

HW6__IS457__33

Do not remove any of the comments. These are marked by #

HW 6 - Due Wednesday Oct 31, 2018 in moodle and hardcopy in class.

(1). Please upload R code and report to Moodle with filename: HW7__IS457__YourCourseID. (2). Turn in hard copy of your report in class.

Please ensure that no identifying information (other than your class ID) is on your paper copy.

Part 1: Unfair Dice Simulation

A die is not necessarily fair, in which case the probabilities for 6 sides are different. We will look at a way to simulate unfair dice rolls in R

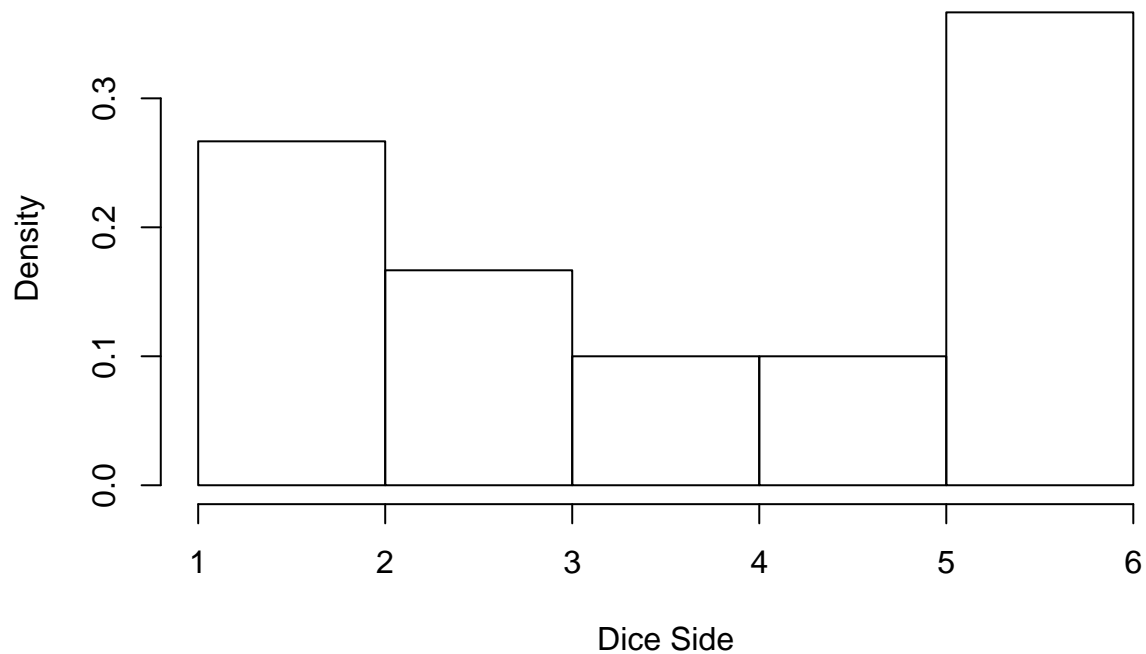
(1) Draw independently from a 6-side die with probability $2/7$ for a six and $1/7$ for others 30 times,

and save your result in a vector called roll1, make a histogram for the empirical density. (2 pt)

Your code here

```
roll1<-sample(1:6, 30, replace=TRUE, prob = c(1/7,1/7,1/7,1/7,1/7,2/7))
hist(roll1, freq = FALSE, xlab = "Dice Side", main = "Histogram of First Dice Experiment")
```

Histogram of First Dice Experiment



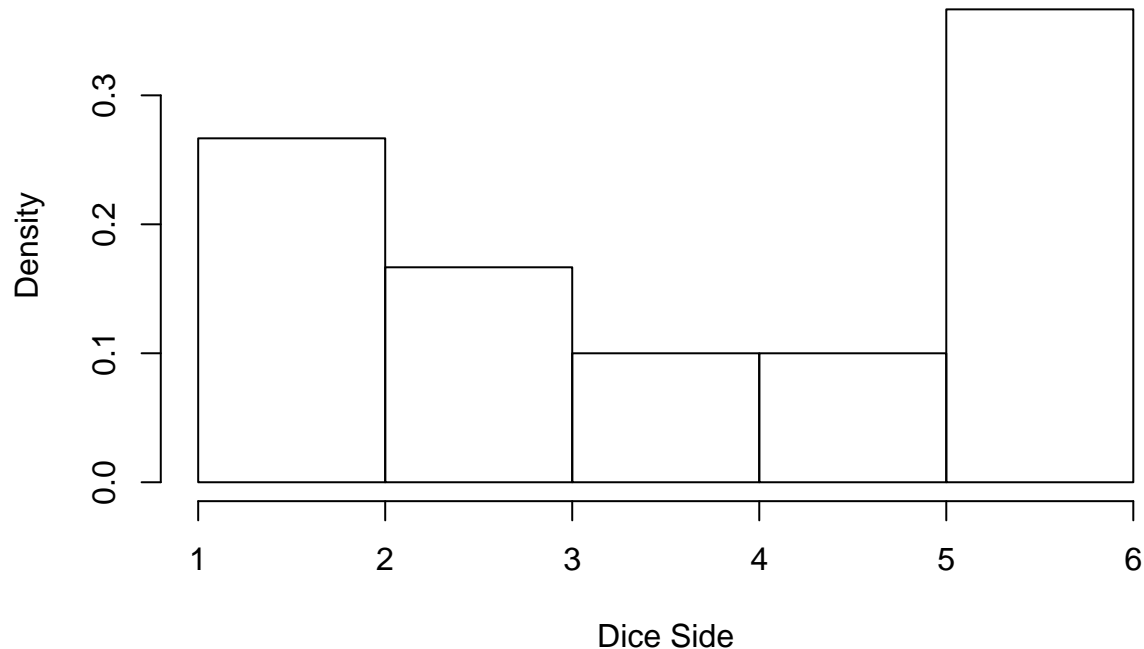
(2) Now, draw independently from a 6-side die with probability $2/7$ for a six and $1/7$ for others 3000 times,

and save your result in vector `roll2`, make a histogram for the empirical density. (2 pt)

Your code here

```
roll2 <- sample(1:6, 3000, replace=TRUE, prob = c(1/7,1/7,1/7,1/7,1/7,2/7))  
hist(roll2, freq = FALSE, xlab = "Dice Side", main = "Histogram of Second Dice Experiment")
```

Histogram of Second Dice Experiment



(3) What do you conclude from comparing these two plots? (2 pts)

Your answer here

From observing the two histograms, I see that even though the trial has been repeated a lot more times, we see the same density due to a consistent “loading” of the “die”.

Part 2: Monte Carlo Simulation

We will use the simulation techniques (Monte Carlo) introduced in class to generate confidence intervals for our estimates of the distribution mean

(1) As we will generate random numbers, to ensure reproducibility, please set the seed as 457.(1 pt)

NOTE: make sure you run the seed command EVERY time you sample something

Your code here

```
set.seed(457)
```

(2) For this simulation problem, we will sample data from the binomial distribution with parameters n and p .

First, we will estimate an individual experiment.

(a) Generate 100 observations of test data from the binomial distribution, with 20 trials and 0.8 probability and name it `test_sample`. (1 pt)

Your code here

```
test_sample <- rbinom(100, 20, 0.8)
```

(b) What is your estimate of the mean for the test data? call your estimate `X_hat`. What is the exact mean (use the formula to calculate mean

for a binomial dist)? are they close? what does this say about our random generation?(4 pts)

Your code here

```
X_hat <- mean(test_sample)
20*0.8
```

```
## [1] 16
```

Your answer

The estimated mean and the actual mean are close. This means our random generation does not completely accurately display the binomial distribution.

(c) What is the 95% confidence interval for `X_hat`? (2 pts)

Your code here

```
X_hat+1.96*sd(test_sample)
```

```
## [1] 19.68722
```

```
X_hat-1.96*sd(test_sample)
```

```
## [1] 12.79278
```

(3) Now use simulation technique to estimate the distribution of \hat{X} and create confidence intervals for it.

(a) Form a set of \hat{X} 's by repeating $B = 1000$ times the individual experiment. (2 pts)

HINT: You may want to create a matrix to save those values.

Your code here

```
set.seed(457)
x_hat_experiment <- matrix(replicate(1000, rbinom(100, 20, 0.8)), nrow = 1000, ncol = 100, byrow = TRUE)
```

(b) Get an estimate for the mean of the \hat{X} 's for each experiment in (3)(a) and save it to a vector $\hat{X}_{\text{estimate}}$ (length B vector). (1 pt)

Your code here

```
x_hat_estimate <- apply(x_hat_experiment, 1, FUN=mean)
```

(c) Now use $\hat{X}_{\text{estimate}}$ to create a “sampling distribution” for \hat{X} , and create a histogram to show the

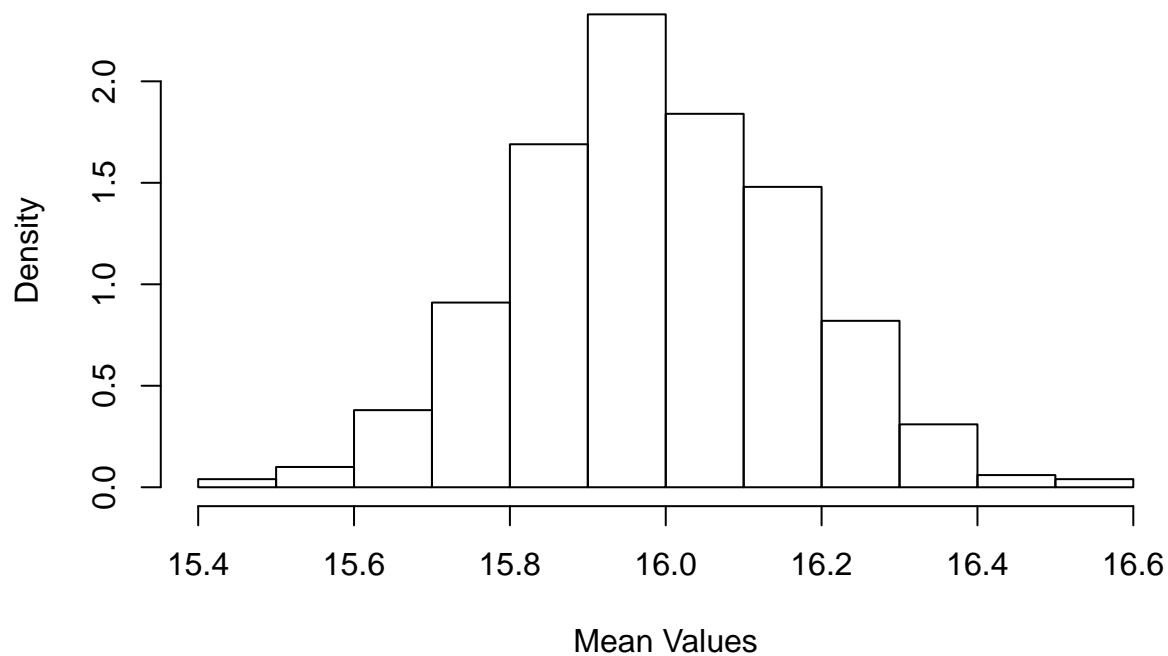
distribution. Does the distribution look normal (what are the essential elements of normal dist)? how can you tell?

if yes, what does it say about our random generation? (4 pts)

Your code here

```
hist(x_hat_estimate, freq = FALSE, xlab = "Mean Values", main = "Histogram of Mean Estimates from 1000 I")
```

Histogram of Mean Estimates from 1000 Repeated Experiments



Your answer here

The histogram of the mean values of the distributions appears to be a normal distribution. This shows that our random generation is pretty consistent in generating sample distributions with similar values that are closely related to each other.

(d) Now as we have a simulated sampling distribution of \hat{X} , we could empirically calculate the standard error using the

$\hat{X}_{\text{estimate}}$. What is your 95% confidence interval?(2 pts)

Notice here the standard error is indeed the standard deviation

Your code here

```
mean(x_hat_estimate)+1.96*sd(x_hat_estimate)
```

```
## [1] 16.34426
```

```
mean(x_hat_estimate)-1.96*sd(x_hat_estimate)
```

```
## [1] 15.64166
```

(4) We made some decisions when we used the simulation above that we can now question.

Repeat the above creation of a confidence interval in (3) for a range of settings

(we had our sample size fixed at 100) and a range of B values (we had B fixed at 1000). Suppose the sample size varies (100, 200, 300, . . . , 1000) and B varies (1000, 2000, . . . , 10000). You will likely find it useful to write functions to carry out these calculations. Your final output should be upper and lower pairs for the confidence intervals produced using the bootstrap method for each value of sample size and B.

(a) Generalize (3) into a function, and vary inputs of sample size and B as we did above. (5 pts)

Your code here

```
custom_CI_function <- function(samplesize, B){
  set.seed(457)
  experiment<-matrix(replicate(B, rbinom(samplesize, 20, 0.8)), nrow = B, ncol = samplesize, byrow = TRUE)
  experiment_means<-apply(experiment, 1, mean)
  CI<-c(mean(experiment_means)+1.96*sd(experiment_means), mean(experiment_means)-1.96*sd(experiment_means))
  return(CI)
}
custom_CI_function(100, 1000)
```

```
## [1] 16.34426 15.64166 15.99296
```

(5) Use the function `errbar()` in `Hmisc` package.

Plot your confidence interval limits to compare the effect of changing the sample size and changing the number of simulation replications B (10 pts). What do you conclude? (4 pts)

Your code here

```
library(Hmisc)

## Loading required package: lattice
## Loading required package: survival
## Loading required package: Formula
## Loading required package: ggplot2
## Warning: package 'ggplot2' was built under R version 3.4.4
##
## Attaching package: 'Hmisc'
##
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```

sampsiz<-c(100,200,300,400,500,600,700,800,900,1000)
B<-c(1000,2000,3000,4000,5000,6000,7000,8000,9000,10000)

CIB<-list()
CISS<-list()
uboundb<-c()
lboundb<-c()
meanb<-c()
ubounds<-c()
lbounds<-c()
means<-c()

for(i in 1:length(B)){
  CIB[[i]]<-custom_CI_function(100, B[i])
}
for(j in 1:length(sampsiz)){
  CISS[[j]]<-custom_CI_function(sampsiz[j],1000)
}

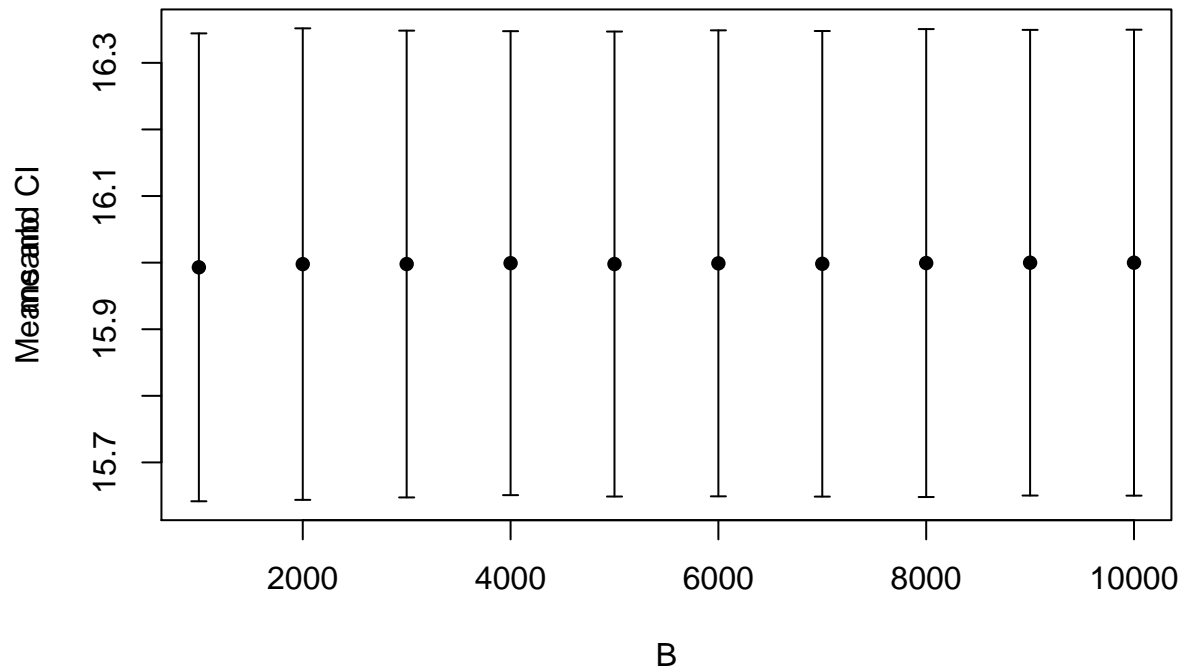
for(k in 1:length(CIB)){
  uboundb[k]<-CIB[[k]][1]
  lboundb[k]<-CIB[[k]][2]
  meanb[k]<-CIB[[k]][3]
}

for(l in 1:length(CISS)){
  ubounds[l]<-CISS[[l]][1]
  lbounds[l]<-CISS[[l]][2]
  means[l]<-CISS[[l]][3]
}

errbar(x=B, y=meanb, yplus=uboundb, yminus = lboundb)
title(main = "Changing B CI Plot", ylab="Means and CI")

```


Changing B CI Plot



```
errbar(x=sampsize, y=means, yplus=ubounds, yminus = lbounds)
title(main= "Changing Sample Size CI Plot", ylab = "Means and CI")
```

Changing Sample Size CI Plot

