CT5102: Programing for Data Analytics

Problem Sheet - Vectors

1. Generate a vector set of random numbers from a normal distribution (note the seed value must be set to 111).

```
set.seed(111)

x<-rnorm(n=101,mean=72,sd=8)

> head(x)
[1] 73.88177 69.35411 69.50701 53.58123 70.63299 73.12223

> tail(x)
[1] 60.79167 82.07299 70.98018 66.16491 62.30911 76.79696
```

   Based on this vector, write your own R code (no loops can be used) to perform the following (results should be stored in separate variables and printed using the cat() function).

   - Calculate the mean
   - Calculate the standard deviation
   - Calculate the range
   - Calculate the median
   - Create a new vector with all values less than the mean
   - Create a new vector with all values greater than or equal to the mean
   - Find the top 5 values
   - Find the bottom 5 values
   - Find the absolute difference from the mean for each value

Where appropriate, you can test your code using standard R functions, e.g. mean(), sd(), and range().

Note, the standard deviation for a sample is:

$$sd = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

2. The goal of this challenge is to use vectors to perform simulation of two dice rolls, and analyse the data (see table above for combinations, outcomes and expected frequencies). The R features that can be used are:

Combinations of Dice						**Sum of two dice**					
(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(2)	(3)	(4)	(5)	(6)	(7)
(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(3)	(4)	(5)	(6)	(7)	(8)
(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(4)	(5)	(6)	(7)	(8)	(9)
(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)	(5)	(6)	(7)	(8)	(9)	(10)
(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)	(6)	(7)	(8)	(9)	(10)	(11)
(6,1)	(6,2)	(6,3)	(6,4)	(6,5)	(6,6)	(7)	(8)	(9)	(10)	(11)	(12)

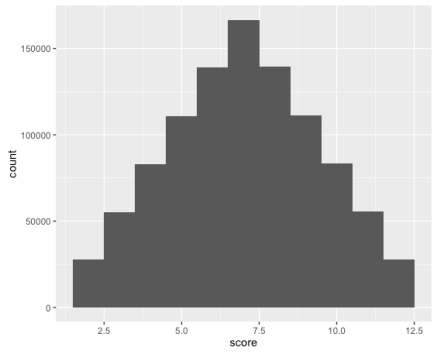
2	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	*2*	*3*	*4*	*5*	*6*	*5*	*4*	*3*	*2*	*1*

- The sample() function, to simulate the roll of a dice, assuming each face has an equal chance of being selected.
- Logical subsetting for atomic vectors
- qplot(data, binwidth=1) to view the simulations
- The mean() and sqrt() functions to calculate the root mean square error which captures the distance between the simulation results and the true probabilities (e.g. 6/36 for the number 7 etc)

For sample sizes of 10^1, 10^2, 10^3, 10^4, 10^5 and 10^6:

- Generate the simulation of two dice rolls
- Plot on the screen to visually verify that they distribution looks as expected
- Extract the frequency of the 11 possible outcomes (note the *table* function cannot be used for this analysis, rather a form of logical subsetting should be used), and the results copied into a new atomic vector
- Calculate the RMSE between the simulated dice rolls and the expected outcomes.
- Plot this RMSE value for each of the sample sizes on the x axis, using qplot (x,y)

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$



3. Generate 5 random passwords of length 10, where a password can have any digit, or letter (upper and lower case). The vector to sample from should be:

```
> both
"a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q"
"r" "s" "t" "u" "v" "w" "x" "y" "z" "A" "B" "C" "D" "E" "F" "G" "H"
"I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S" "T" "U" "V" "W" "X" "Y"
"Z" "0" "1" "2" "3" "4" "5" "6" "7" "8" "9"
```

The test output (using the seed value of 99) should be:

```
> ans2
[1] "wyyEexjOjZ" "rsqvWEOIMU" "BLWZ3MNLdR" "G1Xp7ZimvO" "yUHUrgR57Y"
```

The R Functions to be used include:

- **`set.seed(99),`** to ensure that the results shown below are replicated.
- **`sample()`** – to generate the random samples
- **`paste()`** – for collapsing a vector of characters into a single string.
- The R constants **LETTERS** and **letters**.

4. Demonstrate the idea of vectorization in R, using R's sqrt() function as an example, where the input vector is a sequence of even numbers from 2-10.

5. Create a vector of 100 random numbers, in the range 1-10. From this vector, filter those variables that are divisible by 2 and divisible by 5. Finally, ensure that there are no duplicates in the resulting vector (the R function duplicated() can be used to support this final operation).

6. A quadratic equation has the form ax^2+bx+c. Use sapply() to transform an input vector in the range [-100,+100] using a quadratic equation, where the parameters a, b and c are provided as additional inputs to the transformation.

7. For an input vector of 1000 uniform random numbers, find the difference of each element from the overall mean, and filter out all those resulting elements that are less than zero or equal to zero.

8. Write a script that extracts every fourth element of a vector.

9. Determine the types for each of the following (coerced) vectors:

```r
v1<- c(1L, T, FALSE)

v2<- c(1L, T, FALSE, 2)

v3<- c(T, FALSE, 2, "FALSE")

v4<- c(2L, "FALSE")

v5<- c(0L, 1L, 2.11)
```

10. Create an R vector of squares of 1 to 10. Find the: minimum, maximum and average. Subset all those values greater than the average

11. Create a vector of random numbers between 30 and 90 (random grades for an test) Create a new (parallel) vector that categorises results as follows:
    - Greater or equal to 60 – Honours
    - Between 40 and 59 – Pass
    - Less than 40 – Fail
    - Use the ifelse() (vectorised) function

12. Create a vector of 100 numbers. Set 10 of these to NA (random). Print the locations of the missing values. Hint: Check out the R function ***which***