CSC 120 (R Section)— Lab Exercise 10

This is a non-credit exercise, which you do not hand in.
You may work on your own or together with another student, as you please.

In this lab, you will learn how to run an R script using `knitr::spin`, which produces an HTML file containing the script commands, their output, and any plots they produce. This a convenient way of getting all output and plots in one file, and is how you will produce output for the course assignments. You will also practice in using `sapply`, in how to define functions with defaults, call functions with arguments identified by name, and add things to an existing plot.

The `knitr` package is installed on the CDF computers. If you are using R on your own computer, you will need to install it by giving the following command to R:

```
install.packages("knitr")
```

This may ask you to choose a "mirror" of the R package repository. The rstudio mirror or the local mirror in CANADA (ON) are reasonable choices.

You can now use `knitr::spin` to run an R script, which you could also run using `source`, or by pushing the appropriate button in RStudio.

You can try out `knitr::spin` with a demo script on the course web page. Use your browser to download a local copy of the following file, to some convenient directory (folder):

```
http://www.cs.utoronto.ca/~radford/csc120/demo-knitr.r
```

Then enter the following command (assuming that R's current directory is the one you downloaded to):

```
knitr::spin("demo-knitr.r")
```

This will create a file called `demo-knitr.html` in the same directory. If you view this file in your browser, you should see the commands in the script and the output of these commands, including all the plots produced, at the point where they were finished.

This demo script starts with the following lines:

```
#+ setup,include=FALSE
source("http://www.cs.utoronto.ca/~radford/csc120/options.r")
#+
```

This sets various options the way I like them (and the way you should set them for the third assignment). You might want to copy the `options.r` file to your own computer, and use the local copy rather than the URL above, so that you can work even if you don't have an internet connection.

After these lines, there are some R comment lines that begin with `#'`, as follows:

```
#' This is a demonstration script that is used for showing how the
#' knitr::spin function can be used to run an R script file, producing
#' an HTML file that contains
#'
#'   * The R commands from the script
#'   * The output of these commands
#'   * All the plots produced by these commands
#'
#' This HTML file can then be viewed with a browser.  This procedure
#' has two advantages:
#'
#'   1. All the results get put in a single file.
#'   2. You can put in "markdown" comments, like these, that
#'      become nicely formatted text, with bullet points like above,
#'      or numbered points like here.
```

As it says, these get converted to a nicely formatted block of text in the output HTML file. Regular comments that start with `#` and a space are displayed in normal fashion (just as they are written).

You can compare the result of `knitr::spin` with the result using `source` with the `echo=TRUE` option. How are they similar? How are they different?

You can try modifying the demo script to also read a data frame from the course web page, as follows:

```
df <- read.table("http://www.cs.utoronto.ca/~radford/csc120/data7",head=TRUE)
```

You can then print this data frame, and use `summary(df)` to see some statistics on the variables.

You can also try using `sapply` to find the means and standard deviations of the variables. However, you'll get a warning when you try to find the mean and standard deviation of the non-numeric `name` variable. How could you avoid these warnings?

You can now try using `knitr::spin` for a script of your own. This script should use `source` to read in another script file containing a definition of a `plot_square` function, and then use `plot_square` to plot squares in various places of your choosing.

The definition of your `plot_square` function should start as follows:

```
plot_square <- function (x, y, size=1, col="black", lwd=1)
```

The `x` and `y` arguments specify the centre of the square. The `size` argument specifies the length of a side. The `col` and `lwd` arguments specify the colour and width of the lines.

Notice that the arguments other than `x` and `y` have default values. The `plot_square` function should use `lines` to add lines that draw the specified square to the current plot (*not* make a new plot).

Your script that uses `plot_square` should try calling it with arguments that are specified by name, or by position, sometimes using the default values, and sometimes specifying different values. You will need to create a plot before calling `plot_square`, using the `plot` function. This plot could be empty to start, however, as could be created with a call such as `plot (c(), xlim=c(0,2), ylim=c(1,5))`.

Once you've got your script to draw a bunch of squares, you can try putting text in the squares, using the `text` function that is described in the week 4 lecture slides.

You should be able to run your script using either `source` or `knitr::spin`, though how the results are presented will be different.