Università degli Studi di Milano

Laurea Specialistica in Genomica Funzionale e Bioinformatica

Corso di Linguaggi di Programmazione per la Bioinformatica

Debugging

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

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

1

Debugging in R

- Gli strumenti per il debugging permettono di "congelare" l' esecuzione in particolari punti del codice e di controllare lo stato dell' esecuzione, eseguire il codice istruzione per istruzione e di visualizzare il contentuo delle variabili e dello stack.
- Esistono diversi strumenti per effettuare il debugging di programmi scritti in R:
  1. Funzione browser
  2. Funzione debug
  3. Funzione trace/untrace

2

Browser

- Una chiamata alla funzione browser all' interno di una funzione R ferma l' esecuzione nel punto della chiamata e fornisce all' utente un prompt da cui può eseguire comandi per verificare lo stato dell' esecuzione.
- Alcuni comandi eseguibili dal prompt del browser:
  - <RET> opp. 'c' : continua l' esecuzione
  - 'n' : esegue l'istuzione successiva
  - get("v") : ritorna il valore della variabile v
  - 'where' : ritorna lo stack delle chiamate
  - 'Q' : termina l' esecuzione e salta al prompt di R

3

Utilizzo della funzione browser: esempio

```
golub <- function(x,y)
{
  browser();
  mx <- mean(x);
  my <- mean(y);
  vx <- sd(x);
  vy <- sd(y);
  g <- (mx-my)/(vx+vy);
  g
}
```

```
> golub(x,y)
Called from: golub(x, y)
Browse[1]> c
[1] 0.408709
> golub(x,y)
Called from: golub(x, y)
Browse[1]> n
debug: mx <- mean(x)
Browse[1]> n
debug: my <- mean(y)
Browse[1]> n
debug: vx <- sd(x)
Browse[1]> get("my")
[1] 0.1161415
Browse[1]> where
where 1: golub(x, y)
Browse[1]> Q
>
```

4

Debug

- Il comando debug(fun) invoca il debugger sulla funzione fun. Le successive chiamate alla funzione fun lanciano il debugger
- Il debugger consente la valutazione delle istruzioni nel corpo della funzione sottoposta a debug
- I comandi eseguibili dal prompt sono simili a quelli della funzione browser.
- Il debugging viene inattivato dalla funzione undebug(fun)

5

Utilizzo della funzione debug: esempio

```
> debug(golub)
> golub(x,y)
debugging in: golub(x, y)
debug: {
    mx <- mean(x)
    my <- mean(y)
    vx <- sd(x)
    vy <- sd(y)
    g <- (mx - my)/(vx + vy)
    g
}
Browse[1]>
debug: mx <- mean(x)
Browse[1]>
debug: my <- mean(y)
Browse[1]>
debug: vx <- sd(x)
```

```
Browse[1]> mx
[1] 0.9413439
Browse[1]> get("mx")
[1] 0.9413439
Browse[1]> where
where 1: golub(x, y)

Browse[1]> c
exiting from: golub(x, y)
[1] 0.408709

> undebug(golub)
> golub(x,y)
[1] 0.408709
```

6

Trace/untrace

- Il comando trace(fun) fa sì che ogni volta che la funzione fun venga valutata, la corrispondente chiamata venga stampata sullo schermo.
- *trace* consente quindi di seguire l'esecuzione delle funzioni specificate chiamate all' interno di un determinato programma.
- Il comando untrace(fun) disattiva tale meccanismo di "tracciamento" della funzione.

Esempio:

```
> trace(golub)
> golub(x,y)
trace: golub(x, y)
[1] 0.408709
> untrace(golub)
> golub(x,y)
[1] 0.408709
```

7