Assessment 06 - The Central Limit Theorem

Gabriele Mineo - Harvard Data Science Professional

American Roulette probability of winning money

The exercises in the previous chapter explored winnings in American roulette. In this chapter of exercises, we will continue with the roulette example and add in the Central Limit Theorem.

In the previous chapter of exercises, you created a random variable S that is the sum of your winnings after betting on green a number of times in American Roulette.

What is the probability that you end up winning money if you bet on green 100 times?

Instructions

- Execute the sample code to determine the expected value `avg`and standard error `se` as you have done in previous exercises.
- Use the `pnorm` function to determine the probability of winning money.

```
# Assign a variable `p_green` as the probability of the ball landing in a green pocket
p_green <- 2 / 38

# Assign a variable `p_not_green` as the probability of the ball not landing in a green pocket
p_not_green <- 1-p_green

# Define the number of bets using the variable 'n'
n <- 100

# Calculate 'avg', the expected outcome of 100 spins if you win $17 when the ball lands on green and yo
avg <- n * (17*p_green + -1*p_not_green)

# Compute 'se', the standard error of the sum of 100 outcomes
se <- sqrt(n) * (17 - -1)*sqrt(p_green*p_not_green)

# Using the expected value 'avg' and standard error 'se', compute the probability that you win money be
1 - pnorm(0,avg,se)
```

```
## [1] 0.4479091
```

American Roulette Monte Carlo simulation

Create a Monte Carlo simulation that generates 10,000 outcomes of S, the sum of 100 bets.

Compute the average and standard deviation of the resulting list and compare them to the expected value (-5.263158) and standard error (40.19344) for S that you calculated previously.

Instructions

- Use the `replicate` function to replicate the sample code for `B <- 10000` simulations.
- Use the `sample` function to simulate outcomes of either a loss ($-1) or a win ($17) for the bet.
- Use the `sum` function to add up the winnings over all iterations of the model.
- Use the `mean` function to compute the average winnings.
- Use the `sd` function to compute the standard deviation of the winnings.

```r
# Assign a variable `p_green` as the probability of the ball landing in a green pocket
p_green <- 2 / 38

# Assign a variable `p_not_green` as the probability of the ball not landing in a green pocket
p_not_green <- 1-p_green

# Define the number of bets using the variable 'n'
n <- 100

# The variable `B` specifies the number of times we want the simulation to run. Let's run the Monte Car
B <- 10000

# Use the `set.seed` function to make sure your answer matches the expected result after random samplin
set.seed(1)

# Create an object called `S` that replicates the sample code for `B` iterations and sums the outcomes.
S <- replicate(B, {
  sum(sample(c(17,-1), n, prob=c(p_green, p_not_green), replace=TRUE))
})



# Compute the average value for 'S'
mean(S)
```

```
## [1] -5.9086
```

```r
# Calculate the standard deviation of 'S'

sd(S)
```

```
## [1] 40.30608
```

American Roulette Monte Carlo vs CLT

In this chapter, you calculated the probability of winning money in American roulette using the CLT.

Now, calculate the probability of winning money from the Monte Carlo simulation. The Monte Carlo simulation from the previous exercise has already been pre-run for you, resulting in the variable `S` that contains a list of 10,000 simulated outcomes.

Instructions

- Use the `mean` function to calculate the probability of winning money from the Monte Carlo simulation, S.

```r
# Calculate the proportion of outcomes in the vector `S` that exceed $0
mean(S>0)
```

```
## [1] 0.4232
```

American Roulette Monte Carlo vs CLT comparison

The Monte Carlo result and the CLT approximation for the probability of losing money after 100 bets are close, but not that close. What could account for this?

Possible Answers

- 10,000 simulations is not enough. If we do more, the estimates will match.
- The CLT does not work as well when the probability of success is small. [X]
- The difference is within rounding error.
- The CLT only works for the averages.

American Roulette average winnings per bet

Now create a random variable Y that contains your average winnings per bet after betting on green 10,000 times.

Instructions

- Use `set.seed` to make sure the result of your random operation matches the expected answer for this problem.
- Specify the number of times you want to sample from the possible outcomes.
- Use the `sample` function to return a value from a vector of possible values: winning $17 or losing $1.
- Be sure to assign a probability to each outcome and to indicate that you are sampling with replacement.
- Calculate the average result per bet placed using the `mean` function.

```
# Use the `set.seed` function to make sure your answer matches the expected result after random samplin
set.seed(1)

# Define the number of bets using the variable 'n'
n <- 10000

# Assign a variable `p_green` as the probability of the ball landing in a green pocket
p_green <- 2 / 38

# Assign a variable `p_not_green` as the probability of the ball not landing in a green pocket
p_not_green <- 1 - p_green

# Create a vector called `X` that contains the outcomes of `n` bets
X <- sample(c(17, -1), n, prob=c(p_green,p_not_green), replace=TRUE)

# Define a variable `Y` that contains the mean outcome per bet. Print this mean to the console.
print(Y <- mean(X))
```

```
## [1] 0.008
```

American Roulette per bet expected value

What is the expected value of Y, the average outcome per bet after betting on green 10,000 times?

Instructions

- Using the chances of winning $17 (`p_green`) and the chances of losing $1 (`p_not_green`), calculate the expected outcome of a bet that the ball will land in a green pocket.

```
# Assign a variable `p_green` as the probability of the ball landing in a green pocket
p_green <- 2 / 38

# Assign a variable `p_not_green` as the probability of the ball not landing in a green pocket
p_not_green <- 1 - p_green
```

```r
# Calculate the expected outcome of `Y`, the mean outcome per bet in 10,000 bets
17*p_green + -1*p_not_green
```

```
## [1] -0.05263158
```

American Roulette per bet standard error

What is the standard error of Y, the average result of 10,000 spins?

Instructions

- Compute the standard error of Y, the average result of 10,000 independent spins.

```r
# Define the number of bets using the variable 'n'
n <- 10000

# Assign a variable `p_green` as the probability of the ball landing in a green pocket
p_green <- 2 / 38

# Assign a variable `p_not_green` as the probability of the ball not landing in a green pocket
p_not_green <- 1 - p_green

# Compute the standard error of 'Y', the mean outcome per bet from 10,000 bets.
abs((17 - -1))*sqrt(p_green*p_not_green) / sqrt(n)
```

```
## [1] 0.04019344
```

American Roulette winnings per game are positive

What is the probability that your winnings are positive after betting on green 10,000 times?

Instructions

- Execute the code that we wrote in previous exercises to determine the average and standard error.
- Use the `pnorm` function to determine the probability of winning more than $0.

```r
# We defined the average using the following code
avg <- 17*p_green + -1*p_not_green

# We defined standard error using this equation
se <- 1/sqrt(n) * (17 - -1)*sqrt(p_green*p_not_green)

# Given this average and standard error, determine the probability of winning more than $0. Print the r
1 - pnorm(0,avg,se)
```

```
## [1] 0.0951898
```

American Roulette Monte Carlo again

Create a Monte Carlo simulation that generates 10,000 outcomes of S, the average outcome from 10,000 bets on green.

Compute the average and standard deviation of the resulting list to confirm the results from previous exercises using the Central Limit Theorem.

Instructions

- Use the `replicate` function to model 10,000 iterations of a series of 10,000 bets.
- After each iteration, take the average of the 10,000 outcomes.
- Find the average of the 10,000 average outcomes.
- Compute the standard deviation of the 10,000 simulations.

```
# The variable `n` specifies the number of independent bets on green
n <- 10000

# The variable `B` specifies the number of times we want the simulation to run
B <- 10000

# Use the `set.seed` function to make sure your answer matches the expected result after random number
set.seed(1)

# Generate a vector `S` that contains the the average outcomes of 10,000 bets modeled 10,000 times
S <- replicate(B, {
  mean(sample(c(17, -1), n, prob=c(p_green,p_not_green), replace=TRUE))
})


# Compute the average of `S`
mean(S)
```

```
## [1] -0.05223142
```

```
# Compute the standard deviation of `S`
sd(S)
```

```
## [1] 0.03996168
```

American Roulette comparison

In a previous exercise, you found the probability of winning more than $0 after betting on green 10,000 times using the Central Limit Theorem. Then, you used a Monte Carlo simulation to model the average result of betting on green 10,000 times over 10,000 simulated series of bets.

What is the probability of winning more than $0 as estimated by your Monte Carlo simulation? The code to generate the vector S that contains the the average outcomes of 10,000 bets modeled 10,000 times has already been run for you.

Instructions

- Calculate the probability of winning more than $0 in the Monte Carlo simulation.

```
# Compute the proportion of outcomes in the vector 'S' where you won more than $0
mean(S>0)
```

```
## [1] 0.0977
```

American Roulette comparison analysis

The Monte Carlo result and the CLT approximation are now much closer than when we calculated the probability of winning for 100 bets on green. What could account for this difference?

Possible Answers

- We are now computing averages instead of sums.

- 10,000 Monte Carlo simulations was not enough to provide a good estimate.
- The CLT does works better when the sample size is larger. [X]
- It is not closer. The difference is within rounding error.