NRES 798 — Lab 1

Starting R

Start R. Plain R for now, no RStudio or other aids.

Enter the following commands at the > prompt. Try to predict the results. Experiment with similar entries as needed to understand what is going on.

Basics

```
> 1 + 1
> # This is a comment
> x <- 3 * (6 - 2^2)
> x
> longer.name_2 <- 3.14  # dots in names have no special meaning
> longer.name_2  # tip: type "longer" and press Tab
> camelCaseName <- 25^2
> camelcasename  # R is case-sensitive
> camelCaseName - x
> -4.2e-3
> greet <- 'hello!'
> greet
> 3 > 5
> # Types: numeric, character, logical, complex
> !(3 > 5)
> # Other: <, <=, >=, ==, !=
> 1.414 == 1.414
> sqrt(2)  # a function
> 2 / sqrt(2)  # tip: press up-Arrow to get previous entry, edit it, press Enter
> sqrt(2) == 2 / sqrt(2)  # bad idea!
> abs(sqrt(2) - 2 / sqrt(2)) < 1e-9  # good
> log(5)
> log(5, 2)  # optional arguments, defaults
> help(log)  # shorthand: ?log. Exit help with q (quit)
> log(base=2, x=5)  # naming arguments
> add <- function(a, b) {a + b}
> add(2, 3)
> add <- function(a, b=1) {
+ # Add a and b. Default b = 1
+   a + b
}
> add(2.3, 6.2)
> add(2.3)
> add
> ?help
> ?linear
> ??linear
> help.start()
> ?help.start
> ls()  # list
> rm(great, longer.name_2, camelCaseName)  # remove
> demo()
> demo(graphics)  # just watch the pretty pictures
```

Vectors, matrices, lists

```
> a <- c(2.3, pi, -6, 3)  # c for "concatenate"
> a
> (b <- c(-9, 25, third=-42, last=1.1))  # optional names
> # And trick for forcing output
> c(a, b)
> a + b
> add(a, b)
> add(a)
> a + 1
> mean(b)
> max(b)
> which.max(b)
> 1:5
> 1:-5
> pi:9
> seq(2, 10, 0.2)
> # The mystery of the [1] solved!
> rep(1:3, 4)
> rep(1:3, each=4)
> a[3]
> a[-3]
> b[2] <- 20
> b
> b[2:3]
> b[-2:3]
> b[-(2:3)]
> b['third']
> b[c(F, F, T, F)]  # TRUE and FALSE can be abbreviated
> b[b>0]
> plot(a, b)
> a * b
> a %*% b  # dot product. Result is a 1 x 1 matrix
> matrix(1:4, 3, 4)  # or matrix(x=1:4, nrow=3, ncol=4)
> (m <- matrix(1:4, 3, 4, byrow=TRUE))  # default is to fill by columns
> m[2, c(1, 3)]
> m[, 2]
> cbind(a, b)
> rbind(a, b)
> # Arrays generalize vectors and matrices to more dimensions
> # Lists are like arrays, but elements can be anything:
> (lst <- list(one=a, two='junk', last=T))
> lst$two  # accessing named list elements
> # Used by many statistical functions to return results
```

Data frames

```
> # Most important for us, essentially a list of columns
> (mydata <- data.frame(a, b))
> names(mydata) <- c('var.a', 'var.b')  # or use data.frame(var.a=a, ...
> mydata$sex <- c('F', 'M', 'F', 'F')
> mydata
> summary(mydata)  # tip: summ Tab (my Tab)
> mydata$sex <- as.factor(c('F', 'M', 'F', 'F'))  # up-Arrow 3 times, edit
> summary(mydata)  # factor = categorical variable
> dim(mydata)
> str(mydata)
> mydata$var.b  # like a list
> mydata[, 2]  # like a matrix
> mydata[, 'var.b']
> mydata['var.b']
> str(mydata[, 'var.b'])
> str(mydata['var.b'])
> data()  # quit = q
> data(CO2)
> ls()
> summary(CO2)
```

Quitting R

```
> q()  # asks if you want to save the workspace
```

Or use the GUI menu (MS Windows). The workspace is a chunk of memory with all the variables and functions that you have defined. It is saved in the current folder ("working directory", which can be changed with setwd() or the GUI). The workspace is (usually) automatically restored when starting in the same folder.

Reading-in your data

R can read directly from spreadsheets, data bases, other statistical packages, etc. However, it may be better to output to a text file and read that. E,g., for an Excel spreadsheet with variables in columns and cases/measurements in the rows, save as text.

```
> # Read a table with tab-separated or space-separated values:
> mydata <- read.table('C:/path/fileName.dat')  # use / instead of \
> names(mydata) <- c('var1', 'var2', ...  # variable names. Or easier:
```

```
> mydata <- read.table(’C:/path/filename.dat’, header=TRUE)
> # (takes variable names from the first row)
> mydata <- read.csv(’C:/path/filename.csv’)  # same, for comma-separated values
```

Missing data: Specify the code used in the original file, e.g. `999` or `.`, in the optional argument `na.strings`. For instance,
`> mydata <- read.csv(’C:/path/filename.csv’, na.strings=’.’)`
Obviously (?), for blank cells in a spreadsheet you must save as comma-separated values (csv), and use `na.strings=’’`. In R, missing data is indicated by `NA` (do `?is.na`).

On your own. . .

Experiment some more. Follow the sample session in Appendix A of *An Introduction to R* (`help.start()`).

Close R with `q()`, choosing to save the workspace. Open RStudio. Some things to notice and try (see the *Help* for details): auto-closing of parenthesis, brackets, and quotes. Type a function name (or part of it) and press *Tab* to get help on the arguments. Help and graphs go to the panel on the bottom-right. Workspace contents at the top-right, click on items to display and edit. History of past commands in another tab at the top-right, can be re-executed, possibly after editing. Open a text file on the top-left, use it to keep a log of your work; copy and paste between panels. Explore the menus at the top.

Start reading the rest of *An Introduction to R*.