MT 4113: Computing in Statistics, Practical 1, additional practice material

This sheet contains additional material intended to give you some more practice with control structures, (a bit of) vectorization, and function writing.

Part 1 – Control structures

1. Write R code that simulates a number from a random uniform distribution; if this number is more than or equal to 0.5 print out the word "lucky", and if less than 0.5 print out "unlucky".
2. Here is some code

   ```
   x <- seq(-10, 10, by = 0.01)
   y <- x ^ 2
   plot(x, y, type = 'l')
   ```

   Revise this code so that $y = x^2$ when $x \geq 0$ and $y = 0$ when $x < 0$.
3. Write code to forecast the weather. Your code should print one of three forecasts at random (with equal probablity of each occuring): either "rainy", "hurricane", or "hailstorms". (Hint: if you want to use `switch()` then you should look at the help if the first argument to it will be an integer, as it works differently from the notes in this case.)
4. Extend the above code, so that it repeatedly forecasts the weather until you get a rainy forecast. Print out the number of forecasts output before getting the first rainy forecast.
5. Edit the above code so it also stops if it reaches 4 forecasts without getting "rainy". Have it print out an appropriate message (i.e., if it forecast "rainy" then as before, but if not a message saying it stopped after 4 tries).
6. Simulate a 4x3 matrix of random normal deviates (mean 0 sd 1). Compute the sum of each row.

Part 2 – Functions

1. Take the weather forecast code from question 3 of Part 1 and turn it into a function, `forecast` that has no arguments, but returns the forecast as a character string. Once this function is defined, it should be possible to type `forecast()` and get a forecast. I could save 10 forecasts with

   ```
   n <- 10
   forecasts <- numeric(n)
   for(i in 1:n){
     forecasts[i] <- forecast()
   }
   ```
2. Extend your `forecast` function so that it has one argument, `n`, which is the number of forecasts to produce. Then you should be able to type `forecast(10)` or `forecast(n = 10)` and get a vector of 10 forecasts.
3. Extend the code from question 2 on the last tutorial to turn it into a function, `plot.quadratic(limits)` where `limits` is a vector of length 2 that gives the lower and upper limits for the x-axis. Give the limits argument default values. Add some error checking on the input if you like.
4. (Advanced optional question - only attempt if you found the previous ones easy, or if you like a challenge!) Write a function `plot.function` so you can pass in the limits but also the type of univariate function you want to plot (e.g., x^2 or $\sin(x)$, etc). It will then plot that function within the specified limits. (Hint: in R, you can pass in functions as function arguments.)