This document will describe some of the basic functionality of R. Usually, you interact with R through a command-line interface — you type in a command and R responds. Once you have mastered a few commands, the command-line interface gives you efficient control of an extremely powerful tool for interacting with data. Gaining mastery of a few R commands does take some learning and patience as R is finnicky. You will undoubtedly experience some challenges as you work to learn a new skill that is not wholly intuitive, but it is well worth the effort. Onward!

The following commands give a glimpse of what R can do. You will find it most beneficial, actually, to type these commands into R to see the results yourself.

Calculating with numbers.

```
> 2 + 2
[1] 4
> 12 * 3 - 10/2 + sqrt(16)
[1] 35
> 3^2
[1] 9
> 1:10
 [1]  1  2  3  4  5  6  7  8  9 10
> sum(1:10)
[1] 55
> mean(1:10)
[1] 5.5
> sd(1:10)
[1] 3.027650
```

You can use R like a calculator. The `*` symbol stands for multiplication and the `^` symbol stands for exponentiation. The colon operator `:` creates an array of numbers from the first to the second. R has a number of built-in functions such as `mean` and `sum` that have obvious meaning. The command `sum(1:10)` calculated

$$1 + 2 + 3 + \cdots + 10$$

The symbol `[1]` that precedes the output says that the first row begins with the first number of the output. Long answers are broken across rows, like in this example with similarly useful row labels.

```
> 1:100
  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
 [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
 [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
 [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
 [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
 [91]  91  92  93  94  95  96  97  98  99 100
```

Calculating with arrays. R can do arithmetic operations on arrays. If you multiply an array of numbers by a single number, the multiplication happens separately for each number. You can also add or multiply equal-sized arrays of numbers.

```
> 2 * (1:15)

 [1]  2  4  6  8 10 12 14 16 18 20 22 24 26 28 30

> (1:10) + (10:1)

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

> (1:4)^2

[1]  1  4  9 16
```

Assigning variables. You can use the = sign to create new variables. Typing the name of a variable displays it.

```
> a = 1:10
> a

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

> mean(a)

[1] 5.5
```

(An alternative to the = syntax is to use the key combination <- which was created to look like an arrow. Older documentation may use this instead of the equal sign, but both are valid methods.)

Combining arrays. The c function in R *concatenates* things. (Because c is a reserved function name in R, it is preferable not to use c as the name of a variable lest you or R gets confused. I tend to use cc when I really want to use c.)

```
> a = 1:10
> b = 15:20
> cc = c(b, a, b)
> cc

 [1] 15 16 17 18 19 20  1  2  3  4  5  6  7  8  9 10 15 16 17 18 19 20
```

Using functions. R has many built in functions and you can write your own if you want. To see a function, just type its name. If you want actually *to use* the function, you need to add parentheses at the end, possibly with arguments inbetween.

```
> mean

function (x, ...)
UseMethod("mean")
<environment: namespace:base>

> mean(cc)

[1] 12.04545
```

Here is code to create a function that computes the area of a rectangle.

```
> findArea = function(x, y) {
+     return(x * y)
+ }
> findArea(13, 4)

[1] 52
```

You do not need to type in the `+` prompt. That's how R lets you know that the previous command was incomplete. This can happen when you have a command that begins with a `(` or a `{` and you type a return before the command ends. You can often rectify this by typing the missing `}` or `)`. If you have mistyped and can't get out of the `+` prompt, pressing the `Esc` key usually works to get back to a regular prompt.

Changing your workspace. R keeps all of the variables you keep in memory as it runs. When you end an R session, you may save your workspace. This allows you to have variables you have previously defined available without the need to create them all again from scratch. There will also be times when you will want to read in data sets or read some R code. To do these things more easily, it is often a good idea to change R's workspace.

By default, R will use as its workspace the folder in which the executable program exists. You will most likely want to change the workspace to a new folder where you might keep data from the textbook and your homework. You change the workspace for R under both Windows and Macintosh versions by using the `File` menu and selecting `Change Directory...` with your mouse. My advice is to have a folder where you keep work for this course and to change the worspace to this folder each time you start R.

Quitting R. To quit, you can type `q()` on a command line or you can quit through the `File` menu. R will prompt you if you want to save your workspace.