R Programming: Worksheet 1

1. **Sequences**

   Generate the following sequences and matrices

   (a) $1, 3, 5, 7, \ldots, 21$.

   (b) $50, 47, 44, \ldots, 14, 11$.

   (c) $1, 2, 4, 8, \ldots, 1024$.

   (d)
$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix}$$

2. **Sampling**

   The command `sample()` performs random sampling; for example, to give a random permutation of the numbers 1 to 10, we could do one of:

```
> sample(10)
> sample(1:10)
```

   You can read through `?sample` to understand why these two function calls are the same

   (a) A scientist needs to experiment upon 4 conditions, 5 times each. Generate a vector $(1, 1, 1, 1, 1, 2, \ldots, 4, 4)^T$ of length 20, representing these conditions. *Hint, use the `rep` function, using `?rep` to understand how it works*

   (b) The scientist wants to do the 20 experiments in a completely random order; use `sample()` to reorder the elements of the vector from (a).

   (c) The scientist calls the conditions A, B, C and D. How would you return a character vector with entries `"A", "B", "C", "D"` containing your random permutation?

3. **Random Walks**

   A *random walk* on the integers is a sequence $X_0, X_1, X_2, \ldots$ with $X_0 = 0$, and
$$X_i = X_{i-1} + D_i,$$
   where the D_i are independent with $P(D_i = +1) = P(D_i = -1) = \frac{1}{2}$.

   (a) Have a look at the documentation for the function `sample()`. Use it to generate a vector $(D_1, \ldots, D_{25})^T$.

   (b) Use the command `cumsum()` to generate $(X_0, X_1, \ldots, X_{25})^T$ from this.

   (c) Plot your random walk:

```
> plot(X, type="l")
```

   Try plotting the first 1,000 steps of a random walk.

(d) We can rewrite

$$X_n = \sum_{i=1}^{n} D_i = 2Z_n - n$$

where the distribution of Z_n is binomial (with what parameters?) To generate a random binomial distribution use `rbinom()`:

```
> rbinom(1, 25, 0.5)
```

What does each of the arguments `1`, `25`, and `0.5` do? Remember to use the help file if necessary.

Write some code to generate a realization of X_{25}.

(e) Generate a vector containing the value of X_{25} for 100,000 independent realizations of the symmetric random walk. How could we estimate the probability of X_{25} exceeding 10?

(f) How could we calculate this exactly? Compare to your answer above. [Try looking at `?pbinom`.]

4. **Diagonals**

(a) Create a diagonal matrix whose diagonal entries are $1, \frac{1}{2}, \frac{1}{3}, \ldots, \frac{1}{10}$. Call it `D`. *Check out the `diag` function*

(b) Now define a 10×10 matrix whose entries are all -1, except on the diagonal, where the entries should be 4. Call it `U`

(c) What is the length of the first column vector in `U`? *Not length as in number of entries, but length as in distance from the origin*

Renormalize the entries of `U` so that each column is a unit vector.

Verify that your approach is correct, using the `stopifnot` function, and the `all.equal` function *Check out `?stopifnot` and `?all.equal`*

(d) Calculate the matrix UDU^T, and call it `X`.

(e) Find the eigenvalues of `X` numerically (to search for a term in a function in R, use the double question mark, try typing `??eigenvalue`). Is this what you expected?

(f) Can you use vector recycling to calculate DU^T without using matrix multiplication?

5. **Binary representation**

To better understand rounding problems, here we will convert a non-negative number $x \in [0, 1)$ to its binary reprsentation.

Let b be the binary representation of x to $I \in \mathbb{N}$ binary places.

Starting with $i = 1$, and while $x > 0$, repeat the following.

i Let $y = 2x$

ii If $y \geq 1$ set $b_i = 1$ otherwise set $b_i = 0$

iii Let $x = y - b_i$

iv If $x = 0$ or $i = I$ then stop (as either there are no more non-zero places, or we have reached the limit of our number of digits) v Otherwise increase i by one and repeat

a Implement this algorithm in R using a while loop, and use it to find the binary representation for 0.3 for up to 20 positions ($I = 20$).

b Save your answer for $x = 0.3$ as *bin*_0.3, and compute the above for $0.1+0.1+0.1$, and save it as *bin*_0.1_*three_times*. At what decimal position does the binary representation of 0.3 differ from $0.1 + 0.1 + 0.1$? *Hint, you need to increase I above 20 to a suitable value. You can also use functions here to make this simpler, we will see functions properly in lecture 2*