Introduction to computational programming
Using *R*

David M. Rosenberg
University of Chicago
Committee on Neurobiology

September 14, 2009

Overview

This exercise is designed to serve as a practical introduction to the computational tools that will be used throughout this course. It assumes no previous knowledge of numerical analysis nor experience in computer programming.

In order to help distinguish between *code*, example output, computer commands and textual information, the following conventions will be used (both here and in later computational exercises).

R input

Commands to be entered into the *R* interpreter will be presented in *syntax-highlighted* typewriter font, with the ">" character marking the beginning of each line. Here is an example:

```
> 3 + 5
> help.start()
> load('myData.RData')
```

R output

Output from the *R* interpreter when shown, will be displayed directly after the corresponding input lines using the same font but in a different color and without the leading ">".

```
> 3 + 5
```

```
[1] 8
```

```
> randomData <- rnorm(n=100)
> summary(randomData)
```

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-2.4140  -0.5751   0.2036   0.1135   0.7628   1.8880
```

Computer commands / keyboard keys

Following standard conventions, keyboard commands/shortcuts will be printed inline with the text into black typewriter font. Combinations of keys which must be pressed simultaneously are separated by hyphens. "Modifier keys" (which vary in name from keyboard to keyboard) are denoted as follows

- *Control:* Typically the "control" key abbreviated as `C-`
- *Meta:* Usually the "alt" on standard keyboards and the "command" on apple keyboards, abbreviated as `M-`
- *Enter:* Variously termed "enter", "return", "carriage return", "linefeed", and "newline", abbreviated as `[CR]`
- *Directional arrows:* the arrow keys are represented by `[LEFT]`, `[RIGHT]`, `[UP]`, and `[DOWN]` respectively.
- *Other keys:* Other keys are represented similarly, such as `[Esc]`, `[F1]` and `[TAB]`.

For example `C-c` means to simulaneously press the "Control key" and the letter "c". `C-x C-c` means to first simultaneously press the "Control" key and the letter "x", then to simultaneously press the "Control" key and the letter "c", and `[Esc] : q !` means to sequentially press "escape", the "colon" (requires `[shift]`), "q" and the exclamation point (requires `[shift]`).

Make sure to pay special to similar looking characters such as

- Single- ('), double- (") and "back-" (`) quotes
- Parentheses (( )), brackets ([ ]) and braces ({ })

To navigate graphical menus ...

1 Getting Started

While not strictly necessary, many students find it helpful to have access to *R* and associated tools on their own computers. Fortunately, *R* is *free software*[1], and available for most computing platforms.

[1] By calling *R free software*, we are saying both that:

1. You don't have to pay to use *R* (free as in beer)
2. You are free to examine and improve *R* as you like (free as in speech)

.

1.1 GNU *R*

The R homepage `http://r-project.org` provides compiled binaries for Windows, OS X, and linux platforms as well as the source distributions (for other platforms). The following are platform specific installation instructions for the most common scenarios.

1.1.1 Mac OSX

The Mac OSX binary distribution of *R* can be downloaded from `http://streaming.stat.iastate.edu/CRAN/bin/macosx/` as a `.dmg` file. After downloading the image, simply open the `.dmg` file and drag the `R.app` icon into your `Applications` folder.

Once you have done this, starting *R* is as easy as double-clicking the `R.app` icon in your `Applications` folder. Alternatively, you may run *R* in a console window by opening `Terminal.app` (located in the `Utilities` subfolder of `Applications`) and typing `R`[2].

Running `R.app` provides you with some additional GUI functionality, provided through the menu interface, such as a *R* source editor (`File - New Document`), a package installer (`Packages & Data - Package Installer`) and easy access to package guides (`Help - Vignettes`).

TODO: OSX platform specific notes, gfortran, source vs. binary packages.

1.1.2 Windows

Installing *R* under windows is accomplished by downloading the windows binary installer from `http://streaming.stat.iastate.edu/CRAN/bin/windows/base/`, opening the installer, and following the on-screen directions. Upon completion of the installer (and possibly rebooting), you should have an icon labelled `R 2.9.2` on your desktop (and possibly in the `Start` menu as well).

To start a new *R* session, simply double-click on the `R 2.9.2` icon.

1.1.3 Linux

`http://streaming.stat.iastate.edu/CRAN/bin/linux/ubuntu/`

Installing *R* on a linux system can generally be performed using your distribution-specific package manager (`rpm/yum` for RedHat-type distributions, `apt` for Debian based distributions such as Ubuntu).

TODO: Dependency issues

1.1.4 Other options

Should none of the above options prove successful for you, alternative methods of running *R* ... TODO: other methods

[2] MORE SPECIFIC DIRECTIONS HERE

- Java i.e. Biocep
- remote (ssh)

1.2 Text Editor

1.2.1 Cross-platform

- vi(m) `ftp://ftp.vim.org/pub/vim/unix/vim-7.2.tar.bz2`
- jedit `http://prdownloads.sourceforge.net/jedit/jedit42install.jar`
- emacs
- ESS

1.2.2 Mac OSX

- TextMate `http://macromates.com/`
- MacVim (vi) `http://code.google.com/p/macvim/`
- Aquamacs (emacs) `http://aquamacs.org/`

1.2.3 Windows

- gvim `ftp://ftp.vim.org/pub/vim/pc/gvim72.exe`
- emaics `http://ftp.gnu.org/pub/gnu/emacs/windows/emacs-23.1-bin-i386.zip`
- e-texteditor (TextMate clone) `http://www.e-texteditor.com/`
- notepad++ `http://notepad-plus.sourceforge.net/uk/site.htm`

2 Your first *R* session

2.1 Interpreter

2.1.1 Example 1

```
> x <- rnorm(50, mean=4)
> x

 [1] 5.992187 3.167802 5.059501 4.047898 5.461917 4.922904 5.408302 3.180626
 [9] 4.927753 3.361892 4.485855 4.659267 4.212696 3.369810 4.581721 2.946754
[17] 4.176146 4.403111 4.881706 4.063978 3.291830 4.704465 4.665736 4.663547
[25] 2.641580 4.063530 5.676971 5.754260 3.694409 1.951678 4.517638 2.474433
```

```
[33] 5.086691 2.313378 4.868268 2.258947 3.500221 3.056641 5.184596 5.563314
[41] 5.705071 4.639980 3.662013 2.503130 3.456786 4.180148 5.086807 3.265397
[49] 2.850065 5.945762
```

```
> mean(x)
```

```
[1] 4.170782
```

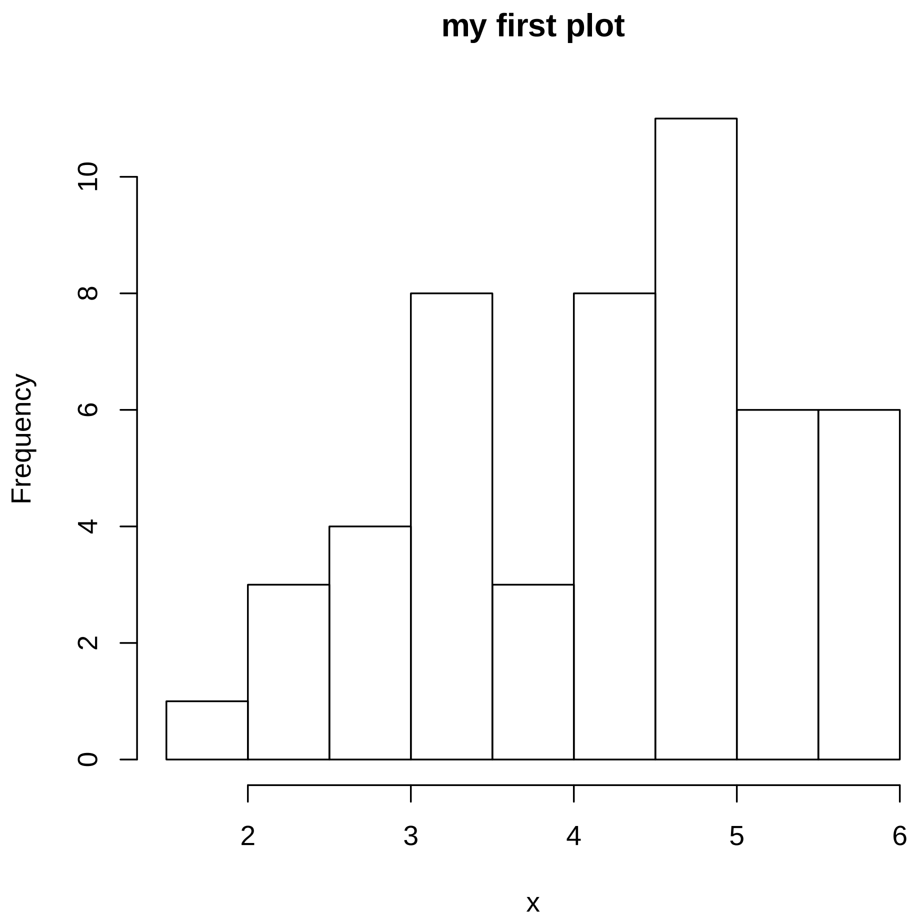
```
> range(x)
```

```
[1] 1.951678 5.992187
```

```
> hist(x)
```

```
> ?hist
```

```
> hist(x, main='my first plot')
```



2.1.2 Tab completion

- function names
- arguments
- file names

2.1.3 History

- up, down arrow
- command 'history'

- file ‘.Rhistory’

2.1.4 Prompt

- Standard
- Continuation
- Other

2.1.5 Getting help

2.1.6 Help browser

- ‘?’ command
- `help.search(`*command*`)`
- `help.start()`

2.1.7 Examples

- `example(`*function*`)`
- `data`

2.2 Session

2.2.1 Saving

- `save.image`
- `save`
- `load`
- `history`

2.2.2 Quitting

- `q()`
- saving/restoring session
- dumping

2.2.3 Aborting

- C-c
- Esc
- kill

3 Exploring *R*

3.1 Example 2: Calculator

```
> # Arithmetic
> 3 / 5

[1] 0.6

> 301 + 50000003

[1] 50000304

> 0.0005 * 0.0001

[1] 5e-08

> -0.0001 ** 9

[1] -1e-36

> -0.0001 ^ 9

[1] -1e-36

> ## exponentiation can be represented with either ** or ^
> 3 + 5 * 2

[1] 13

> (3 + 5) * 2

[1] 16

> ## special operations are called by name
> sin(3)

[1] 0.14112

> sqrt(5)
```

```
[1] 2.236068

> ## complex numbers are supported when written as x + yi
> -1 + 0i

[1] -1+0i

> sqrt(-1 + 0i)

[1] 0+1i

> ## constants can be called by name or expression (varies)
> pi

[1] 3.141593

> exp(1)

[1] 2.718282
```

3.2 Example 3: Variables

- letters, numbers, underscores, '.'
- convention
- reserved

if	TRUE
else	FALSE
repeat	NULL
while	Inf
function	NaN
for	NA
in	NA_integer_
next	NA_real_
break	NA_complex_

```
> x <- 1:20
> y <- x + (x/4 - 2)^3 + rnorm(20, sd=3)
> names(y) <- paste("O",x,sep=".")
> ww <- rep(1,20); ww[13] <- 0
> summary(lmxy <- lm(y ~ x + I(x^2)+I(x^3) + I((x-10)^2), weights = ww), cor = TRUE)

Call:
lm(formula = y ~ x + I(x^2) + I(x^3) + I((x - 10)^2), weights = ww)

Residuals:
    Min      1Q  Median      3Q     Max
-5.8699 -0.7917 -0.1534  1.3804  3.6724
```

```
Coefficients: (1 not defined because of singularities)
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   -6.186503   2.877205  -2.150  0.04826 *
x              3.294472   1.163444   2.832  0.01263 *
I(x^2)        -0.287347   0.128323  -2.239  0.04072 *
I(x^3)         0.012291   0.004052   3.033  0.00838 **
I((x - 10)^2)        NA         NA      NA       NA
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Residual standard error: 2.63 on 15 degrees of freedom
Multiple R-squared: 0.9672,     Adjusted R-squared: 0.9606
F-statistic: 147.4 on 3 and 15 DF,  p-value: 2.371e-11

Correlation of Coefficients:
       (Intercept) x     I(x^2)
x      -0.90
I(x^2)  0.80       -0.97
I(x^3) -0.73        0.93 -0.99

> variable.names(lmxy)

[1] "(Intercept)" "x"           "I(x^2)"      "I(x^3)"

> variable.names(lmxy, full= TRUE)# includes the last

[1] "(Intercept)"   "x"             "I(x^2)"        "I(x^3)"
[5] "I((x - 10)^2)"

> case.names(lmxy)

 [1] "O.1"  "O.2"  "O.3"  "O.4"  "O.5"  "O.6"  "O.7"  "O.8"  "O.9"  "O.10"
[11] "O.11" "O.12" "O.14" "O.15" "O.16" "O.17" "O.18" "O.19" "O.20"

> case.names(lmxy, full = TRUE)# includes the 0-weight case

 [1] "O.1"  "O.2"  "O.3"  "O.4"  "O.5"  "O.6"  "O.7"  "O.8"  "O.9"  "O.10"
[11] "O.11" "O.12" "O.13" "O.14" "O.15" "O.16" "O.17" "O.18" "O.19" "O.20"
```

3.2.1 Types

type	is a	description	notes
integer	numeric	whole number	
double	numeric	floating point number	
numeric		any number	base type
logical		TRUE or FALSE	base type
complex		complex number	
raw		unparsed input string	
character		letters and other characters	
list		collection of other objects	
expression		parsed but unevaluated input	
name		character string referencing an object	
symbol		character string referencing an object	
function (closure)		a function	
pairlist		deprecated linked-list structure	
promise		reference to an expression whose evaluation is delayed by *lazy evaluation* but which is treated as being a value	

3.2.2 Vectors and Matrices

```
> x <- 200
> half.x <- x/2
> threshold <- 95.0
> age <- c(15, 19, 30)
> age[2]        ## [] for accessing element.

[1] 19

> length(age) ## () for calling function.

[1] 3

> y <- c(10, 20, 40)
> y[2]

[1] 20

> length(y)

[1] 3

> x <- 5
> length(x)

[1] 1
```

```
> y <- c(20, 49, 16, 60, 100)
> min(y)

[1] 16

> range(y)

[1]  16 100

> sqrt(y)

[1]  4.472136  7.000000  4.000000  7.745967 10.000000

> log(y)

[1] 2.995732 3.891820 2.772589 4.094345 4.605170

> x <- seq(from=1, to=9, by=2)
> y <- seq(from=2, by=7, length=3)
> z <- 4:8
> a <- seq.int(5)                ## fast for integers
> b <- c(3, 9, 2)
> d <- c(a, 10, b)
> e <- rep( c(1,2), 3)
> f <- integer(7)

> x <- 1:6
> is.matrix(x)

[1] FALSE

> dim(x) <- c(2,3)
> is.matrix(x)

[1] TRUE

> x

     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

> dim(x)

[1] 2 3

> x[2,2]

[1] 4
```

```
> x[1,]                               ## extracting values.

[1] 1 3 5

> x[1:2, 2:3]

     [,1] [,2]
[1,]    3    5
[2,]    4    6

> x[,2]                               ## not column vector!

[1] 3 4

> x[,2,drop=F]                        ## gotcha!

     [,1]
[1,]    3
[2,]    4

> m <- matrix( floor(runif(6, max=50)), nrow=3) ##ncol=2
> x <- rbind( c(1,4,9), c(2,6,8), c(3,2,1))
> y <- cbind( c(1,2,3), 5, c(4,5,6))   # recycling again
> x <- matrix(1:4, 2,2)
> i <- diag(2) ## 2x2 identity matrix
> x %*% i      ## should be x

     [,1] [,2]
[1,]    1    3
[2,]    2    4

> x  *  i      ## not x!

     [,1] [,2]
[1,]    1    0
[2,]    0    4
```

3.3 Example 4: Functions

3.4 Miscellany

4 Source files

4.1 Overview

4.2 Functions

4.3 Scripts

4.4 Style

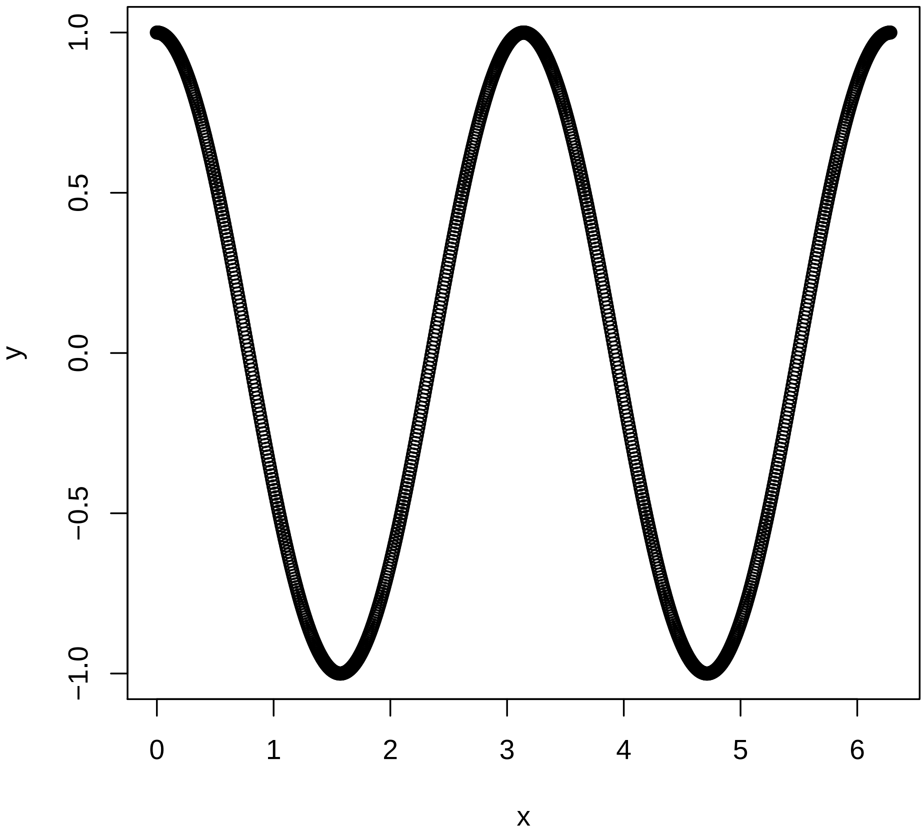
4.4.1 Conventions

4.4.2 Comments

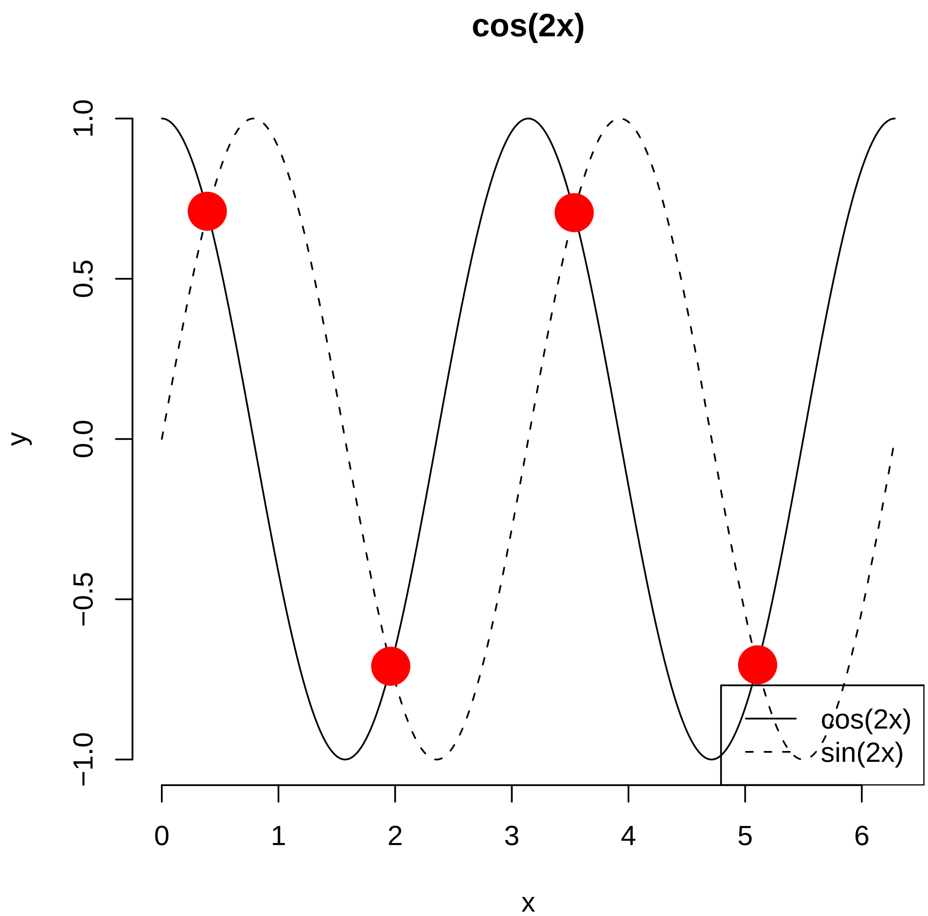
4.4.3 White space

5 Plotting

```
> x <- seq(from=0, to=2*pi, len=1000)
> y <- cos(2*x)
> ## just provide data; sensible labelling
> plot(x,y)
```




```
> ## Expand on previous plot ...
> plot(x,y, main='cos(2x)', type='l', lty=1, bty='n')
> y2 <- sin(2*x)
> lines(x,y2, main='sin(2x)', type='l', lty=2)
> same <- which( abs(y - y2) < 0.01)
> points(x[same], y[same], pch=19, col='red', cex=3)
> legend('bottomright', c("cos(2x)", "sin(2x)"),
+         lty=c(1,2))
```


cos(2x)
1.0
0.5
0.0
−0.5
−1.0
y
0
1
2
3
4
5
6
x
cos(2x)
sin(2x)