[000Z] DEFAD: Introducción a R y Rstudio

Datos: lectura, manejo de observaciones y variables. Funciones

000R Team

2015–16

1. Customización de objetos

Customización de objetos

Datos de ejemplo para trabajar. data.frame()

- Trabajaremos con un *dataset* sencillo

```
# getwd()
id <- c( 1:5 )
date    <- c( "10/07/08", "10/08/08", "10/09/08", "10/10/08", "10/11/08" )
country <- c( "US", "US", "UK", "UK", "UK" )
gender  <- c( "M", "F", "F", "M", "F" )
age     <- c ( 43, 45, 25, 39, 99 )
q1      <- c( 5, 3, 3, 3, 2 )
q2      <- c( 5, 5, 5, 4, 2 )
q3      <- c( 5, 5, 2, 3, 1 )
df     <- data.frame( id, date, country, gender, age, q1, q2, q3,
                      stringsAsFactors = FALSE )
df
```

```
##   id     date country gender age q1 q2 q3
## 1  1 10/07/08      US      M  43  5  5  5
## 2  2 10/08/08      US      F  45  3  5  5
## 3  3 10/09/08      UK      F  25  3  5  2
## 4  4 10/10/08      UK      M  39  3  4  3
## 5  5 10/11/08      UK      F  99  2  2  1
```

Ejercicios

- Crea un nuevo data frame que incluya únicamente las variables q1, q2 y q3.
- Crea de dos formas un nuevo data frame que incluya el país al que pertenecen los sujetos (con el nombre y con el índice)
- Crea un nuevo data frame eliminando las variables q2 y q3
- Selecciona las 3 primeras filas
- ¿Cómo escribirías la condición de ser mujer Y de UK?
- Selecciona las observaciones que cumplan dicha condición

reiniciaamos

devolvemos el df a su estado incial

```
df      <- data.frame( id,
                       date,
                       country,
                       gender,
                       age,
                       q1,
                       q2,
                       q3,
                       stringsAsFactors = FALSE )
```

Transformaciones de variables

transform()

```
# creamos variables con la función transform()
df <- transform( df,
                 sumx  = q1 + q2,
                 meanx = ( q1 + q2 ) / 2 )
df
```

```
##   id     date country gender age q1 q2 q3 sumx meanx
## 1  1 10/07/08      US      M  43  5  5  5   10   5.0
## 2  2 10/08/08      US      F  45  3  5  5    8   4.0
## 3  3 10/09/08      UK      F  25  3  5  2    8   4.0
## 4  4 10/10/08      UK      M  39  3  4  3    7   3.5
## 5  5 10/11/08      UK      F  99  2  2  1    4   2.0
```

eliminar variables

```
ncol( df )
```

```
## [1] 10
```

```
df<-df[ -c(ncol( df )-1 ,ncol( df )) ]
df
```

```
##   id     date country gender age q1 q2 q3
## 1  1 10/07/08      US      M  43  5  5  5
## 2  2 10/08/08      US      F  45  3  5  5
## 3  3 10/09/08      UK      F  25  3  5  2
## 4  4 10/10/08      UK      M  39  3  4  3
## 5  5 10/11/08      UK      F  99  2  2  1
```

crear variables al vuelo

```
df$nueva <- df$q1 * 2
df
```

```
##   id     date country gender age q1 q2 q3 nueva
## 1  1 10/07/08      US      M  43  5  5  5    10
## 2  2 10/08/08      US      F  45  3  5  5     6
## 3  3 10/09/08      UK      F  25  3  5  2     6
## 4  4 10/10/08      UK      M  39  3  4  3     6
## 5  5 10/11/08      UK      F  99  2  2  1     4
```

Recodificación de variables (recordando operadores)

Table 1: Operadores lógicos

Operador	Descripción
`<`	Menor que
`<=`	Menor o igual que
`>`	Mayor que
`>=`	Mayor o igual que
`==`	Exactamente igual a
`!=`	No igual a/que
`!x`	Diferente de x
`x | y`	x o y
`x & y`	x e y
`isTRUE(x)`	Prueba si x es TRUE

Recodificación de variables (2)

```
df$agecat[ df$age > 75 ]                  <- "anciano" # crea la v
df$agecat[ df$age <= 75 & df$age > 44 ] <- "maduro"
df$agecat[ df$age <= 44 ]                 <- "joven"
df
```

```
##   id     date country gender age q1 q2 q3 nueva  agecat
## 1  1 10/07/08      US      M  43  5  5  5    10   joven
## 2  2 10/08/08      US      F  45  3  5  5     6  maduro
## 3  3 10/09/08      UK      F  25  3  5  2     6   joven
## 4  4 10/10/08      UK      M  39  3  4  3     6   joven
## 5  5 10/11/08      UK      F  99  2  2  1     4 anciano
```

Renombrar variables

```
names( df )
```

```
##  [1] "id"      "date"    "country" "gender"  "age"     "q1"      "q2"
##  [8] "q3"      "nueva"   "agecat"
```

```
names( df )[ 1 ] <- "ID"
names( df )[ 3 ] <- "pais"
names( df )[ 4 ] <- "G"

names( df )[ 6:8 ] <- c( "it1", "it2", "it3" )
df
```

```
##   ID     date pais G age it1 it2 it3 nueva  agecat
## 1  1 10/07/08   US M  43   5   5   5    10   joven
## 2  2 10/08/08   US F  45   3   5   5     6  maduro
## 3  3 10/09/08   UK F  25   3   5   2     6   joven
## 4  4 10/10/08   UK M  39   3   4   3     6   joven
## 5  5 10/11/08   UK F  99   2   2   1     4 anciano
```

Valores perdidos (missing values)

Datos de ejemplo para trabajar. data.frame()

- Un data set con NA:

```
# getwd()
id <- c( 1:5 )
date    <- c( "10/07/08", "10/08/08", "10/09/08", "10/10/08", "10/11/08" )
country <- c( "US", "US", "UK", "UK", "UK" )
gender  <- c( "M", "F", "F", NA, "F" )
age     <- c ( NA, 45, 25, 39, 99 )
q1      <- c( 5, 3, 3, 3, 2 )
q2      <- c( 5, 5, 5, NA, 2 )
q3      <- c( 5, 5, 2, NA, 1 )
df     <- data.frame( id, date, country, gender, age, q1, q2, q3,
                      stringsAsFactors = FALSE )
df
```

```
##   id     date country gender age q1 q2 q3
## 1  1 10/07/08      US      M  NA  5  5  5
## 2  2 10/08/08      US      F  45  3  5  5
## 3  3 10/09/08      UK      F  25  3  5  2
## 4  4 10/10/08      UK   <NA>  39  3 NA NA
## 5  5 10/11/08      UK      F  99  2  2  1
```

```
names( df )[ 1 ] <- "ID"
names( df )[ 3 ] <- "pais"
names( df )[ 4 ] <- "G"

names( df )[ 6:8 ] <- c( "it1", "it2", "it3" )
```

NaN y NA

- Valores imposibles: `NaN` (*Not a Number*)
- Los valores perdidos: `NA` (*Not available*)

Hay muchas funciones para identificar estos valores: `is.na()`

is.na()

```
# missings
y <- c( 1, 2, 3, NA )
is.na( y )  # ¿cuáles son NA?
```

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

```
# devuelve un objeto del mismo tamaño que el que recibe
is.na( df[ , 4:8 ] )
```

```
##          G   age   it1   it2   it3
## [1,] FALSE  TRUE FALSE FALSE FALSE
## [2,] FALSE FALSE FALSE FALSE FALSE
## [3,] FALSE FALSE FALSE FALSE FALSE
## [4,]  TRUE FALSE FALSE  TRUE  TRUE
## [5,] FALSE FALSE FALSE FALSE FALSE
```

Excluir valores *missing* del análisis

Emplearemos la opción `na.rm=TRUE` para que **no** considere los valores faltantes

```
# excluir los missings del análisis
x <- c( 1, 2, NA, 3 )
y <- x[ 1 ] + x[ 2 ] + x[ 3 ] + x[ 4 ]
z <- sum( x )
y
```

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

```
z
```

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

```
# ¡¡¡¡Ambos son NA!!!!
sum( x, na.rm = T )  # ¡Ahora no!
```

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

na.omit()

`na.omit()` que elimina cualquier *fila* de un fataframe que tenga valores faltantes.

```
df <- na.omit( df )
df
```

```
##   ID     date pais G age it1 it2 it3
## 2  2 10/08/08   US F  45   3   5   5
## 3  3 10/09/08   UK F  25   3   5   2
## 5  5 10/11/08   UK F  99   2   2   1
```

Recodificar valores a *missing*

```
df$age[ is.na( df$age ) ] <- 99
```

```
# observa qué ocurre
is.na( df[ , 1:8 ] )
```

```
##      ID  date  pais     G   age   it1   it2   it3
## 2 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 3 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 5 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
df$age[ df$age == 99 ] <- NA
```

Conversión de tipos

dos tipos de funciones

Table 2: Conversión de tipos

lógico	convierte
`is.numeric()`	`as.numeric()`
`is.character()`	`as.character()`
`is.vector()`	`as.vector()`
`is.matrix()`	`as.matrix()`
`is.data.frame()`	`as.data.frame()`

Conversión de tipos (2)

```
# conversión de tipos.
a <- c( 1, 2, 3 );a
```

```
## [1] 1 2 3
```

```
is.numeric( a )
```

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

```
is.vector( a )
```

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

```
a <- as.character( a );a
```

```
## [1] "1" "2" "3"
```

Conversión de tipos (2)

```
a <- as.character( a );a
```

```
## [1] "1" "2" "3"
```

```
is.numeric( a )
```

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

```
is.vector( a )
```

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

Ordenar datos

order()

```
df
```

```
##   ID     date pais G age it1 it2 it3
## 2  2 10/08/08   US F  45   3   5   5
## 3  3 10/09/08   UK F  25   3   5   2
## 5  5 10/11/08   UK F  NA   2   2   1
```

ascendente por age

```
df_ordenado <- df[ order( df$age ),  ]
df_ordenado
```

```
##   ID     date pais G age it1 it2 it3
## 3  3 10/09/08   UK F  25   3   5   2
## 2  2 10/08/08   US F  45   3   5   5
## 5  5 10/11/08   UK F  NA   2   2   1
```

order() (2)

Ascendente por gender y descendente segun age

```
df_ordenado2 <- df[ order( df$G , -df$age ),  ]
df_ordenado2
```

```
##   ID     date pais G age it1 it2 it3
## 2  2 10/08/08   US F  45   3   5   5
## 3  3 10/09/08   UK F  25   3   5   2
## 5  5 10/11/08   UK F  NA   2   2   1
```

Ampliar conjuntos de datos

Ampliar/unir conjuntos de datos

- Más variables (columnas): `merge()` y `cbind()`
- Más casos (filas): `rbind()`

Ampliar con variables: merge()

```
dfA <- df  # nos creamos un par de dataframes aunque sean
dfB <- df
df.total <- merge( dfA, dfB,
                   by = "ID" )
```

```
df.total
```

```
##   ID   date.x pais.x G.x age.x it1.x it2.x it3.x   date.y pais.y G.y age.y
## 1  2 10/08/08     US   F    45     3     5     5 10/08/08     US   F    45
## 2  3 10/09/08     UK   F    25     3     5     2 10/09/08     UK   F    25
## 3  5 10/11/08     UK   F    NA     2     2     1 10/11/08     UK   F    NA
##   it1.y it2.y it3.y
## 1     3     5     5
## 2     3     5     2
## 3     2     2     1
```

Ampliar con variables: cbind()

```
# unir matrices
A         <- matrix( 1:9, 3, 3 )
B         <- matrix( 1:9, 3, 3 )
AB.total <- cbind( A, B )
AB.total
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    4    7    1    4    7
## [2,]    2    5    8    2    5    8
## [3,]    3    6    9    3    6    9
```

Ampliar con observaciones: rbind()

Ambos *dataframes* han de tener las mismas columnas

```
df.total2 <- rbind( dfA, dfB )
df.total2
```

```
##    ID     date pais G age it1 it2 it3
## 2   2 10/08/08   US F  45   3   5   5
## 3   3 10/09/08   UK F  25   3   5   2
## 5   5 10/11/08   UK F  NA   2   2   1
## 21  2 10/08/08   US F  45   3   5   5
## 31  3 10/09/08   UK F  25   3   5   2
## 51  5 10/11/08   UK F  NA   2   2   1
```