Parcial primer corte

Santiago Lozano

14 de marzo de 2020

```
base11 <- read.table("base11.txt",header = T,sep = " ")
base11
```

```
##        id   sexo año.de.nacimiento fecha.de.confirmación
## 539   539 female              1975            2020-02-23
## 1190 1190 female              1960            2020-02-26
## 457   457 female              1963            2020-02-23
## 230   230 female              1961            2020-02-22
## 117   117 female              1980            2020-02-21
## 487   487 female              1967            2020-02-23
## 217   217 female              1962            2020-02-22
## 532   532   male              1956            2020-02-23
```

```
base22 <- read.table("base22.txt",header = T,sep = " ")
base22
```

```
##       país   sexo   Estado   id
## 531  Korea female isolated  531
## 457  Korea female isolated  457
## 481  Korea female isolated  481
## 117  Korea female isolated  117
## 1184 Korea   male isolated 1184
## 539  Korea female isolated  539
## 224  Korea female isolated  224
## 217  Korea female isolated  217
```

1

Escriba falso o verdadero según corresponda

a. si tengo el vector `x <- c(5,7,3,1,4,3,8)` y a este le aplico `x[-c(1,2)]`, obtengo el último y el penúltimo vector respectivamente (Falso)

```
x <- c(5,7,3,1,4,3,8)
x[-c(1,2)]
```

```
## [1] 3 1 4 3 8
```

b. Teniendo la siguiente lista `L<-list(12,c(34,67),Mat,1:15,list(10,11))`, donde `Mat<-matrix(1:12,nrow=4,byrow=TRUE)`, si aplico `L[[4]][7:10]` obtengo los números del 7 al 10 (verdadero)

```
Mat<-matrix(1:12,nrow=4,byrow=TRUE)
L<-list(12,c(34,67),Mat,1:15,list(10,11))
L[[4]][7:10]
```

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

c. si en `ymatrix = matrix(data = c(6,34,923,5,0, 112:116, 5,9,34,76,2, 545:549), nrow = 5)` aplico `ymatrix[c(1,5),c(1,3)]` obtengo la matriz (Falso)

```
ymatrix = matrix(data = c(6,34,923,5,0, 112:116, 5,9,34,76,2, 545:549), nrow = 5)
ymatrix[c(1,5),c(1,3)]
```

```
##      [,1] [,2]
## [1,]    6    5
## [2,]    0    2
```

```
##      [,1] [,2]
## [1,]  923  114
## [2,]    5  115
```

d. los dataframes son una estructura de datos (Verdadero)

2

¿Cuál es el output de las siguientes líneas de código

a.

```
vecA <- c(1,3,6,2,7,4,8,1,0)
vecB <- c(vecA, 4, 1)
vecC <- c(vecA[1:4], 8, 5, vecA[5:9])
vecC[vecB>4]
```

```
## [1] 6 8 7
```

b.

```
a <- c()
a <- c(a,2)
a <- c(a,7)
a
```

```
## [1] 2 7
```

c.

```
y1 <- c(1,2,3,NA)
y2 <- c(5,6,NA,8)
y3 <- c(9,NA,11,12)
y4 <- c(NA,14,15,16)
full.frame <- data.frame(y1,y2,y3,y4)
reduced.frame <- full.frame[!is.na(full.frame$y1),]
reduced.frame
```

```
##   y1 y2 y3 y4
## 1  1  5  9 NA
## 2  2  6 NA 14
## 3  3 NA 11 15
```

d. de

```
base11
```

```
##        id   sexo año.de.nacimiento fecha.de.confirmación
## 539   539 female              1975            2020-02-23
## 1190 1190 female              1960            2020-02-26
## 457   457 female              1963            2020-02-23
## 230   230 female              1961            2020-02-22
```

```
## 117   117 female              1980            2020-02-21
## 487   487 female              1967            2020-02-23
## 217   217 female              1962            2020-02-22
## 532   532   male              1956            2020-02-23
```

que obtengo si ejecuto

```
base11$id[base11$sexo=="female"]
```

```
## [1]  539 1190  457  230  117  487  217
```

Sugerencia: Recuerde que si escribo `base11$id` estoy accediendo a la variable id del dataframe base11

e. nombre 3 tipos de datos

3

Tome los dos siguientes dataframes

```
base11
```

```
##        id   sexo año.de.nacimiento fecha.de.confirmación
## 539   539 female              1975            2020-02-23
## 1190 1190 female              1960            2020-02-26
## 457   457 female              1963            2020-02-23
## 230   230 female              1961            2020-02-22
## 117   117 female              1980            2020-02-21
## 487   487 female              1967            2020-02-23
## 217   217 female              1962            2020-02-22
## 532   532   male              1956            2020-02-23
```

```
base22
```

```
##       país   sexo   Estado   id
## 531  Korea female isolated  531
## 457  Korea female isolated  457
## 481  Korea female isolated  481
## 117  Korea female isolated  117
## 1184 Korea   male isolated 1184
## 539  Korea female isolated  539
## 224  Korea female isolated  224
## 217  Korea female isolated  217
```

3.1. Realice un merge (combinación) entre los dos dataframes de manera ordinaria

```
merge(base11,base22)
```

```
##    id   sexo año.de.nacimiento fecha.de.confirmación  país   Estado
## 1 117 female              1980            2020-02-21 Korea isolated
## 2 217 female              1962            2020-02-22 Korea isolated
## 3 457 female              1963            2020-02-23 Korea isolated
## 4 539 female              1975            2020-02-23 Korea isolated
```

3.2. En el caso anterior por defecto R identifica las columnas comunes por el nombre común de la variable, recordemos que podemos forzar las variables comunes con el argumento **by**, haga un merge forzando a id como única columna común y despliegue el posible resultado

```
merge(base11,base22,by=c("id"))
```

```
##    id sexo.x año.de.nacimiento fecha.de.confirmación  país sexo.y   Estado
```

```
## 1 117 female              1980            2020-02-21 Korea female isolated
## 2 217 female              1962            2020-02-22 Korea female isolated
## 3 457 female              1963            2020-02-23 Korea female isolated
## 4 539 female              1975            2020-02-23 Korea female isolated
```

4

La función apply(), aplica una función dada (con el argumento FUN) a todas la filas (MARGIN=1) o todas las columnas (MARGIN=2), vamos a querer aplicar una operación masiva a una matriz usando apply()

2.1. Escriba la codificación en R para obtenener la siguiente matriz

```
##      [,1] [,2] [,3] [,4]
## [1,]    6  112    5  545
## [2,]   34  113    9  546
## [3,]  923  114   34  547
## [4,]    5  115   76  548
## [5,]    0  116    2  549
```

2.2. Escriba la codificación en R para obtener la media de cada fila

```
apply(mymatrix, MARGIN = 1,FUN = mean)
```

```
## [1] 167.00 175.50 404.50 186.00 166.75
```

2.3. Escriba la codificación en R para obtener la media de cada columna

```
apply(mymatrix, MARGIN = 2,FUN = mean)
```

```
## [1] 193.6 114.0  25.2 547.0
```

2.4. la función sort nos permite ordenar los elemento de un vector, por ejemplo

```
sort(c(7,2,6,0,4,8,5,9))
```

```
## [1] 0 2 4 5 6 7 8 9
```

De esta manera escriba la codificación para ordenar los elementos de cada columna y como quedarían

```
apply(mymatrix, MARGIN = 2,FUN = sort)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    0  112    2  545
## [2,]    5  113    5  546
## [3,]    6  114    9  547
## [4,]   34  115   34  548
## [5,]  923  116   76  549
```