Guia de R - PASE

Amilkar Gazque

9/22/2020

Ayuda en R

Para ver la documentación se puede usar el comando *help()*

```
help("svd")
```

Con el comando *apropos()* aparecerán las expresiones que contienen el término introducido.

```
apropos("mean")
```

```
##  [1] ".colMeans"     ".rowMeans"     "colMeans"      "kmeans"
##  [5] "mean"          "mean.Date"     "mean.default"  "mean.difftime"
##  [9] "mean.POSIXct"  "mean.POSIXlt"  "rowMeans"      "weighted.mean"
```

Operaciones aritméticas

Para la potencia usamos el acento circunflejo (^)

```
3^2
```

```
## [1] 9
```

Para el modulo hay:

```
31%%7
```

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

Para la asignación de valores se usa <-

```
un_valor <- 2*4+5-6/8
un_valor
```

```
## [1] 12.25
```

Ciclos

La sintaxis del ciclo *for* es:

for(indice in valor_inicial:valor_final){intrucciones}. Ejemplo

```
for(i in 1:5){
  print(i^2)
}
```

```
## [1] 1
## [1] 4
## [1] 9
## [1] 16
## [1] 25
```

O para sumar los elementos de un vector:

```
x <- c(1,2,3,4,5,6,7)
s <- 0
for(e in x){
  s=s+e}
s
```

```
## [1] 28
```

Los mismos ejemplos con el ciclo while:

```
i = 1
while(1){
  print(i^2)
  i = i + 1
  if(i == 6){break()}
}
```

```
## [1] 1
## [1] 4
## [1] 9
## [1] 16
## [1] 25
```

```
i = 1
s = 0
while(i <= length(x)){
  s = s + x[i]
  i = i + 1
}
s
```

```
## [1] 28
```

Vectores

Para crear un vector vacio:

```
vector <- c()
```

Para agregar un elemento:

```
vector <- c()
vector[1] = 0
vector
```

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

Tambien podemos agregar varios elementos

```
vector <- c(vector,1,2,3,4)
vector
```

```
## [1] 0 1 2 3 4
```

Para crear un vector de la forma $[n, n+1, n+2, ..., m]$

```
v <- 1:100
v
```

```
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##  [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
##  [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
##  [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
##  [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
##  [91]  91  92  93  94  95  96  97  98  99 100
```

En general podemos crear vectores de la forma $[n, n+r, n+2r, ...]$ con el comando *seq()*

```
v <- seq(1,20,2)
v
```

```
##  [1]  1  3  5  7  9 11 13 15 17 19
```

Si agregamos el parámetro length se generan r números entre n y m, igualmente espaciados

```
v<-seq(4,10, length = 10)
v
```

```
##  [1]  4.000000  4.666667  5.333333  6.000000  6.666667  7.333333  8.000000
##  [8]  8.666667  9.333333 10.000000
```

Con la instrucción rep(x,r) se genera una lista de r valores todos iguales a x.

```
v <- rep(0,10)
v
```

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

Para extraer elementos se usan corchetes []

```
v <- 1:10
v[5]
```

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

```
v[1:3]
```

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

Las operaciones con vectores son entrada a entrada

```
v**2
```

```
##  [1]   1   4   9  16  25  36  49  64  81 100
```

```
v/3
```

```
##  [1] 0.3333333 0.6666667 1.0000000 1.3333333 1.6666667 2.0000000 2.3333333
##  [8] 2.6666667 3.0000000 3.3333333
```

```
sqrt(v)
```

```
##  [1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751 2.828427
##  [9] 3.000000 3.162278
```

Para que las operaciones con dos vectores esten bien definidas es necesario que tengan la misma logitud, esto se puede revisar con el comando length()

Otras comandos utiles son sum(), cumsum(),mean(),var(), max(), min(), sort(), rev()

Tambien podemos hacer operaciones logicas como

```
v <- 1:20
v < 15
```

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

O bien

```
sum(v<7)
```

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

La instrucción x[x<r] devuelve los valores del vector x que verifican la condición impuesta x<r.

```
 v[v<7]
```

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

Matrices

Para crear matrices

```
matrix(1:12, nrow = 3, ncol = 4)
```

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

Para unir vectores, usando cada uno como una columna podemos usar cbind() y para unir vectores, usando cada uno como un renglón podemos usar rbind()

```
v1 <- 1:4
v2 <- 5:8
v3 <- 9:12
v4 <- 13:16
```

```
matriz_r <- rbind(v1, v2, v3, v4)
matriz_r
```

```
##    [,1] [,2] [,3] [,4]
## v1    1    2    3    4
## v2    5    6    7    8
## v3    9   10   11   12
## v4   13   14   15   16
```

```
matriz_c <- cbind(v1, v2, v3, v4)
matriz_c
```

```
##      v1 v2 v3 v4
## [1,]  1  5  9 13
## [2,]  2  6 10 14
## [3,]  3  7 11 15
## [4,]  4  8 12 16
```

Aunque tambien podemos crear una matriz llena de cero y despues cambiar sus valores,con los cochetes. Para revisar la forma de la matriz esta el comando dim().

```r
dim(matriz_r)
```

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

Algunas operaciones utiles con matrices son t(), det(), solve(), diag()

Funciones

La sintaxis para crear una función es:

nombre <- function(parametros){intrucciones} Ejemplo

```r
foo <- function(x){
  return(x - mean(x))
  }
foo(1:20)
```

```
##  [1] -9.5 -8.5 -7.5 -6.5 -5.5 -4.5 -3.5 -2.5 -1.5 -0.5  0.5  1.5  2.5  3.5  4.5
## [16]  5.5  6.5  7.5  8.5  9.5
```

Para hacer gráficas simples podemos usar el comando plot()

```r
x <- seq(0,2*pi,length = 1000,)
plot(x,sin(x))
```