# Il ciclo for

Vedi anche "Cicli for", in *R for Data Science - Edizione italiana*

Quando ci troviamo a ripetere uno stesso comando più di una volta (tipicamente con il copia-incolla), è possibile utilizzare il ciclo `for`.

La forma generale di `for` è

```
for(var in seq) {
  codice
}
```

R costruisce una variabile (`var`) che assume sequenzialmente tutti i valori indicati in `seq`. In questo modo, possiamo scrivere il codice facendo riferimento alla variabile stessa. Ad esempio:

```
for(i in 1:10) {
  print(3 * i)
}
```

```
## [1] 3
## [1] 6
## [1] 9
## [1] 12
## [1] 15
## [1] 18
## [1] 21
## [1] 24
## [1] 27
## [1] 30
```

oppure

```
for(i in seq(from = 1, to = 10, by = 1)) {
  print(3 * i)
}
```

```
## [1] 3
## [1] 6
## [1] 9
## [1] 12
## [1] 15
## [1] 18
## [1] 21
```

```
## [1] 24
## [1] 27
## [1] 30
```

o anche:

```
for(i in seq(1, 10, 1)) {
  print(3 * i)
}
```

La variabile non deve chiamarsi necessariamente `i`.

Calcoliamo le medie di una serie di variabili:

```
library(LabRS)
data(MYSLID)

for (i in 2:4) {
  print(mean(MYSLID[[i]], na.rm = TRUE))
}
```

```
## [1] 15.55308
## [1] 12.49608
## [1] 43.98276
```

Costruiamo un risultato un po' più leggibile:

```
for (i in 2:4) {
  print(c(colnames(MYSLID[i]),
          mean(MYSLID[[i]], na.rm = TRUE))
  )
}
```

```
## [1] "Retribuzione"     "15.5530817458404"
## [1] "Istruzione"       "12.4960841694537"
## [1] "Eta"              "43.9827609427609"
```

# Risultati in oggetti

Possiamo anche costruire un oggetto con i risultati. L'oggetto va però dichiarato prima (costruito e definito prima del ciclo):

```
res <- array()                # dichiaro l'oggetto come vettore
for (i in 2:4) {
```

```
  res[[i-1]] <- mean(MYSLID[[i]], na.rm = TRUE)
}

res
```

```
## [1] 15.55308 12.49608 43.98276
```

```
class(res)
```

```
## [1] "numeric"
```

L'oggetto può essere di qualunque tipo.

# Indicizzazione

Come si nota nell'esempio, una delle cose a cui prestare attenzione è l'indicizzazione degli oggetti. Dal momento che le tre variabili hanno indici 2, 3 e 4, l'indicizzazione del vettore va indicata con `i-1`, si verrebbe a creare un elemento vuoto:

```
res <- array()
for (i in 2:4) {
  res[[i]] <- mean(MYSLID[[i]], na.rm = TRUE)
}

res
```

```
## [1]       NA 15.55308 12.49608 43.98276
```

Non ci può essere infatti un elemento 2 senza un elemento 1, in quanto l'indice indica l'ordine nella sequenza (vedi Indicizzazione).

# Cicli annidati (nested)

Vediamo un semplice esempio di ciclo annidato per costruire una matrice (tabella di Pitagora):

```
# dichiaro l'oggetto
pitagora <- matrix(nrow = 10,
                   ncol = 10)

# ciclo for
# righe
```

```
for(i in 1:10) {
  # colonne
  for(j in 1:10) {
    pitagora[i, j] <- i * j
  }
}

pitagora
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]    1    2    3    4    5    6    7    8    9    10
##  [2,]    2    4    6    8   10   12   14   16   18    20
##  [3,]    3    6    9   12   15   18   21   24   27    30
##  [4,]    4    8   12   16   20   24   28   32   36    40
##  [5,]    5   10   15   20   25   30   35   40   45    50
##  [6,]    6   12   18   24   30   36   42   48   54    60
##  [7,]    7   14   21   28   35   42   49   56   63    70
##  [8,]    8   16   24   32   40   48   56   64   72    80
##  [9,]    9   18   27   36   45   54   63   72   81    90
## [10,]   10   20   30   40   50   60   70   80   90   100
```

Esempio di indicizzazione degli elementi, con metà tabella:

```
# dichiaro l'oggetto
pitagora <- matrix(nrow = 10,
                   ncol = 5,
                   # in questo caso aggiungo i nomi di riga e colonna
                   dimnames = list(seq(1, 10, 1),
                                   seq(6, 10, 1)))

# ciclo for
for(i in 1:10) {
  for(j in 6:10) {
  # i valori di j non corrispondono alle posizioni degli elementi,
  # quindi 'j - 5'
    pitagora[i, j-5] <- i * j
  }
}

pitagora
```

```
##     6  7  8  9  10
## 1   6  7  8  9  10
## 2  12 14 16 18  20
## 3  18 21 24 27  30
## 4  24 28 32 36  40
```

```
## 5  30 35 40 45  50
## 6  36 42 48 54  60
## 7  42 49 56 63  70
## 8  48 56 64 72  80
## 9  54 63 72 81  90
## 10 60 70 80 90 100
```

# Vedi

- I cicli con lapply e sapply
- purrr_map

- Control - RDocumentation
- seq - RDocumentation
- Iterare una sequenza di istruzioni: il ciclo for - InsulaR
- How to write the first for loop in R - RBloggers

Funzioni, Coding