**STAT321-19B – Assignment 1** **Due: Tuesday 6th August, 5.00pm (40 marks total)**

Please prepare answers to these questions as a word-processed document in pdf format (for example, using Word – and either Save as Adobe PDF, or 'print' the document using Microsoft Print to PDF – or better yet using LaTeX). Please submit the document to the appropriate link on Moodle. Please also upload to Moodle your **commented** `R` code as a separate text file called `SID_familyname_321_1.R` where SID is your student ID and familyname is your family name, or surname.

**Question 1** **(12 marks)**

A random walk is defined as a path described by a sequence of random movements or steps over a mathematical space. Random walks are important processes in many statistical applications.

Consider a simple one-dimensional random walk on the real number line, where we start at 0, then take a number of steps $n$ which could be in the negative or positive direction, where each step is of a random length between 0 and some maximum value $k$ (that is, the next position is equally likely to be any value between $-k$ and $k$ away from the current position).

(a) Write a function that generates such a random walk. This function will need two arguments – the number of steps to take, $n$, and the maximum size of a step, $k$. You can call these arguments anything you like (you do not have to call them `n` and `k`, in fact it is good practice to use *meaningful variable names* such as `numSteps` and `stepRange`). Include comments in your `R` code that describe what you are doing at each step. Your function should return a vector of $n+1$ values showing the point on the real number line you have reached after each step (the first value in this vector will be the starting point, 0). Copy your R code to your report. [4 marks]

*Hint*: You should be able to perform all the operations you need to inside the function with loops, like the `for` loop. However, some of the operations could take a few seconds. If you would like a challenge, you might be able to improve the speed of the simulation by using the ideas of vectorisation in `R` or functions like `sapply()` instead of loops. This is purely optional though – so long as your code works, I don't mind how long it takes.

R Functions You Might Find Useful

- `runif()` – random draws from a uniform distribution. This takes three arguments: the number of draws you want to take (which can be 1, if you want), and the lower and upper limit of the range of values you want to draw from.
- `numeric(n)` – creates a vector of zeroes of length `n`.
- `matrix(x,n,m)` – creates a matrix with `n` rows and `m` columns, with each matrix entry containing the number `x`.

(b) Use your function to create a path of 100 steps, where the maximum step-size is 1, then draw a line-plot of this data. Include the code and the plot in your report. [2 marks]

It can be shown that after a sufficiently large number of steps (typically more than 30 or so), the distribution of signed-distances our random walk ends up away from its starting point is approximately normally distributed with a mean of 0 and a standard deviation equal to the maximum step-size multiplied by the square-root of the number of steps divided by 3, i.e. $k \times \sqrt{n/3}$.

(c) Consider a random walk of 50 steps where the maximum step-size is 5. Save the final position from 1000 of such walks in a vector. Copy the code you use to do this to your report (you do not need to copy the resulting vector to your report!). [3 marks]

*Hint*: Again, you could use a loop for this, or you could use a function like `sapply()`. If you use a loop, you will need to create the vector you are going to store the results in before you start the loop.

(d) Plot the output after using the `density()` function on your vector of 1000 end-points, then superimpose a red line showing the density of a normal distribution with a mean of zero and a standard deviation equal to $k \times \sqrt{n/3}$. Copy the commands you use and the graph you produce to your report. [3 marks]

R Functions You Might Find Useful

- `density()` – like a smoothed histogram. Note that this function does not produce a plot itself, but if you plot the result of using this function, you get a nice plot of the empirical probability density function of your data.
- `lines()` – this function superimposes a line on top of an existing plot. You need to tell it the x and y values of the points you want to draw lines between. You can also change the colour of these lines with the `col` argument.
- `dnorm()` – the density of the normal distribution at a specified x-value. Like most `R` functions, you can specify a vector of x-values to compute simultaneously, rather than 1 x-value at a a time. You also have to provide the mean and the standard deviation of the normal distribution you are working with.
- `seq()` – create a sequence of values between a lower limit and an upper limit. After specifying these limits, you can either use the `by` argument to specify the distance between successive values, or the `length.out` argument to specify how many values you want in your sequence.

**Question 2** **(28 marks)**

The data in `blood.csv` (available from Moodle) are blood measurements from 50 subjects taken by the US Department of Health and Human Services. Variables recorded are

| | |
|---|---|
| `Sex` | The sex of the patient (note: when `R` treats the levels as numeric, `F=1` and `M=2`) |
| `Age` | The age of the patient, in years |
| `RedBCC` | Red Blood Cell Count, in millions of cells per microlitre |
| `Haemoglobin` | Haemoglobin level, in grams per decilitre |

(a) Produce a matrix of scatterplots between the pairs of variables using the `plot()` function (or `pairs()` function) and briefly describe the relationships you see. [3 marks]

(b) Produce a plot of side-by-side boxplots to show the difference in distributions of `Haemoglobin` between females and males. Describe and compare the distributions. [3 marks]

(c) Use the `lm()` function to model `Haemoglobin` as a function of `Sex`, saving the model as an `R` object. Produce a summary of the model. Interpret the output from this model. [3 marks]

(d) Use the `lm()` function to model `Haemoglobin` as a function of `Sex`, `RedBCC`, and the interaction between these variables, saving the model as an `R` object. Produce a summary of the model. Interpret the output from this model. [4 marks]

**The remaining parts of this question all refer to the model in part (d).**

(e) Plot a four-in-one diagnostic plot of your model, and describe whether it shows any causes for concern. [4 marks]

(f) Perform a Shapiro Test to test the assumption of normality of the residuals from your model and interpret the outcome. [2 marks]

(g) Use your model to predict the `Haemoglobin` level at 100 equally spaced values of `RedBCC` between 3.5 and 6.0 cells per microlitre, once for females, and then for males, storing the predictions in two vectors. Plot these predictions as line graphs on the same plot (`Haemoglobin` predictions on the y-axis, new `RedBCC` values on the x-axis), using different line types for males and females (start with the females, because their predictions are over a larger range than the males. Include axis labels and a legend to differentiate between the line for males and the line for females. [6 marks]

(h) Save the model matrix of your linear model as `X` and the vector of values of `Haemoglobin` as `y` in `R`. Use these objects to calculate the following. Show your commands.

i) The least squares estimates of the regression coefficients, $\hat{\beta}$.

ii) The hat matrix $H$, and save this matrix as `H` in `R`. Show the first three rows and columns of this matrix in your report, along with the commands to generate it.

iii) Using only the vector of responses `y` and the hat matrix `H`, calculate the residual sum of squares using matrix multiplication (i.e. not using the `sum()` function)

[1 mark each]