Exercise 1 to 6 are for practice purpose.

#To make your results reproducible, you can specify the value of the seed using set.seed(‘any number’) before calling a random function. (If you are not familiar with seeds, think of them as the tracking number of your random number process.) We will use set.seed(1).

**Exercise 1**

Generating random numbers. Set your seed to 1 and generate 10 random numbers using runif and save it in an object called random\_numbers.

**Exercise 2**

Using the function ifelse and the object random\_numbers simulate coin tosses. Hint: If random\_numbers is bigger than .5 then the result is head, otherwise is tail.

Another way of generating random coin tosses is by using the rbinom function. Set the seed again to 1 and simulate with this function 10 coin tosses. Note: The value you will obtain is the total number of heads of those 10 coin tosses.

**Exercise 3**

Using the function rbinom to generate 10 unfair coin tosses with probability success of 0.3. Set the seed to 1.

**Exercise 4**

We can simulate rolling a die in R with runif. Save in an object called die\_roll 1 random number with min = 0 and max = 6. This mean that we will generate a random number between 1 and 6.

Apply the function ceiling to die\_roll. Don’t forget to set the seed to 1 before calling runif.

**Exercise 5**

Simulate normal distribution values. Imagine a population in which the average height is 1.70 m with an standard deviation of 0.1, using rnorm simulate the height of 100 people and save it in an object called heights.

To get an idea of the values of heights applying the function summary to it.

**Exercise 6**

a) What’s the probability that a person will be smaller or equal to 1.90 m ? Use pnorm  
b) What’s the probability that a person will be taller or equal to 1.60 m? Use pnorm

**Exercise 7**

Generating dice rolls:  Set your seed to 1 and generate 30 random numbers using runif. Save it in an object called random\_numbers. Then use the ceiling function to round the values. These values represent rolling dice values.

**Exercise 8**

Simulate normal distribution values. Imagine a population in which the average height is 1.70 m with a standard deviation of 0.1. Using rnorm, simulate the height of 100 people and save it in an object called heights.

To get an idea of the values of heights, use the function summary.

**Exercise 9**

90% of the population is smaller than \_\_\_\_\_\_\_\_\_\_\_\_?

**Exercise 10**

Which percentage of the population is bigger than 1.60 m?

**Exercise 11**

Let’s assume that we want to simulate a game in which we throw an unfair coin (success probability is 0.48) 10 times and you win $10 every time the result is tails and lose $10 when the result is heads. Simulate this game 1000 times using rbinom, and find the expected amount of money you will gain or lose in this game using the simulated values.

**Exercise 12**

Simulate normal distribution values. Imagine a population in which the average height is 1.70 m with a standard deviation of 0.1. Use rnorm to simulate the height of 1000 people and save it in an object called heights.

a) Plot the density of the simulated values.  
b) Generate 10000 values with the same parameters and plot the respective density function.

This plot will show you how much a sample with 10000 simulations approximate to the real normal distribution.

**Exercise 13**

Find the 90% interval of a population with mean = 1.70 and standard deviation = .1

**Exercise 14**

Select ten random numbers between one and three using uniform distrbution.

**Exercise 15**

Assume that the test scores of a college entrance exam fits a normal distribution. Furthermore, the mean test score is 72, and the standard deviation is 15.2. What is the percentage of students scoring 84 or more in the exam?

**Exercise 16**

Suppose there are twelve multiple choice questions in an English class quiz. Each question has five possible answers, and only one of them is correct. Find the probability of having four or less correct answers if a student attempts to answer every question at random.