

# Appendix A: A Sample R Session

The purpose of this appendix is to become accustomed to the R and the way it responds to line commands. R can be downloaded from

`http://cran.r-project.org/`

Be sure to download the version of R corresponding to your operating system - linux, MacOS, or windows.

As you progress, you will learn the statistical ideas behind the commands that ask for graphs or computation. Note that R only prints output when it is requested. On a couple of occasions, you will see a plus (+) sign at the beginning of the line. This is supplied by R and you will not type this on your R console.

- Learn how to access help. Type

```
> help.start()
```

to access on-line manuals, references, and other material.

- Find fundamental constants

```
> pi
> exp(1)
> round(exp(1), 4)
```

- The `<-` is used to indicate an assignment. Type

```
> x<-rnorm(50)
> length(x)
> hist(x)
> mean(x)
> sd(x)
> summary(x)
```

- To see what the `sort` command does type

```
> ?sort
```

Next, sort the values in `x`, first in increasing and then decreasing order.

```
> sort(x)
> sort(x, decreasing=TRUE)
```

The first command gives 50 independent standard normal random variables and stores them as a vector `x`. It then gives the number of entries in `x`, creates a histogram, computes the mean and standard deviation, and gives a summary of the data. The last two commands give the values of `x` sorted from bottom to top and then from top to bottom

- To prepare for a scatterplot, enter

```
> (y<-rnorm(x))
```

This gives 50 additional independent standard normal random variables and stores them as a vector  $y$ . When the command is placed in parentheses, R prints out the value of the variable.

- To make a scatterplot of these data, type

```
> plot(x,y)
```

A graphics window will appear automatically.

- To find the correlation between  $x$  and  $y$ .

```
> cor(x,y)
```

- To preform a  $t$ -test, type

```
> t.test(x,y)
> t.test(x,y,alternative="greater")
```

Notice the difference in  $p$ -value.

- To check to see what is in your workspace, type

```
> ls()
```

- To remove a variable  $x$

```
> rm(x)
```

Now type `ls()` again to see that  $x$  has been removed.

- To make a variety of graphs of  $\sin(\theta)$

```
> theta<-seq(0,2*pi,length=100)
> plot(theta,sin(theta))
> par(new=TRUE)
> plot(theta,sin(theta),type="h")
> plot(theta,sin(theta),type="l")
> plot(theta,sin(theta),type="s")
> theta<-seq(0,2*pi,length=10)
> plot(theta,sin(theta),type="l")
> plot(theta,sin(theta),type="b")
```

To see what these commands mean, type

```
> help(plot)
```

- To make some simple arithmetic and repeating sequences, type

```
> 1:25
> seq(1, 25)
> seq(25, 1, -1)
> seq(1, 25, 2)
> seq(1, 25, length=6)
> seq(0, 2, 0.1)
> rep(0, 25)
> rep(1, 25)
```

- Make a vector of integers from 1 to 25

```
> n<-1:25
```

- Randomly shuffle these 25 numbers

```
> sample(n)
```

- Choose 10 without replacement.

```
> sample(n, 10)
```

- Choose 30 with replacement.

```
> samp<-sample(n, 30, replace=TRUE)
> samp
```

- Turn this into a  $3 \times 10$  matrix and view it.

```
> (A<-matrix(samp, ncol=10))
> (B<-matrix(samp, nrow=3))
```

Notice that these give the same matrix. The entries are filled by moving down the columns from left to right.

- Check the dimension.

```
> dim(A)
```

- View it as a spreadsheet.

```
> fix(A)
```

You will need to close the window before entering the next command into R.

- Find the transpose.

```
> t(A)
```

- View the first row.

```
> A[1, ]
```

the second, third and fourth column,

```
> A[, 2:4]
```

all but the second, third and fourth column,

```
> A[, -(2:4)]
```

and the 1,4 entry

```
> A[1, 4]
```

- Turn this into a  $3 \times 10$  matrix.

```
> matrix(samp, ncol=3)
```

- Make a segmented bar plot of these numbers.

```
> data<-matrix(samp, nrow=3)
> barplot(data)
```

- Perform a chi-squared test..

```
> chisq.test(data)
```

- Make a column of weight vectors equal to the square root of n.

```
> w<-sqrt(n)
```

- Simulate some response variables, and display them in a table.

```
> r<- n + rnorm(n)*w
> data.frame(n, r)
```

- Create a regression line, display the results, create a scatterplot, and draw the regression line on the plot in red.

```
> regress.rn<-lm(r~n)
> summary(regress.rn)
> plot(n, r)
> abline(regress.rn, col="red")
```

Note that the order of r and n for the regression line is reversed from the order in the plot.

- Plot the residuals and put labels on the axes.

```
> plot(fitted(regress.rn), resid(regress.rn), xlab="Fitted values",
+ ylab="Residuals", main="Residuals vs Fitted")
```

- Simulate 100 tosses of a fair coin and view the results

```
> x<-rbinom(100, 1, 0.5)
> x
```

Next, keep a running total of the number of heads, plot the result with steps (`type = "s"`)

```
> c<-cumsum(x)
> plot(c,type="s")
```

- Roll a fair dice 1000 times, look at a summary, and make a table.

```
> fair<-sample(c(1:6),1000,replace=TRUE)
> summary(fair)
> table(fair)
```

- Roll a biased dice 1000 times, look at a summary, and make a table.

```
> biased<-sample(c(1:6),1000,replace=TRUE,prob=c(1/12,1/12,1/12,1/4,1/4,1/4))
> summary(biased)
> table(biased)
```

- The next data set arise from the famous Michaelson-Morley experiment. To see the data set, type

```
> morley
```

There are five experiments (column Expt) and each has 20 runs (column Run) and Speed is the recorded speed of light minus 290,000 km/sec.

- The data in the first two columns are labels, type

```
> morley$Expt <- factor(morley$Expt)
```

so that the experiment number will be a factor

- Now make a labeled boxplot of the speed in column 3

```
> boxplot(morley[,3]~morley$Expt,main="Speed of Light Data", xlab="Experiment",
+ ylab="Speed")
```

- Perform an analysis of variance to see if the speed are measured speeds are significantly different between experiments. .

```
> anova.mm<-aov(Speed~Expt,data=morley)
> summary(anova.mm)
```

- Draw a cubic.

```
> x<-seq(-2,2,0.01)
> plot(x,x^3-3*x,type="l")
```

- Draw a bell curve.

```
> curve(dnorm(x),-3,3)
```

- Look at the probability mass function for a binomial distribution.

```
> x<-c(0:100)
> prob<-dbinom(x,100,0.5)
> plot(x,prob,type="h")
```

- To plot a parameterized curve, start with a sequence and give the  $x$  and  $y$  values.

```
> angle<-seq(-pi, pi, 0.01)
> x<-sin(3*angle)
> y<-cos(4*angle)
> plot(x, y, type="l")
```

The `type = "l"` (the letter ell, not the number one) command connects the values in the sequence with lines.

- Now we will plot contour lines and a surface. First, we give a sequence of values. This time we specify the number of terms.

```
> x<-seq(-pi, pi, len=150)
> y<-x
```

Then, we define a function for these  $x$  and  $y$  values and draw a contour map. Then, choose the number of levels.

```
> f<-outer(x, y, function(x, y) cos(y) / (1+x^2))
> contour(x, y, f)
> contour(x, y, f, nlevels=20)
```

- For a color coded “heat map”,

```
> image(x, y, f)
```

- To draw a surface plot,

```
> persp(x, y, f, col="orange")
```

and change the viewing angle

```
> persp(x, y, f, col="orange", theta=-30, phi=45)
```