

Università degli Studi di Milano

Laurea Specialistica in Genomica Funzionale e Bioinformatica

Corso di Linguaggi di Programmazione per la Bioinformatica

Leggere e scrivere dati da file

Giorgio Valentini

e –mail: *valentini@dsi.unimi.it*

DSI – Dipartimento di Scienze dell' Informazione

Università degli Studi di Milano

Lettura e scrittura di dati da file

- I dati utilizzati in bioinformatica sono usualmente di *grandi dimensioni* (ad es: file PDB che memorizzano la struttura tridimensionale delle proteine, file per la memorizzazione di dati di espressione genica, etc)
- Oggetti di grandi dimensioni sono usualmente memorizzati in *file esterni su memoria di massa*
- In R esistono diverse *funzioni di I/O* per la lettura e scrittura di file
- Esistono anche funzioni e facility per importare/esportare dati verso altri ambienti/linguaggi di programmazione
- Per maggiori dettagli si consulti il manuale *R Data Import/Export* disponibile on-line

Caricare e salvare oggetti in formato binario

- Caricare e salvare oggetti arbitrari in formato binario:
 - Salvare oggetti in formato binario:

```
> x <- runif(20);  
> y <- list(a = 1, b = TRUE, c = "oops");  
> save(x, y, file = "xy.Rdata");  
> rm(x, y)  
> x  
Errore: oggetto "x" non trovato
```
 - Caricare oggetti in formato binario:

```
> load("xy.Rdata");  
> ls()  
[1] "x"      "y"
```
- Caricare e salvare oggetti relativi ad un'intera sessione di lavoro:

```
> save.image();  
> load(".RData");
```

Scrittura su file di data frame

La funzione **write.table** *memorizza un data frame in un file.*

Sintassi: **write.table** (*x*, file=*"data"*)

data è il nome del file su cui verrà scritto il data frame *x*.

La funzione `write.table` possiede molti altri argomenti che permettono di modularne opportunamente la semantica.

Esempio:

```
> m1 <-matrix(1:12,nrow=2); v <- c("A", "C")
> daf3<-data.frame(m1,v); daf3
  X1 X2 X3 X4 X5 X6 v
1  1  3  5  7  9 11 A
2  2  4  6  8 10 12 C
> write.table(daf3,file="data.df") # memorizza nel file
# "data.df" il data frame daf3
```

Lettura di data frame da file

La funzione **read.table** legge un file memorizzato su disco, inserendo i dati direttamente in un data frame.

Il file esterno deve essere memorizzato nel modo seguente:

- La prima riga del file deve avere un nome per ciascuna variabile del data frame
- Le righe successive del file memorizzano le osservazioni che saranno memorizzate nel data frame
- Ciascuna di queste righe può avere come primo valore l' etichetta di riga (che sarà memorizzata nel' attributo row.names del data frame)
- Ciascun valore sulla riga è separato da un blank (spazio, tabulazione, etc)
- Possono essere selezionati altri separatori
- read.table dispone di molti altri parametri che si possono settare per esigenze particolari (vedi help).

Lettura di data frame da file: esempi

Il seguente data frame è memorizzato sul file “data.df”:

	X1	X2	X3	X4	X5	X6	v
1	1	3	5	7	9	11	A
2	2	4	6	8	10	12	C

La lettura viene effettuata tramite la funzione **read.table**:

```
daf4<-read.table("data.df")
```

```
> daf4
```

	X1	X2	X3	X4	X5	X6	v
1	1	3	5	7	9	11	A
2	2	4	6	8	10	12	C

Il file può naturalmente essere generato da altri programmi (purchè in ASCII), ad es: tramite un qualsiasi text editor, ed essere letto tramite read.table.

Lettura e scrittura di data frame : esempi

Sia `read.table`, sia `write table` possono avere altri argomenti opzionali:

```
> m1 <-matrix(1:12,nrow=2); v <- c("A","C")
> daf3<-data.frame(m1,v)
> write.table(daf3,file="data.df",col.names=paste("col",1:7,sep=""))
> read.table("data.df")
      col1 col2 col3 col4 col5 col6 col7
1       1     3     5     7     9    11    A
2       2     4     6     8    10    12    C

> write.table(daf3,file="data.df",sep = ",") # file memorizzato
# utilizzando la virgola come separatore: controllare con un editor
> read.table("data.df",sep=",")
  X1 X2 X3 X4 X5 X6 v
1  1  3  5  7  9 11 A
2  2  4  6  8 10 12 C
```

Funzioni generali per lettura/scrittura di file

- In R sono presenti diverse funzioni generali per lettura e scrittura di file in formato ASCII o binario.
- Ad es: la funzione **file** può aprire, creare o chiudere file e più in generale *connessioni*: ad es: file in scrittura e/o lettura, connessioni di rete tramite socket o descritte da URL.
- Ci occuperemo solo dell' insieme di funzioni per la scrittura/lettura di file.

Scrittura di file: esempio

```
> ff <- file("ex.data", "w") # apertura di un file in
  scrittura
>      cat("TITLE extra line", "2 3 5 7", "", "11 13 17",
  file = ff, sep = "\n") # scrittura d 4 linee di testo
>      cat("One more line\n", file = ff)
>      close(ff) # chiude la connessione al file
>      readLines("ex.data") # lettura delle righe dal file
[1] "TITLE extra line" "2 3 5 7"          ""
     "11 13 17"      "One more line"
>      unlink("ex.data") # cancella il file dal disco
```

Per scrivere dati su file si può usare anche la funzione `write` (utilizzata
usualmente per scrivere matrici)

Le funzioni di I/O si possono usare anche per il download/upload di file in rete:

```
x <- readLines("http://homes.dsi.unimi.it/~valenti/DATA/MICROARRAY-
DATA/Leukemia/Readme.Leukemia");
```

Lettura di file: esempio

```
> ff <- file("ex.data", "r") # apertura file in lettura
> readLines(ff) # lettura di tutto il file
[1] "TITLE extra line" "2 3 5 7"          ""          "11 13
    17"          "One more line"
> seek(ff,0) # "rewind" del file
[1] 54
> readLines(ff,n=1) # lettura d una riga alla volta
[1] "TITLE extra line"
> readLines(ff,n=1)
[1] "2 3 5 7"
> readLines(ff,n=1)
[1] ""
> readLines(ff,n=1)
[1] "11 13 17"
> readLines(ff,n=1)
[1] "One more line"
> readLines(ff,n=1) # esaurite le righe del file
character(0)
> close (ff) # chiusura file
```

La funzione scan

La funzione **scan** legge un file di input e memorizza i dati in un vettore o una lista.

Esempi:

A. Memorizzazione dati in un vettore

```
> x <- matrix(1:10, nrow=2)
> write (x, "data")
# scrittura della matrice
# su file
> xread <- scan ("data",0)
Read 10 items
> xread
[1] 1 2 3 4 5 6 7 8
9 10
```

B. Memorizzazione dati in una lista

Si supponga di avere un file "data" composto dalle seguenti linee:

```
A    0.1  0.2  Q
B    0.5  0.4  M
A    1.1  1.2  Q
Q    0.3  0.9  P
```

```
> inp <- scan("data",list("",
0,0,"")) # lettura file e
memorizzazione # in una lista:
si noti la # lettura "per
colonne"
```

Read 4 records

```
> inp
[[1]] "A" "B" "A" "Q"
[[2]] 0.1 0.5 1.1 0.3
[[3]] 0.2 0.4 1.2 0.9
[[4]] "Q" "M" "Q" "P"
```

Accesso a data set built-in

- Molti data set sono disponibili con R (data set built-in) ed altri sono contenuti nei package.
- Per listare i data set built-in si utilizza la funzione `data()`.
- Per caricare un data set built-in la sintassi è:
`> data (nome-data-built-in)`

Esempio:

```
> data(iris)
```

```
> iris
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
....					

Editing dei dati

- E' possibile utilizzare la funzione `edit` per effettuare cambiamenti “manuali” su matrici e data frame
- E' possibile anche utilizzare la funzione `edit` per costruire ex novo nuove matrici e data frame
- La funzione `edit` fornisce un ambiente di editing simile a quello di un foglio elettronico

Esempi:

```
> edit(iris) # editing di un data frame  
# esistente  
  
> new.data.frame <- edit (data.frame())  
# creazione di un nuovo data frame
```

Importare, esportare file in Excel

A. Usare write.table e read.table e le funzioni di conversione di Excel:

```
> data(iris)
> write.table(iris, file="iris.txt",
row.names=F)
```

Aprire “iris.txt” con Excel ed utilizzare le conversioni formato.

Per salvare un file Excel usare “Salva formato testo con tabulazioni”.

Per aprire il file in R:

```
> iris2<- read.table("iris2.txt", header=T)
```

B. Leggere e scrivere direttamente file Excel: il package xlsReadWrite:

```
> library(xlsReadWrite)
> data(iris)
> write.xls( iris, file="iris.xls",
colNames=TRUE);
> iris2 <- read.xls("iris.xls")
```

Esercizi

- Costruire un data frame *df1* di 5 righe con 6 variabili di cui 4 numeriche e 2 a caratteri. Memorizzare su file il data frame e quindi leggerlo, assegnandolo alla variabile *df2*.
- Costruire una matrice numerica utilizzando la funzione *edit*. Scriverla su file tramite la funzione *write*. Ricaricare quindi la matrice in memoria. Si potrebbero utilizzare altre funzioni per memorizzare la matrice?
- Scrivere su file il data frame *df1* dell' es. 1 separando però gli elementi con virgole, ed omettendo il nome delle variabili.
- Effettuare tramite R il download del file “colon128.train” dal sito <http://homes.dsi.unimi.it/~valenti/DATA/Colon>. Il file e' un data set con 31 campioni (righe), ognuno costituito da 129 feature separate da virgole.
- Carica dal package *Biobase* il data set *aaMap*. A cosa si riferisce? Tramite quale struttura dati è rappresentato?