Probability Homework 1: Simulation

Simulation

What is simulation? Why use simulations?

Simulation is a way to model random events, such that simulated outcomes closely match real-world outcomes. By observing simulated outcomes, researchers gain insight on the real world. Some of the benefits of simulations are:

- Analysis where observable data is not available
- Mimicking the repeated sampling framework of classical frequentist statistics
- Provide solutions where analytic solutions are not available or are intractable

What is the difference between a true probability and simulated probability?

Suppose I toss a coin in the air. What is the probability of the coin landing on heads?

- We can define a probability as a logical consequence based on assumptions. In this case we make an assumption about the coin being fair and assert based on that assumption that the probability is *0.5*.
- How do we know this is true? Statistician Karl Pearson once tossed a coin 24,000 times, obtaining 12,012 heads, resulting in an observed proportion of 0.5005.
- This assertion relies on *the law of large numbers*, which states that by increasing the number of observations, the observed proportion of any event occurring will converge to the true probability that the event occurs.
- To mimic this experiment, let's use R function `sample()` to simulate the experiment.

```
set.seed(12345)
# Set the seed for reproducible results.
# This allows everyone running this document to get exactly the same numbers.
```

Tell R to flip a coin 1, 2, and 3 time(s)

```
sample(0:1, 1, replace = T)
```

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

```
# Flip one coin, and get an outcome of 0 = Tails or 1 = Heads

sample(0:1, 2, replace = T)
```

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

```
# Flip two coins.

sample(0:1, 3, replace = T)
```

```
## [1] 1 0 0
```

```
# Flip three coins.
```

In this case, an outcome of a 1 indicates that a heads was flipped, and an outcome of a 0 indicates that a tail was flipped. Therefore, the sum of the sample vector is the number of heads that was flipped.

```r
x = sample(0:1, 3, replace = T)
x
```

```
## [1] 0 1 1
```

```r
sum(x)
```

```
## [1] 2
```

The mean of the vector is the observed proportion of heads that were flipped, which is an estimate of the probability of observing a head.

```r
mean(x)
```

```
## [1] 0.6666667
```

We can also write a function to flip a coin any specified number of times.

```r
# Write a function that flips a fair coin n times
Flip1Coin = function(n) {
  sample(0:1, n, replace=T)
}
```

This is what the outcome looks like:

```r
Flip1Coin(30)
```

```
##  [1] 1 0 0 1 0 0 0 0 0 0 1 0 0 1 1 1 0 1 1 0 0 1 0 0 1 0 0 1 1 1
```

Here we write another function to calculate the observed proportion coins landing on heads:

```r
Prob.Flip1Coin = function(n) {
  x <- sample(0:1, n, replace=T)
  return(sum(x)/n) # this line calculates the observed proportion of heads
}
```

The outcome of fliping a coin 1/2/5/10/20/50/100 times:

```r
Prob.Flip1Coin(1)
```

```
## [1] 0
```

```r
Prob.Flip1Coin(2)
```

```
## [1] 0.5
```

```r
Prob.Flip1Coin(5)
```

```
## [1] 0.4
```

```r
Prob.Flip1Coin(10)
```

```
## [1] 0.5
```

```r
Prob.Flip1Coin(20)
```

```
## [1] 0.55
```

```r
Prob.Flip1Coin(50)
```

```
## [1] 0.66
```

```r
Prob.Flip1Coin(100)
```

```
## [1] 0.56
```

As the number of coin flips goes to infinity, we expect this observed proportion to tend to 0.5. The following code will show this fact. However, first we need to introduce the idea of a *for loop*.

When running code manually is to cumbersome, it is possible to use loops to automate the process. A loop does the same thing over and over again while some condition is true. This will allow us to repeat experiments many times and record the result.

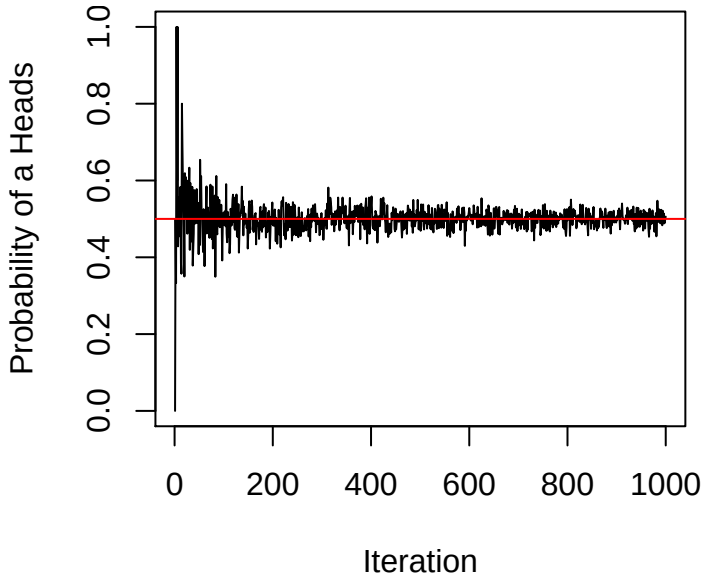
A *for loop* is the most basic type of loop. For some set of indices, the loop will iterate. In this case, the following loop will look at the result of 1 flip, then 2 flips, then 3, continuing until it reaches the number in the ITER variable (currently set to 1000).

```
ITER <- 1000
Prob <- rep(0, ITER)
# Creates a vector of length ITER to store all the different results.

for (i in 1:ITER) {
  Prob[i] <- Prob.Flip1Coin(i)
}
```

Finally, we plot the probability of the coin landing on heads against the number of observations, and see that the observed probability of a heads converges to 0.5.

```
plot(1:ITER,Prob, type = "l", ylab = "Probability of a Heads", xlab = "Iteration")
abline(h=0.5,col=2)
```



Second Example: Roll a Die

Suppose we wanted to roll one die 10 times. Since the output of any die roll is a number between 1 and 6, we could try:

```
sample(1:6, 10, replace = T)
```

```
##  [1] 2 2 4 1 6 6 1 1 6 5
```

```
#Select a number from 1-6, ten times with replacement
```

or we could write a function:

```
RollDie <- function(n) {
  sample(1:6, n, replace = T)
}

RollDie(20)
```

```
##  [1] 3 3 6 2 1 3 5 3 5 1 2 3 3 6 6 2 2 3 3 6
```

```
# Rolls a die 20 times.
```

Now we can 'RollDie' as many times as we want.

Here we practice a for-loop. Say we want to roll 2 dice and record the numbers; then we want to repeat this 100 times. Here's how we do it.

```
roll1 = rep(0, 100)
roll2 = rep(0, 100)
for (i in 1:100) {
  roll1[i] = RollDie(1)
  roll2[i] = RollDie(1)
}
```

How would you calculate the proportion of rolls that were doubles?

```
#ANSWER:
roll1 == roll2
```

```
##   [1]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [12] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [23] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
##  [34] FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE
##  [45] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
##  [56] FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
##  [67] FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE
##  [78] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [89] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE
## [100] FALSE
```

```
## a vector of TRUEs and FALSEs that tell you whether the observed rolls were equal.
## TRUE = 1, FALSE = 0, so the mean of this vector is the observed proportion of doubles.

mean(roll1 == roll2)
```

```
## [1] 0.17
```

Next, increase the number of rolls to 10,000. Recalculate the proportion of rolls that were doubles.

```
roll1 = rep(0, 10000)
roll2 = rep(0, 10000)
for (i in 1:10000) {
  roll1[i] = RollDie(1)
  roll2[i] = RollDie(1)
}

mean(roll1 == roll2)
```

```
## [1] 0.1633
```

We expect this number to tend to 0.16666 since 1/6 of the possible outcomes (out of 36 total) are doubles.

Exercises

Question 1:

A) If we flip a coin, what is the expected probability of getting a head?

ANSWER: 1/2

B) If we flip a coin 10 times, what is the expected number of heads?

ANSWER: 5

C) Have R flip a coin 10 times and count the number of heads. Repeat this 8 times and store the results for each one. (Hint: you can't have a vector of other vectors, but you can have a list of vectors or a matrix)

```
## ANSWER:
flipcoin  <- function(n){
  sample(c(1,0),n, replace = TRUE)
}

flips <- list(length=8)
for (i in 1:8) {
  flips[[i]] <- flipcoin(10)
}

print(flips)
```

```
## $length
##  [1] 0 0 0 1 0 1 0 1 1 1
##
## [[2]]
##  [1] 0 0 1 0 0 1 0 0 0 1
##
## [[3]]
##  [1] 1 1 0 0 1 1 0 0 0 1
##
## [[4]]
##  [1] 1 0 0 0 0 1 1 0 1 0
##
## [[5]]
##  [1] 0 0 1 0 0 1 1 0 1 0
##
## [[6]]
##  [1] 0 1 0 1 1 1 0 1 0 1
##
## [[7]]
##  [1] 1 1 1 0 1 0 1 1 0 1
##
## [[8]]
##  [1] 0 0 1 0 1 0 0 0 1 0
```

D) Have R flip a coin 10 times, and count the number of heads of the 10 flips. Store this number and repeat for 1000 iterations. Print a table of the number of heads.

```
## Answer
nHeads <- rep(0, 1000)

for (i in 1:1000) {
```

```
  nHeads[i] <- sum(flipcoin(10))
}

print(table(nHeads))
```

```
## nHeads
##   0   1   2   3   4   5   6   7   8   9
##   1   8  53 110 193 250 228 113  38   6
```

Homework (Due after 1st week of classes)

Note: This assignment should be submitted as a knitted PDF file. You may delete all notes above the Homework heading before submitting.

A friend proposes that you play a game with dice. You will roll 2 die. If the sum of those two rolls is a 7, your friend will pay you $3; if the sum is an 11, your friend will pay you $5. If the two rolls add up to any other value, you have to pay $0.70

A) Simulate taking the sum of 2 dice using the `RollDie()` function. Do this 1000 times and store the results.

```
## Answer (note: this has been completed for you)

# Re-print the RollDie function for reference
RollDie <- function(n) {
  sample(1:6, n, replace = T)
}

# Make a vector of 1000 0s called twoDiceRolls (to save the results of each game play)
twoDiceRolls <- rep(0,1000)

# Roll 2 die 1000 times and store their sum in the vector twoDiceRolls
for (i in 1:1000) {
  twoDiceRolls[i] <- sum(RollDie(2))
}
```

Calculate the following from the data.

B) Based on the data you collected in part A, what is the (approximate) probability of rolling a seven? Try using the *table* command.

```
## Insert your answer here
```

C) What is the probability of rolling an eleven?

```
## Insert your answer here
```

D) What is the probability of rolling a seven or an eleven?

```
## Insert your answer here
```

E) You play the game 10 times and win $30. You think, boy this game is easy to win at! I should keep playing! Is this the correct assumption? Prove it with a simulation by calculating your mean earning over 10000 iterations, as well as the your net earning (the sum over the 10000 iterations). Hint: Write a for loop where you keep track of the sum of the two dice rolls, and then in the Earning vector, keep track of the monetary transaction that would occur. Then take the mean and sum of the Earning vector. You may want to look up how to use the *if*, *else if* and *else* commands.

```r
# Set the number of iterations to 10,000
nIter <- 10000

# Create an vector of nIter 0s called Earning to store your earnings on each iteration
Earning <- #insert code here

# play the game nIter times and store your earnings each time
for (i in 1:nIter){
  #fix code below to roll two die, take their sum, and save the result as: "Roll"
  Roll <- 2
  if(Roll==7){
    #insert code here to save (in Earning[i]) the amount earned if Roll = 7
  }
  #insert "else if" and "else" statements here to record earnings for other possible outcomes of Roll
}
```

Summarize the results from running the code above:

Should you keep playing the game?

What was the average payout or loss per game?

What were your total winnings over the course of 10,000 iterations?