Taller de R. Clase 04

Dr. Isaías Moreno Cruz

02-10-2024

Vectorized Operations

Operaciones vectoriales

Muchas operaciones en R son vectorizadas haciendo el codigo más eficientes

R

```
x <- 1:4; y<-6:9
x + y
x > 2
x >= 2
y==8
x*y
x/y
```

Operaciones vectoriales con matrices

R

```
x <- matrix(1:4,2,2); y <- matrix(rep(10,4),2,2)
x*y
x/y
```

R

Verdadera multipliación matricial

```
x%*%y
```

Reading and Writing Data

Reading Data

Hay algunas funciones principales para leer datos en R

- `read.table`, `read.csv`, para leer datos tabulares
- `readLines`, para leer lineas de texto del archivo
- `source`, para leer archivos de código en R (`file.R`, inverso de `dump`)
- `dget`, para leer archivos de código en R (inverso de `dput`)
- `load`, para leer espacios de trabajo salvados (`file.rda`)
- `unserialize`, para leer objetos de R in forma binaria

Writing Data

Existen funciones analogas para la escritura de datos

- write.table
- writeLines
- dump
- dput
- save (file.rda)
- serialize

read.table

La función `read.table` es una de las funciones usadas más comunes para lear datos. Tienes pocos argumentos importantes:

- file
- header
- sep
- colClassess
- nrows
- comment.char
- skip
- stringsAsFactors

```
head ./data/preciosBCSday.csv
```

```
file <- "./data/preciosBCSday.csv"
data <- read.table(file, sep=',', header=TRUE)
head(data)
```

readLines

```
file <- "./data/homicides.txt"
data <- readLines(file)
data[2]
```

Control Structures

Estructuras de control en R. Estructuras comunes:

- `if, else`
- `for`
- `while`
- `repeat`
- `break`
- `next`
- `return`

if

R

```
if(<condition>) {
## do something
} else {
## do something
}
```

R

```
if(<condition>) {
## do something
} else if(<condition2>){
## do something
} else {
## do something
}
```

if II

R

```r
x <- 10

if(x>3){
  y <- 10
} else {
  y<-0
}

print(y)
```

for

R

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

R

```r
x <- c("a", "b", "c", "d")
for(i in 1:4) {
  print(x[i])
}
```

R

```r
for(i in seq_along(x)) {
  print(x[i])
}
```

Matrices

R

```
x <- matrix(1:6, 2,3)
for(i in seq_len(nrow(x))){
  for(j in seq_len(ncol(x))){
    print(x[i,j])
    }
  }
```

while

R

```r
count <- 0
while(count < 10){
  print(count)
  count <- count + 1
  }
```

Functions

R

```
f <- function(<arguments>){
## Do something
  }
```

Argument matching

R

```
args(lm)
```

Definir una función

R

```
d2r <- function(x){
  return(x*pi/180)
  }

d2r(180)
```

Argumento de "..."

El argumento de "..." indica un numero de variables de argumentos que son usadas en otras funciones

R

```
myplot <- function(x,y,type="l", ...){
plot(x,y,type=type, ...)
}
```

Funciones de funciones

R

```
make.power <- function(n){
  pow <- function(x){
    x^n
  }
  pow
}

cube <- make.power(3)
square <- make.power(2)

cube(3)
square(3)
```

Loop Function

Existen algunas funciones que implementan bucles para facilitar la programación

- `lapply`, loop sobre una lista y evalua una función en cada elemento
- `sapply`, lo mismo que `lapply` pero trata de simplificar el resultado
- `apply`, aplica una función sobre los margenes de un arreglo
- `tapply`, aplica una función sobre los subconjuntos de un vector
- `mapply`, versión multivariable de lapply

  Una función auxiliar es `split` también útil, en particular con `lapply`.

lapply

`lapply` siempre regresa una lista.

R

```r
x <- list(a=1:5, b=rnorm(10))
lapply(x, mean)
```

lapply II

`lapply`, y las demás, usa funciones anónimas

R

```
x <- list(a = matrix(1:4,2,2), b = matrix(1:6,3,2))
x
```

R

Y se desea extraer la primera columna de ambas matrices.

```
lapply(x, function(elt) elt[,1])
```

sapply

`sapply` trata de simplificar el resultado de `lapply`.

R

```
x <- list(a=1:4, b=rnorm(10), c=rnorm(20,1), d=rnorm(100,5))
lapply(x, mean)
```

R

```
sapply(x, mean)
```

apply

`apply` es usado para evaluar una función (frecuentemente una anónima) sobre el margen de un arreglo.

R

```
x <- matrix(rnorm(20), 5,4)
x
```

R

```
apply(x,2,mean)
```

R

```
apply(x,1,mean)
```

apply II

Para la suma y el promedio de las dimensiones de matrices, se tiene algunos atajos

- rowSums = apply(x,1,sum)
- rowMeans = apply(x,1,mean)
- colSums = apply(x,2,sum)
- colMeans = apply(x,2,mean)

Cuartiles en filas de matrices

R

```r
x<- matrix(rnorm(160), 8, 20)
apply(x, 1, quantile, probs=c(0.25, 0.5, 0.75))
```

tapply

`tapply` es usado para aplicar a una función sobre subconjuntos de un vector.

R

```
str(tapply)
```

- X es un vector
- INDEX es un factor o una lista de factores
- FUN es una función para ser aplicada
- ... contiene otros argumentos de la función FUN
- simplify, debe simplificar el resultado?

R

```
x <- c(rnorm(10), runif(10), rnorm(10,1))
f <- gl(3,10) ## Genera niveles de factores
f
```

R

split

`split` toma un vector o otros objetos y separarlo en grupos determinados por un factor o lista de factores.

```
x <- c(rnorm(10), runif(10), rnorm(10,1))
f <- gl(3,10)
split(x,f)
```

```
lapply( split(x,f), mean)
```

mapply

`mapply` es una aplicación multivariable.

```
str(mapply)
```

Ejercicio

Leer los archivos en dbeta y concatenarlos en un único archivo

```
rut <- "./data/dbeta"
files <- list.files(path=rut, pattern=".dat")
path <- paste(rut, "/", files[1], sep="")
df <- read.table(path, sep=",", header=FALSE, skip=1)
for(f in files[-1]){
  path <- paste(rut, "/", f, sep="")
  aux <- read.table(path, sep=",", header=FALSE, skip=1)
  df <- rbind(df, aux)
  }
```