

Il linguaggio R
Un invito ad approfondire

Giulio D'Agostini

Università di Roma La Sapienza e INFN
Roma, Italy

Sommario

- Perché?
- Elementi del linguaggio
- Qualche esempio

Perché R?

È un linguaggio di scripting:

- non necessita di compilazione
- adatto ad uso interattivo
- funzioni ad alto livello

Perché R?

È un linguaggio di scripting:

- non necessita di compilazione
- adatto ad uso interattivo
- funzioni ad alto livello
- multiplatforma

Perché R?

È un linguaggio di scripting:

- non necessita di compilazione
- adatto ad uso interattivo
- funzioni ad alto livello
- multiplatforma
- gratuito ('clone' di S-Plus)

Perché R?

È un linguaggio di scripting:

- non necessita di compilazione
- adatto ad uso interattivo
- funzioni ad alto livello
- multiplatforma
- gratuito ('clone' di S-Plus)
- open source (ci si può guardare dentro!)

Perché R?

È un linguaggio di scripting:

- non necessita di compilazione
- adatto ad uso interattivo
- funzioni ad alto livello

- multiplatforma
- gratuito ('clone' di S-Plus)
- open source (ci si può guardare dentro!)
- specializzato per uso statistico

R: pro e contro

R: pro e contro

- alto livello: funzioni potenti

R: pro e contro

- alto livello: funzioni potenti
- ... per fare quello che sa fare

R: pro e contro

- alto livello: funzioni potenti
- ... per fare quello che sa fare
- ma non abbastanza potente/fessibile per altro

R: pro e contro

- alto livello: funzioni potenti
- ... per fare quello che sa fare
- ma non abbastanza potente/fessibile per altro
- → (Fortran), C, C++, Java

R: pro e contro

- alto livello: funzioni potenti
- ... per fare quello che sa fare
- ma non abbastanza potente/fessibile per altro
- → (Fortran), C, C++, Java
- → Perl, Php, Python

R: pro e contro

- alto livello: funzioni potenti
- ... per fare quello che sa fare
- ma non abbastanza potente/fessibile per altro
- → (Fortran), C, C++, Java
- → Perl, Php, Python

⇒ ottimo per piccole/medie applicazioni matematiche e di analisi dati

R: pro e contro

- alto livello: funzioni potenti
- ... per fare quello che sa fare
- ma non abbastanza potente/fessibile per altro
- → (Fortran), C, C++, Java
- → Perl, Php, Python

⇒ ottimo per piccole/medie applicazioni matematiche e di analisi dati

Cominciamo con qualche esempio (⇒ `basi.R`)

R: basi del linguaggio - 1

- Sintassi nomi, operazioni elementari

R: basi del linguaggio - 1

- Sintassi nomi, operazioni elementari

R: basi del linguaggio - 1

- Sintassi nomi, operazioni elementari

R: basi del linguaggio - 1

- Sintassi nomi, operazioni elementari

R: basi del linguaggio - 1

- Sintassi nomi, operazioni elementari

R: basi del linguaggio - 1

- Sintassi nomi, operazioni elementari

R: basi del linguaggio - 1

- Sintassi nomi, operazioni elementari

R: basi del linguaggio - 1

- Sintassi nomi, operazioni elementari

R: basi del linguaggio - 2

- Strutture di controllo

R: basi del linguaggio - 2

- Strutture di controllo

R: basi del linguaggio - 2

- Strutture di controllo

R: basi del linguaggio - 2

- Strutture di controllo

R: basi del linguaggio - 2

- Strutture di controllo

R: basi del linguaggio - 2

- Strutture di controllo

R: basi del linguaggio - 2

- Strutture di controllo

Ambiente di lavoro e 'help'

Help molto potente

Ambiente di lavoro e 'help'

Help molto potente

- `help()` (notare parentesi!)

Ambiente di lavoro e 'help'

Help molto potente

- `help()` (notare parentesi!)
- `help('comando')`, ovvero `? 'comando'` (per molti comandi gli apicetti possono essere ignorati)

Ambiente di lavoro e 'help'

Help molto potente

- `help()` (notare parentesi!)
- `help('comando')`, ovvero `? 'comando'` (per molti comandi gli apicetti possono essere ignorati)
- `help` \neq `help()`
⇒ le funzioni senza argomento danno loro stesse! (con eccezione delle *primitive*)

Ambiente di lavoro e 'help'

Help molto potente

- `help()` (notare parentesi!)
- `help('comando')`, ovvero `? 'comando'` (per molti comandi gli apicetti possono essere ignorati)
- `help` \neq `help()`
⇒ le funzioni senza argomento danno loro stesse! (con eccezione delle *primitive*)
- `ls()`: variabili/funzioni della sessione

Ambiente di lavoro e 'help'

Help molto potente

- `help()` (notare parentesi!)
- `help('comando')`, ovvero `? 'comando'` (per molti comandi gli apicetti possono essere ignorati)
- `help` \neq `help()`
⇒ le funzioni senza argomento danno loro stesse! (con eccezione delle *primitive*)
- `ls()`: variabili/funzioni della sessione
- `ls.str()`: fornisce ulteriori dettagli

Ambiente di lavoro e 'help'

Help molto potente

- `help()` (notare parentesi!)
- `help('comando')`, ovvero `? 'comando'` (per molti comandi gli apicetti possono essere ignorati)
- `help` \neq `help()`
⇒ le funzioni senza argomento danno loro stesse! (con eccezione delle *primitive*)
- `ls()`: variabili/funzioni della sessione
- `ls.str()`: fornisce ulteriori dettagli
- `help.start()` ⇒ help su browser!

Sessione - 1

R si ricorda quello che abbiamo fatto!

Sessione - 1

R si ricorda quello che abbiamo fatto!

- ```
> q()
Save workspace image? [y/n/c]:
> y
```

## Sessione - 1

R si ricorda quello che abbiamo fatto!

- `> q()`  
Save workspace image? [y/n/c]:  
`> y`
- Informazioni della sessione su due file  
.RData  
.Rhistory

## Sessione - 1

R si ricorda quello che abbiamo fatto!

- ```
> q()  
Save workspace image? [y/n/c]:  
> y
```
- Informazioni della sessione su due file

```
.RData  
.Rhistory
```
- La volta successiva che R è chiamato, vengono ricaricati

```
.RData
```

 e

```
.Rhistory
```

 contenuti nella directory da cui lo si chiama

Sessione - 1

R si ricorda quello che abbiamo fatto!

- ```
> q()
Save workspace image? [y/n/c]:
> y
```
- Informazioni della sessione su due file  

```
.RData
.Rhistory
```
- La volta successiva che R è chiamato, vengono ricaricati `.RData` e `.Rhistory` contenuti nella directory da cui lo si chiama
- ⇒ creare diverse directory per diversi progetti

# Pulizia

Sovrascrittura e cancellazione degli oggetti

# Pulizia

## Sovrascrittura e cancellazione degli oggetti

- Ogni assegnazione sovrascrive l'eventuale oggetto avente stesso nome:

```
half <- function(x) x/2 (quasi come in C...)
```

```
half(pi)
```

```
half <- 1/2
```

# Pulizia

## Sovrascrittura e cancellazione degli oggetti

- Ogni assegnazione sovrascrive l'eventuale oggetto avente stesso nome:

```
half <- function(x) x/2 (quasi come in C...)
```

```
half(pi)
```

```
half <- 1/2
```

- Gli oggetti possono essere rimossi con `rm(oggetto)`, ad esempio `rm(half)`  
(attenzione: non chiede conferma!)

# Pulizia

## Sovrascrittura e cancellazione degli oggetti

- Ogni assegnazione sovrascrive l'eventuale oggetto avente stesso nome:

```
half <- function(x) x/2 (quasi come in C...)
```

```
half(pi)
```

```
half <- 1/2
```

- Gli oggetti possono essere rimossi con `rm(oggetto)`, ad esempio `rm(half)`  
(attenzione: non chiede conferma!)
- rimozioni multiple: `rm(x,y,pippo,ora)`

# Comandi di sistema

---

Interazione con il sistema operativo

# Comandi di sistema

---

## Interazione con il sistema operativo

- `system('comando di sistema')`  
esempio: `system('ls -la')`

# Comandi di sistema

---

## Interazione con il sistema operativo

- `system('comando di sistema')`  
esempio: `system('ls -la')`
- `system('date')`

# Comandi di sistema

---

## Interazione con il sistema operativo

- `system('comando di sistema')`  
esempio: `system('ls -la')`
- `system('date')`
- `oggi <- system('date +%d')`  
`ora <- system('date +%H')`  
`minuti <- system('date +%M')`

# Comandi di sistema

## Interazione con il sistema operativo

- `system('comando di sistema')`  
esempio: `system('ls -la')`
- `system('date')`
- `oggi <- system('date +%d')`  
`ora <- system('date +%H')`  
`minuti <- system('date +%M')`
- **Per eseguire *scripts*:**  
`source('file_di_comandi')`  
tipicamente `nome_file.R`

## Vettori, arrays, liste

---

Uno dei vantaggi di linguaggi tipo *R* rispetto a C/C++ etc. consiste nella gestione compatta di strutture di dati

- I '*vettori*' possono contenere solo dati omogenei (numerici o caratteri)
- Gli *arrays* possono estensioni multidimensionali (a molti indici) dei vettori. Il caso a due indici è una matrice
- Le *liste* sono 'collezioni ordinate di oggetti', anche di tipo diverso.

## Vettori: esempi ( $\Rightarrow$ `vettori.R`)

---

## Vettori: esempi ( $\Rightarrow$ `vettori.R`)

---

- `x <- c(0., 4.5, 1.3, 3.2)`

## Vettori: esempi ( $\Rightarrow$ `vettori.R`)

---

- `x <- c(0., 4.5, 1.3, 3.2)`
- `length(x), min(x), max(x), sum(x), prod(x), mean(x), sd(x)`

## Vettori: esempi ( $\Rightarrow$ `vettori.R`)

---

- `x <- c(0., 4.5, 1.3, 3.2)`
- `length(x), min(x), max(x), sum(x), prod(x), mean(x), sd(x)`
- `1/sqrt(x), sum(x^2)`

## Vettori: esempi ( $\Rightarrow$ `vettori.R`)

---

- `x <- c(0., 4.5, 1.3, 3.2)`
- `length(x)`, `min(x)`, `max(x)`, `sum(x)`, `prod(x)`,  
`mean(x)`, `sd(x)`
- `1/sqrt(x)`, `sum(x^2)`
- dato `y <- c(2, 5.2, 3, 0)`,  
si può calcolare `x+y`, `x/y`, etc.

## Vettori: esempi ( $\Rightarrow$ `vettori.R`)

---

- `x <- c(0., 4.5, 1.3, 3.2)`
- `length(x)`, `min(x)`, `max(x)`, `sum(x)`, `prod(x)`,  
`mean(x)`, `sd(x)`
- `1/sqrt(x)`, `sum(x^2)`
- dato `y <- c(2, 5.2, 3, 0)`,  
si può calcolare `x+y`, `x/y`, etc.
- R gestisce anche valori *vuoti*, *infiniti* o *indefiniti*

## Vettori: esempi ( $\Rightarrow$ `vettori.R`)

---

- `x <- c(0., 4.5, 1.3, 3.2)`
- `length(x)`, `min(x)`, `max(x)`, `sum(x)`, `prod(x)`,  
`mean(x)`, `sd(x)`
- `1/sqrt(x)`, `sum(x^2)`
- dato `y <- c(2, 5.2, 3, 0)`,  
si può calcolare `x+y`, `x/y`, etc.
- R gestisce anche valori *vuoti*, *infiniti* o *indefiniti*
- dato `z <- c(0, NA, 3, 1/0)`,  
quanto fa `x/z`?

## Vettori: esempi ( $\Rightarrow$ `vettori.R`)

- `x <- c(0., 4.5, 1.3, 3.2)`
- `length(x)`, `min(x)`, `max(x)`, `sum(x)`, `prod(x)`,  
`mean(x)`, `sd(x)`
- `1/sqrt(x)`, `sum(x^2)`
- dato `y <- c(2, 5.2, 3, 0)`,  
si può calcolare `x+y`, `x/y`, etc.
- R gestisce anche valori *vuoti*, *infiniti* o *indefiniti*
- dato `z <- c(0, NA, 3, 1/0)`,  
quanto fa `x/z`?
- $\Rightarrow$  NaN NA 0.4333333 0.0000000

## Vettori: esempi ( $\Rightarrow$ `vettori.R`)

- `x <- c(0., 4.5, 1.3, 3.2)`
- `length(x)`, `min(x)`, `max(x)`, `sum(x)`, `prod(x)`,  
`mean(x)`, `sd(x)`
- `1/sqrt(x)`, `sum(x^2)`
- dato `y <- c(2, 5.2, 3, 0)`,  
si può calcolare `x+y`, `x/y`, etc.
- R gestisce anche valori *vuoti*, *infiniti* o *indefiniti*
- dato `z <- c(0, NA, 3, 1/0)`,  
quanto fa `x/z`?
- $\Rightarrow$  `NaN NA 0.4333333 0.0000000`
- Funzioni per gestire NA, NaN e Inf:  
`is.na()`, `is.nan()`, `is.infinite()`, `is.finite()`

## Funzioni di probabilità ( $\Rightarrow$ `prob.R`)

---

R conosce le principali funzioni di probabilità

Dato nome, es. `binom`, `pois`, `norm`, `unif`, `geom`, `chisq`, ...

## Funzioni di probabilità ( $\Rightarrow$ `prob.R`)

---

R conosce le principali funzioni di probabilità

Dato nome, es. `binom`, `pois`, `norm`, `unif`, `geom`, `chisq`, ...

- `dnome(x, )`  $\Rightarrow$  funzione/densità di probabilità  $f(x)$ ;

## Funzioni di probabilità ( $\Rightarrow$ prob. R)

---

R conosce le principali funzioni di probabilità

Dato nome, es. `binom`, `pois`, `norm`, `unif`, `geom`, `chisq`, ...

- `dnome`(`x`, ...)  $\Rightarrow$  funzione/densità di probabilità  $f(x)$ ;
- `pnome`(`x`, ...)  $\Rightarrow$  cumulativa  $F(x)$ ;

## Funzioni di probabilità ( $\Rightarrow$ prob. R)

R conosce le principali funzioni di probabilità

Dato nome, es. `binom`, `pois`, `norm`, `unif`, `geom`, `chisq`, ...

- `dnome`( $x, \dots$ )  $\Rightarrow$  funzione/densità di probabilità  $f(x)$ ;
- `pnome`( $x, \dots$ )  $\Rightarrow$  cumulativa  $F(x)$ ;
- `qnome`( $p, \dots$ )  $\Rightarrow$  'quantili'  $x$  tale che  $p = F(x)$ ;

## Funzioni di probabilità ( $\Rightarrow$ prob. R)

R conosce le principali funzioni di probabilità

Dato nome, es. `binom`, `pois`, `norm`, `unif`, `geom`, `chisq`, ...

- `dnome`( $x$ , ...)  $\Rightarrow$  funzione/densità di probabilità  $f(x)$ ;
- `pnome`( $x$ , ...)  $\Rightarrow$  cumulativa  $F(x)$ ;
- `qnome`( $p$ , ...)  $\Rightarrow$  'quantili'  $x$  tale che  $p = F(x)$ ;
- `rnome`( $n$ , ...)  $\Rightarrow$   $n$  numeri (pseudo-)casuali.

## Funzioni di probabilità ( $\Rightarrow$ prob. R)

R conosce le principali funzioni di probabilità

Dato nome, es. `binom`, `pois`, `norm`, `unif`, `geom`, `chisq`, ...

- `dnome(x, )`  $\Rightarrow$  funzione/densità di probabilità  $f(x)$ ;
- `pnome(x, )`  $\Rightarrow$  cumulativa  $F(x)$ ;
- `qnome(p, )`  $\Rightarrow$  'quantili'  $x$  tale che  $p = F(x)$ ;
- `rnome(n, )`  $\Rightarrow$   $n$  numeri (pseudo-)casuali.
- Nota:  $x$  e  $p$  possono anche dei vettori.

## Grafica ( $\Rightarrow$ `plots.R`)

---

R ha molte funzioni per la grafica: `plot`, istogrammi, etc

## Grafica ( $\Rightarrow$ `plots.R`)

---

R ha molte funzioni per la grafica: `plot`, istogrammi, etc

- `plot(x, )` plot di `x` Vs numero d'ordine

## Grafica ( $\Rightarrow$ `plots.R`)

---

R ha molte funzioni per la grafica: `plot`, istogrammi, etc

- `plot(x, )` plot di `x` Vs numero d'ordine
- `plot(x,y, )` scatter plot `y` Vs `x`

## Grafica ( $\Rightarrow$ `plots.R`)

---

R ha molte funzioni per la grafica: `plot`, istogrammi, etc

- `plot(x, )` plot di `x` Vs numero d'ordine
- `plot(x,y, )` scatter plot `y` Vs `x`
- varie opzione di grafica, didascalie, etc.

## Grafica ( $\Rightarrow$ `plots.R`)

---

R ha molte funzioni per la grafica: plot, istogrammi, etc

- `plot(x, )` plot di `x` Vs numero d'ordine
- `plot(x,y, )` scatter plot `y` Vs `x`
- varie opzione di grafica, didascalie, etc.
- `hist(x, )`: istogramma

## Grafica ( $\Rightarrow$ `plots.R`)

---

R ha molte funzioni per la grafica: `plot`, istogrammi, etc

- `plot(x, )` plot di `x` Vs numero d'ordine
- `plot(x,y, )` scatter plot `y` Vs `x`
- varie opzione di grafica, didascalie, etc.
- `hist(x, )`: istogramma
- `curve(funzione(x), )`

## Altri esempi (e concetti)

## Altri esempi (e concetti)

---

- Piccolo esempio di uso di R come 'calcolatorino' dotato di grafica  $\Rightarrow$  RC.R

## Altri esempi (e concetti)

---

- Piccolo esempio di uso di R come 'calcolatorino' dotato di grafica  $\Rightarrow$  `RC.R`
- Data frame  $\Rightarrow$  `voti.R`

## Altri esempi (e concetti)

---

- Piccolo esempio di uso di R come 'calcolatorino' dotato di grafica  $\Rightarrow$  `RC.R`
- Data frame  $\Rightarrow$  `voti.R`
- Matrici  $\Rightarrow$  `matrici.R`

## Altri esempi (e concetti)

---

- Piccolo esempio di uso di R come 'calcolatorino' dotato di grafica  $\Rightarrow$  `RC.R`
- Data frame  $\Rightarrow$  `voti.R`
- Matrici  $\Rightarrow$  `matrici.R`
- Uso batch (calcoli, plot, etc. senza dover aprire una sessione):

## Altri esempi (e concetti)

---

- Piccolo esempio di uso di R come 'calcolatorino' dotato di grafica  $\Rightarrow$  `RC.R`
- Data frame  $\Rightarrow$  `voti.R`
- Matrici  $\Rightarrow$  `matrici.R`
- Uso batch (calcoli, plot, etc. senza dover aprire una sessione):

```
echo 'rnorm(10)' | R --slave
```

## Referenze

---

- Sito ufficiale  $\Rightarrow$  *googlare* R  
(così si impara anche che esiste un altro super-calcolatorino...)
- Download sorgenti/rpm per Unix, binary per Windows
- Molta documentazione: manuale, intro's, etc. inclusi nell'installazione; esempi in giro
- Possibilità di aggiungere pacchetti (molti!) da repositories in modo pressoché automatico

## Conclusioni

Vale la pena di provare  $\Rightarrow$  **buon divertimento!**

Ulteriori esempi:

- `mc1.R`: MC stima  $\pi$  e Teorema Limite Centrale
- `mc_lotto.R`: MC estrazioni al lotto
- `mc_gassiane.R`: MC gaussiane
- `mc_hit_miss.R`: MC, inversione  $F(x)$  Vs hit-miss
- `triang.R`: *dtriang*, *ptriang* e *rtriang*; simulazione somma di triangolari asimmetriche.
- `prop_cov.R`: propagazione varianze/covarianze (assume linearizzazione — derivate fatte da  $R$ )