BEDU APUNTE 2

Victor Miguel Terrón Macias

13/1/2021

INTRODUCCIÓN

En el presente documento presento mis notas correspondientes con la sesión 2 de BEDU-Manipulación de datos.

Se estudian conceptos básicos de estadística para contextualizar al alumno y sentar las bases de la estructura del curso. Se proporcionan algunas herramientas básicas y muy útiles en la práctica para que el alumno pueda importar datos a `R` desde diferentes tipos de archivos, con estos datos en `R`, podrá filtrar filas, seleccionar variables, transformar variables y en general manipular los datos para llevarlos a una forma deseada.

En ésta sesion estudiaremos temas relacionados con los siguientes puntos: * Medidas de posición y de dispersión * Funciones para observar algunas características de objetos en R * Funciones para filtrar filas, seleccionar variables y transformar un data frame en `R` * Funciones para combinar data frames en R por filas o por columnas * Importación de datos a `R` desde diferentes tipos de archivos

Los ejemplos y retos de la sesión son los siguientes:

EJEMPLO 1 MEDIDAS DE TENDENCIA CENTRAL, DE POSICIÓN Y DE DISPERSIÓN

OBJETIVO

- Aprender a calcular medidas de tendencia central, de posición y de dispersión para conjuntos de datos, con ayuda de funciones de R.
- Interpretar los resultados obtenidos

MEDIDAS DE TENDENCIA CENTRAL

Para sacar las medidas de tendencia central tenemos que utilizar las siguientes funciones:

```
x=c(1:10,23:54,1)
mean(x)
```

```
[1] 29.95349
```

```
print("Como pudimos ver sacamos el promedio de un vector en este caso pero lo mismo se puede hacer con
```

```
[1] "Como pudimos ver sacamos el promedio de un vector en este caso pero lo mismo se puede hacer con ot
```

```r
print("")
```

```
[1] ""
```

```r
print("Si queremos sacar la mediana tenemos que utilizar la función: ")
```

```
[1] "Si queremos sacar la mediana tenemos que utilizar la función: "
```

```r
median(x)
```

```
[1] 33
```

```r
print("Si deseamos obtener la moda de un conjunto de datos entonces descargamos la librería DescTools c
```

```
[1] "Si deseamos obtener la moda de un conjunto de datos entonces descargamos la librería DescTools con
```

```r
#install.packages("DescTools",dependencies = T)
library(DescTools)
Mode(x)
```

```
[1] 1
attr(,"freq")
[1] 2
```

Así que en resumen tenemos lo siguiente: * $media = mean(datos)$ * $mediana = median(datos)$ * $moda = Mode(x)$

MEDIDAS DE POSICIÓN

Las medidas de posición son los cuartiles, deciles y percentiles.

Para obtener los cuantiles se utiliza la función $quantile(datos)$, aplicandolo tenemos lo siguiente:

```r
x<-c(1,2,3,4,5,6,7,8)
quantile(x,0.15)#PERCENTIL
```

```
 15%
2.05
```

```r
quantile(x,0.5)#SEGUNDO CUARTIL
```

```
50%
4.5
```

```r
quantile(x,seq(0.1,1,by = 0.1))#DECILES
```

```
 10%  20%  30%  40%  50%  60%  70%  80%  90% 100%
 1.7  2.4  3.1  3.8  4.5  5.2  5.9  6.6  7.3  8.0
```

MEDIDAS DE DISPERSIÓN

Podemos calcular el rango intercualítico en R con la función $IQR()$ por ejemplo:

```
IQR(x)
```

```
[1] 3.5
```

Recordemos a su vez que el rango intercualítico es el tercer cuartil menos el primer cuartil por lo que otra forma de calcularlo es:

```
quantile(x,0.75)-quantile(x,0.25)
```

```
## 75%
## 3.5
```

La varianza y la desviación estandar muestral en R se calculan con las siguientes funciones respectivamente:

```
var(x)#VARIANZA
```

```
## [1] 6
```

```
sd(x)#DESVIACIÓN ESTANDAR
```

```
## [1] 2.44949
```

RETO 1

- INSTRUCCIONES:
    * Calcule, la media, mediana y moda de los valores en x
    * Obtenga los deciles de los números en x
    * Encuentre la rango intercuartílico, la desviación estándar y varianza muestral de las mediciones en x

```
set.seed(134)
x <- round(rnorm(1000, 175, 6), 1)
mean(x)#Media
```

```
## [1] 174.9169
```

```
median(x)#Mediana
```

```
## [1] 174.8
```

```
Mode(x)#Moda
```

```
## [1] 174.1 175.9
## attr(,"freq")
## [1] 12
```

```r
quantile(x,seq(0.1,1,by = 0.1))
```

```
##    10%    20%    30%    40%    50%    60%    70%    80%    90%   100%
## 167.30 170.00 171.90 173.50 174.80 176.30 178.03 180.00 182.70 196.60
```

```r
IQR(x)
```

```
## [1] 8
```

EJEMPLO 2 CARACTERISTICAS DE STR, SUMMARY HEAD Y VIEW

OBJETIVO

- Conocer mejor conjuntos de datos guardados como data frames en R de una forma rápida, mediante algunas funciones útiles y de uso común.

DESARROLLO

FUNCIÓN *str*

str es una función que muestra de manera compacta la estructura interna de un objeto de R. Por ejemplo, si usamos como argumento de str el conjunto de datos iris que podemos encontrar en R tenemos lo siguiente:

```r
str(iris)
```

```
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

La salida de la instrucción nos muestra el tipo de objeto, número de observaciones y de variables, así como el tipo de dato al que corresponde cada variable.

FUNCION *summary*

La función *summary*() es una función genérica para obtener resumenes de diferentes objetos de R y aplicandola tenemos lo siguiente:

```r
summary(x)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  151.8   171.0   174.8   174.9   179.0   196.6
```

```r
summary(mtcars)
```

```
      mpg             cyl             disp             hp       
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
      drat             wt             qsec             vs        
 Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
 1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
 Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
 Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
 3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
 Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
       am              gear            carb      
 Min.   :0.0000   Min.   :3.000   Min.   :1.000  
 1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
 Median :0.0000   Median :4.000   Median :2.000  
 Mean   :0.4062   Mean   :3.688   Mean   :2.812  
 3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
 Max.   :1.0000   Max.   :5.000   Max.   :8.000  
```

Es útil para obtener resumenes de los resultados de diferentes modelos, por ejemplo:

```r
set.seed(57)
x <- rnorm(35)
e <- rnorm(35)
y <- 5 + 2*x + e
modelo <- lm(y~x)
summary(modelo)
```

```

Call:
lm(formula = y ~ x)

Residuals:
     Min       1Q   Median       3Q      Max 
-2.87555 -0.46075 -0.04895  0.74275  1.73543 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   5.1586     0.1823   28.30  < 2e-16 ***
x             1.8628     0.1438   12.96 1.71e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.078 on 33 degrees of freedom
Multiple R-squared:  0.8357,    Adjusted R-squared:  0.8308 
F-statistic: 167.9 on 1 and 33 DF,  p-value: 1.707e-14
```

FUNCION *head*

La función *head* devuelve la primera parte de un data frame, tabla, matriz, vector o función. Por ejemplo, al usar el data frame mtcars como argumento de la función head, se devolverán únicamente las primeras seis filas del data frame.

La función *tail* devuelve los ultimos 6 datos del data frame.

La función *view* devuelve en otra ventana el dataframe completo, estilo como hoja de calculo.

```
head(mtcars)
```

```
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
tail(mtcars)
```

```
                mpg cyl  disp  hp drat    wt qsec vs am gear carb
Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.7  0  1    5    2
Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.9  1  1    5    2
Ford Pantera L 15.8   8 351.0 264 4.22 3.170 14.5  0  1    5    4
Ferrari Dino   19.7   6 145.0 175 3.62 2.770 15.5  0  1    5    6
Maserati Bora  15.0   8 301.0 335 3.54 3.570 14.6  0  1    5    8
Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.6  1  1    4    2
```

```
View(mtcars)
```

RETO 2

Considere el data frame mtcars de `R`. * INSTRUCCIONES: Utilice las funciones

* *str*
* *summary*
* *head*
* *View*

```
str(mtcars)
```

```
'data.frame':   32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
```

```
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

```
summary(mtcars)
```

```
      mpg             cyl             disp             hp       
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
      drat             wt             qsec             vs        
 Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
 1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
 Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
 Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
 3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
 Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
       am              gear            carb      
 Min.   :0.0000   Min.   :3.000   Min.   :1.000  
 1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
 Median :0.0000   Median :4.000   Median :2.000  
 Mean   :0.4062   Mean   :3.688   Mean   :2.812  
 3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
 Max.   :1.0000   Max.   :5.000   Max.   :8.000  
```

```
head(mtcars)
```

```
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
View(mtcars)
```

EJEMPLO 3 PAQUETE *dplyr* y aplicaciones

OBJETIVO

- Utilizar funciones del paquete dplyr para seleccionar columnas, filtrar filas y en general manipular o transformar datos en un data frame y llevarlos a una forma deseada

DESARROLLO

El paquete *dplyr* cuenta con varias funciones muy útiles para manipular y transformar data frames. Una vez instalado el paquete dplyr, puede cargarlo en R de la siguiente manera (sin mensajes ni advertencias)

```
suppressMessages(suppressWarnings(library(dplyr)))
```

Vamos a descargar archivos csv que contienen datos del covid-19 para mostrar como funcionan algunas funciones del paquete dplyr. Las url desde las cuales descargamos los datos son las siguientes

```
url1 <- "https://data.humdata.org/hxlproxy/data/download/time_series_covid19_confirmed_global_narrow.csv
url2 <- "https://data.humdata.org/hxlproxy/data/download/time_series_covid19_deaths_global_narrow.csv?d
#DESCARGA DE CSV
download.file(url = url1, destfile = "st19ncov-confirmados.csv", mode = "wb")
download.file(url = url2, destfile = "st19ncov-muertes.csv", mode = "wb")
#IMPORTACIÓNDEDATOS
conf <- read.csv("st19ncov-confirmados.csv")
mu <- read.csv("st19ncov-muertes.csv")
str(conf);str(mu)
head(conf);head(mu)
#AHORA SELECCIONAMOS TODAS LAS FILAS EXCEPTO LA PRIMERA, PARA EXCLUIR UTILIZAMOS
sconf<-conf[-1,]
smu<-mu[-1,]
# Con la función select del paquete dplyr, del data frame de casos confirmados
#seleccionamos únicamente las columnas de país, fecha y número acumulado de
#casos
sconf <- select(sconf, Country.Region, Date, Value) # Va el origen de datos, País, fecha y acumulado de
#Con la función rename, renombramos las columnas correspondientes al país y al
#número acumulado de infectados por covid-19
sconf <- rename(sconf, Country = Country.Region, Infectados = Value)
str(sconf)
```

Como cada una de las columnas del último data frame aparecen como factor, con la función mutate transformamos las columnas correspondientes a fechas y a número de infectados, esto para que R reconozca como fechas la columna correspondiente y como números los elementos de la columna que indica el acumulado de casos. Es decir al aplicar str podemos ver el tipo de datos que tiene el data frame y todos fueron convertidos a char, por ello se requiere usar mutate.

```
sconf <- mutate(sconf, Date = as.Date(Date, "%Y-%m-%d"), Infectados = as.numeric(as.character(Infectados
```

Hacemos algo similar con el data frame correspondiente al número acumulado de muertos.

```
smu <- select(smu, Country.Region, Date, Value) # Seleccionamos país, fecha y acumulado de muertos
smu <- rename(smu, Country = Country.Region, Muertos = Value) # Renombramos
smu <- mutate(smu, Date = as.Date(Date, "%Y-%m-%d"), Muertos = as.numeric(as.character(Muertos))) # La
```

Unimos infectados, muertos acumulados para cada fecha.

```
scm<-merge(smu,sconf)# Unimos infectados y muertos acumulados para cada fecha
```

Aplicando filtros tenemos lo siguiente:

```r
mex <- filter(Scm, Country == "Mexico") # Seleccionamos sólo a México
mex <- filter(mex, Infectados != 0) # Primer día de infectados
```

Podemos crear otras columnas de interés o variables con ayuda de la función *mutate*. * LAG Y DIFF ES EQUIVALENCIA, LAG DETECTA QUE ANTERORMENTE NO HUBO NADA ENTONCES DESPLAZA UNA LINEA EL CONTEO * CUANDO SE UTILIZA EL COMANDO RENAME(OBJETO_PROVENIENTEDEDATOS) * MUTATE TRANSFORMA DE UNA VARIABLE A OTRA

```r
mex <- mutate(mex, NI = c(1, diff(Infectados))) # Nuevos infectados por día
mex <- mutate(mex, NM = c(0, diff(Muertos))) # Nuevos muertos por día

mex <- mutate(mex, Letalidad = round(Muertos/Infectados*100, 1)) # Tasa de letalidad

mex <- mutate(mex, IDA = lag(Infectados), MDA = lag(Muertos)) # Valores día anterior
mex <- mutate(mex, FCI = Infectados/IDA, FCM = Muertos/MDA) # Factores de Crecimiento
mex <- mutate(mex, Dia = 1:dim(mex)[1]) # Días de contingencia
head(mex);tail(mex)
```

EJEMPLO 4-FUNCIONES *cbind*() y *rbind*()

Las funciones *cbind*() & *rbind*() sirven para combinar elementos.

FUNCION *cbind*()

La función *cbind* toma una sucesión de argumentos que pueden ser vectores, matrices o data frames y los combina por columnas, por ejemplo:

```r
cbind(1:10, 11:20, 21:30)
```

```
      [,1] [,2] [,3]
 [1,]    1   11   21
 [2,]    2   12   22
 [3,]    3   13   23
 [4,]    4   14   24
 [5,]    5   15   25
 [6,]    6   16   26
 [7,]    7   17   27
 [8,]    8   18   28
 [9,]    9   19   29
[10,]   10   20   30
```

```r
cbind(1:10, matrix(11:30, ncol =2))
```

```
      [,1] [,2] [,3]
 [1,]    1   11   21
 [2,]    2   12   22
 [3,]    3   13   23
 [4,]    4   14   24
```

```
 [5,]    5   15   25
 [6,]    6   16   26
 [7,]    7   17   27
 [8,]    8   18   28
 [9,]    9   19   29
[10,]   10   20   30
```

```r
cbind(data.frame(x = 1:10, y = 11:20), z = 21:30)
```

```
    x  y  z
1   1 11 21
2   2 12 22
3   3 13 23
4   4 14 24
5   5 15 25
6   6 16 26
7   7 17 27
8   8 18 28
9   9 19 29
10 10 20 30
```

FUNCION *rbind*()

La función *rbind* funciona de manera similar a *cbind*, pero en lugar de combinar los objetos por columnas, los combina por filas, como ejemplo tenemos lo siguiente:

```r
df1 <- data.frame(x = 1:5, y = 6:10, z = 16:20)
df2 <- data.frame(x = 51:55, y = 101:105, z = 151:155)
df1; df2
```

```
  x  y  z
1 1  6 16
2 2  7 17
3 3  8 18
4 4  9 19
5 5 10 20
```

```
   x   y   z
1 51 101 151
2 52 102 152
3 53 103 153
4 54 104 154
5 55 105 155
```

```r
rbind(df1, df2)
```

```
    x   y   z
1   1   6  16
2   2   7  17
3   3   8  18
4   4   9  19
```

```
5   5  10  20
6  51 101 151
7  52 102 152
8  53 103 153
9  54 104 154
10 55 105 155
```

EJEMPLO 5 FUNCIONES APPLY, LAPPLY Y DO.CALL

OBJETIVO

- Realizar operaciones por filas o columnas en un arreglo
- Aplicar funciones a elementos de vectores o listas y obtener una lista con los resultados
- Combinar múltiples data frames en un único data frame con la ayuda de funciones de una manera fácil y rápida

DESARROLLO

Primero comenzaremos con la función *apply*:

La función *apply* regresa un vector, arreglo o lista de valores obtenidos al aplicar una función a los márgenes de un arreglo o matriz. Por ejemplo

```
X <- matrix(1:49, ncol = 7)
X
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    1    8   15   22   29   36   43
[2,]    2    9   16   23   30   37   44
[3,]    3   10   17   24   31   38   45
[4,]    4   11   18   25   32   39   46
[5,]    5   12   19   26   33   40   47
[6,]    6   13   20   27   34   41   48
[7,]    7   14   21   28   35   42   49
```

```
apply(X, 1, mean) # cálculo de la media para las filas 1 es filas
```

```
[1] 22 23 24 25 26 27 28
```

```
apply(X, 2, median) # cálculo de la mediana para las columnas 2 es columnas
```

```
[1]  4 11 18 25 32 39 46
```

La función *lapply* se usa de la siguiente manera lapply(X, FUN, ...) donde X puede ser un vector o una lista, FUN es una función que será aplicada a cada elemento de X y ... representa argumentos opcionales para FUN. lapply regresa una lista de la misma longitud que X, en donde cada elemento de la lista es el resultado de aplicar FUN al elemento que corresponde de X.

Vamos a utilizar *lapply* para leer un conjunto de archivos csv de manera consecutiva y "rápida", para esto debemos especificar un directorio de trabajo y descargar los archivos csv en nuestro directorio, por ejemplo, puede crear la carpeta soccer para descargar los datos. El directorio debe estar vacío

```r
u1011 <- "https://www.football-data.co.uk/mmz4281/1011/SP1.csv"
u1112 <- "https://www.football-data.co.uk/mmz4281/1112/SP1.csv"
u1213 <- "https://www.football-data.co.uk/mmz4281/1213/SP1.csv"
u1314 <- "https://www.football-data.co.uk/mmz4281/1314/SP1.csv"

download.file(url = u1011, destfile = "SP1-1011.csv", mode = "wb")
download.file(url = u1112, destfile = "SP1-1112.csv", mode = "wb")
download.file(url = u1213, destfile = "SP1-1213.csv", mode = "wb")
download.file(url = u1314, destfile = "SP1-1314.csv", mode = "wb")
```

Podemos guardar los csv descargados en una lista y leerlos a la vez:

```r
lista <- lapply(dir(), read.csv) # Guardamos los archivos en lista
```

Los elementos de lista son los archivos csv leidos y se encuentran como data frames:

```r
library(dplyr)
lista <- lapply(lista, select, Date:FTR) # seleccionamos solo algunas columnas de cada data frame
head(lista[[1]]); head(lista[[2]]); head(lista[[3]]); head(lista[[4]])
```

Cada uno de los data frames que tenemos en lista, los podemos combinar en un único data frame utilizando las funciones rbind y do.call de la siguiente manera.

Para combinar algunos data.frames podemos utilizar rbind() y do.call de la siguiente forma:

```r
data <- do.call(rbind, lista)
head(data)
dim(data)
```

RETO 3

OBJETIVO

- Importar múltiples archivos csv a R y combinar el contenido de estos archivos como un único data frame

DESARROLLO

1. Descargue los archivos csv que corresponden a las temporadas 2017/2018, 2018/2019, 2019/2020 y 2020/2021 de la Bundesliga 1 y que se encuentran en el siguiente enlace https://www.football-data.co.uk/germanym.php
2. Importe los archivos descargados a `R`
3. Usando la función *select* del paquete *dplyr*, seleccione unicamente las columnas:
    * Date
    * HomeTeam
    * AwayTeam
    * FTHG
    * FTAG
    * FTR
4. Combine cada uno de los data frames en un único data frame con ayuda de las funciones:
    * rbind
    * do.call

```r
setwd("C:/Users/Victor Miguel Terron/Downloads/reto3")
library(dplyr)
u1011 <- "https://www.football-data.co.uk/mmz4281/1718/D1.csv"
u1112 <- "https://www.football-data.co.uk/mmz4281/1718/D2.csv"
u1213 <- "https://www.football-data.co.uk/mmz4281/1819/D1.csv"
u1314 <- "https://www.football-data.co.uk/mmz4281/1819/D2.csv"
u1415 <- "https://www.football-data.co.uk/mmz4281/1920/D1.csv"
u1516 <- "https://www.football-data.co.uk/mmz4281/1920/D2.csv"
u1617 <- "https://www.football-data.co.uk/mmz4281/2021/D1.csv"
u1718 <- "https://www.football-data.co.uk/mmz4281/2021/D2.csv"


download.file(url = u1011, destfile = "SP1-1011.csv", mode = "wb")
download.file(url = u1112, destfile = "SP1-1112.csv", mode = "wb")
download.file(url = u1213, destfile = "SP1-1213.csv", mode = "wb")
download.file(url = u1314, destfile = "SP1-1314.csv", mode = "wb")
download.file(url = u1415, destfile = "SP1-1415.csv", mode = "wb")
download.file(url = u1516, destfile = "SP1-1516.csv", mode = "wb")
download.file(url = u1617, destfile = "SP1-1617.csv", mode = "wb")
download.file(url = u1718, destfile = "SP1-1718.csv", mode = "wb")
lista <- lapply(list.files(), read.csv)
lista <- lapply(lista, select, Date,HomeTeam:FTR,AwayTeam:FTR,HomeTeam:FTAG,AwayTeam:FTAG) # selecciona
data <- do.call(rbind, lista)
head(data);tail(data)
```

EJEMPLO 6 LECTURA DE JSON Y XML

OBJETIVO

- Aprender a importar archivos json y xml a R y guardar los datos como data frames.

DESARROLLO

Para comenzar necesimos instalar los paquetes *rjson*, y XML, por ejemplo, utilizando la función install.packages. Una vez que hemos instalado los paquetes, podemos cargarlos en R mediante la instrucción:

```r
#install.packages("rjson",dependencies = T)
#install.packages("XML",dependencies = T)
library(rjson)
library(XML)
```

LECTURA DE JSON

Podemos leer un JSON de la siguiente manera:

```r
URL1 <- "https://tools.learningcontainer.com/sample-json-file.json"
JsonData <- fromJSON(file = URL1)
class(JsonData)
length(JsonData)
str(JsonData)
```

LECTURA DE UN XML

Podemos leer un XML de la siguiente manera:

```
URL2 <- "http://www-db.deis.unibo.it/courses/TW/DOCS/w3schools/xml/cd_catalog.xml"
xmlfile <- xmlTreeParse(URL2) # Parse the XML file. Analizando el XML
topxml <- xmlSApply(xmlfile, function(x) xmlSApply(x, xmlValue)) # Mostrando los datos de una forma ami
xml_df <- data.frame(t(topxml), row.names= NULL) # Colocandolos en un Data Frame
str(xml_df) # Observar la naturaleza de las variables del DF
head(xml_df)
```

O también de la siguiente manera:

```
url3 <- URL2 # cargue el URL del XML
data_df <- xmlToDataFrame(url3)
head(data_df)
# Datos obtenidos de: https://datos.gob.mx/busca/dataset/saldo-de-bonos-de-proteccion-al-ahorro-bpas
```

EJEMPLO 7 FUNCIONES *na.omit* y *complete.cases*

OBJETIVO

- Tener una herramienta para identificar filas con valores perdidos (NA)
- Filtrar filas sin valores perdidos cuando estas se hallan identificados

DESARROLLO

Ahora vamos a considerar el conjunto de datos *airquality*, observamos primero algunas de sus filas

```
head(airquality)
```

```
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
```

```
library(dplyr)
str(airquality)
```

```
'data.frame':   153 obs. of  6 variables:
 $ Ozone  : int  41 36 12 18 NA 28 23 19 8 NA ...
 $ Solar.R: int  190 118 149 313 NA NA 299 99 19 194 ...
 $ Wind   : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
 $ Temp   : int  67 72 74 62 56 66 65 59 61 69 ...
 $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
 $ Day    : int  1 2 3 4 5 6 7 8 9 10 ...
```

```
dim(airquality)
```

```
[1] 153   6
```

Con la función *complete.cases* podemos averiguar cuales son aquellas filas que no contienen ningún valor perdido (NA) y cuales son aquellas filas que tienen al menos un valor perdido.

```
bien <- complete.cases(airquality)
```

La variable bien, es un vector lógico con TRUE en las posiciones que representan filas de airquality en donde no hay NA's y con FALSE en las posiciones que representan aquellas filas de airquality en donde se encontraron NA's.

Por tanto, podemos contar el número de filas en donde no hay NA´s de la siguiente manera:

```
sum(bien)
```

```
## [1] 111
```

Podemos filtrar aquellas filas sin NA's de la siguiente manera

```
airquality[bien,]
```

```
    Ozone Solar.R Wind Temp Month Day
1      41     190  7.4   67     5   1
2      36     118  8.0   72     5   2
3      12     149 12.6   74     5   3
4      18     313 11.5   62     5   4
7      23     299  8.6   65     5   7
8      19      99 13.8   59     5   8
9       8      19 20.1   61     5   9
12     16     256  9.7   69     5  12
13     11     290  9.2   66     5  13
14     14     274 10.9   68     5  14
15     18      65 13.2   58     5  15
16     14     334 11.5   64     5  16
17     34     307 12.0   66     5  17
18      6      78 18.4   57     5  18
19     30     322 11.5   68     5  19
20     11      44  9.7   62     5  20
21      1       8  9.7   59     5  21
22     11     320 16.6   73     5  22
23      4      25  9.7   61     5  23
24     32      92 12.0   61     5  24
28     23      13 12.0   67     5  28
29     45     252 14.9   81     5  29
30    115     223  5.7   79     5  30
31     37     279  7.4   76     5  31
38     29     127  9.7   82     6   7
40     71     291 13.8   90     6   9
41     39     323 11.5   87     6  10
```

44	23	148	8.0	82	6	13
47	21	191	14.9	77	6	16
48	37	284	20.7	72	6	17
49	20	37	9.2	65	6	18
50	12	120	11.5	73	6	19
51	13	137	10.3	76	6	20
62	135	269	4.1	84	7	1
63	49	248	9.2	85	7	2
64	32	236	9.2	81	7	3
66	64	175	4.6	83	7	5
67	40	314	10.9	83	7	6
68	77	276	5.1	88	7	7
69	97	267	6.3	92	7	8
70	97	272	5.7	92	7	9
71	85	175	7.4	89	7	10
73	10	264	14.3	73	7	12
74	27	175	14.9	81	7	13
76	7	48	14.3	80	7	15
77	48	260	6.9	81	7	16
78	35	274	10.3	82	7	17
79	61	285	6.3	84	7	18
80	79	187	5.1	87	7	19
81	63	220	11.5	85	7	20
82	16	7	6.9	74	7	21
85	80	294	8.6	86	7	24
86	108	223	8.0	85	7	25
87	20	81	8.6	82	7	26
88	52	82	12.0	86	7	27
89	82	213	7.4	88	7	28
90	50	275	7.4	86	7	29
91	64	253	7.4	83	7	30
92	59	254	9.2	81	7	31
93	39	83	6.9	81	8	1
94	9	24	13.8	81	8	2
95	16	77	7.4	82	8	3
99	122	255	4.0	89	8	7
100	89	229	10.3	90	8	8
101	110	207	8.0	90	8	9
104	44	192	11.5	86	8	12
105	28	273	11.5	82	8	13
106	65	157	9.7	80	8	14
108	22	71	10.3	77	8	16
109	59	51	6.3	79	8	17
110	23	115	7.4	76	8	18
111	31	244	10.9	78	8	19
112	44	190	10.3	78	8	20
113	21	259	15.5	77	8	21
114	9	36	14.3	72	8	22
116	45	212	9.7	79	8	24
117	168	238	3.4	81	8	25
118	73	215	8.0	86	8	26
120	76	203	9.7	97	8	28
121	118	225	2.3	94	8	29
122	84	237	6.3	96	8	30

```
123    85     188  6.3   94     8  31
124    96     167  6.9   91     9   1
125    78     197  5.1   92     9   2
126    73     183  2.8   93     9   3
127    91     189  4.6   93     9   4
128    47      95  7.4   87     9   5
129    32      92 15.5   84     9   6
130    20     252 10.9   80     9   7
131    23     220 10.3   78     9   8
132    21     230 10.9   75     9   9
133    24     259  9.7   73     9  10
134    44     236 14.9   81     9  11
135    21     259 15.5   76     9  12
136    28     238  6.3   77     9  13
137     9      24 10.9   71     9  14
138    13     112 11.5   71     9  15
139    46     237  6.9   78     9  16
140    18     224 13.8   67     9  17
141    13      27 10.3   76     9  18
142    24     238 10.3   68     9  19
143    16     201  8.0   82     9  20
144    13     238 12.6   64     9  21
145    23      14  9.2   71     9  22
146    36     139 10.3   81     9  23
147     7      49 10.3   69     9  24
148    14      20 16.6   63     9  25
149    30     193  6.9   70     9  26
151    14     191 14.3   75     9  28
152    18     131  8.0   76     9  29
153    20     223 11.5   68     9  30
```

```
data <- select(airquality, Ozone:Temp)
apply(data, 2, mean)
```

```
    Ozone   Solar.R      Wind      Temp
       NA        NA  9.957516 77.882353
```

```
apply(data, 2, mean, na.rm = T)
```

```
     Ozone    Solar.R       Wind       Temp
 42.129310 185.931507   9.957516  77.882353
```

La función *na.omit* devuelve el objeto con casos incompletos eliminados:

```
(m1 <- apply(na.omit(data), 2, mean))
```

```
    Ozone   Solar.R      Wind      Temp
 42.09910 184.80180   9.93964  77.79279
```

```
b <- complete.cases(data)

(m2 <- apply(data[b,], 2, mean))
```

```
    Ozone   Solar.R      Wind      Temp
 42.09910 184.80180   9.93964  77.79279
```

```r
identical(m1, m2)
```

```
[1] TRUE
```

POSTWORK SESION 2

OBJETIVO

- Importar múltiples archivos csv a R
- Observar algunas características y manipular los data frames
- Combinar múltiples data frames en un único data frame

DESARROLLO

Ahora vamos a generar un cúmulo de datos mayor al que se tenía, esta es una situación habitual que se puede presentar para complementar un análisis, siempre es importante estar revisando las características o tipos de datos que tenemos, por si es necesario realizar alguna transformación en las variables y poder hacer operaciones aritméticas si es el caso, además de sólo tener presente algunas de las variables, no siempre se requiere el uso de todas para ciertos procesamientos.

1. Importa los datos de soccer de las temporadas 2017/2018, 2018/2019 y 2019/2020 de la primera división de la liga española a R, los datos los puedes encontrar en el siguiente enlace: https://www.football-data.co.uk/spainm.php
2. Obten una mejor idea de las características de los data frames al usar las funciones: str, head, View y summary
3. Con la función select del paquete dplyr selecciona únicamente las columnas Date, HomeTeam, AwayTeam, FTHG, FTAG y FTR; esto para cada uno de los data frames. (Hint: también puedes usar lapply).
4. Asegúrate de que los elementos de las columnas correspondientes de los nuevos data frames sean del mismo tipo (Hint 1: usa as.Date y mutate para arreglar las fechas). Con ayuda de la función rbind forma un único data frame que contenga las seis columnas mencionadas en el punto 3 (Hint 2: la función do.call podría ser utilizada).

```r
# POST-WORK SESION 2
#CARGAMOS LIBRERIA DPLYR
suppressWarnings(suppressMessages(library(dplyr)))
#IMPORTANDO LOS DATOS
U1 <- "https://www.football-data.co.uk/mmz4281/1718/SP1.csv"
U2 <- "https://www.football-data.co.uk/mmz4281/1819/SP1.csv"
U3 <- "https://www.football-data.co.uk/mmz4281/1920/SP1.csv"
U4 <- "https://www.football-data.co.uk/mmz4281/1718/SP2.csv"
U5 <- "https://www.football-data.co.uk/mmz4281/1819/SP2.csv"
U6 <- "https://www.football-data.co.uk/mmz4281/1920/SP2.csv"
#IMPORTANDO LOS DATASETS
dataset1 <- read.csv(file = U1)
dataset2 <- read.csv(file = U2)
dataset3 <- read.csv(file = U3)
```

```r
dataset4 <- read.csv(file = U4)
dataset5 <- read.csv(file = U5)
dataset6 <- read.csv(file = U6)
# VERIFICAMOS EL TIPO DE DATOS QUE HAY EN CADA DATASET USANDO STR, HEAD, VIEW, SUMMARY
str(dataset1)
str(dataset2)
str(dataset3)
str(dataset4)
str(dataset5)
str(dataset6)

head(dataset1)
head(dataset2)
head(dataset3)
head(dataset4)
head(dataset5)
head(dataset6)

View(dataset1)
View(dataset2)
View(dataset3)
View(dataset4)
View(dataset5)
View(dataset6)

summary(dataset1)
summary(dataset2)
summary(dataset3)
summary(dataset4)
summary(dataset5)
summary(dataset6)

#PUNTO 3 SELECT DATE,HOMETEAM,AWAYTEAM,FTGH, FTAG Y FTR PARA CADA DATAFRAME

lista <-list(dataset1,dataset2,dataset3,dataset4,dataset5,dataset6)
camposelect <- lapply(lista, select,Date,HomeTeam,AwayTeam,FTHG,FTAG,FTR)
View(camposelect)
#COMPROBAMOS LOS CAMBIOS QUE HIZO A LAS VARIABLES, DATE ES UN CHAR Y NO DATE
str(camposelect)
#HACEMOS LA CONVERSION DE DATE DE CHAR A DATE
camposelect[[1]]<-mutate(camposelect[[1]],Date=as.Date(Date,"%d/%m/%y"))
camposelect[[2]]<-mutate(camposelect[[2]],Date=as.Date(Date,"%d/%m/%y"))
camposelect[[3]]<-mutate(camposelect[[3]],Date=as.Date(Date,"%d/%m/%y"))
camposelect[[4]]<-mutate(camposelect[[4]],Date=as.Date(Date,"%d/%m/%y"))
camposelect[[5]]<-mutate(camposelect[[5]],Date=as.Date(Date,"%d/%m/%y"))
camposelect[[6]]<-mutate(camposelect[[6]],Date=as.Date(Date,"%d/%m/%y"))
#VERIFICAMOS EL CAMBIO DEL CAMPO DATE
str(camposelect)
#UNIMOS LOS DATAFRAMES EN UNO SOLO CON RBIND Y DO.CALL
unidos<-do.call(rbind,camposelect)
View(unidos)
#PODEMOS OBTENER MÁS INFORMACIÓN DEL DATAFRAME UNIDO POR EJEMPLO LOS PRIMEROS 6
head(unidos)
```

```r
#LOS ULTIMOS 6
tail(unidos)
#RESUMEN
summary(unidos)
#DIMENSIONES
dim(unidos)
#VISUALIZARLO
View(unidos)
```