Simulating a Deck of Cards in R

Salman Quaiyum

1. Introduction

This is the final project for Udacity's Intro to Descriptive Statistics course. It has been implemented using **R**.

2. Problem Statement

This project simulates a deck of **52** (fifty two) playing cards divided into four suits: **spades**, **hearts**, **diamonds**, and **clubs**. Each suit contains **13** (thirteen) cards. A value is assigned to each individual card: the **numbered cards** are assigned their **respective numbers (2,3,4 etc.)**, **1** is assigned to **Ace** and **10** is assigned to **Jack**, **Queen**, and **King** of each suit.

3. Generating the population

3.1 Coding

At first, we impport the required libraries

```
library(dplyr)
library(tidyr)
```

Then we declare the card suits, card names and assign each card its respective value.

```
card_suits <- c("Spades", "Hearts", "Diamonds", "Clubs")

card_names <- c("Ace", "King", "Queen", "Jack", "Ten", "Nine", "Eight",
"Seven", "Six", "Five", "Four", "Three", "Two")

card_values <- c(1, 10, 10, 10, 10:2)
```

Then we manipulate the three vectors (`card_names`, `card_suits`, `card_values`) to create a deck containing **52** cards along with their associated values.

```
n_suits <- length(card_suits)

card_deck <- expand.grid(card_names, card_suits)

card_deck <- card_deck %>%
              unite("Cards", c(1, 2), sep = "_", remove = TRUE) %>%
              mutate(Values = rep(card_values, n_suits))
```

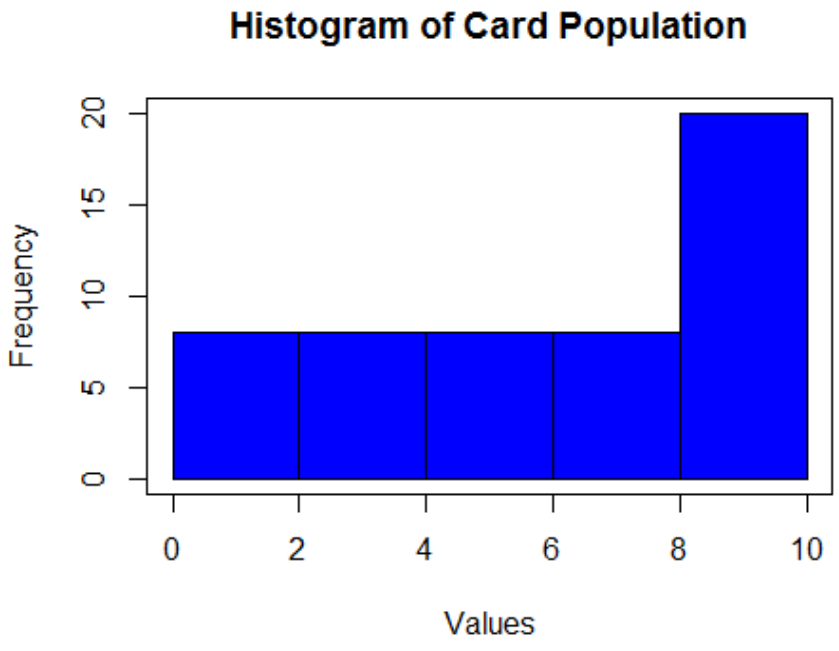
3.2 Visualization

Now, our deck of cards has been built.

Cards	Values
Ace_Spades	1
King_Spades	10
Queen_Spades	10
Jack_Spades	10
Ten_Spades	10
Nine_Spades	9
Eight_Spades	8
Seven_Spades	7
Six_Spades	6
Five_Spades	5
Four_Spades	4
Three_Spades	3
Two_Spades	2
Ace_Hearts	1
King_Hearts	10
Queen_Hearts	10
Jack_Hearts	10
Ten_Hearts	10
Nine_Hearts	9
Eight_Hearts	8
Seven_Hearts	7
Six_Hearts	6
Five_Hearts	5
Four_Hearts	4
Three_Hearts	3
Two_Hearts	2
Ace_Diamonds	1
King_Diamonds	10
Queen_Diamonds	10
Jack_Diamonds	10
Ten_Diamonds	10
Nine_Diamonds	9
Eight_Diamonds	8
Seven_Diamonds	7
Six_Diamonds	6

Five_Diamonds	5
Four_Diamonds	4
Three_Diamonds	3
Two_Diamonds	2
Ace_Clubs	1
King_Clubs	10
Queen_Clubs	10
Jack_Clubs	10
Ten_Clubs	10
Nine_Clubs	9
Eight_Clubs	8
Seven_Clubs	7
Six_Clubs	6
Five_Clubs	5
Four_Clubs	4
Three_Clubs	3
Two_Clubs	2

Here is the histogram of our population:



3.3 Analysis

We would like to see some statistical information from our population. We could have just run the built in `summary` function. Unfortunately, it doesn't give us everything we

need. Therefore, we create our own summary function called `sum_spc`. And use this function to observe the descriptive statistics.

```
##       min      mean       med       max       var        sd
##  1.000000  6.538462  7.000000 10.000000 10.135747  3.183669
```

4. Generating the Samples

4.1 Coding

Now, let's work on creating our sampling distribution. First, we assign the sample size (`sample_size`) and number of samples to be taken (`sampling_number`).

```
sample_size = 3

sampling_number = 30
```

Now, we will simulate taking **3** random samples from the population. At first, we set the seed, so we will always generate the same samples.

```
set.seed(110)
```

The following chunks of code simulate randomly taking **3** cards from a deck and add up their values. We simulate this using the **R** built in function `sample`. We call this function with four arguments: `x` which is our population values, `size` which is equal to `sample_size` and finally `replace` which equals `FALSE`. The last argument ensures that the samples will be drawn without any replacements. And this process is repeated **30** times, which is evident from the `for` loop.

```
for(i in 1:sampling_number){
  cards[i] <- list(sample(card_deck$Cards, size = sample_size, replace =
FALSE))
  sample_sums[i] <- sum(card_deck$Values[card_deck$Cards %in% cards[[i]]])
}
```

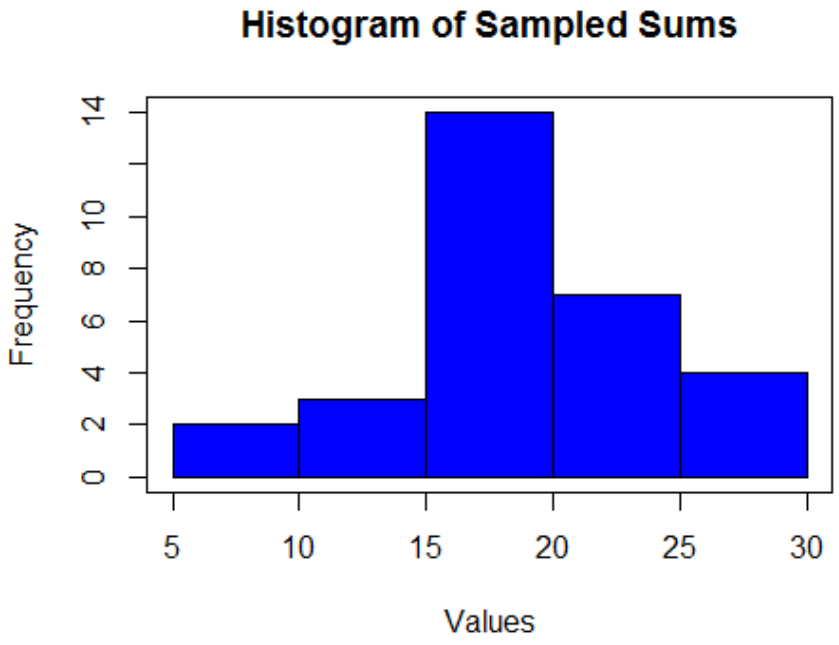
4.2 Visualization

Let's view our samples:

Cards	Sums
Nine_Diamonds + Three_Spades + Eight_Clubs	20
Three_Diamonds + Two_Clubs + Eight_Spades	13
Five_Clubs + Four_Clubs + Seven_Diamonds	16
Ten_Diamonds + Five_Diamonds + Seven_Spades	22
Two_Hearts + Four_Diamonds + Seven_Clubs	13
Eight_Spades + Queen_Spades + Ace_Diamonds	19
Ten_Diamonds + Ten_Hearts + Seven_Clubs	27
Ten_Diamonds + Jack_Clubs + Nine_Spades	29
Three_Clubs + Two_Spades + Two_Clubs	7

Seven_Diamonds + Four_Clubs + Nine_Clubs	20
Ten_Diamonds + Five_Spades + Jack_Spades	25
King_Hearts + Three_Diamonds + Six_Hearts	19
Five_Clubs + Ace_Clubs + King_Spades	16
Six_Clubs + Seven_Diamonds + Ten_Diamonds	23
Jack_Spades + Three_Clubs + Ace_Clubs	14
Two_Diamonds + Four_Spades + Three_Hearts	9
King_Clubs + Eight_Hearts + Five_Clubs	23
Seven_Diamonds + Nine_Spades + Four_Diamonds	20
Ten_Spades + Ace_Hearts + Nine_Hearts	20
Three_Diamonds + Seven_Hearts + Eight_Spades	18
Ten_Diamonds + Jack_Clubs + Eight_Diamonds	28
Seven_Diamonds + Nine_Hearts + Ace_Diamonds	17
Four_Spades + Ten_Spades + Five_Spades	19
Four_Hearts + Ten_Diamonds + Jack_Clubs	24
King_Clubs + Jack_Hearts + Nine_Diamonds	29
Nine_Hearts + Three_Spades + Eight_Diamonds	20
Jack_Hearts + Seven_Diamonds + Six_Spades	23
Ten_Hearts + Two_Diamonds + Queen_Diamonds	22
Four_Diamonds + Queen_Spades + Two_Spades	16
Four_Hearts + Seven_Spades + Seven_Hearts	18

Here is the histogram of our Sampling Distribution:



4.3 Analysis

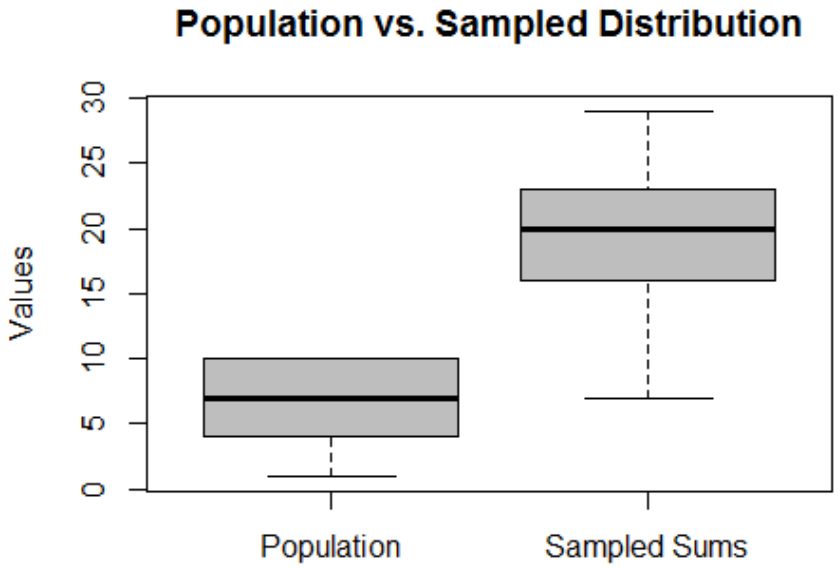
We again make use of the `sum_spc` function to see relevant statistical information of the samples

```
##       min      mean       med       max       var        sd
##  7.000000 19.633333 20.000000 29.000000 28.929885  5.378651
```

5. Comparison

If we compare the histograms of both the pouplation and sum of sampled distribution, it is evident that their distribution is completely different. The histogram of population is negatively skewed. Whereas, the histogram of the sampled sums has a more uniform distibution.

Finally, we observe a boxplot comparing the values obtained from population and the samples



The box plot illustrates the differences between the population and the sample. The sample values do overlap with the population values - precisely between the population median and the upper quartile.

6. Estimating future values

6.1 Approximate range of 90% values

From the **Z Table**, we look for the probability **90% (0.9)**. The closest we get is **0.9015** for **z = 1.29**. Now, we have to reverse calculate the value.

```
z_90 = 1.29
x_90 = mu_samp + (sigma_samp * z_90)
```

Which gives us **26.57179**. Which means approximately **90%** values should be less than **26.57**.

6.2 Probablity for getting a draw value of at least 20

First we calculate the corresponding **z value** for **20** using **mean** and **standard deviation** of the sampled distribution.

```
z_20 <- (20- mu_samp)/sigma_samp
```

Using **Z Table**, the probability for **z = 0.0681 (0.07)** is **0.5279**. Therefore the probability of getting at least 20 per draw is **(1-0.5279) = 0.4721**.

Or, inversely we could look for the probability of **z = -0.07**, which directly gives us **0.4721**.

We could also directly calculate the probability in R using the `pnorm` function.

```
prob_20 <- (1 - pnorm(z_20))
```

Which gives us 0.4728.

7. Conclusion

This brings us to the end. I hope the approach I took for solving this problem was the correct one and I have succeeded in providing a logical solution. I have learnt a lot while doing this project. I will definitely recommend taking Udacity's course on descriptive statistics for anyone looking to refresh their statistical knowledge. Feel free to provide any suggestion / comments / feedbacks.

https://about.me/salmanq