Università degli Studi di Milano

Laurea Specialistica in Genomica Funzionale e Bioinformatica

Corso di Linguaggi di Programmazione per la Bioinformatica

Vettori ed assegnamenti in R

Giorgio Valentini
e –mail: *valentini@dsi.unimi.it*

DSI – Dipartimento di Scienze dell' Informazione
Università degli Studi di Milano

1

Strutture dati in R

Nei linguaggi di programmazione ad alto livello non si ha accesso diretto alla memoria, ma ad una sua astrazione (*struttura dati*).

Le principali strutture dati fornite da R sono:

1. Vettori
2. Array e matrici
3. Fattori
4. Liste
5. Data frame

2

Vettori

Rappresentano sequenze ordinate di dati omogenei.

Ad es: sequenze ordinate di numeri o di caratteri.

Esempio:

```
> c(1,4,5)  # crea un vettore di interi
> c("A","B","C") # crea un vettore di
  caratteri
> c("gatto", "topo", "12") # crea un
  vettore di stringhe
```

La *funzione* c(arg1, arg2, arg3, arg4) combina i suoi argomenti in vettore.

3

Variabili ed assegnamenti - 1

Un vettore può essere *assegnato* ad una *variabile*.

Esempio 1

```
>X <- c(1,4,5) # il vettore <1 4 5> è assegnato alla
   variabile X
> X
[1] 1 4 5
```

La variabile X rappresenta ora il vettore <1 4 5>: si può pensare come un "contenitore" del vettore y

Es. 2

```
>X <- c(4,7) # il vettore <4 7> è assegnato alla
   variabile X
> X
[1] 4 7
```

Un nuovo assegnamento cancella il contenuto precedente

Es.3

```
> Y <- 100  # vettore formato da 1 solo elemento
> Y
[1] 100
```

4

Variabili ed assegnamenti - 2

Altri modi per rappresentare l' assegnamento:

Es.1

```
> c(1,4,5) -> x
> x = c(1,4,5)
> assign(x, c(1,4,5))
```

I valori di una variabile possono essere assegnati ad altre variabili:

Es.2

```
> y <- 2
> z <- y
> z
[1] 2
```

Non possono essere assegnati valori ad una costante:

```
> 2 <- x
Error in 2 <- x : invalid (do_set) left-hand side
  to assignment
```

5

Concatenazione di vettori

I vettori possono essere concatenati attraverso l' operatore c di concatenazione:

```
> x <- c(1,2,3)
> y <- c(4,5,6)
> z <- c(x,y)
> z
[1] 1 2 3 4 5 6
> w <- c(z,x,9,y)
> w
 [1] 1 2 3 4 5 6 1 2 3 9 4 5 6
```

6

Tipi elementari di vettori

I vettori sono sequenze ordinate i cui elementi possono essere di 3 tipi base:

- Numerici: numeri interi o in virgola mobile (floating point)
- Caratteri: singoli caratteri o stringhe (sequenze) di caratteri
- Logici: TRUE o FALSE

7

Operazioni con vettori aritmetici

Le operazioni usuali dell' aritmetica vengono eseguite sui vettori elemento per elemento:

Addizione e sottrazione

```
> x <- c(1,2,3)
> y <- c(4,5,6)
> z <- x + y
> z
[1] 5 7 9
> d <- y - x
> d
[1] 3 3 3
```

Moltiplicazione e divisione

```
> x <- c(1,2,3)
> y <- c(4,5,6)
> p <- x * y
> p
[1]  4 10 18
> q <- y / x
> q
[1] 4.0 2.5 2.0
```

8

Funzioni matematiche

Sono disponibili diversi operatori e funzioni matematiche (che operano sempre elemento per elemento). Ad esempio:

```
> x <- c(1,2,3)
> x^3
[1]  1  8 27
> log(x)
[1] 0.0000000 0.6931472 1.0986123
> exp(x)
[1]  2.718282  7.389056 20.085537
> sin(x)
[1] 0.8414710 0.9092974 0.1411200
> sqrt(x)
[1] 1.000000 1.414214 1.732051
```

9

Altre funzioni di uso comune

```
> x <- c(1,2,3)
> mean(x)
[1] 2
> var(x)
[1] 1
> max(x)
[1] 3
> min(x)
[1] 1
> range(x)
[1] 1 3
> sum(x)
[1] 6
> prod(x)
[1] 6
```

```
> y <-rnorm(10) # generazione ~N(0,1)
> y
 [1] -1.5171592  0.5538263 -0.9505327
-0.6218845  0.5113505  0.7547935
 [7] -1.5403415  2.3607231  1.3177109
-1.3993465
> sort(y) # ordinamento
 [1] -1.5403415 -1.5171592 -1.3993465
-0.9505327 -0.6218845  0.5113505
 [7]  0.5538263  0.7547935  1.3177109
2.3607231
> order(y)  # indici corrispondenti
            # agli elementi ordinati
 [1]  7  1 10  3  4  5  2  6  9  8
```

10

La “regola del riciclo” in R (1)

Se si sommano due vettori di diversa lunghezza in R, il vettore più corto viene ripetuto tante volte fino a raggiungere la lunghezza del vettore di maggior lunghezza.
Es:

```
> x <- c(2,4)
> y <- c(3,5,7,9)
> x + y  # il vettore x viene ripetuto due
         # volte (regola del riciclo)
[1]  5  9  9 13
```

La somma precedente equivale cioè a:

```
> xx <- c(x,x)  # duplicazione del vettore x
> xx + y
[1]  5  9  9 13
```

11

La “regola del riciclo” in R (2)

La regola vale in generale in R, anche per altre strutture dati e per altre operazioni.

Es:

```
> x <- c(2,4)
> xx <- c(x,x)
> y <- c(3,5,7,9)
> xx * y
[1]  6 20 14 36
> x * y # il vettore x viene ripetuto due
        # volte (regola del riciclo)
[1]  6 20 14 36
```

12

Generazione di sequenze regolari

R dispone di diversi comandi per generare automaticamente sequenze di numeri:

```
> c(1:10)
 [1]  1  2  3  4  5  6  7  8  9 10
> c (5:1)
[1] 5 4 3 2 1
> seq (1,10)
 [1]  1  2  3  4  5  6  7  8  9 10
> seq (from=1, to=4, by=0.5)
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0
```

La funzione seq() può avere 5 argomenti (si veda l' help). Un' altra funzione per generare repliche di vettori è rep():

```
> rep( c(1,2), times=4)
[1] 1 2 1 2 1 2 1 2
```

13

Vettori di caratteri

Gli elementi sono caratteri o stringhe di caratteri:

```
> x <- c("A","T","G")
> x
[1] "A" "T" "G"
> y <- c("ATA","TTTG","GCTCG")
> y
[1] "ATA"   "TTTG"  "GCTCG"
```

La funzione paste concatena 1 o piu' argomenti separandoli di default con degli spazi o con i caratteri specificati dall' argomento sep:

```
> paste("A","T","G")
[1] "A T G"
> paste("A","T","G", sep="")
[1] "ATG"
> paste("A","T","G", sep="C")
[1] "ACTCG"
> paste (x,y,sep="--")
[1] "A--ATA"   "T--TTTG"  "G--GCTCG"
```

14

Vettori logici

Sono vettori i cui elementi possono assumere valore TRUE o FALSE.

Es:

```
> x <- c(TRUE, FALSE, TRUE, FALSE)
> x
[1]  TRUE FALSE  TRUE FALSE
```

I vettori logici possono essere generati da condizioni e operazioni logiche.

Es:

```
> x <- 1:5
> x
[1] 1 2 3 4 5
> l <- x > 3  # la condizione logica è valutata
   # elemento per elemento
> l
[1] FALSE FALSE FALSE  TRUE  TRUE
```

15

Operatori logici

Operatori logici:

`<, <=, >, >=, ==` (uguaglianza), `!=` (disuguaglianza)

Es:

```
> x<- 3:8
> x >5
[1] FALSE FALSE FALSE  TRUE  TRUE  TRUE
> x <= 5
[1]  TRUE  TRUE  TRUE FALSE FALSE FALSE
> x == 5
[1] FALSE FALSE  TRUE FALSE FALSE FALSE
> x != 5
[1]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
> x != c(5,6) # vale la "regola del riciclo"!
[1]  TRUE  TRUE FALSE FALSE  TRUE  TRUE
```

16

Connettivi logici

Se c1 e c2 sono espressioni logiche, allora possono essere connesse tramite:

- & (AND)
- | (OR)
- ! (NOT)

Es 1:

```
> c1 <- 5>3
> c1
[1] TRUE
> c2 <- "gatto" == "topo"
> c2
[1] FALSE
```

```
> c3 <- c1 & c2
> c3
[1] FALSE
> c3 <- c1 | c2
> c3
[1] TRUE
> c3 <- !c1
> c3
[1] FALSE
```

Es. 2: I connettivi logici operano sui vettori elemento per elemento :

> c1 <- c(3,4) > c(2,6)

> c2 <- c(1,2) < c(2,8)

> c1 & c2

[1] TRUE FALSE

> c1 | c2

[1] TRUE TRUE

17

Dati mancanti

- In molte situazioni reali i componenti di un vettore possono essere "non noti" o comunque non disponibili.
- In questi casi R riserva il valore speciale NA ("Not Available").
- In generale qualunque operazione che coinvolga valori NA ha come risultato NA.

Es:

```
> x <- c(1:4,NA)
> x + 2
[1]  3  4  5  6 NA
```

Si noti che NA non è un valore ma un "marcatore" di una quantità non disponibile:

```
> x == NA
[1] NA NA NA NA NA
```

Per individuare quali elementi siano effettivamente NA in un vettore si deve usare la funzione is.na():

```
> is.na(x)
[1] FALSE FALSE FALSE FALSE  TRUE
```

18

Vettori: selezione e accesso a sottoinsiemi di elementi

Esistono diverse modalità di accesso a singoli elementi o a sottoinsiemi di elementi di un vettore. In generale la selezione e l' accesso avviene tramite l' operatore [] (parentesi quadre) : sottoinsiemi di elementi di un vettore sono selezionati collegando al nome del vettore un vettore di indici in parentesi quadre. Esistono *4 diverse modalità di selezione/accesso*:

- Vettori di indici interi positivi
- Vettore di indici interi negativi
- Vettore di indici logici
- Vettori di indici a caratteri

19

Selezione ed accesso tramite vettori di indici interi positivi (1)

Gli elementi di un vettore *x* sono selezionati tramite un vettore *y* di indici positivi racchiuso fra parentesi quadre : x[y]
i corrispondenti elementi sono selezionati e concatenati.

Es:

```
> x <- 5:10
> x[1] # selezione di un singolo elemento
[1] 5
> x[5]
[1] 9
> length(x) # lunghezza del vettore
[1] 6
> x[7] # accesso ad un elemento "fuori range"
[1] NA
```

20

Selezione ed accesso tramite vettori di indici interi positivi (2)

Si possono selezionare più elementi utilizzando vettori di indici positivi. Ad es:

```
> x <- 5:10
> x[2:4]
[1] 6 7 8
> x[c(1,3,5)]
[1] 5 7 9
> x[1:8]  # il vettore contiene solo 6 elementi!
[1]  5  6  7  8  9 10 NA NA
```

Un esempio un pò più complicato:

```
> s <- c("A","T")[rep(c(1,2,2,1), times=3)]
> s
 [1] "A" "T" "T" "A" "A" "T" "T" "A" "A" "T" "T" "A"
```

Si possono anche assegnare sottoinsiemi di elementi ad un vettore:

```
> y <- c("G","C","G","C")
> s[1:3] <- y[2:4]
> s
 [1] "C" "G" "C" "A" "A" "T" "T" "A" "A" "T" "T" "A"
```

21

Selezione ed accesso tramite vettori di indici interi negativi

Sono selezionati gli elementi di un vettore *x* che devono essere esclusi tramite un vettore *y* di indici negativi racchiuso fra parentesi quadre : x[y]
Es:

```
> y <- rep(c("G","A","T","T"), times=3)
> y
 [1] "G" "A" "T" "T" "G" "A" "T" "T" "G" "A" "T" "T"
> z <- y[-(1:5)] # selezionati tutti gli elementi di y
                 # eccetto primi 5
> z
[1] "A" "T" "T" "G" "A" "T" "T"
> z <- z[-length(z)]  # cancellazione dell' ultimo
                      # elemento di z
> z
[1] "A" "T" "T" "G" "A" "T"
```

22

Selezione ed accesso tramite vettori indice logici

Il vettore indice deve essere della stessa lunghezza del vettore i cui elementi devono essere selezionati. Sono selezionati gli elementi corrispondenti a TRUE nel vettore degli indici ed omessi quelli corrispondenti a FALSE.

Es:

```
> x <- c(1:5,NA,NA)
> x
[1]  1  2  3  4  5 NA NA
> i <- c(rep(TRUE,times=3),rep(FALSE,times=4))
> i # i è il vettore indice logico
[1]  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE
> x[i]  # selez.  elementi tramite vett. indice logico
[1] 1 2 3
> x[!is.na(x)] # selezione elementi che non sono NA
[1] 1 2 3 4 5
> x[!is.na(x) & x > 2]
[1] 3 4 5
```

23

Selezione ed accesso tramite vettori di indici a caratteri

E' applicabile quando un vettore possiede un attributo[1] names per identificare le sue componenti. In questo caso un sottovettore del vettore names può essere utilizzato per selezionare le componenti

Es:

```
> campione <- c(45,210,5.12,73.22,0.82)
> names(campione) <- c("Eta", "Conc.ciclosporina",
"Dose.giornaliera", "Conc.urea", "Conc.bilirubina")
> campione
             Eta Conc.ciclosporina  Dose.giornaliera
           45.00            210.00              5.12
Conc.urea   Conc.bilirubina
    73.22              0.82
> Eta.Dose <- campione[c("Eta","Dose.giornaliera")]
> Eta.Dose
             Eta Dose.giornaliera
           45.00              5.12
```

[1] Gli attributi dei vettori e degli oggetti in R verranno trattati nella prossima lezione

24

Esercizi

1. Generare un vettore contenente i primi 100 interi positivi: Calcolare la media, la varianza e la deviazione standard dei suoi elementi.
2. Ripetere il precedente esercizio con un vettore di 100 numeri random estratti da una distibuzione normale standard (si veda la funzione *rnorm*)
3. (a) Costruire una sequenza *s* costituita da 3 ripetizioni in sequenza della stringa "CGCT".
   (b) Estrarre dalla sequenza ottenuta una sottosequenza *sub* in cui compaiano tutti gli elementi di *s* eccetto la lettera C.
   (c) Aggiungere in coda alla sequenza ottenuta 3 valori NA.
   (d) Riottenere la sequenza *sub* in (b) tramite la funzione *is.na*()
4. Cosa accade se si prova a costruire un vettore eterogeneo di numeri e caratteri ? E di numeri e di valori logici ?
5. Generare un vettore *vet* di 100 elementi casuali estratti secondo la distribuzione uniforme in [0,1] (vedi *runif*). Estarre da *vett* un vettore *subvet* i cui elementi abbiano valore v>0.2 e valore v<0.6. Estrarre da *subvet* gli elementi di indice pari ed assegnarli al vettore *w.* Trasformare *w* in modo che i suoi elementi siano normalizzati rispetto al loro valore medio ed alla deviazione standard.

25