Simulation practise and redoing the class code with some extra stuff

Uzair Mirza

05/05/2021

Seed

What is the use of seed

set.seed(x) function where x is a number is used to make the sampling consistent for every run. Although the samples are still random but set.seed() makes it consistent when lets say the marker will run the code chunk with the same seed will get the same randomly generated values. Without setting seed each execution of the code will result in a different generated values for the sample.

```
            ##NEED TO FIND A SOLUTION TO THIS, NOT WORKING CURRENTLY

# If a certain option needs to be frequently set to a value in multiple code
# chunks, you can consider setting it globally in the first code chunk of your
# document. Hence over here we set this chunk as a "setup" to make all the to be
# generated samples consistent with the given set seed.

## setting up seed
set.seed(6969)
```

sample()

working with sample(), what it does

sample(x, size, replace=, prob=) is used to randomly sample from a vector, size is the amount of entries required, replace = F is by default and prob = is assigning prob to the entries in the vector

```
set.seed(6969)   ## FIND CODE REPETETION SOLUTION and update

# make a vector and take the sample
test.1 <- c(1:10)
samp.1 <- sample(test.1, 5)
# making a dataframe and tibble()
data.1 <- tibble(vec_sample = samp.1)
data.1
```

```
## # A tibble: 5 x 1
##   vec_sample
##        <int>
## 1         10
## 2          4
```

```
## 3          6
## 4          1
## 5          2
```

runif()

working with runif(), what it does

runif(n, min = 0, max = 1) is used to take samples of size n from numbers between "min=" and "max=". **NOTE** the distribution used to take these samples is Uniform distribution **HENCE** we see the samples which are generated are decimals(dbl data type). We can use round() to round the generated sample to integer form or round it to required places if required

```
set.seed(6969)   ## FIND CODE REPETETION SOLUTION and update

# trying out runif() and round()
samp.2 <- runif(5, min = 0, max =10) %>% round(digits = 2)

# adding this result with the previous result to a dataframe using cbind() OR mutate()
        # used mutate() bc easier and cleaner to declare column
data.2 <- data.1 %>%
  mutate(runif_sample = samp.2)
data.2
```

```
## # A tibble: 5 x 2
##   vec_sample runif_sample
##        <int>        <dbl>
## 1         10         6.09
## 2          4         3.17
## 3          6         7.36
## 4          1         1.86
## 5          2         2.75
```

rnorm()

working with rnorm(), what it does

rnorm(n, mean = 0, sd = 1) is used to generates "n" samples from normal distribution with the provided "mean" and "sd"

```
set.seed(6969)   ## FIND CODE REPETETION SOLUTION and update

# trying out rnorm()
samp.3 <- rnorm(5, mean = 2, sd = 3)

# adding this result with the previous result to THE PREVIOUS dataframe
data.2 <- data.2 %>%
  mutate(rnorm_sample = samp.3)
data.2
```

```
## # A tibble: 5 x 3
```

```
##   vec_sample runif_sample rnorm_sample
##        <int>        <dbl>        <dbl>
## 1         10         6.09        2.83
## 2          4         3.17        3.89
## 3          6         7.36        0.208
## 4          1         1.86       -4.04
## 5          2         2.75        1.66
```

SUMMARY

This document contains details and is practice for using and generating different samples techniques for simulation. On top of that making dataframes, mutating and apending to existing data frames and rounding the entries in the dataframe.