A Brief Introduction to R[1]

This semester most of our computing will be done with the statistical programming language **R**. **R** is an extremely powerful language — in this semester we will not be using all of its features, but will be focusing on tools for analyzing spatial data. In this handout and in assignments, we'll outline several of the tools available in **R**. To learn about other aspects of **R**, see reference books such as Dalgaard (2002) (*Introductory Statistics with R*. Springer, New York), as well as the online help feature in **R** described below.

I encourage you to obtain your own copy of **R**. Versions for Windows, the Macintosh, Linux, and several other operating systems can be downloaded from the Comprehensive R Archive Network. The U.S. mirror, `http://cran.us.r-project.org`, may be the fastest download site. Depending on which platform you are working on, your output might vary slightly. The material that I have copied below is from my laptop computer.

Upon starting, **R** will provide some startup information, and then supply you with the **R** prompt, `>` , as follows:

```
R version 2.15.1 (2012-06-22) -- "Roasted Marshmallows"
Copyright (C) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: i386-pc-mingw32/i386 (32-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

The first thing to know is that if you want to quit **R**, type `q()` at the prompt. At one level, **R** functions like a calculator (similar to **python or Matlab**). Here are some simple examples that you should try. Note the assignment operator `=` which is used to assign a value to a variable name.

```
> 3+6
[1] 9
> 8*4
[1] 32
> 3^4
[1] 81
> x = 9
> y = 12
> x - y
[1] -3
> z = (x+y)/2
```

[1]Adapted from course materials for STAT575 University of Wisconsin–Madison

```
> z
[1] 10.5
```

Of course, we want to be able to do more than just simple arithmetic. Here are some examples of some more sophisticated operations.

```
> x = c(15, 18, 22, 24, 31, 17, 18, 21, 14, 27, 16, 25, 24, 12)
> stem(x)

  The decimal point is 1 digit(s) to the right of the |

  1 | 24
  1 | 56788
  2 | 1244
  2 | 57
  3 | 1
```

In the above example, `x` is a *vector* of data values. We can perform numerous operations on this vector:

```
> mean(x)
[1] 20.28571
> sum(x)
[1] 284
> median(x)
[1] 19.5
> sd(x)
[1] 5.469194
> var(x)
[1] 29.91209
```

We can also manipulate the values of `x` :

```
> x
 [1] 15 18 22 24 31 17 18 21 14 27 16 25 24 12
> x+5
 [1] 20 23 27 29 36 22 23 26 19 32 21 30 29 17
> x*2
 [1] 30 36 44 48 62 34 36 42 28 54 32 50 48 24
```

and we can add and subtract vectors:

```
> w
[1] 12 19 15
> z
[1] 1 4 3
> w+z
[1] 13 23 18
```

Here's a trick that can be of value at times. Note the output from the command `x < 20` :

```
> x
 [1] 15 18 22 24 31 17 18 21 14 27 16 25 24 12
> x < 20
 [1]  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE
[13] FALSE  TRUE
> sum(x < 20)
[1] 7
```

The command `x < 20` determines, for each value in `x`, whether it is less than 20. The `sum` command then indicates the total number of `x` values less than 20.

To get to a particular element in a vector, we use brackets, `[]`, as follows:

```
> x
 [1] 15 18 22 24 31 17 18 21 14 27 16 25 24 12
> x[1]
[1] 15
> x[4]
[1] 24
```

Note that if you type a return before you have finished a command, then R gives a `+` prompt, indicating that it is waiting for further input before executing the command.

```
> x = c(15, 18, 22, 24, 31, 17, 18,
+        21, 14, 27, 16, 25, 24, 12)
```

Here's another example. Here we will generate 50 values from a standard normal distribution, and then create a stem and leaf display, a histogram, and a normal scores plot for these data.

```
> mydat = rnorm(50)
> mydat
 [1] -0.08672556  0.40661701  0.37533153 -0.71380071 -0.81837253  0.35515607
 [7]  2.53024662  0.63466582  1.31550240 -0.47121679 -0.15151223 -0.86226648
[13]  0.02948572  0.93543393 -0.80803132  0.54928505 -1.13450004 -0.15258614
[19]  0.27756430  2.32088894  1.55977274  1.74429986  1.13511691 -0.08055711
[25]  0.04392383  1.03889201  0.60292234  0.45285342 -0.72932885  1.02688882
[31] -1.80438213 -0.61015291  0.83085749  0.84271298 -1.28580882 -0.86395998
[37]  0.91301588  1.98792503  0.19818193  0.92497281 -2.12226657 -0.34069130
[43]  0.29402143 -0.35328211 -0.70236025 -0.93149113  1.63494051  0.96289022
[49]  1.42927791 -0.66853615
> stem(mydat)

  The decimal point is at the |

  -2 | 1
  -1 | 8
  -1 | 31
  -0 | 99988777765
  -0 | 432211
   0 | 00233444
   0 | 556688999
   1 | 000134
   1 | 667
   2 | 03
   2 | 5
```

The function `hist(mydat)` will activate a graphics window, and produce a histogram; `qqnorm(mydat)` will produce a normal scores plot.

To get help in **R**, you can find pdf versions of the manuals for the various packages on the **R** web page. There are also some introductory guides there. Within **R**, type `help.start()` which will start up the on-screen help system within a browser. (I have variable success with this, however.) You can also type `help(hist)`, for example, at the prompt to get help on the `hist` function. Incidentally, the most useful part of the help output for any given command might be the list of examples at the end. Scroll down to the end, if necessary, to see these.

You'll see from the above examples that *all* **R** functions end with parentheses. Sometimes there are arguments in the parentheses, like `rnorm(50)`, and sometimes not, like `q()`. Try typing a function name like `q` without the parentheses — you'll get an outline of how the `q` (or other) function is structured. This is helpful for more advanced users, but we won't be needing this information at this point.

Finally, a general comment on the choice of software packages. There are many other statistical analysis packages available, some free, and some quite expensive. Some are very reliable (SAS, for example), and some are not. **R** is free, well-written, faster, has a more extensive collection of packages available (click on the "Package Sources" link on the **R** web page to see these), and is available on more platforms than other software packages such as Splus. Both Splus and **R** have the additional advantage that they can perform most of the other data analysis and graphics that you need.