Combinación y extracción en estructuras de datos

Santiago Lozano

28 de febrero de 2020

Combinación de matrices o data.frames

Para concatenar matrices o data.frames usamos las funciones cbind() para concatenar columnar y rbind() para concatenar filas

```
cbind(1:4,5:8)
```

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

Combinación de matrices o data.frames

Sin embargo esta función no es óptima en el siguiente ejemplo

```
X1 <- data.frame(Id=1:4,Genero=c("M","F","F","M"),
                 Peso=c(75,68,48,72))
X1
```

```
##   Id Genero Peso
## 1  1      M   75
## 2  2      F   68
## 3  3      F   48
## 4  4      M   72
```

Combinación de matrices o data.frames

```
X2 <- data.frame(Id=1:4,Genero=c("M","F","F","M"),
                 Altura=c(182,165,160,178))
X2
```

```
##   Id Genero Altura
## 1  1      M    182
## 2  2      F    165
## 3  3      F    160
## 4  4      M    178
```

Combinación de matrices o data.frames

```
cbind(X1,X2)
```

```
##   Id Genero Peso Id Genero Altura
## 1  1      M   75  1      M    182
## 2  2      F   68  2      F    165
## 3  3      F   48  3      F    160
## 4  4      M   72  4      M    178
```

Combinación de matrices o data.frames

Para estos casos la función que permite combinar elementos en bases de datos es merge()

```
merge(X1,X2)
```

```
##   Id Genero Peso Altura
## 1  1      M   75    182
## 2  2      F   68    165
## 3  3      F   48    160
## 4  4      M   72    178
```

Combinación de matrices o data.frames

Ahora suonga que tenemos

```
X3 <- data.frame(Id=c(2,1,4,3),Genero=c("F","M","M","F"),
                 Altura=c(165,182,178,160))
X3
```

```
##   Id Genero Altura
## 1  2      F    165
## 2  1      M    182
## 3  4      M    178
## 4  3      F    160
```

Combinación de matrices o data.frames

de igual forma merge() es la función adecuada para combinar X1 y X3

```
merge(X1,X3)
```

```
##   Id Genero Peso Altura
## 1  1      M   75    182
## 2  2      F   68    165
## 3  3      F   48    160
## 4  4      M   72    178
```

Combinación de matrices o data.frames

Como hemos visto la función merge() combinados data.frames. La combinación está basada en las columnas de esos data.frames que tienen los mismos nombres. Las cuales se denominarán "Columnas comunes". El argumento **by** puede ser usado para forzar cuales columnas son comunes. El valor de este argumento puede ser un vector de nombres, vector de índices o vectores lógicos y las otras colunas son tratadas como distintas. La función merge() trbaja de la siguiente manera, entonces sea X y Y dos data.frames y veamos como trabaja

- Toda fila de un del data.frame X, en la función merge() compara los elementos de esta fila con respecto a todas las filas de Y, pero solo sobre el subconjunto de columnas en común

Combinación de matrices o data.frames

- si encuentra un match perfect, considera que es un mismo individuo: este individuo es agregado al nuevo data.frame resultado de la combinación y completa con los valores de las columnas no comunes de X y Y
- Si no hay match perfecto, el individuo es igualmente agregado al data.frame resultante con NA's (si el argumento all() toma el valor TRUE) o removido (si el argumento all() es FALSE)
- La operación se repite a la segunda fila y así sucesivamente

Combinación de matrices o data.frames

Veamos un ejemplo

```
X <- data.frame(Genero=c("F","M","M","F"),
                Altura=c(165, 82,178,160),
                Peso=c(50,65,67,55),
                Ingreso=c(80,90,60,50))
X
```

```
##   Genero Altura Peso Ingreso
## 1      F    165   50      80
## 2      M     82   65      90
## 3      M    178   67      60
## 4      F    160   55      50
```

Combinación de matrices o data.frames

```
Y <- data.frame(Genero=c("F","M","M","F"),
                Altura=c(165, 82,178,160),
                Peso=c(55,65,67,85),
                Salario=c(70,90,40,40) ,
                row.names=4:7)
Y
```

```
##   Genero Altura Peso Salario
## 4      F    165   55      70
## 5      M     82   65      90
## 6      M    178   67      40
## 7      F    160   85      40
```

Combinación de matrices o data.frames

```
merge(X,Y,by=c("Genero","Peso"))
```

```
##   Genero Peso Altura.x Ingreso Altura.y Salario
## 1      F   55      160      50      165      70
## 2      M   65       82      90       82      90
## 3      M   67      178      60      178      40
```

Combinación de matrices o data.frames

```
merge(X,Y,by=c("Genero","Peso"),all=TRUE)
```

```
##   Genero Peso Altura.x Ingreso Altura.y Salario
## 1      F   50      165      80       NA      NA
## 2      F   55      160      50      165      70
## 3      F   85       NA      NA      160      40
## 4      M   65       82      90       82      90
## 5      M   67      178      60      178      40
```

Combinación de matrices o data.frames

Tengamos en cuenta que la función merge() no tiene en cuenta los nombres de los individuos que asigna R, cuando determina los individuos comunes. Los nombres pueden ser incluidos agregando una columna Id a los dos data.frames, para identificar los individuos usando "row.names" como el valor en el argumento by

```
merge(X,Y,by=c("row.names","Peso"))
```

```
##   Row.names Peso Genero.x Altura.x Ingreso Genero.y
## 1         4   55        F      160      50        F
## Altura.y Salario
##      165      70
```

Combinación de matrices o data.frames

```
merge(X,Y,by=c("row.names","Peso"),all=TRUE)
```

```
##   Row.names Peso Genero.x Altura.x Ingreso Genero.y Altu
## 1         1   50        F      165      80     <NA>
## 2         2   65        M       82      90     <NA>
## 3         3   67        M      178      60     <NA>
## 4         4   55        F      160      50        F
## 5         5   65     <NA>       NA      NA        M
## 6         6   67     <NA>       NA      NA        M
## 7         7   85     <NA>       NA      NA        F
```

Combinación de matrices o data.frames

```
##    Altura.y Salario
## 1        NA      NA
## 2        NA      NA
## 3        NA      NA
## 4       165      70
## 5        82      90
## 6       178      40
## 7       160      40
```

Combinación de matrices o data.frames

Para concatenar filas usamos rbind()

```
rbind(1:4,5:8)
```

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

Combinación de matrices o data.frames

veamos una función útil en algunos casos

```
df1 <- data.frame(A=1:5, B=LETTERS[1:5])
df1
```

```
##   A B
## 1 1 A
## 2 2 B
## 3 3 C
## 4 4 D
## 5 5 E
```

Combinación de matrices o data.frames

```
df2 <- data.frame(A=6:10, E=letters[1:5])
df2
```

```
##    A E
## 1  6 a
## 2  7 b
## 3  8 c
## 4  9 d
## 5 10 e
```

Combinación de matrices o data.frames

```
smartbind(df1, df2)
```

```
##       A     B    E
## 1:1   1     A <NA>
## 1:2   2     B <NA>
## 1:3   3     C <NA>
## 1:4   4     D <NA>
## 1:5   5     E <NA>
## 2:1   6  <NA>    a
## 2:2   7  <NA>    b
## 2:3   8  <NA>    c
## 2:4   9  <NA>    d
## 2:5  10  <NA>    e
```

Operaciones sobre listas

Las funciones lapply() y sapply() son similares a la función apply(), aplicando una función a todos los elementos de una lista, el output corresponde a una lista, y cada elemento de la lista genera un vector si es posible

Operaciones sobre listas

```r
x <- list(a = 1:10, beta = exp(-3:3),
          logic = c(TRUE,FALSE,FALSE,TRUE))
x
```

```
## $a
##  [1]  1  2  3  4  5  6  7  8  9 10
##
## $beta
## [1]  0.04978707  0.13533528  0.36787944  1.00000000
## [5]  2.71828183  7.38905610  2.71828183  7.38905610
## [9] 20.08553692
##
## $logic
## [1]  TRUE FALSE FALSE  TRUE
```

Operaciones sobre listas

```
lapply(x,mean)
```

```
## $a
## [1] 5.5
##
## $beta
## [1] 4.535125
##
## $logic
## [1] 0.5
```

Operaciones sobre listas

```
lapply(x,quantile,probs=(1:3)/4)
```

```
## $a
##  25%  50%  75%
## 3.25 5.50 7.75
##
## $beta
##       25%       50%       75%
## 0.2516074 1.0000000 5.0536690
##
## $logic
## 25% 50% 75%
## 0.0 0.5 1.0
```

Operaciones sobre listas

```
sapply(x, quantile)
```

```
##           a         beta logic
## 0%     1.00  0.04978707   0.0
## 25%    3.25  0.25160736   0.0
## 50%    5.50  1.00000000   0.5
## 75%    7.75  5.05366896   1.0
## 100% 10.00 20.08553692   1.0
```

Operaciones sobre listas

Si desea aplicar una función a un vector (en lugar de al margen de una matriz), use sapply(. Aquí está el código para generar una lista de secuencias.

Operaciones sobre listas

```
i36 <- sapply(3:6, seq)# Crea una lista de vectores
i36
```

```
## [[1]]
## [1] 1 2 3
##
## [[2]]
## [1] 1 2 3 4
##
## [[3]]
## [1] 1 2 3 4 5
##
## [[4]]
## [1] 1 2 3 4 5 6
```

Operaciones sobre listas

```
sapply(i36, sum)
```

```
## [1]  6 10 15 21
```

Ejercicios (1)

```
setwd("~/Progrmación en R/2020-I/PR05-
        "Combinación y extracción en estrucutura de datos"
        "de datos")
(matdata <- read.table(sweepdata.txt"))
```

```
##    V1 V2  V3  V4
## 1   3 12 0.4 125
## 2   5 12 0.7 166
## 3   7 15 0.8 174
## 4   7 14 0.7 128
## 5   5 18 0.3 136
## 6   9 13 0.2 155
## 7   7 15 0.5 115
## 8   2 13 0.5 169
## 9   1 10 0.1 182
## 10  0 11 0.2 166
```

Ejercicios (1)

En este ejemplo, queremos expresar una matriz en términos de las desviaciones de cada valor de su media de columna. Primero aplique la función apply() para sacar la media a cada una de sus columnas

```
(cols <- apply(matdata,MARGIN=2,mean))
```

```
##      V1      V2      V3      V4
##    4.60   13.30    0.44 151.60
```

Ejercicios (1)

Ahora a cada valor de la columna le resto la media correspondiente usando sweep()

```
sweep(matdata,MARGIN=2,STATS=cols,FUN="-")
```

```
##       V1   V2    V3    V4
## 1  -1.6 -1.3 -0.04 -26.6
## 2   0.4 -1.3  0.26  14.4
## 3   2.4  1.7  0.36  22.4
## 4   2.4  0.7  0.26 -23.6
## 5   0.4  4.7 -0.14 -15.6
## 6   4.4 -0.3 -0.24   3.4
## 7   2.4  1.7  0.06 -36.6
## 8  -2.6 -0.3  0.06  17.4
## 9  -3.6 -3.3 -0.34  30.4
## 10 -4.6 -2.3 -0.24  14.4
```

Ejercicios (2)

Suponga que desea obtener los subíndices para un barrido de datos en columnas o en filas. Establezca los subíndices de fila repetidos en cada columna usando apply():

```
apply(matdata,MARGIN=2,FUN=function (x) 1:10)
```

```
##       V1 V2 V3 V4
##  [1,]  1  1  1  1
##  [2,]  2  2  2  2
##  [3,]  3  3  3  3
##  [4,]  4  4  4  4
##  [5,]  5  5  5  5
##  [6,]  6  6  6  6
##  [7,]  7  7  7  7
##  [8,]  8  8  8  8
##  [9,]  9  9  9  9
```

Ejercicios (2)

y para las columnas usando apply()

```
t(apply(matdata,MARGIN=1,FUN=function (x) 1:4))
```

```
##       [,1] [,2] [,3] [,4]
##  [1,]    1    2    3    4
##  [2,]    1    2    3    4
##  [3,]    1    2    3    4
##  [4,]    1    2    3    4
##  [5,]    1    2    3    4
##  [6,]    1    2    3    4
##  [7,]    1    2    3    4
##  [8,]    1    2    3    4
##  [9,]    1    2    3    4
## [10,]    1    2    3    4
```

Ejercicios (3)

Realice lo misimo con la función sweep()

```
sweep(matdata,MARGIN=1,STATS=1:10,FUN=function(a,b) {b})
```

```
##       [,1] [,2] [,3] [,4]
##  [1,]    1    1    1    1
##  [2,]    2    2    2    2
##  [3,]    3    3    3    3
##  [4,]    4    4    4    4
##  [5,]    5    5    5    5
##  [6,]    6    6    6    6
##  [7,]    7    7    7    7
##  [8,]    8    8    8    8
##  [9,]    9    9    9    9
## [10,]   10   10   10   10
```

Ejercicios (4)

```
sweep(matdata,2,1:4,function(a,b) b)
```

```
##       [,1] [,2] [,3] [,4]
##  [1,]    1    2    3    4
##  [2,]    1    2    3    4
##  [3,]    1    2    3    4
##  [4,]    1    2    3    4
##  [5,]    1    2    3    4
##  [6,]    1    2    3    4
##  [7,]    1    2    3    4
##  [8,]    1    2    3    4
##  [9,]    1    2    3    4
## [10,]    1    2    3    4
```

Ejercicios (5)

Tenga en cuenta que en ambos casos, la respuesta producida por apply() es un vector en lugar de una matriz. Puede aplicar funciones a los elementos individuales de la matriz en lugar de a los márgenes. El margen que especifique solo influye en la forma de la matriz resultante.

```
(X <- matrix(1:24,nrow=4))
```

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

Ejercicios (5)

Aplique raíz cuadrada a todos los elementos de la matriz y muestremelos por filas

```
apply(X,MARGIN=1,FUN=sqrt)
```

```
##          [,1]     [,2]     [,3]     [,4]
## [1,] 1.000000 1.414214 1.732051 2.000000
## [2,] 2.236068 2.449490 2.645751 2.828427
## [3,] 3.000000 3.162278 3.316625 3.464102
## [4,] 3.605551 3.741657 3.872983 4.000000
## [5,] 4.123106 4.242641 4.358899 4.472136
## [6,] 4.582576 4.690416 4.795832 4.898979
```

Ejercicios (5)

ahora por columnas

```
apply(X,2,sqrt)
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,] 1.000000 2.236068 3.000000 3.605551 4.123106
## [2,] 1.414214 2.449490 3.162278 3.741657 4.242641
## [3,] 1.732051 2.645751 3.316625 3.872983 4.358899
## [4,] 2.000000 2.828427 3.464102 4.000000 4.472136

##          [,6]
## [1,] 4.582576
## [2,] 4.690416
## [3,] 4.795832
## [4,] 4.898979
```

Extracción e inserción de elementos

Es esta sección veremos los diferentes recursos que tiene R para extraer componentes de n vector, pues algunas veces quisieramos usar no todos los contenidos de un vector. En R podemos usar la función "["(), o podemos usar "[]" en el cual establecemos los subíndices que queremos extraer ó podemos tomar los siguientes argumentos

- un vector de índices de elementos a extraer
- un vector de índices de elementos a no extraer
- un vector de valores lógicos indicando cuales elementos se extraen

Extracción e inserción de elementos en vectores

```
vec <- c(2,3,4,8,3)
vec
```

Extracción e inserción de elementos en vectores

```
vec <- c(2,3,4,8,3)
vec
```

```
## [1] 2 3 4 8 3
```

```
vec[2]
```

Extracción e inserción de elementos en vectores

```
vec <- c(2,3,4,8,3)
vec
```

```
## [1] 2 3 4 8 3
```

```
vec[2]
```

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

Extracción e inserción de elementos en vectores

```
"["(vec,2)
```

Extracción e inserción de elementos en vectores

```
"["(vec,2)
```

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

```
vec[-2]
```

Extracción e inserción de elementos en vectores

```
"["(vec,2)
```

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

```
vec[-2]
```

```
## [1] 2 4 8 3
```

Extracción e inserción de elementos en vectores (slicing)

Podemos realizar un barrido para seleccionar una cantidad masiva de subíndices a extraer, debe tenerse en cuente que R maneja los subíndices empezando en 1

```
vec[2:5]
```

Extracción e inserción de elementos en vectores (slicing)

Podemos realizar un barrido para seleccionar una cantidad masiva de subíndices a extraer, debe tenerse en cuente que R maneja los subíndices empezando en 1

```
vec[2:5]
```

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

```
vec[-c(1,5)]
```

Extracción e inserción de elementos en vectores (slicing)

Podemos realizar un barrido para seleccionar una cantidad masiva de subíndices a extraer, debe tenerse en cuente que R maneja los subíndices empezando en 1

```
vec[2:5]
```

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

```
vec[-c(1,5)]
```

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

Extracción e inserción de elementos en vectores (slicing)

```
vec[c(T,F,F,T,T)]
```

Extracción e inserción de elementos en vectores (slicing)

```
vec[c(T,F,F,T,T)]
```

```
## [1] 2 8 3
```

```
vec>4
```

Extracción e inserción de elementos en vectores (slicing)

```
vec[c(T,F,F,T,T)]
```

```
## [1] 2 8 3
```

```
vec>4
```

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

```
vec[vec>4]
```

Extracción e inserción de elementos en vectores (slicing)

```
vec[c(T,F,F,T,T)]
```

```
## [1] 2 8 3
```

```
vec>4
```

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

```
vec[vec>4]
```

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

Extracción e inserción de elementos en vectores

Es importante notar que la simplicidad sintáctica de una instrucción como x[y>0], el cual extrae de x todos los elementos del subíndice i tales que $y_i > 0$

```
x <- 1:5
y <- c(-1,2,-3,4,-2)
x[y>0]
```

Extracción e inserción de elementos en vectores

Es importante notar que la simplicidad sintáctica de una instrucción como x[y>0], el cual extrae de x todos los elementos del subíndice i tales que $y_i > 0$

```
x <- 1:5
y <- c(-1,2,-3,4,-2)
x[y>0]
```

```
## [1] 2 4
```

a menudo necesitamos usar tantas construcciones como sean posibles, las cuales son llamadas máscaras lógicas, en estos existen dos ventajs: el código es sencillo para leer y rápido para ejecutar

Note que las funciones which(),which.min(), which.max() serán útiles

Extracción e inserción de elementos en vectores

```
mask <- c(TRUE,FALSE,TRUE,NA,FALSE,FALSE,TRUE)
which(mask)
```

Extracción e inserción de elementos en vectores

```
mask <- c(TRUE,FALSE,TRUE,NA,FALSE,FALSE,TRUE)
which(mask)
```

```
## [1] 1 3 7
```

```
x <- c(0:4,0:5,11)
which.min(x)
```

Extracción e inserción de elementos en vectores

```
mask <- c(TRUE,FALSE,TRUE,NA,FALSE,FALSE,TRUE)
which(mask)
```

```
## [1] 1 3 7
```

```
x <- c(0:4,0:5,11)
which.min(x)
```

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

```
which.max(x)
```

Extracción e inserción de elementos en vectores

```
mask <- c(TRUE,FALSE,TRUE,NA,FALSE,FALSE,TRUE)
which(mask)
```

```
## [1] 1 3 7
```

```
x <- c(0:4,0:5,11)
which.min(x)
```

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

```
which.max(x)
```

```
## [1] 12
```

Remplazamiento en vectores

Para reemplazar elementos en un vector se hace de una manera similar a la extracción. Todo lo que necesitamos es seleccionar las elementos como si usted quisiera extraerlos, y usando <- se sigue de los elementos para reemplazar. Por supuesto, usted necesita especificar el mismo número de elementos entrantes que salientes

```
z <- c(0,0,0,2,0)
z[c(1,5)] <- 1
z
```

Remplazamiento en vectores

Para reemplazar elementos en un vector se hace de una manera similar a la extracción. Todo lo que necesitamos es seleccionar las elementos como si usted quisiera extraerlos, y usando <- se sigue de los elementos para reemplazar. Por supuesto, usted necesita especificar el mismo número de elementos entrantes que salientes

```
z <- c(0,0,0,2,0)
z[c(1,5)] <- 1
z
```

```
## [1] 1 0 0 2 1
```

```
z[which.max(z)] <- 0
z
```

Remplazamiento en vectores

Para reemplazar elementos en un vector se hace de una manera similar a la extracción. Todo lo que necesitamos es seleccionar las elementos como si usted quisiera extraerlos, y usando <- se sigue de los elementos para reemplazar. Por supuesto, usted necesita especificar el mismo número de elementos entrantes que salientes

```
z <- c(0,0,0,2,0)
z[c(1,5)] <- 1
z
```

```
## [1] 1 0 0 2 1
```

```
z[which.max(z)] <- 0
z
```

```
## [1] 1 0 0 0 1
```

Remplazamiento en vectores

Para reemplazar elementos en un vector se hace de una manera similar a la extracción. Todo lo que necesitamos es seleccionar las elementos como si usted quisiera extraerlos, y usando <- se sigue de los elementos para reemplazar. Por supuesto, usted necesita especificar el mismo número de elementos entrantes que salientes

```
z <- c(0,0,0,2,0)
z[c(1,5)] <- 1
z
```

```
## [1] 1 0 0 2 1
```

```
z[which.max(z)] <- 0
z
```

```
## [1] 1 0 0 0 1
```

Agregar o insertar elementos a un vector preexistente

Usamos la función c()

```
vecA <- c(1,3,6,2,7,4,8,1,0)
vecA
```

Agregar o insertar elementos a un vector preexistente

Usamos la función c()

```
vecA <- c(1,3,6,2,7,4,8,1,0)
vecA
```

```
## [1] 1 3 6 2 7 4 8 1 0
```

```
(vecB <- c(vecA, 4, 1))
```

Agregar o insertar elementos a un vector preexistente

Usamos la función c()

```
vecA <- c(1,3,6,2,7,4,8,1,0)
vecA
```

```
## [1] 1 3 6 2 7 4 8 1 0
```

```
(vecB <- c(vecA, 4, 1))
```

```
##  [1] 1 3 6 2 7 4 8 1 0 4 1
```

```
(vecC <- c(vecA[1:4], 8, 5, vecA[5:9]))
```

Agregar o insertar elementos a un vector preexistente

Usamos la función c()

```
vecA <- c(1,3,6,2,7,4,8,1,0)
vecA
```

```
## [1] 1 3 6 2 7 4 8 1 0
```

```
(vecB <- c(vecA, 4, 1))
```

```
##  [1] 1 3 6 2 7 4 8 1 0 4 1
```

```
(vecC <- c(vecA[1:4], 8, 5, vecA[5:9]))
```

```
##  [1] 1 3 6 2 8 5 7 4 8 1 0
```

Agregar o insertar elementos a un vector preexistente

Este mecanismo provee la habilidad para completar un vector cuyo tamaño no es fijo en el principio

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

Agregar o insertar elementos a un vector preexistente

Este mecanismo provee la habilidad para completar un vector cuyo tamaño no es fijo en el principio

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

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

Ejercicio

Cree un vecctor altura <- c(182,150,160,140.5,191) y vector género genero <- c(0,1,1,1,0) donde la altura seexpresa en cms y el género 1 mujer y 0 hombre. extraiga del vector altura, las altura de los hombres. Use el método de extracción de variables por subíndices, repitiendo la tarea con una máscara lógica

Ejercicio

Extraiga del siguiente vector todos los números entre 2 y 3

```
x <- c(0.1,0.5,2.1,3.5,2.8,2.7,1.9,2.2,5.6)
```

Extracción e inserción en matrices

-Extracción via X[índice fila,índice columna], omitir la primera componente significa que todas las filas son seleccionada, o en su debido caso las columnas, cuando las componentes son negativas indican que elementos no extraer

- Extracción vía máscara lógica X[máscara], sabiendo que la matriz es de valores lógicos de mismo tamaño que X el cual indica que elementos extraer

Extracción e inserción en matrices

```
Mat <- matrix(1:12,nrow=4,ncol=3,byrow=TRUE)
Mat
```

Extracción e inserción en matrices

```
Mat <- matrix(1:12,nrow=4,ncol=3,byrow=TRUE)
Mat
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
## [4,]   10   11   12
```

```
Mat[2,3]
```

Extracción e inserción en matrices

```
Mat <- matrix(1:12,nrow=4,ncol=3,byrow=TRUE)
Mat
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
## [4,]   10   11   12
```

```
Mat[2,3]
```

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

Extracción e inserción en matrices

```
Mat[,1]
```

Extracción e inserción en matrices

```
Mat[,1]
```

```
## [1]  1  4  7 10
```

```
Mat[c(1,4),]
```

Extracción e inserción en matrices

```
Mat[,1]
```

```
## [1]  1  4  7 10
```

```
Mat[c(1,4),]
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]   10   11   12
```

```
Mat[3,-c(1,3)]
```

Extracción e inserción en matrices

```
Mat[,1]
```

```
## [1]  1  4  7 10
```

```
Mat[c(1,4),]
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]   10   11   12
```

```
Mat[3,-c(1,3)]
```

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

Extracción e inserción en matrices

```
MatLogical <- matrix(c(TRUE,FALSE),nrow=4,ncol=3)
MatLogical
```

Extracción e inserción en matrices

```
MatLogical <- matrix(c(TRUE,FALSE),nrow=4,ncol=3)
MatLogical
```

```
##       [,1]  [,2]  [,3]
## [1,]  TRUE  TRUE  TRUE
## [2,] FALSE FALSE FALSE
## [3,]  TRUE  TRUE  TRUE
## [4,] FALSE FALSE FALSE
```

```
Mat[MatLogical]
```

Extracción e inserción en matrices

```
MatLogical <- matrix(c(TRUE,FALSE),nrow=4,ncol=3)
MatLogical
```

```
##       [,1]  [,2]  [,3]
## [1,]  TRUE  TRUE  TRUE
## [2,] FALSE FALSE FALSE
## [3,]  TRUE  TRUE  TRUE
## [4,] FALSE FALSE FALSE
```

```
Mat[MatLogical]
```

```
## [1] 1 7 2 8 3 9
```

Extracción e inserción en matrices

```
ind <- c(2,4,6,8,3)
Mat[ind]
```

Extracción e inserción en matrices

```
ind <- c(2,4,6,8,3)
Mat[ind]
```

```
## [1]  4 10  5 11  7
```

Extracción e inserción en matrices

Algunas veces la función de extracción cambia la estrucutura de manipulación

```
m <- matrix(1:6,nrow=2) ; m
```

Extracción e inserción en matrices

Algunas veces la función de extracción cambia la estrucutura de manipulación

```
m <- matrix(1:6,nrow=2) ; m
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```
m[,1]
```

Extracción e inserción en matrices

Algunas veces la función de extracción cambia la estrucutura de manipulación

```
m <- matrix(1:6,nrow=2) ; m
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

```
m[,1]
```

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

Extracción e inserción en matrices

Pero resulta que yo quería el vector como columna, este problema se puede arreglar

```
m[,1,drop=FALSE]
```

Extracción e inserción en matrices

Pero resulta que yo quería el vector como columna, este problema se puede arreglar

```
m[,1,drop=FALSE]
```

```
##      [,1]
## [1,]    1
## [2,]    2
```

Extracción e inserción en matrices

Usando la función which yo puedo alternar subíndices de los elementos de una matriz los cuales son verificados con la condición

```
m <- matrix(c(1,2,3,1,2,3,2,1,3),3,3)
m
```

Extracción e inserción en matrices

Usando la función which yo puedo alternar subíndices de los elementos de una matriz los cuales son verificados con la condición

```
m <- matrix(c(1,2,3,1,2,3,2,1,3),3,3)
m
```

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

```
which(m == 1)
```

Extracción e inserción en matrices

Usando la función which yo puedo alternar subíndices de los elementos de una matriz los cuales son verificados con la condición

```
m <- matrix(c(1,2,3,1,2,3,2,1,3),3,3)
m
```

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

```
which(m == 1)
```

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

Extracción e inserción en matrices

para poner los índices como parejas

```
which(m == 1,arr.ind=TRUE)
```

Extracción e inserción en matrices

para poner los índices como parejas

```
which(m == 1,arr.ind=TRUE)
```

```
##      row col
## [1,]   1   1
## [2,]   1   2
## [3,]   2   3
```

Extracción e inserción en matrices

Para realizarr inserción de elementos en una matriz

```
m[m!=2] <- 0
m
```

Extracción e inserción en matrices

Para realizarr inserción de elementos en una matriz

```
m[m!=2] <- 0
m
```

```
##      [,1] [,2] [,3]
## [1,]    0    0    2
## [2,]    2    2    0
## [3,]    0    0    0
```

```
Mat <- Mat[-4,] ; Mat
```

Extracción e inserción en matrices

Para realizarr inserción de elementos en una matriz

```
m[m!=2] <- 0
m
```

```
##      [,1] [,2] [,3]
## [1,]    0    0    2
## [2,]    2    2    0
## [3,]    0    0    0
```

```
Mat <- Mat[-4,] ; Mat
```

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

Extracción e inserción en matrices

```
m[Mat>7] <- Mat[Mat>7]
m
```

Extracción e inserción en matrices

```
m[Mat>7] <- Mat[Mat>7]
m
```

```
##      [,1] [,2] [,3]
## [1,]    0    0    2
## [2,]    2    2    0
## [3,]    0    8    9
```

Ejercicio

```
m1 <- matrix(c(0,22,0,23,34,0,0,0,28),ncol=3)
m2 <- matrix(c(10,1,4,10,9,9,2,6,4),ncol=3)
```

reemplace todos los valores distintos de cero de m1 con el correspondiente valor en m2,despues remueva la segunda columna de m1

Extracción e inserción en listas

```
L <- list(12,c(34,67,8),Mat,1:15,list(10,11))
L
```

Extracción e inserción en listas

```
L <- list(12,c(34,67,8),Mat,1:15,list(10,11))
L
```

```
## [[1]]
## [1] 12
##
## [[2]]
## [1] 34 67  8
##
## [[3]]
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
```

Extracción e inserción en listas

Extracción e inserción en listas

```
## [[4]]
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
##
## [[5]]
## [[5]][[1]]
## [1] 10
##
## [[5]][[2]]
## [1] 11
```

Extracción e inserción en listas

```
L[2]
```

Extracción e inserción en listas

```
L[2]
```

```
## [[1]]
## [1] 34 67  8
```

```
class(L[2])
```

Extracción e inserción en listas

```
L[2]
```

```
## [[1]]
## [1] 34 67  8
```

```
class(L[2])
```

```
## [1] "list"
```

Extracción e inserción en listas

```
L[c(3,4)]
```

Extracción e inserción en listas

```
L[c(3,4)]
```

```
## [[1]]
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
##
## [[2]]
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
```

Extracción e inserción en listas

Para acceder a los elementos de una lista usamos

```
L[[2]]
```

Extracción e inserción en listas

Para acceder a los elementos de una lista usamos

```
L[[2]]
```

```
## [1] 34 67  8
```

```
L[[2]][1]
```

Extracción e inserción en listas

Para acceder a los elementos de una lista usamos

```
L[[2]]
```

```
## [1] 34 67  8
```

```
L[[2]][1]
```

```
## [1] 34
```

```
L[[5]][[2]]
```

Extracción e inserción en listas

Para acceder a los elementos de una lista usamos

```
L[[2]]
```

```
## [1] 34 67  8
```

```
L[[2]][1]
```

```
## [1] 34
```

```
L[[5]][[2]]
```

```
## [1] 11
```

Extracción e inserción en listas

Para usar indexación recursiva usamos

```
L[[c(2,3)]]
```

Extracción e inserción en listas

Para usar indexación recursiva usamos

```
L[[c(2,3)]]
```

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

Extracción e inserción en listas

Para accerder a los elementos de una lista con nombres en cada elemento usamos

```
L <- list(cars=c("FORD","PEUGEOT"),climate=
            c("Tropical","Temperate"))
L
```

Extracción e inserción en listas

Para accerder a los elementos de una lista con nombres en cada elemento usamos

```
L <- list(cars=c("FORD","PEUGEOT"),climate=
            c("Tropical","Temperate"))
L
```

```
## $cars
## [1] "FORD"    "PEUGEOT"
##
## $climate
## [1] "Tropical"  "Temperate"
```

Extracción e inserción en listas

```
L["cars"][2]
```

Extracción e inserción en listas

```
L[["cars"]][2]
```

```
## [1] "PEUGEOT"
```

```
L$cars
```

Extracción e inserción en listas

```
L[["cars"]][2]
```

```
## [1] "PEUGEOT"
```

```
L$cars
```

```
## [1] "FORD"    "PEUGEOT"
```

Extracción e inserción en listas

```
L$climate[2] <- "Continental"
L
```

Extracción e inserción en listas

```
L$climate[2] <- "Continental"
L
```

```
## $cars
## [1] "FORD"    "PEUGEOT"
##
## $climate
## [1] "Tropical"    "Continental"
```

Extracción e inserción en listas

El nombre de una columna puede incluir espacios. Para acceder a ella, usted necesita una marca especial

```
L <- list("pretty cars"=c("FORD","PEUGEOT"))
L
```

Extracción e inserción en listas

El nombre de una columna puede incluir espacios. Para acceder a ella, usted necesita una marca especial

```
L <- list("pretty cars"=c("FORD","PEUGEOT"))
L
```

```
## $`pretty cars`
## [1] "FORD"    "PEUGEOT"
```

```
L$"pretty cars"
```

Extracción e inserción en listas

El nombre de una columna puede incluir espacios. Para acceder a ella, usted necesita una marca especial

```
L <- list("pretty cars"=c("FORD","PEUGEOT"))
L
```

```
## $`pretty cars`
## [1] "FORD"    "PEUGEOT"
```

```
L$"pretty cars"
```

```
## [1] "FORD"    "PEUGEOT"
```

Ejercicios

```
peas <- c(4,7,6,5,6,7)
```

```
peas[4]
```

Ejercicios

```
peas <- c(4,7,6,5,6,7)
```

```
peas[4]
```

```
## [1] 5
```

```
pods <- c(2,3,6)
peas[pods]
```

Ejercicios

```
peas <- c(4,7,6,5,6,7)
```

```
peas[4]
```

```
## [1] 5
```

```
pods <- c(2,3,6)
peas[pods]
```

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

Ejercicios

```
peas[c(2,3,6)]
```

Ejercicios

```
peas[c(2,3,6)]
```

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

```
peas[-1]
```

Ejercicios

```
peas[c(2,3,6)]
```

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

```
peas[-1]
```

```
## [1] 7 6 5 6 7
```

```
peas[-length(peas)]
```

Ejercicios

```
peas[c(2,3,6)]
```

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

```
peas[-1]
```

```
## [1] 7 6 5 6 7
```

```
peas[-length(peas)]
```

```
## [1] 4 7 6 5 6
```

Ejercicios

```
peas[1:3]
```

Ejercicios

```
peas[1:3]
```

```
## [1] 4 7 6
```

```
peas[seq(2,length(peas),2)]
```

Ejercicios

```
peas[1:3]
```

```
## [1] 4 7 6
```

```
peas[seq(2,length(peas),2)]
```

```
## [1] 7 5 7
```

```
peas[1:length(peas) %% 2 == 0]
```

Ejercicios

```
peas[1:3]
```

```
## [1] 4 7 6
```

```
peas[seq(2,length(peas),2)]
```

```
## [1] 7 5 7
```

```
peas[1:length(peas) %% 2 == 0]
```

```
## [1] 7 5 7
```

Ejercicios

```r
y <- 4.3
z <- y[-1]
length(z)
```

Ejercicios

```
y <- 4.3
z <- y[-1]
length(z)
```

```
## [1] 0
```

```
y <- c(8,3,5,7,6,6,8,9,2,3,9,4,10,4,11)
```

```
rev(sort(y))[1:3]
```

Ejercicios

```
y <- 4.3
z <- y[-1]
length(z)
```

```
## [1] 0
```

```
y <- c(8,3,5,7,6,6,8,9,2,3,9,4,10,4,11)
```

```
rev(sort(y))[1:3]
```

```
## [1] 11 10  9
```

Ejercicios

```
sum(rev(sort(y)))[1:3]
```

Ejercicios

```
sum(rev(sort(y)))[1:3]
```

```
## [1] 95 NA NA
```

```
names <- c("Williams","Jones","Smith","Williams",
           "Jones","Williams")
```

```
table(names)
```

Ejercicios

```
sum(rev(sort(y)))[1:3]
```

```
## [1] 95 NA NA
```

```
names <- c("Williams","Jones","Smith","Williams",
           "Jones","Williams")
```

```
table(names)
```

```
## names
##    Jones    Smith Williams
##        2        1        3
```

Ejercicios

```
unique(names)
```

Ejercicios

```
unique(names)
```

```
## [1] "Williams" "Jones"    "Smith"
```

```
duplicated(names)
```

Ejercicios

```
unique(names)
```

```
## [1] "Williams" "Jones"    "Smith"
```

```
duplicated(names)
```

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

```
names[!duplicated(names)]
```

Ejercicios

```
unique(names)
```

```
## [1] "Williams" "Jones"    "Smith"
```

```
duplicated(names)
```

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

```
names[!duplicated(names)]
```

```
## [1] "Williams" "Jones"    "Smith"
```

Ejercicios

```
salary <- c(42,42,48,42,42,42)
salary[!duplicated(names)]
```

Ejercicios

```
salary <- c(42,42,48,42,42,42)
salary[!duplicated(names)]
```

```
## [1] 42 42 48
```

Ejercicios

```
mean(salary[!duplicated(names)])
```

Ejercicios

```
mean(salary[!duplicated(names)])
```

```
## [1] 44
```

```
mean(salary[!duplicated(salary)])
```

Ejercicios

```r
mean(salary[!duplicated(names)])
```

```
## [1] 44
```

```r
mean(salary[!duplicated(salary)])
```

```
## [1] 45
```