Manipolazione di dati in R

Corso di Bioinformatica

Nicola Vitacolonna
Corso di Laurea in Biotecnologie

Alcune funzioni d'utilità generale

```
class(oggetto)  # classe di un oggetto (character, numeric, logical, etc…)
length(oggetto)  # numero di elementi o componenti di un oggetto
summary(oggetto)  # alcune sintesi statistiche sull'oggetto
names(oggetto)  # etichette di un oggetto
str(oggetto)  # struttura di un oggetto
args(funzione)  # elenca gli argomenti di una funzione
ls()  # Elenca gli oggetti attualmente in memoria
rm(oggetto)  # Rimuove un oggetto dalla memoria
```

Accedere ai dati

- Le parentesi quadre `[ ]` sono usate per estrarre un sottoinsieme dei dati (uno o piú elementi): producono sempre un oggetto della stessa classe dell'oggetto di partenza
- Le parentesi quadre doppie `[ [ ] ]` sono usate per estrarre un singolo elemento da una lista o da un data frame e la classe dell'oggetto risultante non è necessariamente la stessa dell'oggetto di partenza
- il dollaro `$` è usato per accedere agli elementi di una lista o di un data frame *per nome*; la semantica è simile a quella di `[ [ ] ]`

Vettori

```
x <- c("a", "b", "c", "c", "d", "a")
x[1]
```

```
## [1] "a"
```

```
x[2]
```

```
## [1] "b"
```

```
x[1:4]
```

```
## [1] "a" "b" "c" "c"
```

Filtrare con condizioni logiche

```
x[x > "a"]
```

```
## [1] "b" "c" "c" "d"
```

```
u <- (x > "a")
u
```

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

```
x[u]
```

```
## [1] "b" "c" "c" "d"
```

Rimuovere i valori nulli da un vettore

```
x <- c(1, 2, NA, 4, NA, 5)
valoriNulli <- is.na(x)  # Vettore logico
y <- x[!valoriNulli]
y
```

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

O combinando le operazioni in un comando:

```
y <- x[!is.na(x)]
y
```

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

Operazioni vettoriali

R opera con i vettori in modo intuitivo

```
x <- 1:4
y <- 6:9
x + y
```

```
## [1]  7  9 11 13
```

```
x - y
```

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

```
x * y  # Moltiplicazione elemento per elemento
```

```
## [1]  6 14 24 36
```

Operazioni vettoriali

```
x <- 1:4
y <- 6:9
x/y  # Divisione elemento per elemento
```

```
## [1] 0.1667 0.2857 0.3750 0.4444
```

```
x * 2  # Moltiplicazione per uno scalare
```

```
## [1] 2 4 6 8
```

```
x - 1  # Sottrazione di uno scalare da ciascun elemento del vettore
```

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

Operazioni vettoriali

```
x <- 1:4
x > 2
```

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

```
x >= 2
```

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

```
x == 4
```

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

Matrici

```
x <- matrix(1:6, 2, 3)
x
```

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

```
x[1, 2]
```

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

```
x[2, 1]
```

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

Gli indici di riga o gli indici di colonna possono essere omessi

```
x <- matrix(1:6, 2, 3)
x
```

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

```
x[1, ]  # Notate la virgola!
```

```
## [1] 1 3 5
```

```
x[, 2]  # Notate la virgola!
```

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

Quando si accede a un singolo elemento di una matrice, il risultato è normalmente un vettore di lunghezza 1. Se si vuole una matrice 1x1 bisogna specificare `drop = FALSE`:

```
x <- matrix(1:6, 2, 3)
x[1, 2]
```

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

```
x[1, 2, drop = FALSE]
```

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

Lo stesso vale per l'estrazione di una singola riga o colonna di una matrice:

```
x <- matrix(1:6, 2, 3)
x[1, ]
```

```
## [1] 1 3 5
```

```
x[1, , drop = FALSE]  # Prestate attenzione alle virgole!
```

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

Moltiplicazione di matrici

```
x <- matrix(1:4, 2, 2)
y <- matrix(rep(10, 4), 2, 2)
x * y  # Moltiplicazione elemento per elemento
```

```
##      [,1] [,2]
## [1,]   10   30
## [2,]   20   40
```

```
x %*% y  # Prodotto di matrici
```

```
##      [,1] [,2]
## [1,]   40   40
## [2,]   60   60
```

Liste

```
x <- list(primoElemento = 1:4, secondoElemento = 0.6)
x
```

```
## $primoElemento
## [1] 1 2 3 4
##
## $secondoElemento
## [1] 0.6
```

```
x[1]  # Produce una sotto-lista
```

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

```
x[[1]]  # Produce un vettore dal primo elemento della lista
```

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

```
x$primoElemento  # Idem
```

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

- L'accesso per nome è conveniente perché non è necessario conoscere la *posizione* dell'elemento desiderato
- `$` si può usare *soltanto* con *nomi* (ad esempio, scrivere `x$1` per selezionare il primo elemento della lista è un errore)

Data frame

```
data(airquality)
airquality[2:4, ]  # Notate la virgola!
```

```
##   Ozone Solar.R Wind Temp Month Day
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
```

```
airquality[c(1, 3, 4), c("Temp", "Month", "Day")]
```

```
##   Temp Month Day
## 1   67     5   1
## 3   74     5   3
## 4   62     5   4
```

Data frame

```
airquality[, c(1, 3)]  # Seleziona la prima e terza colonna (notate la virgola!)
```

```
##     Ozone Wind
## 1      41  7.4
## 2      36  8.0
## 3      12 12.6
## 4      18 11.5
## 5      NA 14.3
## 6      28 14.9
## 7      23  8.6
## 8      19 13.8
## 9       8 20.1
## 10     NA  8.6
## 11      7  6.9
## 12     16  9.7
## 13     11  9.2
## 14     14 10.9
## 15     18 13.2
## 16     14 11.5
## 17     34 12.0
```

Rimuovere i valori nulli da un data frame

Con la funzione `complete.cases()`

```
ok <- complete.cases(airquality)  # Produce un vettore logico
airquality[ok, ][1:5, ]
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 7    23     299  8.6   65     5   7
```

Filtrare gli indici

La funzione `which()` costruisce un vettore che risponde alla domanda: quali righe (di un altro vettore o di un data frame) soddisfano una data condizione?

```
# In quali righe la velocità del vento è minore di 2 mph?
indici <- which(airquality$Wind < 2)
indici
```

```
## [1] 53
```

Si può filtrare un data frame usando il vettore di indici cosí costruito:

```
airquality[indici, ]  # Non dimenticare la virgola!
```

```
##    Ozone Solar.R Wind Temp Month Day
## 53    NA      59  1.7   76     6  22
```

Campionamento casuale da un vettore

La funzione `sample()` estrae dati in modo casuale da un determinato insieme di dati

```
x <- 1:10
sample(x, 4)
```

```
## [1] 6 8 9 1
```

```
sample(x, 4)
```

```
## [1]  3  5 10  4
```

Campionamento casuale con ripetizioni

```
sample(x, 4, replace = TRUE)  # Campionamento con ripetizioni
```

```
## [1] 2 8 6 2
```

Se si vuole ottenere un campionamento riproducibile, bisogna usare la funzione `set.seed()` (usando come argomento un numero intero) *immediatamente prima* di usare `sample()`

```
set.seed(2)
sample(x, 4, replace = TRUE)
```

```
## [1] 2 8 6 2
```

```
set.seed(2)
sample(x, 4, replace = TRUE)
```

```
## [1] 2 8 6 2
```

Campionamento casuale da un data frame

La funzione `nrow()` calcola il numero di righe di un data frame.

```
data(women)  # Carica un data set predefinito con altezze e pesi di 15 donne
indici <- sample(1:nrow(women), 3)
women[indici, ]  # Filtra il data frame
```

```
##    height weight
## 15     72    164
## 14     71    159
## 2      59    117
```

La funzione `ncol()` calcola il numero di colonne:

```
ncol(women)
```

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