Monte Carlo Simulation

Gustavo R Santos

5/22/2022

Creating the Lake with 100 marked fish

```
fishes <- data.frame(num = seq(1:500),
                     marked = rep(c(1,1,0,0,0,0,0,0,0,0), times=50)
                     )

# Confirming 100 marked fishes
marked_fish_total <- sum(fishes[fishes$marked==1,2])
marked_fish_total
```

```
## [1] 100
```

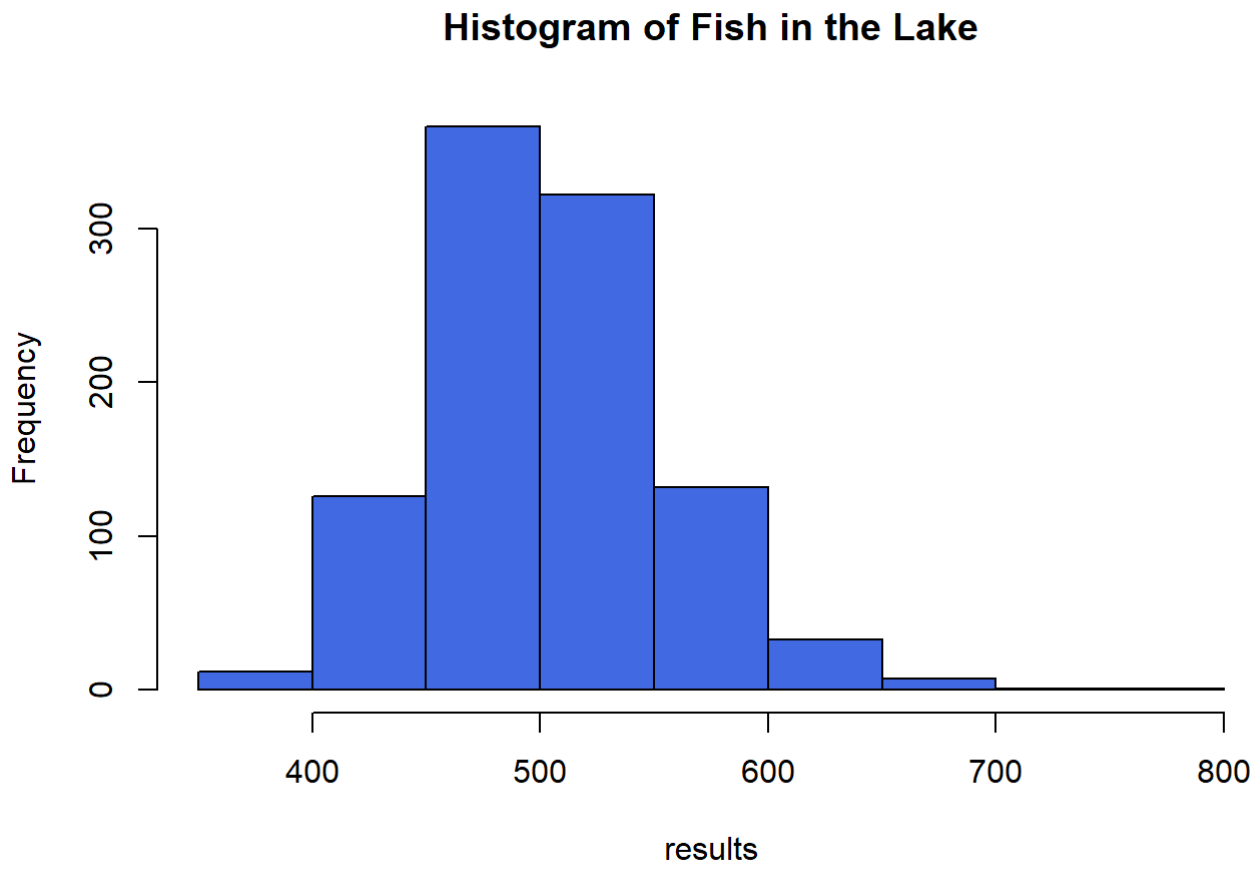
Monte Carlo Simulation

Next we are going to create a function to simulate 100 fish experiments and then calculate the number of fishes in the lake out of the results.

```
monte_carlo <- function (reps, dataset){
      "Run the same experiment for n reps and return the average proportion
      * reps: [int] How many repetitions of the same experiment
      * dataset: [data.frame] it is the dataframe with the fish data
      "

# Creating a vector to collect the marked fishes proportion
results <- c()
marked_prop <- c()

#Let's simulate a fishing of 100 fishes, repeated 1,000 times
  for (experiment in seq(1,reps)) {

    # Shuffle the dataset
    shuffled <- sample(nrow(dataset))
    dataset <- dataset[shuffled, ]

    # Create a random index before catching the fish
    index <- sample(1:500, size=100)
    fishing <- dataset[index,]
    # Calculate the proportion
    p <- mean(fishing$marked)

    # Store result in the vector
    marked_prop <- c(marked_prop, p)
    marked_prop <- mean(marked_prop)

    "If we know the percentage of marked fish and we know that the total
    of marked fish in that lake is 100, then we can calculate the 100%
    using this formula:

      Marked Fish Total / Marked Fish Proportion "
      # Total Fish in the lake
      total_fish_lake <- marked_fish_total / marked_prop
      results <- c(results, total_fish_lake)

    } # close for Loop
    # Plot Histogram
  hist(results, col='royalblue', main='Histogram of Fish in the Lake')

} #close the function
```

Running the function created above, here's the result.

```
# Running the Monte Carlo Simulation
marked_fish_proportion <- monte_carlo(1000, fishes)
```



Second Part

In this second part, we're going to simulate the number of customers to attend a restaurant on an average evening.

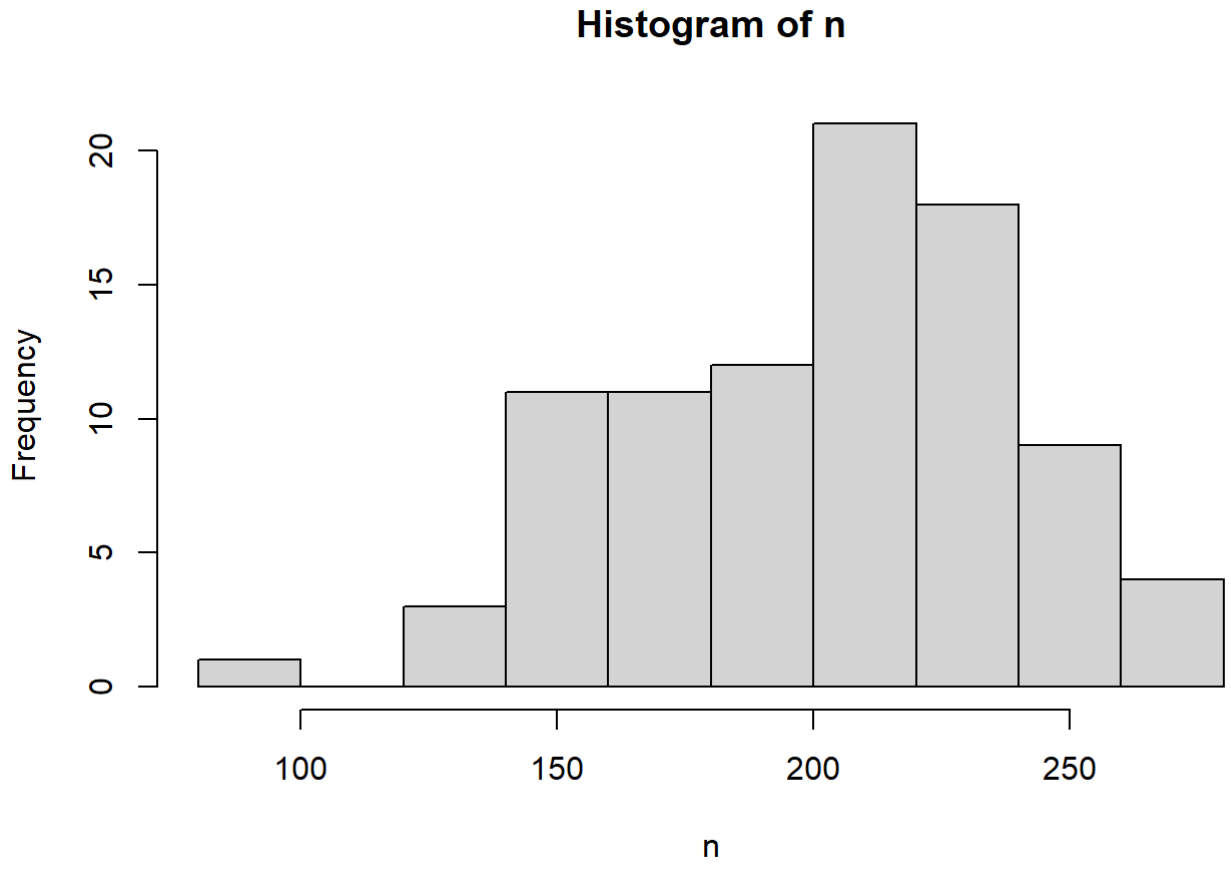
Assuming we are talking about average numbers, the Central Limit Theorem would approximate the averages to a normal curve. So, let's create a sample for 90 days based on a normal distribution.

The dataset

```
# Normal Distribution
n <- rnorm(90, mean=200, sd=40)

# Transform to Interger numbers
n <- as.integer(n)

# Calculate mean and Standar Dev
customer_avg <- mean(n)
customer_std <- sd(n)
print( paste('Mean:', customer_avg, ' | Std:', customer_std) )
```

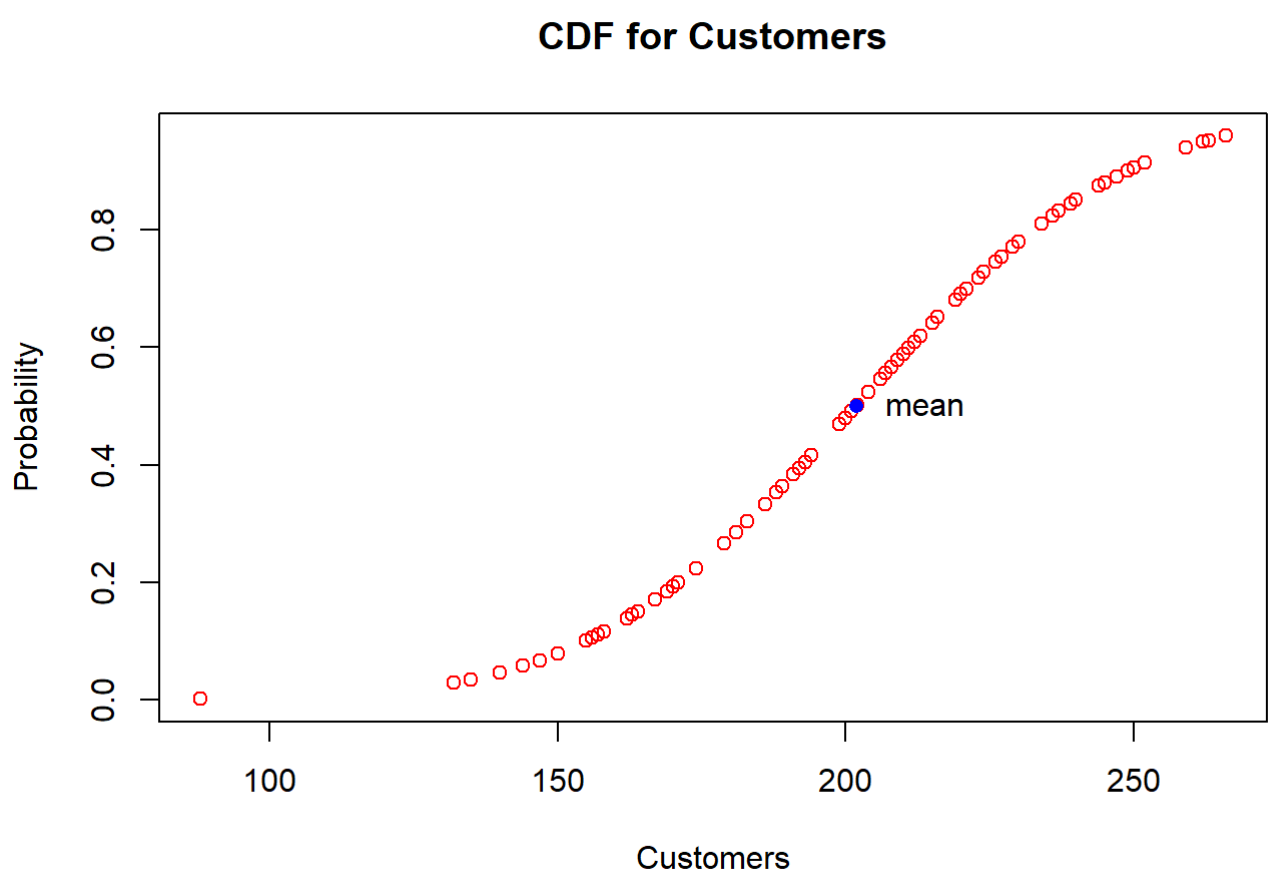
```
## [1] "Mean: 201.8  | Std: 36.540019618329"
```

```
# Plot Histogram
hist(n)
```



Create a CDF for the simulations.

```
# Plot the CDF
plot( x= n,
      xlab= 'Customers',
      y= pnorm(n, mean=mean(n), sd=customer_std),
      ylab= 'Probability',
      col='red',
      main='CDF for Customers')
# Plot the mean
points(x= customer_avg,
     y= 0.5,
     col='blue', pch=16)
# Add annotation
text('mean',x=customer_avg + 12,y=0.5)
```



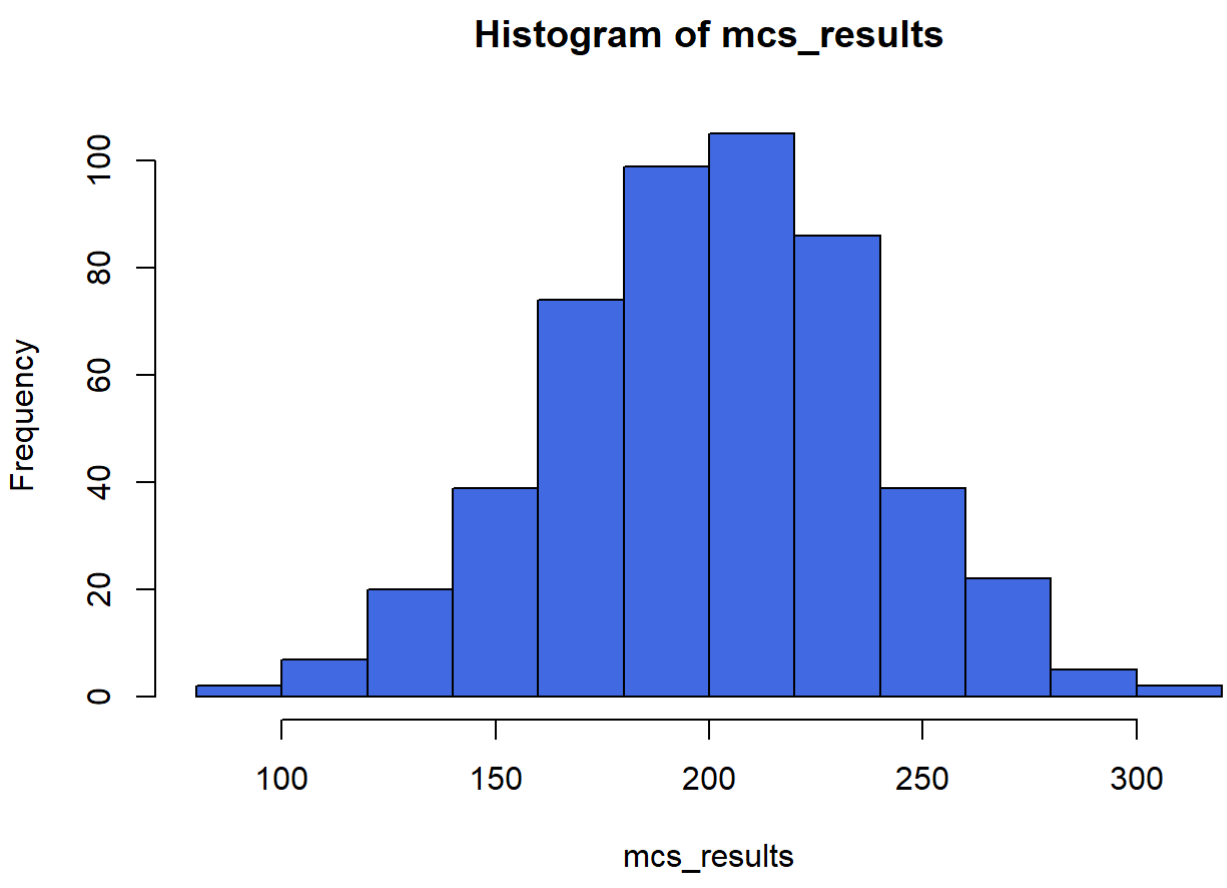
Simulating

We can simulate a random number between 0 and 1, which will translate to a simulated percentage. `runif()` can be used for to create that. The numbers convert to a simulated number of customers.

```
# Simulated attendance
mcs_customers <- function(simulations){
  "This function takes an integer number and generates that amount of repetitions of the simulation"

  # Create a list to store the results
  mcs_results <- c()

  # Loop
  for (n in 1:simulations){
    # Generate a random number
    r <- runif(1)
    # Use our CDF to capture the simulated quantity of customers
    simulated <- qnorm(r, mean=customer_avg, sd= customer_std)
    # Take the lowest integer rounded
    simulated <- floor(simulated)

    #Store result
    mcs_results <- c(mcs_results, simulated)

  } #end loop

  # Plot histogram
  hist(mcs_results, col='royalblue')

  # Return vector
  return(mcs_results)

}# end function
```

Now, it's time to run 500 simulations.

```
mcs <- mcs_customers(500)
```


Histogram of mcs_results
Frequency
0
20
40
60
80
100
100
150
200
250
300
mcs_results