Introducción a R

Modelos no paramétricos y de regresión

Enrique Reyes

01 de febrero de 2018

Tipos de objetos

R trabaja con distintos tipos de objetos, estos pueden ser del tipo estructurado o no estructurado; hablando de los objetos de tipo estructurado trabaja con: vectores, matrices, arreglos y bases de datos (data frame); en el caso de datos no estructurados trabaja principalmente con listas.

Vectores

Los vectores sólo pueden almacenar información de un sólo tipo (numérico, carácter o lógico), en caso de combinar un vector de tipo númerico con lógico, obtendremos un vector de tipo numérico como resultado final, en caso de combinar variables de tipo carácter con lógicos o numéricos, el resultado final será un vector de tipo carácter.

```
#Vector numérico
x<-c(1,2,3,4); x; str(x)
```

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

```
##  num [1:4] 1 2 3 4
```

```
#vector de texto
y<-c("R","S","T","W"); y; str(y)
```

```
## [1] "R" "S" "T" "W"
```

```
##  chr [1:4] "R" "S" "T" "W"
```

```
#vector lógico
z<-c(T,F,F,T); z; str(z)
```

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

```
##  logi [1:4] TRUE FALSE FALSE TRUE
```

```
#vector númerico
u=c(T,F,T,5); u; str(u)
```

```
## [1] 1 0 1 5
```

```
##  num [1:4] 1 0 1 5
```

```
#vector carácter
v=c("0",1,2,3,4); v; str(v)
```

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

```
##  chr [1:5] "0" "1" "2" "3" "4"
```

```
w=c("T",T,F,F,T); w; str(w)
```

```
## [1] "T"     "TRUE"  "FALSE" "FALSE" "TRUE"
```

```
##  chr [1:5] "T" "TRUE" "FALSE" "FALSE" "TRUE"
```

```r
#Se crean dos vectores
x<-c(1,2,3,11,12,20); x
```

```
## [1]  1  2  3 11 12 20
```

```r
y<-c(4,5,6,10,14,21); y
```

```
## [1]  4  5  6 10 14 21
```

```r
#Combinación de dos vectores
z<-c(x,y); z
```

```
##  [1]  1  2  3 11 12 20  4  5  6 10 14 21
```

```r
#Si quisieramos extraer las entradas 1,3,5
#error
#z(1,3,5)
#error
#z[1,3,5]
#Devuelve los elementos del vector seleccionadas
z[c(1,3,5)]
```

```
## [1]  1  3 12
```

```r
#Datos condicionados de un vector
z[z>10]
```

```
## [1] 11 12 20 14 21
```

```r
z[z<15]
```

```
##  [1]  1  2  3 11 12  4  5  6 10 14
```

```r
#error no acepta intervalos declarados de esta manera
#z[18>z>1]
#pero si de esta forma
z[z>1&z<18]
```

```
## [1]  2  3 11 12  4  5  6 10 14
```

```r
z[z>1|z<18]
```

```
##  [1]  1  2  3 11 12 20  4  5  6 10 14 21
```

```r
#actualiza el vector completo
z=z/6; z
```

```
##  [1] 0.1666667 0.3333333 0.5000000 1.8333333 2.0000000 3.3333333 0.6666667
##  [8] 0.8333333 1.0000000 1.6666667 2.3333333 3.5000000
```

```r
#Regresa las posiciones en x/y presentes en z
z[x]
```

```
## [1] 0.1666667 0.3333333 0.5000000 2.3333333 3.5000000        NA
```

```r
z[y]
```

```
## [1] 1.833333 2.000000 3.333333 1.666667       NA       NA
```

```r
#definamos un vector
g<-c(1,2,3,4); g
```

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

```r
#Total
sum(g)
```

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

```r
#Prodcuto
prod(g)
```

```
## [1] 24
```

```r
#Gasto semanal
gs<-c(12,45,0,9,6,25,30); gs
```

```
## [1] 12 45  0  9  6 25 30
```

```r
#Etiquetas
names(gs)<-c("Lunes","Martes","Miercoles","Jueves","Viernes","Sabado","Domingo"); gs
```

```
##     Lunes    Martes Miercoles    Jueves   Viernes    Sabado   Domingo
##        12        45         0         9         6        25        30
```

```r
gs[4]
```

```
## Jueves
##      9
```

```r
gs[gs==max(gs)]
```

```
## Martes
##     45
```

```r
names(gs[gs==max(gs)])
```

```
## [1] "Martes"
```

```r
max(gs)
```

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

```r
#Función secuencia
seq(from=7, to=38, by=3)
```

```
##  [1]  7 10 13 16 19 22 25 28 31 34 37
```

```r
c=seq(from=12, to=50); c
```

```
##  [1] 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
## [24] 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
```

```r
seq(from=100, to=10, by=-10)
```

```
##  [1] 100  90  80  70  60  50  40  30  20  10
```

```r
d=seq(from=1, to=10, by=0.1); d
```

```
##  [1]  1.0  1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9  2.0  2.1  2.2  2.3
## [15]  2.4  2.5  2.6  2.7  2.8  2.9  3.0  3.1  3.2  3.3  3.4  3.5  3.6  3.7
## [29]  3.8  3.9  4.0  4.1  4.2  4.3  4.4  4.5  4.6  4.7  4.8  4.9  5.0  5.1
## [43]  5.2  5.3  5.4  5.5  5.6  5.7  5.8  5.9  6.0  6.1  6.2  6.3  6.4  6.5
## [57]  6.6  6.7  6.8  6.9  7.0  7.1  7.2  7.3  7.4  7.5  7.6  7.7  7.8  7.9
## [71]  8.0  8.1  8.2  8.3  8.4  8.5  8.6  8.7  8.8  8.9  9.0  9.1  9.2  9.3
## [85]  9.4  9.5  9.6  9.7  9.8  9.9 10.0
```

```r
e=seq(from=1, to=10, length.out = 22); e
```

```
##  [1]  1.000000  1.428571  1.857143  2.285714  2.714286  3.142857  3.571429
##  [8]  4.000000  4.428571  4.857143  5.285714  5.714286  6.142857  6.571429
## [15]  7.000000  7.428571  7.857143  8.285714  8.714286  9.142857  9.571429
## [22] 10.000000
```

```r
#Función repetir
rep(1,5)
```

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

```r
rep(6,9)
```

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

```r
x<-c(1,2,3);x
```

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

```r
rep(x,2)
```

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

```r
rep(c(2,6,9),9)
```

```
##  [1] 2 6 9 2 6 9 2 6 9 2 6 9 2 6 9 2 6 9 2 6 9 2 6 9 2 6 9
```

```r
rep(1:5,2)
```

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

```r
rep(c(1,2,3),c(1,4,8))
```

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

```r
rep(1:4,c(1,2,3,4))
```

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

```r
#remplazar o actualizar uno o varios elementos
x
```

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

```r
x[1]=0; x
```

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

```r
x[c(1,3)]=9;x
```

```
## [1] 9 2 9
```

```r
#Valores faltantes
f=c(1,5,9,NA,5,NA,0); f
```

```
## [1]  1  5  9 NA  5 NA  0
```

```r
#Detecta los elementos vacios o faltantes del vector
sum(is.na(f))
```

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

```r
#Elimina los elementos vacios del vector
f<-f[!is.na(f)]; f
```

```
## [1] 1 5 9 5 0
```

```
#Función de longitud de un vector
length(f)
```

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

```
#Minimo
min(f)
```

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

```
#Maximo
max(f)
```

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

```
#Media o promedio
mean(f)
```

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

```
#Crea un vector de tipo caracter
v=c("a","d","g","h","a","x","V","a","d","g","h","a","x","V"); v
```

```
##  [1] "a" "d" "g" "h" "a" "x" "V" "a" "d" "g" "h" "a" "x" "V"
```

```
#Nos dice los elementos diferentes que tenemos y cuantas veces se repiten
table(v)
```

```
## v
## a d g h V x
## 4 2 2 2 2 2
```

```
#Calcula el tamaño del vector
length(v)
```

```
## [1] 14
```

```
#Creamos un vector de tipo categórico para hacer analisis de sus elementos
v1=as.factor(v); v1
```

```
##  [1] a d g h a x V a d g h a x V
## Levels: a d g h V x
```

```
#Se extraen los elementos del vector sin repetir
levels(v1)
```

```
## [1] "a" "d" "g" "h" "V" "x"
```

```
#Me dice cuantos elementos hay de cada elemento del vector
summary(v1)
```

```
## a d g h V x
## 4 2 2 2 2 2
```

```
#generamos un vector numerico
estaruras<-c(1.7,1.5,1.9,2,1.45,1.7,1.5,1.9,2,1.45,1.68,1.6,1.45,1.72); estaruras
```

```
##  [1] 1.70 1.50 1.90 2.00 1.45 1.70 1.50 1.90 2.00 1.45 1.68 1.60 1.45 1.72
```

```
length(estaruras)
```

```
## [1] 14
```

```
#Hace un analisis por clase, en este caso calcula la media
tapply(estaruras,v1,mean)
```

```
##      a      d      g      h      V      x
## 1.6625 1.7500 1.6750 1.8400 1.6100 1.5750
```

```
#Ordenar vectores numericos
#ordena en orden ascendente
estaturas=sort(estaruras); estaturas
```

```
##  [1] 1.45 1.45 1.45 1.50 1.50 1.60 1.68 1.70 1.70 1.72 1.90 1.90 2.00 2.00
```

```
#Ordena en orden decreciente
e1=sort(estaruras,decreasing = TRUE); e1
```

```
##  [1] 2.00 2.00 1.90 1.90 1.72 1.70 1.70 1.68 1.60 1.50 1.50 1.45 1.45 1.45
```

```
#Ordenar vector tipo caracter
calidad=c("media","baja","media","alta","media","baja","alta","baja"); calidad
```

```
## [1] "media" "baja"  "media" "alta"  "media" "baja"  "alta"  "baja"
```

```
#Le decimos que baja es peor calidad, media es el intermedio y alta es la mejor calidad
calidad1=ordered(calidad,c("baja","media","alta")); calidad1
```

```
## [1] media baja  media alta  media baja  alta  baja
## Levels: baja < media < alta
```

```
#ordena de manera ascendente
sort(calidad1)
```

```
## [1] baja  baja  baja  media media media alta  alta
## Levels: baja < media < alta
```

```
#Ordena de manera descendente
sort(calidad1,decreasing=TRUE)
```

```
## [1] alta  alta  media media media baja  baja  baja
## Levels: baja < media < alta
```

```
#Creamos dos vectores
y <- c("Aguascalientes", "Baja California", "Baja California", "Chihuahua",
       "Zacatecas", "Zacatecas", "Zacatecas", "Baja California", "Chihuahua")
z <- c("H", "M", "M", "M", "H", "M", "M", "M", "M")
cbind(EDO = y, SEXO = z) # forma una base de datos a partir de 2 vectores
```

```
##       EDO               SEXO
##  [1,] "Aguascalientes"  "H"
##  [2,] "Baja California" "M"
##  [3,] "Baja California" "M"
##  [4,] "Chihuahua"       "M"
##  [5,] "Zacatecas"       "H"
##  [6,] "Zacatecas"       "M"
##  [7,] "Zacatecas"       "M"
##  [8,] "Baja California" "M"
##  [9,] "Chihuahua"       "M"
```

```
table(y)    # Tabla de frecuencias
```

```
## y
##  Aguascalientes Baja California       Chihuahua       Zacatecas
##               1               3               2               3
```

```
table(z)
```

```
## z
## H M
## 2 7
```

```
table(z, y) # Tabla de contingencia
```

```
##    y
## z   Aguascalientes Baja California Chihuahua Zacatecas
##   H              1               0         0         1
##   M              0               3         2         2
```

```
#Remplazamos uno o algunos elementos del vector
replace(letters, c(1, 5, 9, 15, 21), c("A", "E", "I", "O", "U"))
```

```
##  [1] "A" "b" "c" "d" "E" "f" "g" "h" "I" "j" "k" "l" "m" "n" "O" "p" "q"
## [18] "r" "s" "t" "U" "v" "w" "x" "y" "z"
```

```
# Creamos un vector numérico
x <- c(4, 1.5, 6, 4, 10, 20, 1, 15, 0); x
```

```
## [1]  4.0  1.5  6.0  4.0 10.0 20.0  1.0 15.0  0.0
```

```
# Extraemos la longitud de un vector
n <- length(x); n
```

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

```
#Creamos un vector tipo lógico
id <- x > 6; id; id*1
```

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

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

```
sum(id)
```

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

```
(sum(id)/n)*100
```

```
## [1] 33.33333
```

```
(1:n)[id]
```

```
## [1] 5 6 8
```

```
x[id]
```

```
## [1] 10 20 15
```

```
x[(1:n)[id]]
```

```
## [1] 10 20 15
```

```
indices<-which(id) #visualiza los indices en los que se cumple id
indices
```

```
## [1] 5 6 8
```

```r
id2 <- y == "Baja California"; id2
```

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

```r
y[id2]
```

```
## [1] "Baja California" "Baja California" "Baja California"
```

```r
z[id2]
```

```
## [1] "M" "M" "M"
```