>      <int>  <dbl>
##  1  1880 F     Mary        7065 0.0724
##  2  1880 F     Anna        2604 0.0267
##  3  1880 F     Emma        2003 0.0205
##  4  1880 F     Elizabeth   1939 0.0199
##  5  1880 F     Minnie      1746 0.0179
##  6  1880 F     Margaret    1578 0.0162
##  7  1880 F     Ida         1472 0.0151
##  8  1880 F     Alice       1414 0.0145
##  9  1880 F     Bertha      1320 0.0135
## 10  1880 F     Sarah       1288 0.0132
## # ... with 1,138,283 more rows
```

Un ejemplo práctico

```
baby_names %>%
  filter(sex == 'F' & name == 'Ida')
```

```
## # A tibble: 138 x 5
##     year sex   name      n   prop
##    <dbl> <chr> <chr> <int>  <dbl>
##  1  1880 F     Ida    1472 0.0151
##  2  1881 F     Ida    1439 0.0146
##  3  1882 F     Ida    1673 0.0145
##  4  1883 F     Ida    1634 0.0136
##  5  1884 F     Ida    1882 0.0137
##  6  1885 F     Ida    1854 0.0131
##  7  1886 F     Ida    2049 0.0133
##  8  1887 F     Ida    1929 0.0124
##  9  1888 F     Ida    2229 0.0118
## 10  1889 F     Ida    2122 0.0112
## # ... with 128 more rows
```

Un ejemplo práctico

```
baby_names %>%
  group_by(name) %>%
  summarise(media = mean(n))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 97,310 x 2
##    name       media
##    <chr>      <dbl>
##  1 Aaban      10.7
##  2 Aabha       7
##  3 Aabid       5
##  4 Aabir       5
##  5 Aabriella   6.4
##  6 Aada        5
##  7 Aadam       9.77
##  8 Aadan      11.8
##  9 Aadarsh    11.7
## 10 Aaden     259.
## # ... with 97,300 more rows
```

Un ejemplo práctico

```
baby_names %>%
  filter(sex == 'F' & name == 'Ida') %>%
  group_by(year) %>%
  summarise(media = mean(n))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 138 x 2
##     year media
##    <dbl> <dbl>
##  1  1880  1472
##  2  1881  1439
##  3  1882  1673
##  4  1883  1634
##  5  1884  1882
##  6  1885  1854
##  7  1886  2049
##  8  1887  1929
##  9  1888  2229
## 10  1889  2122
## # ... with 128 more rows
```

Ahora ustedes

1. Asigna la base de datos `ChickWeight` a una variable y transformal a un *data frame* con `as_tibble()` (Recuerda que necesitas llamar *magritter* y *dplyr*)

2. Revisa la documentación (`?función/paquete/objeto`) sobre la base de datos

3. Realiza un exploratorio de las variables en tu *df* con `glimpse()` y luego con `summary()`

4. *Filtra* por el tiempo mayor a 0, *agrupa* por tipo de dieta y haz un *resúmen* de la media y desviación estandar del peso

5. ¿De acuerdo a tus análisis que dieta parece ser más efetiva en términos de peso promedio del pollo?

05:00

Solución

```
pollos <- ChickWeight %>%
  as_tibble() %>%
  janitor::clean_names()

pollos %>%
  filter( time > 0) %>%
  group_by(diet) %>%
  summarise(mean_weight = mean(weight))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 4 x 2
##   diet  mean_weight
##   <fct>       <dbl>
## 1 1            109.
## 2 2            130.
## 3 3            152.
## 4 4            144.
```

Limpieza de Datos

¿Cuál es el problema de estos datos en el contexto de Tidy Data

```
crimes <- USArrests
crimes$state <- rownames(crimes)
crimes <- crimes %>%
  as_tibble() %>%
  janitor::clean_names()

crimes
```

```
## # A tibble: 50 x 5
##    murder assault urban_pop  rape state
##     <dbl>   <int>     <int> <dbl> <chr>
##  1   13.2     236        58  21.2 Alabama
##  2   10       263        48  44.5 Alaska
##  3    8.1     294        80  31   Arizona
##  4    8.8     190        50  19.5 Arkansas
##  5    9       276        91  40.6 California
##  6    7.9     204        78  38.7 Colorado
##  7    3.3     110        77  11.1 Connecticut
##  8    5.9     238        72  15.8 Delaware
##  9   15.4     335        80  31.9 Florida
## 10   17.4     211        60  25.8 Georgia
```

Limpiea de Datos

Tenemos 3 variables **Estado**, **Crímen** y **Población**

Sin embargo, la estrucutura de los datos no hace sentido con la semántica de los mismos

¿Cómo corregimos esto?

La mayoría de las veces nos vamos a encontrar este tipo de bases de datos y lo que queremos es pasarlas a formato *long* de tal manera que se vean así:

Limpiea de Datos

```
## # A tibble: 200 x 3
##    state   crime     count
##    <chr>   <chr>     <dbl>
##  1 Alabama murder     13.2
##  2 Alabama assault   236
##  3 Alabama urban_pop  58
##  4 Alabama rape       21.2
##  5 Alaska  murder     10
##  6 Alaska  assault   263
##  7 Alaska  urban_pop  48
##  8 Alaska  rape       44.5
##  9 Arizona murder      8.1
## 10 Arizona assault   294
## # ... with 190 more rows
```

Ahora ustedes (2)

1. Busquen la documentación de `tidyr::pivot_longer()` y realicen la misma transformación que yo acabo de hacer

2. Guarden este nuevo *data frame* como `crimes_long`

3. *Agrupen* por **estado** y por **crímen** y hagan un *resúmen* del *número promedio* por crímen

4. ¿Qué estado es el más violento en términos de asesinato y robo?

10:00

Solución

```
library(tidyr)
crimes_long <- crimes %>%
  pivot_longer(-state, names_to = 'crime', values_to = 'count')

violence <- crimes_long %>%
  filter(crime %in% c('murder','assault')) %>%
  group_by(crime, state) %>%
  summarise(mean_crimes = mean(count))

max_assault <- violence %>%
  filter(crime  == 'assault' & mean_crimes == max(mean_crimes))

max_murder <- violence %>%
  filter(crime  == 'murder' & mean_crimes == max(mean_crimes))
```

Solución

```
max_assault
```

```
## # A tibble: 1 x 3
## # Groups:   crime [1]
##   crime   state          mean_crimes
##   <chr>   <chr>                <dbl>
## 1 assault North Carolina         337
```

Solución

```
max_murder
```

```
## # A tibble: 1 x 3
## # Groups:   crime [1]
##   crime  state   mean_crimes
##   <chr>  <chr>         <dbl>
## 1 murder Georgia        17.4
```