HW6_IS457_85

Mon Oct 31, 2018

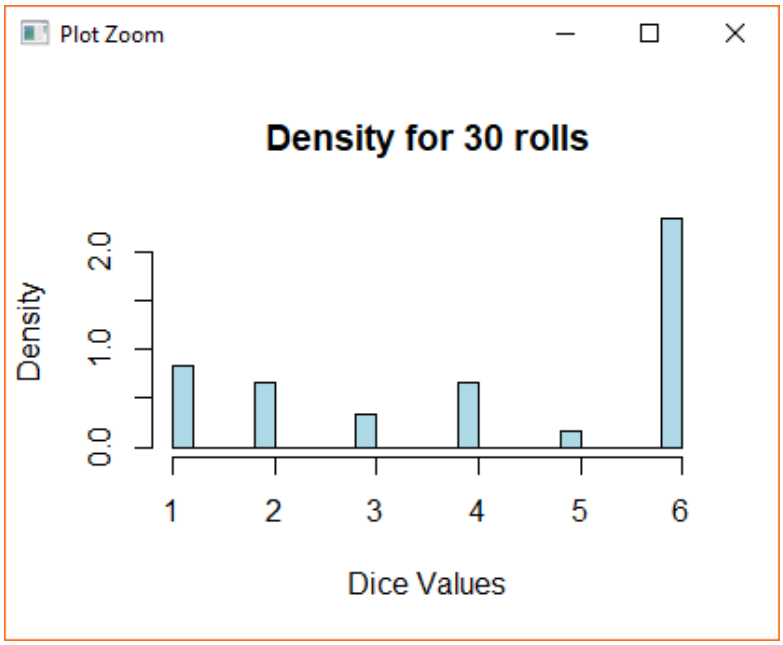
Part 1: Unfair Dice Simulation

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)

Ans :

```
set.seed(457)

roll1 = sample(1:6,30, replace = TRUE, prob = c(rep(1/7,5),2/7))

head(roll1,10)

hist(roll1, freq = FALSE, breaks = 18, col = "lightblue", xlab = "Dice Values", main = "Density for 30 rolls")
```

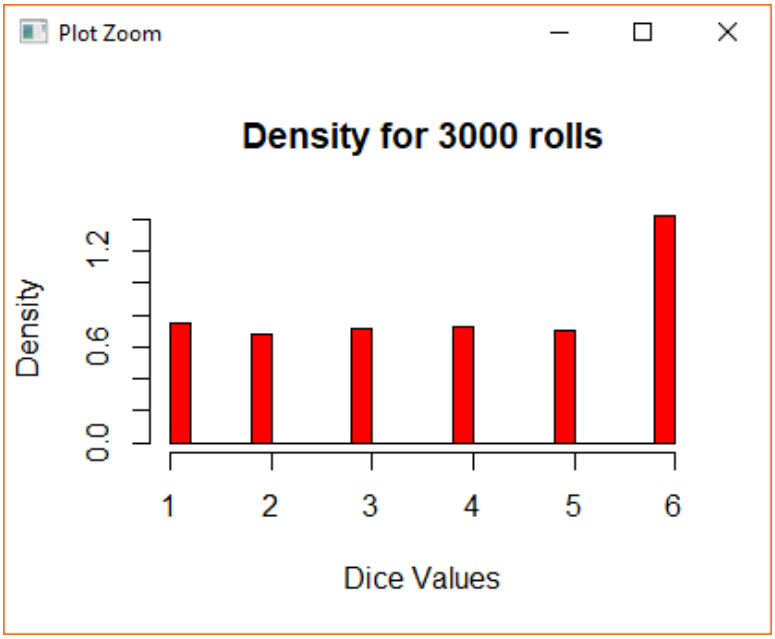


(2) Now, draw independently from a 6-side die with probability 2/7 for a six and 1/7 for others 3000 times, and save your result in vector roll2, make a histogram for the empirical density. (2 pt)

Ans :

```
set.seed(457)

roll2 = sample(1:6,3000, replace = TRUE, prob = c(rep(1/7,5),2/7))

head(roll2, 10)
```

hist(roll2, freq = FALSE, breaks = 18, col = "red", xlab = "Dice Values", main = "Density for 3000 rolls")



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

Ans :

As we can see from the above histograms as we increase the number the number of trails the probability for each element approaches their theoretical value. We can see from the second graph that the bias for the sixth value in clearly evident.

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.

Ans :

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)

Ans :

test_sample = rbinom(100, 20, 0.8)

head(test_sample,10)

```
 [1] 17 19 17 17 17 16 14 19 18 13
```

(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)

Ans :

x_hat = mean(test_sample)

n = 20

p = 0.8

binom_mean = n*p

x_hat

```
[1] 16.24
```

binom_mean

```
[1] 16
```

The mean value for the binomial distribution is given by the formula m = n*p and it comes out to be 16 in our case. The mean value for our sample is 16.24 which is close to the theoretical value. The shows that our random generation process to be unbiased.

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

Ans :

x_hat_error = sd(test_sample)/sqrt(20)

ci_low = x_hat - 1.96 * x_hat_error

ci_high = x_hat + 1.96* x_hat_error

ci = c(ci_low,ci_high)

x_hat_error

```
[1] 0.4407237
```

ci

```
[1] 15.37618 17.10382
```

(3) Now use simulation technique to estimate the distribution of X_hat and create confidence intervals for it.

(a) Form a set of X_hat's by repeating B = 1000 times the individual experiment. (2 pts)
HINT: You may want to create a matrix to save those values.

Ans :

```
x_hat_dis = replicate(1000, rbinom(100,20,0.8),set.seed(457))

x_hat_mat = matrix(x_hat_dis, nrow =100, ncol = 1000)
```

(b) Get an estimate for the mean of the X_hat's for each experiment in (3)(a) and save it to a vector X_hat_estimate (length B vector).(1 pt)

Ans :

```
x_hat_estimate = apply(x_hat_mat, 2 , mean)

length(x_hat_estimate)
```

```
[1] 1000
```

```
head(x_hat_estimate,10)
```

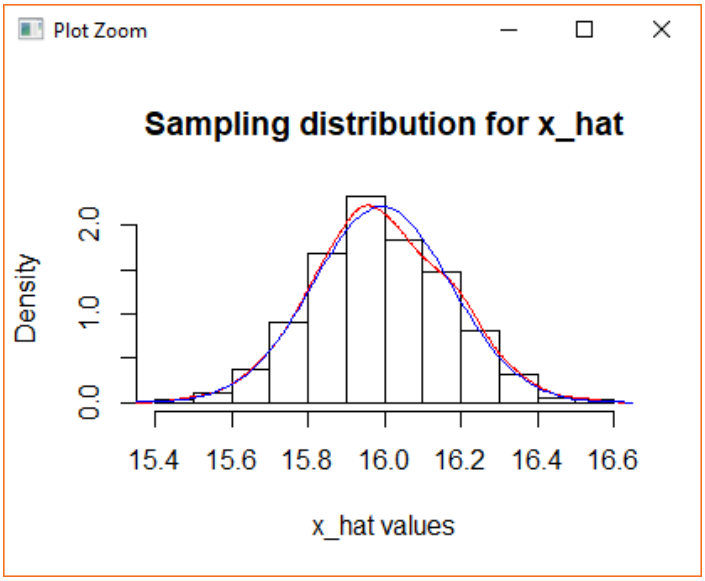
```
[1] 16.24 16.01 15.88 15.88 16.10 16.25 15.83 16.17
[9] 16.11 16.10
```

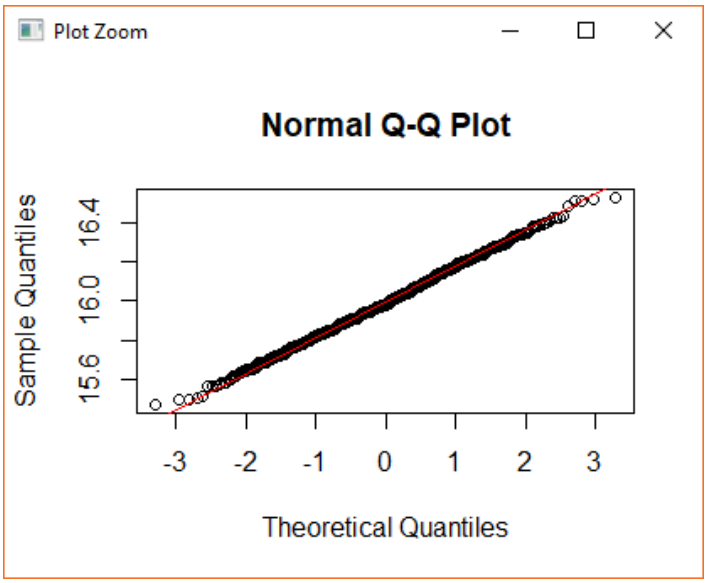
(c) Now use X_hat_estimate to create a "sampling distribution" for X_hat, and create a histogram to show the distribution. Does the distribution look normal (what are the essential elements of normal dist)? how can you tell? if yes, what does it say about our random generation? (4 pts)

Ans :

```
hist(x_hat_estimate, freq = FALSE, xlab = "x_hat values", main = "Sampling distribution for x_hat")

lines(density(x_hat_estimate), col = 'Red')

curve(dnorm(x,mean = mean(x_hat_estimate), sd = sd(x_hat_estimate)), from = 15, to = 17, add = TRUE, col = "blue")

qqnorm(x_hat_estimate)

qqline(x_hat_estimate ,col = "red")

shapiro.test(x_hat_estimate)
```

```
Shapiro-Wilk normality test

data:  x_hat_estimate
W = 0.9983, p-value = 0.4301
```





As we can see from the above graphs our sampling distribution does look like a normal distribution. It also has a high p-value i.e. > 0.05 on the Shapiro – Wilk normality test which implies it to have follow normal distribution. As per the central limit theorem the sampling distribution of the sample having more than 30 element follow a normal distribution. As our sampling distribution follows the same we can say that the sampling process is unbiased.

(d) Now as we have a simulated sampling distribution of X_hat, we could empirically calculate the standard error using the X_hat_estimate. What is your 95% confidence interval?(2 pts)
Notice here the standard error is indeed the standard deviation

Ans :

x_hat_estimate_error = sd(x_hat_estimate)

ci_low_estimate = mean(x_hat_estimate) - 1.96 * x_hat_estimate_error

ci_high_estimate = mean(x_hat_estimate) + 1.96 * x_hat_estimate_error

```
ci_estimate= c(ci_low_estimate,ci_high_estimate)

ci_estimate
```

```
[1] 15.64166 16.34426
```

(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)

Ans :

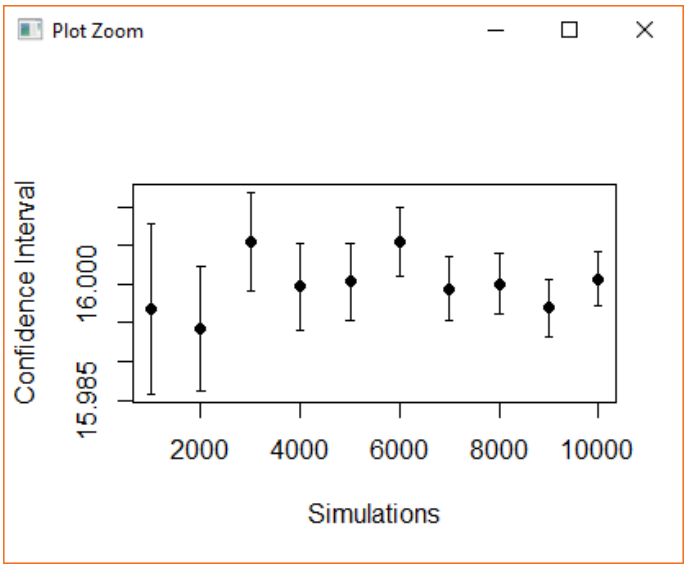
```
b_vector = c()

CI_function <- function(n, B){
 for(i in 1:B){
   b_vector[i] = mean(rbinom(n,20,0.8))
 }
 mean_B = mean(b_vector)
 sd_B = sd(b_vector)
 SE = sd_B/sqrt(B)
 ci_low_B = mean_B - 1.96*SE
 ci_high_B = mean_B + 1.96*SE
 ci_B = c(ci_low_B,ci_high_B)
 return(ci_B)
}

## n = 100 constant, b varies

b_values = seq(1000,10000, by = 1000)

ci_values_b = list()

for(i in 1:length(b_values)){
 ci_values_b[[i]] = CI_function(100, b_values[i])
}

ci_low_values_b = c()

ci_high_values_b = c()

mean_values_b = c()

for(i in 1:length(b_values)){
 ci_low_values_b[i] = c(ci_values_b[[i]][1])
 ci_high_values_b[i] = c(ci_values_b[[i]][2])
 mean_values_b[i] = c(mean(ci_values_b[[i]]))
}
```

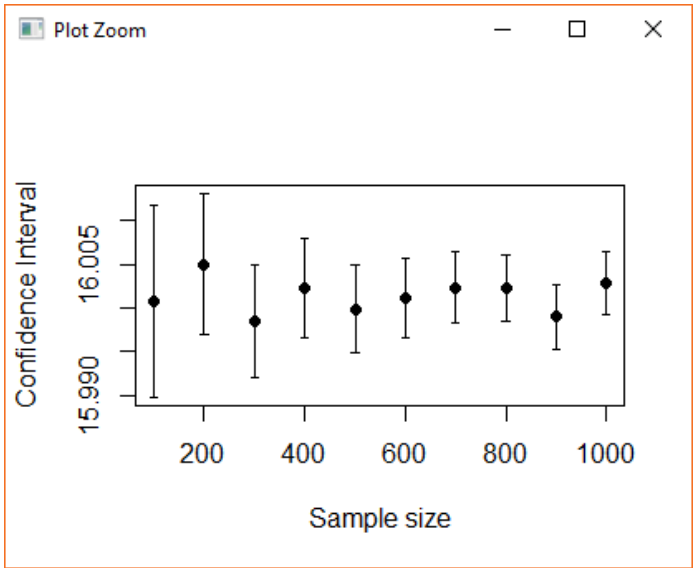
```
## n varies, b = 1000 constant

n_values = seq(100, 1000, by = 100)

ci_values_n = list()

for(i in 1:length(b_values)){
  ci_values_n[[i]] = CI_function(n_values[i], 1000)
}

ci_low_values_n = c()

ci_high_values_n = c()

mean_values_n = c()

for(i in 1:length(n_values)){
  ci_low_values_n[i] = c(ci_values_n[[i]][1])
  ci_high_values_n[i] = c(ci_values_n[[i]][2])
  mean_values_n[i] = c(mean(ci_values_n[[i]]))
}
```

(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)

Ans :

```
library(Hmisc)

errbar(b_values,mean_values_b,ci_high_values_b,ci_low_values_b,main = "Confidence Interval",
      xlab = "Simulations", ylab = "Confidence Interval")

errbar(n_values,mean_values_n,ci_high_values_n,ci_low_values_n,main = "Confidence Interval",
      xlab = "Sample size", ylab = "Confidence Interval")
```





As we can see from the above graphs the variation in the mean values is larger when we change the no. of simulations and becomes stable as the number goes on increasing. When we change the sample size, the mean is concentrated around the centre with less variation among mean values. The mean value is larger when we vary the simulations than the median values for change in sample size. The confidence interval goes on decreasing and converges to the true parameter value which in this case is the mean of the binomial distribution as the samples size as well as the simulation number increases.