CSC 121 — Lab Exercise 2

These are non-credit excercises, which you do not hand in.
You may work own your own or together with another student, as you please.

There are three parts to this lab. First, I've given you some things to explore in R, and some simple exercises. Once you've done these, you can explore some data I've put on the course web page about deaths each day in Toronto in 1993–1995 and the average temperature each of those days. You can look at whether number of deaths seems to be related to the temperature, and whether it's also related to the season, or perhaps only to the season (not temperature).

You may find it useful to look at this lab handout and the lecture slides in windows on your screen next to the RStudio window, or you can use both a laptop and a CS teaching computer to get more screen space.

Playing around with R.

Try typing things to the R console (the lower-left panel in RStudio) to find out the answers to these questions:

1) What happens when you try to divide a non-zero number by zero? How about when you try to divide zero by zero?

2) What happens when you try to add two character strings, such as "Fred" and "Mary"?

3) What happens when you try to use the `paste` function to combine two numbers (rather than two character strings)?

4) What happens when you apply the `substring` function to a number rather than a character string?

5) What happens when you ask `substring` to find a substring that ends after the last character of the string? How about when you as it to fnd a substring that both begins and ends after the last character of the string?

6) What does `substring` do when the told to extract a substring that ends before it begins?

7) What does `substring` do if you give it only two arguments — the string and the beginning character position, without saying where the substring it extracts should end?

8) Does `help(substring)` answer some of these questions?

Some simple exercises.

For this part, you will write several small R functions that manipulate vectors. You should put the definitions of these functions in an R script file called `lab2defs.R`, which you can create and edit in the upper left pane of the RStudio window. You can "source" this script to define the functions, and then test them by typing calls of these functions with various arguments. You should start with just the first function below, test it and modify it until it is working, then go on to writing the next function, and so on.

You should write all these functions using only the facilities of R that we have covered in lectures, or that are suggested below, even though some of these problems could be solved more easily using facilities of R that we haven't covered yet.

1) Write a function `sum_first_last` that takes a numeric vector as its only argument and computes the sum of the first and last elements of this vector. For example, the value returned by `sum_first_last(c(3,1,2,1,8))` should be `11`.

   For this exercise, you'll need to use the `length` function, which gives how many elements are in a vector.

2) Write a function `midvalue`, which takes a numeric vector as its only argument. If the length of the vector is odd, it returns the middle element of this vector. If the length of the vector is even, it returns the average of the two middle elements. For example, `midvalue(c(7,1,3,7,2))` should be `3`, and `midvalue(c(7,3,7,2))` should be `5`. You can assume that the vector has at least one element.

   For this exercise, you'll need to use the `%%` operator, which finds the remainder of a division. For example, `8%%3` is `2`, and `8%%2` is `0`.

3) Write a function `swap_first_last` that takes a vector (of either numbers or strings) as its only argument, and returns a modified version of this vector in which the first and last elements are swapped. For example:

```
> x <- c("fred","mary","george","bert","helen")
> swap_first_last(x)
[1] "helen"  "mary"   "george" "bert"   "fred"
```

   (Don't worry about what to do if the vector has fewer than two elements.)

The temperature data.

You can read in two sets of numbers (deaths and temperature) for days in Toronto from 1993 to 1995 in the way we've done before:

```
deaths <- scan("http://www.cs.utoronto.ca/~radford/csc121/deaths.txt")
temp <- scan("http://www.cs.utoronto.ca/~radford/csc121/temp.txt")
```

You'll also need to create a vector with numbers indicating the day of the year (from 1 to 365, since none of these years are leap-years). You can do this using the ":" operator, which produces a vector of consecutive integer — eg, 1:10 gives numbers 1 through 10 — and the "rep" operator that repeats a vector some number of times — eg, rep(c(3,4),2) gives the same vector as c(3,4,3,4).

You can start by reading the data and looking at it (perhaps plotting it) just to be sure you know what it's like. But you should then try to write an R Script that will produce four plots:

- Deaths for each day, with the horizontal axis being just numbers from 1 up. You can get a plot like this with just `plot(deaths)`.
- Deaths on the vertical axis versus temperature on the horizontal axis. You can get a plot like this with `plot(temp,deaths)`.
- Temperature versus day of year.
- Deaths versus day of year.

To make all four plots visible at once, you can use the command par(mfrow=c(2,2)). Each plot will then go into the next slot in a two-by-two array of plots. You might want to use the "pch" option (eg, `pch="."` or `pch=20`) to make the points smaller — for example, `plot(deaths,pch=20)`.

Once you've written a script that does this, you can try to modify it so that the points in the plots other than the first show what year they are for by colour — red for 1993, green for 1994, and blue for 1995. To do this, you can use the "col" option to plot, which can be a vector, giving the colour of each point plotted.

You can then write a second script, in which you try to predict the number of deaths based on the day of the year (ie, the season). You should try a prediction equation of the following form:

```
prediction <- a + b*cos((day_of_year-s)*2*pi/365)
```

You'll need to set the variables a, b, and s before doing this. Play around to see what values seem to be best. One hint: It's probably a good idea for `mean(prediction)` to be about the same as `mean(deaths)`.

You can test your predictions by plotting deaths versus day of the year, and then plotting your predictions on top of this plot. You can add points to a plot using the `points` function.

You can put the commands for setting a, b, and s, making the predictions, and plotting the data and the prediction in an R Script, which you can easily run again and again with different values for a, b, and s.

Once you've found what seem to be good values for a, b, and s, you can try seeing whether the error in this prediction is related to temperature. Since the prediciton is based only on the season, not the actual temperature, this might reveal whether temperature itself is related to deaths, or whether it's just that deaths are related to the season, and temperature is related to the season.

(Note: To draw any firm conclusions about the real relationships among death, temperature, and season, you'd want to look at more than three years of data, in more places than Toronto. This is just an exercise!)