R Programming: Worksheet 1

Try to focus on questions 1–3; there are a couple extra for those who finish quickly. Bits with an asterisk (*) are slightly harder, and are either non-examinable or will be covered later.

Functions used:

```
seq(), rep()
sample(), rnorm()
matrix(), t(), solve(), ncol()
apply()
sd(), var(), cumsum()
rbinom(), pbinom(), diag(), %*%
plot() # 1-dimensional data
```

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)
```

   (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.

   (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. **Matrices and `apply()`**

   Remember that a matrix can be created with the command `matrix()`, and that it fills in by column first:

```
> A = matrix(1:12, nrow = 3)
```

The command `apply()` allows you to neatly perform an operation on each row (or column) of a matrix. For example, if you want the row-by-row averages of the matrix A, you could use

```
> apply(A, 1, mean)
```

or for the column means, use

```
> apply(A, 2, mean)
```

(a) Create a 10×11 matrix of independent standard normal random variables; call it `A`.

(b) How would you find the maximum entry in each row of A?

(c) Calculate the standard deviation of each column of `A` (the command you need is `sd()`).

(d) Select the last column of `A`, and call it `b`. Then remove the last column from the original `A`. Do this using the function `ncol()`.

(e) Solve the system of linear equations $Ax = b$.

(f) Find a vector containing the sums of each row of `A`.
Can you think of (or find) any other ways of achieving this?

(g) * Create a second matrix `B`, where the ith column of `B` is the sum of the first i columns of `A`.

4. **Random Walks**

A *random walk* on the integers is a sequence $X_0, X_1, X_2, \dots$ 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 $(D_1, \dots, D_{25})^T$.

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

5. **Diagonals**

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

(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`?

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

Check directly that your approach is correct.

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

(e) Find the eigenvalues of `X` numerically (try typing `??eigenvalue`). Is this what you expected?

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