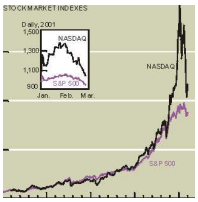

Playing with the stock market

The goal of this project is to begin learning how to do data analysis with R. Along the way we'll have some fun with stock market data. The R software can make many of the calculations in statistics seem easy. Think of it as a calculator for statistics. However, it requires some time to learn.

In Windows, you start R by clicking on the icon for it. This opens up a large window that will contain various subwindows. In particular a command console for typing commands, a graphics window and a help page window.

Interacting with R is done in a **question-and-answer** manner. You ask questions and R answers. You ask these questions by **typing** them in at the prompt which looks like

```
>
```

For example, to see that R is a calculator let's check it out. Type the following and hit the `enter` key.

```
> 2+2
[1] 4
> 5*6
[1] 30
> 3^2
[1] 9
```

Notice in each case R answers correctly. The `[1]` will be clear later.

0.0.1 Problems

Use R to answer the following

1. $52 \cdot 51$, 14^3, $2004/4002$, 6^3

0.1 Storing data

Statistics is about data though which usually means more than one number. R works naturally with more than 1 number at a time. First, we need to learn how to store some numbers.

The price of a share of stock fluctuates on a daily basis. Some stocks more so than most. In January of 2004, The AT&T wireless stock (symbol AWE) for AT&T's cellphone services had been having a big decline. In late January though, word of a possible merger was released changing how investor's view the stock.

Data for Friday's close of AT&T wireless for a few months are:

```
23-Jan-04    10.61 12-Dec-03    7.13
16-Jan-04    9.99   5-Dec-03    7.27
 9-Jan-04    8.15  28-Nov-03    7.50
 2-Jan-04    8.08  21-Nov-03    7.00
26-Dec-03    7.63  14-Nov-03    6.81
19-Dec-03    7.35   7-Nov-03    7.02
```

What can we say about the data?

Before doing anything, let's store the data into the computer for January and December.

There are two ways to store data into R: using `c()` and `scan()`. The function `c()` combines data into a single object. For example, typing the command (and hitting the enter key) produces the output.

```
> c(2,3,5,7,11)
[1]  2  3  5  7 11
```

The numbers were combined and then printed – then they were forgotten! Again, the `[1]` appears. This helps keep track of how many numbers are in the data vector. (We call a variable that stores data a **data vector**.)

We need to store the data so we can reuse it. To do this, we *assign* the data to a *variable* using an equals sign. The following will store the values into the variable called `x`.

```
> x =  c(2,3,5,7,11)
>
```

Notice, only the prompt is returned. `R` is quiet after an assignment. However, it was busy. Now whenever we type `x` we get this dataset.

```
> x
[1]  2  3  5  7 11
```

We go ahead now and enter in the data for AT&T into the variable `awe`.

```
> awe = c(10.61, 9.99, 8.15, 8.08, 7.63, 7.35, 7.13, 7.27)
```

Save time typing Typing can get tedious. Fortunately, there are a few shortcuts to help reuse what you've previously typed. First, you can cut-and-paste which you are likely familiar with, but `R` allows you to copy and paste at the same time, do a right-mouse popup to see. Second, the up and down arrows scroll through your previous commands. So if you like what you typed, hit the up arrow to recall the command, and then edit it. Finally, the HOME key will take you to the beginning of the line, where as the side arrows move you left or right.

0.1.1 Problems

Type in the following data sets.

1. The numbers 1,1,2,3,5,8,13 into the variable `fib`
2. The numbers 7.97, 7.51, 5.94, 5.62 into the variable `pcs`

1 Manipulating data

`R` has many functions that work quite easily on data vectors.

For example, we can make a simple plot of the data using the command `plot()`. To apply a function to a data vector you type the name of the function, open parentheses, the variable name, and close parentheses. It's easier to see than read:

```
> plot(awe)
```

A plot window should open up showing our admittedly boring plot. By default this plots the numbers in the order they are typed in. Seems like our stock is dropping doesn't it?

Well not really, that's because the stock numbers were typed in reverse chronological order. How can we reverse the numbers? `R` has a built in function `rev()` to do so:

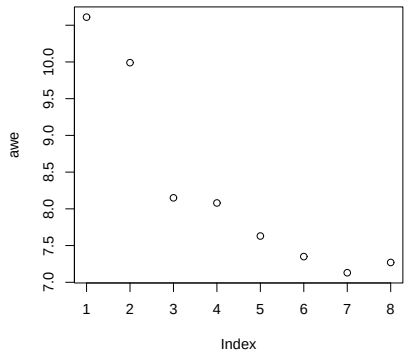

Figure 1: The result of `plot(awe)`

```
> rev(awe)
[1]  7.27  7.13  7.35  7.63  8.08  8.15  9.99 10.61
```

Okay, it works, but we need to store it to do any good. We store it in the same variable

```
> awe = rev(awe)
> plot(awe)                       # not shown
```

If you make the plot now you see an increasing trend. We all should have bought some shares.

1.0.2 Problems

1. Another plotting function is a `barplot()`. Make a barplot of `awe`. Can you interpret the graph?
2. The variable `pcs` from above is stock data for Sprint PCS, only it too is reversed in time. First unreverse, and then plot.

1.1 Other functions

R has quite a few functions to help explore a data set. For a small one like `awe` these things can be easily done by hand, but for larger ones it is much better to use the functions.

For example, R can find the smallest and largest with `min()` and `max()`

```
> min(awe)
[1] 7.13
> max(awe)
[1] 10.61
```

Or both at a time with `range()`

```
> range(awe)
[1]  7.13 10.61
```

The values can be sorted from smallest to largest with `sort()`

```
> sort(awe)
[1]  7.13  7.27  7.35  7.63  8.08  8.15  9.99 10.61
```

In statistics we will be interested in the average value of some data vector. The average is called the **mean** in statistics so the function name is `mean()`

```
> mean(awe)
[1] 8.276
```

Of course, you could find this your self as follows

```
> (7.27+7.13+7.35+7.63+8.08+8.15+9.99+10.61)/8
[1] 8.276
```

But isn't that alot of typing? Besides, why type the data in again if we already have it stored? `R` provides a convenient function for adding the values in a data vector called `sum()`. SO we could do

```
> sum(awe)/8
[1] 8.276
```

That's less typing, but we still cheated. Where did 8 come from? I counted the number of entries in the data vector. If there were 508 I would have been pretty tired after doing this. Instead, I would use the `length()` function which gives the length of a data vector

```
> length(awe)
[1] 8
> sum(awe)/length(awe)
[1] 8.276
```

Still, `mean()` is easiest to type, but it is good to know we can do things conveniently ourselves.

1.1.1 Problems

Okay, now we are going to look at bigger datasets. But rather than type it in, we are going to let the computer do the work for us. However, you need to teach the computer how by typing the following lines exactly as shown (there are 4 capital letters):

```
> where = "http://www.math.csi.cuny.edu/st/R/downloadStockData.R"
> source(url(where))
```

This creates a new function that will get a years worth of data on a stock. For example, the following will get data for AT&T wireless for one year and store in into `x`

```
> x = downloadStockData("AWE")
> plot(x)
```

1. AT&T wireless's stock jumped up recently when talk of it being aquired by Cingular appeared. Can you tell when from the graph?
2. Sprint PCS has stock market symbol PCS. Download a year's worth of their data, and find the minimum stock price, the maximum stock price and average stock price.
3. Repeat with Starbucks (symbol SBUX). Buy or sell?
4. Make a plot of Microsoft (symbol MSFT). What happened at the beginning?