

Statistics 120

Statistical Computing With R

The R System

- This course uses the R computing environment for practical examples.
- R serves both as a statistical package and as a general programming environment.
- R contains a large number of predefined graphical techniques and it is extensible so that new techniques can be easily added to it.
- R was developed at the University of Auckland by Ross Ihaka and Robert Gentleman, but has now matured into an internationally supported system.

Free Software

- R is an example of *free* software.
- Note the word free has two English meanings:
 - “free beer” (gratis in French)
 - “free speech” (libre in French)
- R is free in both senses. It is available free of charge, and you are free to copy it and give it away to your friends.
- R is a official part of the Free Software Foundation’s GNU suite of software.



An R Development Meeting (Vienna 2001)

Resources

- R is available from one of Comprehensive R Archive Network (CRAN) Web sites.

`http://cran.stat.auckland.ac.nz`

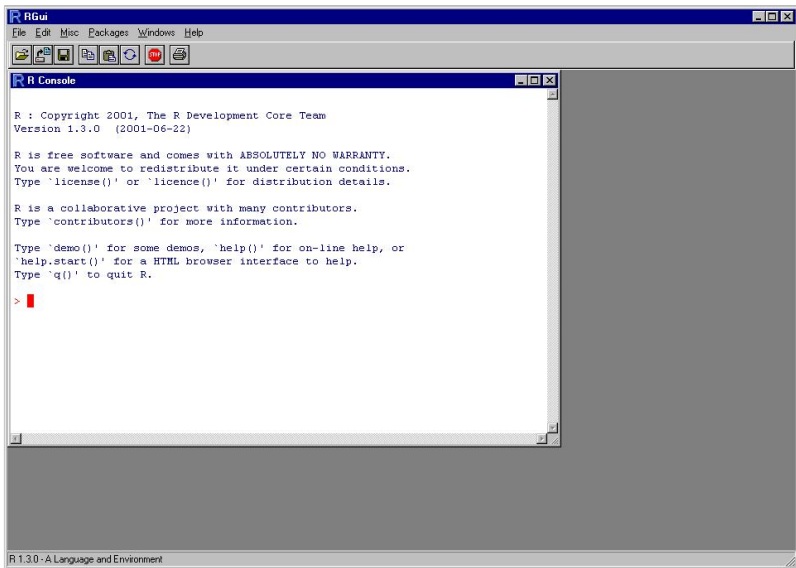
- At this site you can find:
 - The R Software (including source code)
 - Extension Packages
 - Manuals and FAQs
 - Newsletters
 - Mailing List Access

Teaching Laboratories

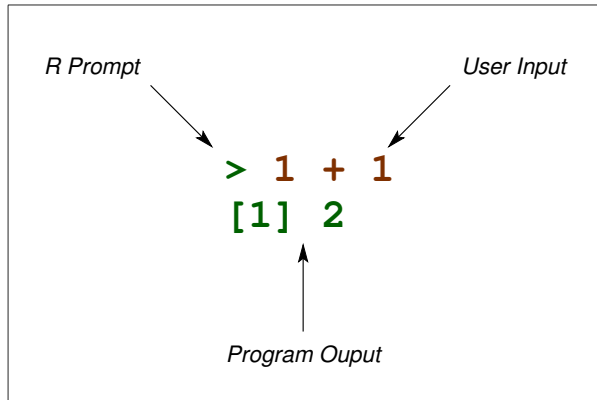
- The Statistics Department teaching laboratories have R (and a variety of other software) installed.
- Because this course is relatively new, and quite small, many demonstrators are not familiar with its contents.
- The lab demonstrators can show you how to get started with R, but they may not be able to help you much with assignments.
- The class has 303.175 reserved on Fridays from Noon to 2pm.

Starting R

- R can be started from the *Start* menu under Windows (look under *Programs*).
- When R is started it opens a *listener* window. This is where you type R commands.
- You should begin by typing a few commands in the listener window and seeing what the response is.



R Command Structure



A Simple Dialog

```
> 1 + 1
```

```
[1] 2
```

```
> 1/2
```

```
[1] 0.5
```

```
> sqrt(9)
```

```
[1] 3
```

```
> pi
```

```
[1] 3.141593
```

```
> cos(pi)
```

```
[1] -1
```

R Commands

- R commands look like mathematical expressions.
- You have to type the “Enter” key after each command.
- If a command executes successfully it may or may not print some results.
- If a command fails an error message is printed.
- Error messages can be obscure. e.g.

```
Error: syntax error
```

Assignment

- Values computed by R are stored for later use by assigning them a name.
- Assignment is indicated by an equals sign.

`x = 42`

- This means that the value 42 is stored with the name “x”.

Legal Names

- After assignment, the value associated with a name can be recalled by just typing that name.
- Legal names consist of strings of characters from:
 - A – Z (upper-case letters)
 - a – z (lower-case letters)
 - 0 – 9 (digits)
 - .
 - (full stops)
- Names cannot start with any string of letters which looks like a number.

Assignment & Variables

```
> x = 100
```

```
> x
```

```
[1] 100
```

```
> x + 17
```

```
[1] 117
```

```
> y = x + 1
```

```
> a.big.name = 123
```

Case Sensitivity

```
> x = 100
```

```
> X = 200
```

```
> x
```

```
[1] 100
```

```
> X
```

```
[1] 200
```

Checking Name Use

- At any time, you can check which names are in use by using the “objects” command. This lists any names currently in use.
- You can restrict to just printing those names which contain a particular pattern, e.g.

```
> objects()
```

```
> objects(pat = "xxx")
```


Using the “objects” Function

```
> objects()
```

```
[1] "a.big.name" "x"           "X"
```

```
[4] "y"
```

```
> objects(pat = "big")
```

```
[1] "a.big.name"
```

```
> objects(pat = "x|y")
```

```
[1] "x" "y"
```

Removing Objects

```
> rm(x)
```

```
> rm(x, y)
```

```
> rm(list = objects(pat = "x|y"))
```

```
> rm(list = objects())
```

Vectors

- R is designed to work on collection of values called vectors.
- A simple way to create a vector is by using the `c()` function.
- The command:

```
> x = c(1, 2, 3, 4)
```

creates a vector containing the four values, 1, 2, 3, 4, and stores it with the name “x”.

Vector Examples

```
> x = c(1, 2, 3, 4)
```

```
> x
```

```
[1] 1 2 3 4
```

```
> length(x)
```

```
[1] 4
```

```
> y = c(x, x)
```

```
> length(y)
```

```
[1] 8
```

```
> y
```

```
[1] 1 2 3 4 1 2 3 4
```

Vector Examples

```
> x + 10
```

```
[1] 11 12 13 14
```

```
> x/10
```

```
[1] 0.1 0.2 0.3 0.4
```

```
> sqrt(x)
```

```
[1] 1.000000 1.414214 1.732051 2.000000
```

```
> min(x)
```

```
[1] 1
```

```
> max(x)
```

```
[1] 4
```

```
> range(x)
```

```
[1] 1 4
```

Vector Examples

```
> mean(x)
```

```
[1] 2.5
```

```
> median(x)
```

```
[1] 2.5
```

```
> sd(x)
```

```
[1] 1.290994
```

Regular Patterns

- R has facilities for generating patterned vectors.
- The main functions for doing this are “`seq()`” (and its shorthand operator “`:`”) and “`rep()`”.
- Using these functions together makes it possible to generate some quite general sequences.

Generating Sequences

```
> seq(0, 10)
```

```
[1] 0 1 2 3 4 5 6 7 8 9 10
```

```
> seq(0, 10, length = 6)
```

```
[1] 0 2 4 6 8 10
```

```
> seq(0, 10, by = 2.5)
```

```
[1] 0.0 2.5 5.0 7.5 10.0
```

```
> 0:10
```

```
[1] 0 1 2 3 4 5 6 7 8 9 10
```

```
> 10:0
```

```
[1] 10 9 8 7 6 5 4 3 2 1 0
```


Repetition

```
> x = 1:4
```

```
> rep(x, 3)
```

```
[1] 1 2 3 4 1 2 3 4 1 2 3 4
```

```
> rep(x, c(2, 3, 3, 2))
```

```
[1] 1 1 2 2 2 3 3 3 4 4
```

```
> rep(x, 3)
```

```
[1] 1 2 3 4 1 2 3 4 1 2 3 4
```

```
> rep(x, rep(3, length(x)))
```

```
[1] 1 1 1 2 2 2 3 3 3 4 4 4
```

Combining Vectors

- It is possible to combine vectors using arithmetic operations like $+$ or $-$.
- When the vectors have the same length, corresponding elements are combined.
- When the vectors have different lengths, the shorter vector is first “recycled” to the length of the longer one.

Combining Vectors

```
> x = c(1, 2, 3, 4)
```

```
> y = c(10, 20)
```

```
> x
```

```
[1] 1 2 3 4
```

```
> y
```

```
[1] 10 20
```

```
> rep(y, 2)
```

```
[1] 10 20 10 20
```

```
> x + y
```

```
[1] 11 22 13 24
```

Combining Vectors

```
> 2 * (1:10)
```

```
[1] 2 4 6 8 10 12 14 16 18 20
```

```
> round(1/3, 1:6)
```

```
[1] 0.300000 0.330000 0.333000 0.333300 0.333330
```

```
[6] 0.333333
```

```
> 1:10 + 10:1
```

```
[1] 11 11 11 11 11 11 11 11 11 11
```

Comments

- The character # provides a “comment facility” for R.
- Any input on a line which follows # is ignored.
- This can be used to document your code.
- There is no multiple line comment symbol.

Miscellaneous

```
> prod(1:10)
```

```
[1] 3628800
```

```
> prod(1:10)/prod(1:3, 1:7)
```

```
[1] 120
```

```
> choose(10, 3)
```

```
[1] 120
```

```
> pnorm(1.96) - pnorm(-1.96)
```

```
[1] 0.9500042
```

```
> sum(1:100)
```

```
[1] 5050
```