Group 12 Lab 5

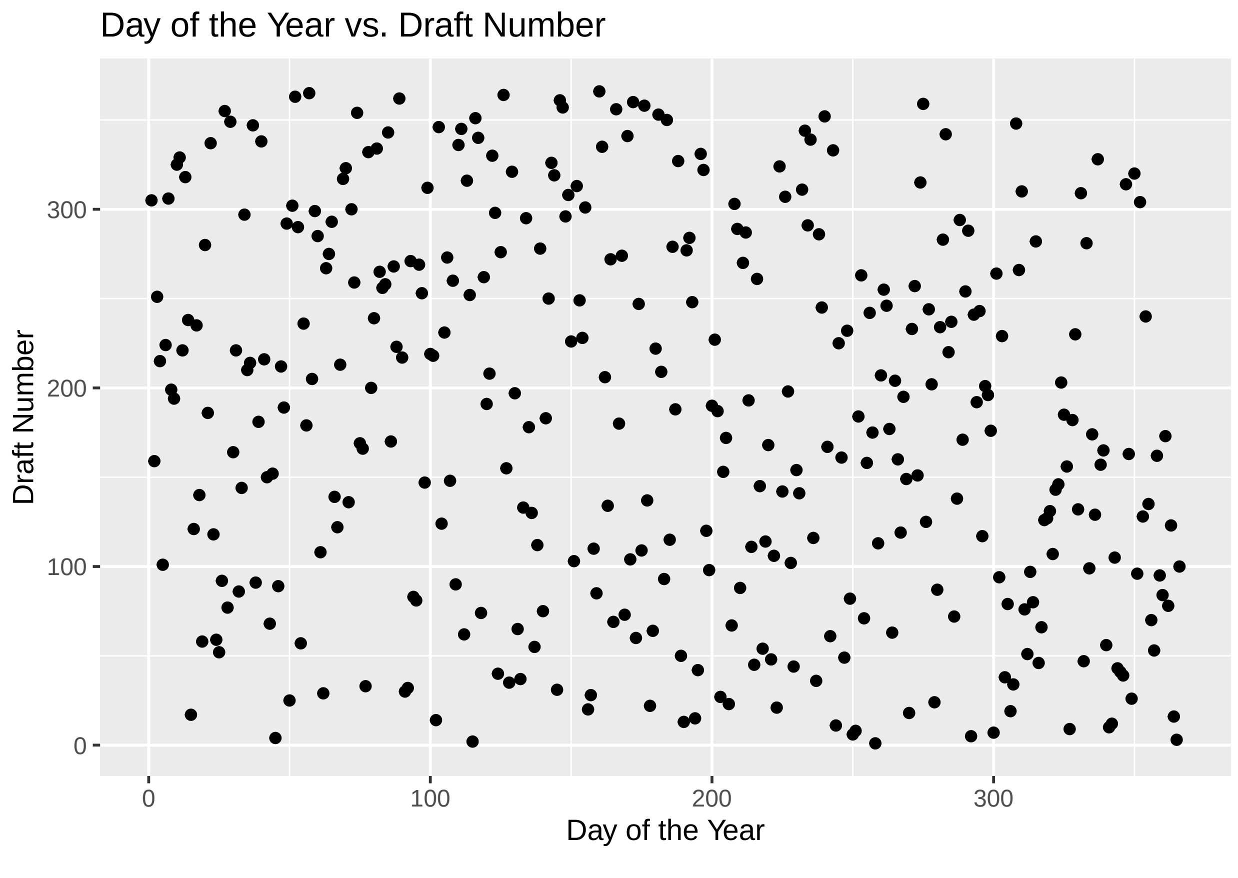
varsi146, aswma317

2022-12-23

Question 1: Hypothesis Testing

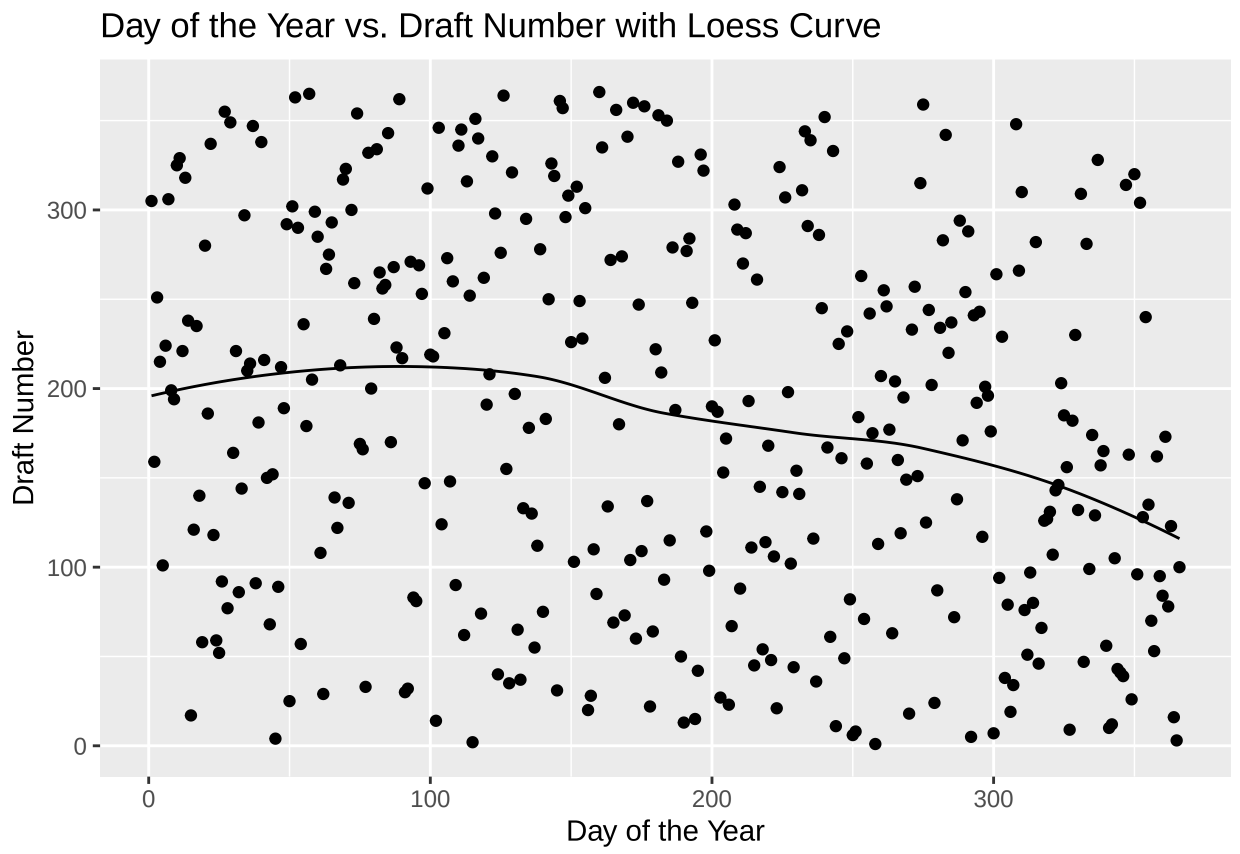
Sub Question 1:

Based on the scatter-plot of the response variable(Draft_No) vs. predictor variable(Day_of_Year), we can say that the data looks random and that the draft numbers are randomly selected from the days of the year without any bias or unfairness.



Sub Question 2:

We now fit a loess smoother to compute an estimate $\hat{Y}$ as a function of X and on putting a curve $\hat{Y}$ vs. X we can see from the scatter-plot below that the lottery looks random, however, based on the curve, we can say that the relationship between X(Day of Year) and $\hat{Y}$ changes towards the end of the year in the sense that, there is a chance that those born towards the end of the year are less likely to be drafted.



Sub Question 3:

We now compute a test statistic T, that is,

$$T = \frac{\hat{Y}(X_b) - \hat{Y}(X_a)}{X_b - X_a}$$

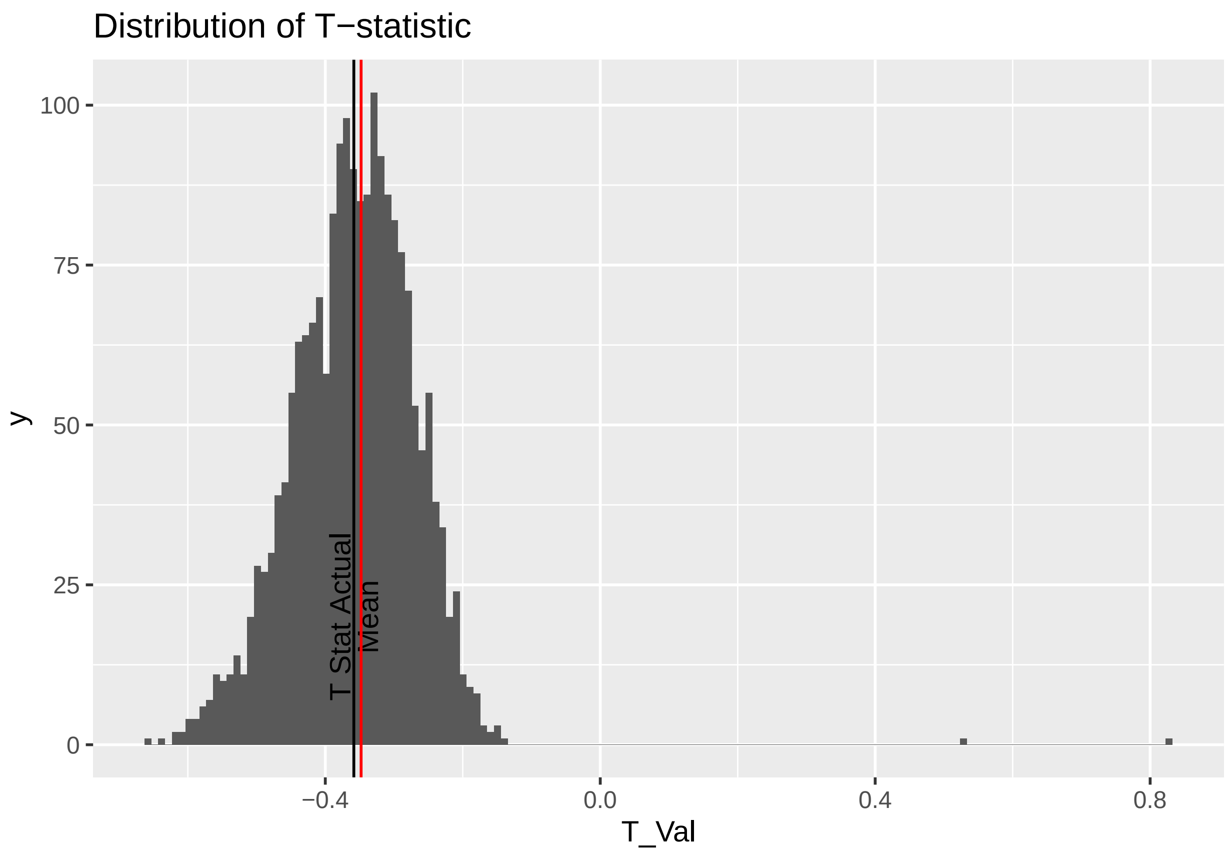
where,

$$X_b = argmax_X \hat{Y}(X), X_a = argmin_X \hat{Y}(X)$$

We can see below the T statistic value for the data, and based on the value of T statistic being significantly different from 0, we can say that there is a trend(based on the loess curve) that the data is not random.

```
## The T-statistic value for the data is  -0.3479163
```

We now try to estimate the distribution of T by using a non-parametric bootstrap with B = 2000 and below we see the distribution of T:



The p-quantile for the test at $T = 0$ can be shown as:

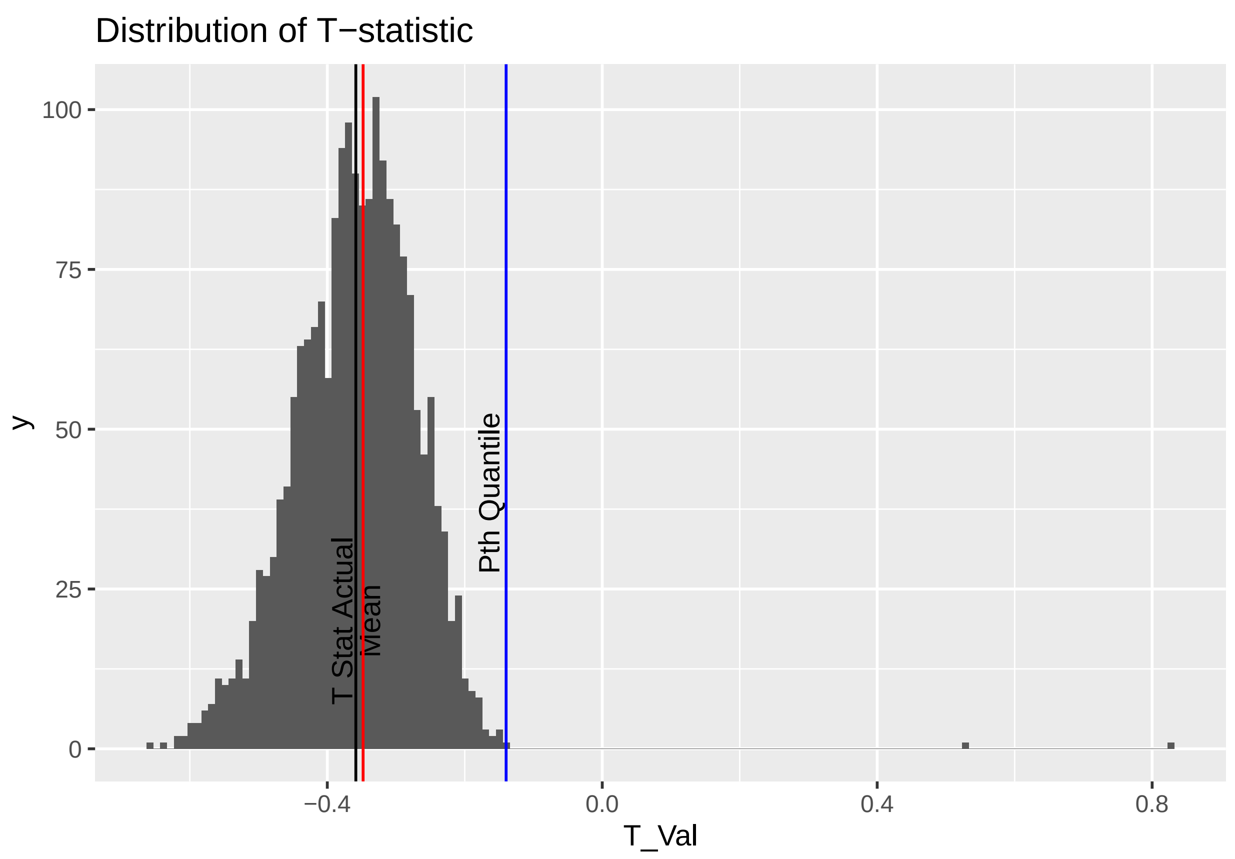
```
## The p-quantile at T=0 using the quantile function is estimated to be 99.9%
```

We see that, the p-quantile at $T \approx 0$ is 99.9%.

Alternatively, we can determine the p-quantile in the following method as well:

```
p_quantile <- (sum(boot_res$t < 0)/length(boot_res$t))*100
cat('The p-quantile at T = 0 can be calculated in an alternative method
    and estimated to be', p_quantile)
```

```
## The p-quantile at T = 0 can be calculated in an alternative method
##     and estimated to be 99.9
```



Therefore, we can conclude that the p^{th} quantile at $T = 0$ is 99.9%.

We can conclude that the data is non-random since the actual T-statistic value is significantly different from 0 as we see from the plot above.

Sub Question 4:

We are now asked to implement a function depending on data and B that tests the hypothesis:

H_0: Lottery is random

versus

H_1: Lottery is non-random

by using a permutation test with statistics T and return the p-value of this test.

Upon performing a permutation test for the given dataset and for B = 2000, we obtain the p-value as:

```
## The p-value of this test is 0.1445
```

Sub Question 5:

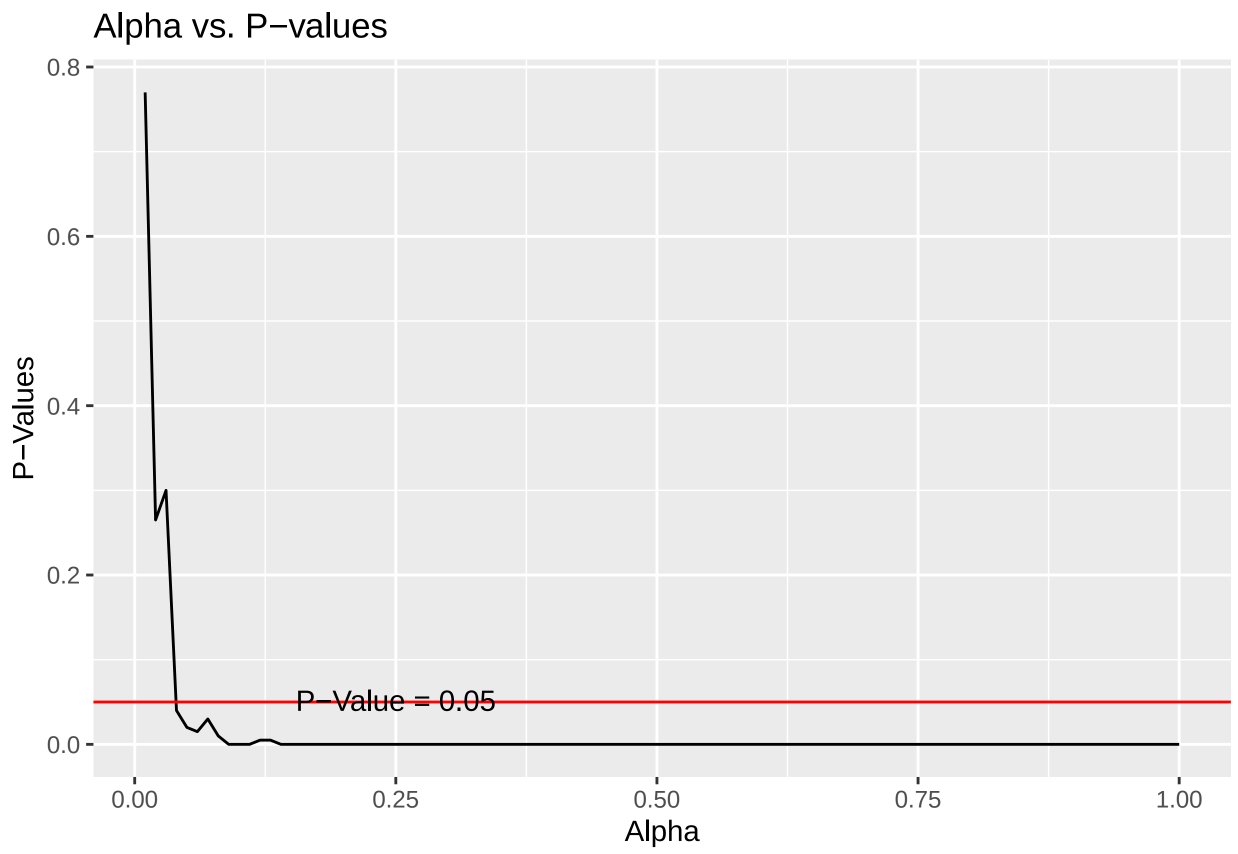
a). We are asked to generate a dataset with n = 366, using same X, as in original data set and $Y(x) = max(0, min(\alpha x + \beta, 366))$, where $\alpha = 0.01$ and $\beta \sim \mathcal{N}(183, sd = 10)$

b). We then plug the data generated from the procedure above into the permutation test function we have constructed and obtain a p-value as shown below:

```
##   Alpha P_Val
## 1  0.01 0.405
```

Since the p-value above is greater than 0.05, the test is deemed statistically "not" significant and we do not reject the null hypothesis that says that the data for the lottery draft is random.

c). We then repeat steps 5a-5b, but for $\alpha = 0.01, 0.02, ..., 1$ and below we see a plot for p-values vs. α and note that given that we are generating non-random data-sets, most of the datasets when passed into the permutation test return a p-value less than 0.05 we reject the corresponding datasets.



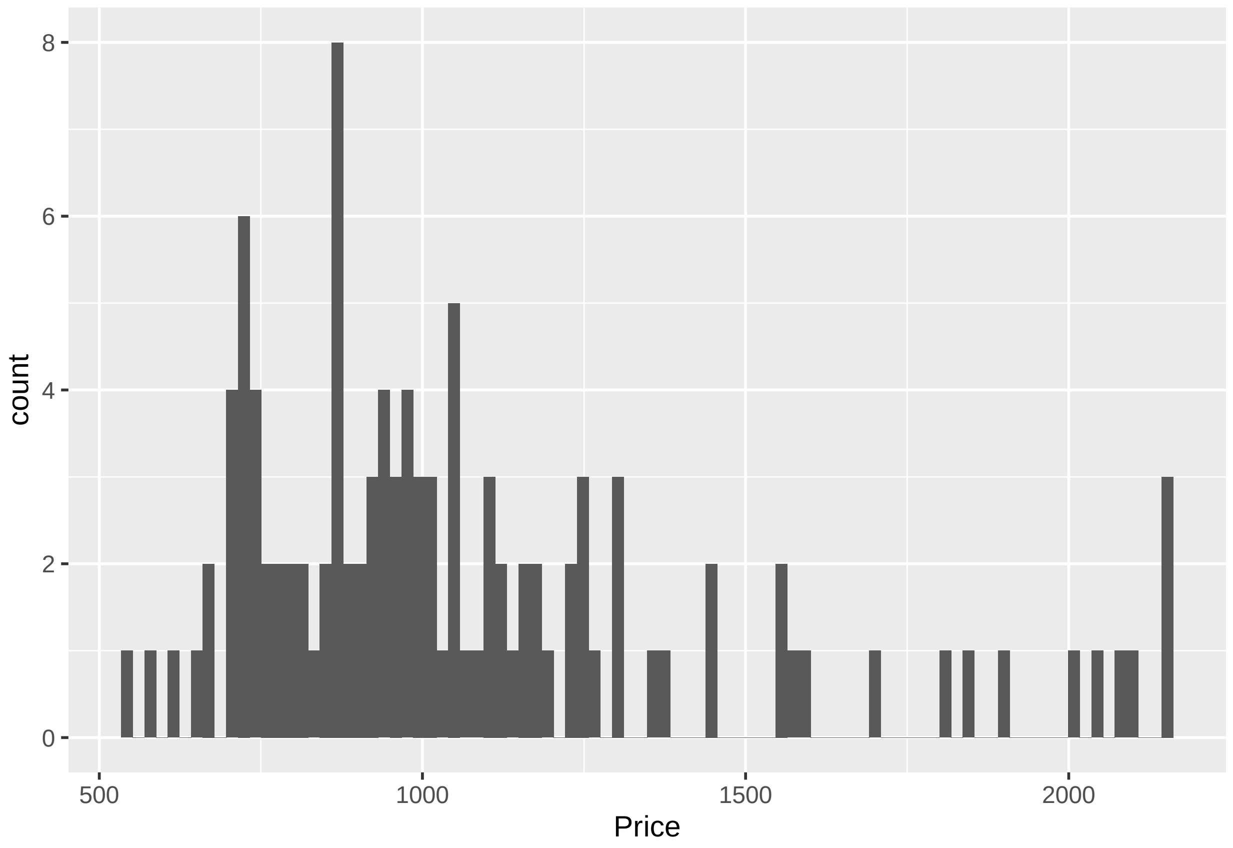
We can say that the quality of the test statistic is very good given that it is able to reject more than 95% of the given datasets that is generated under the ***non-random*** model, meaning that the datasets generated are in fact non-random and that the test statistic is able to reject almost 95% of them.

```
## The power of the test is calculated to be  0.97
```

Question 2: Bootstrap, Jackknife and Confidence Interval

Sub Question 1:

Part a. Plot the histogram of prices



Part b. Does it remind of a conventional distribution?

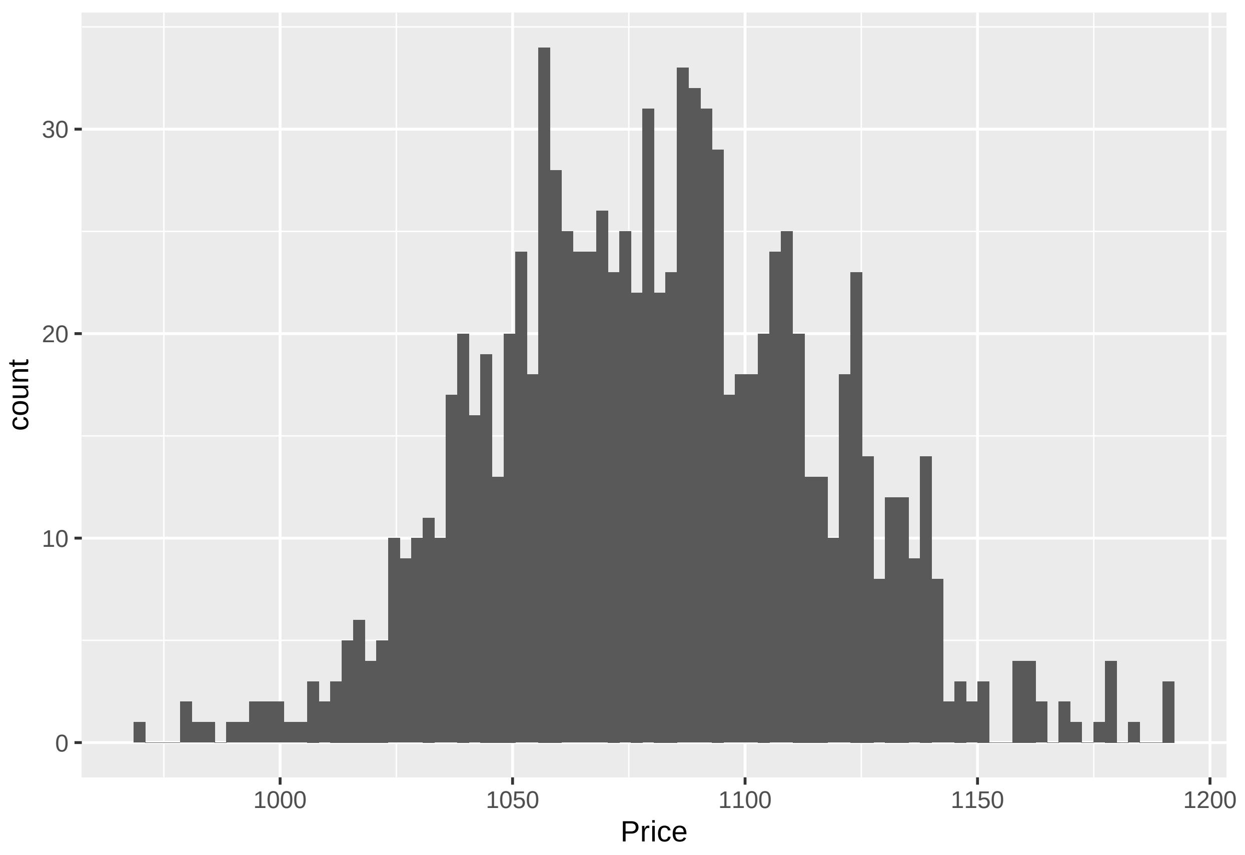
It almost looks like a Gamma Distribution except at the right tail. At the right tail, there are a higher frequency of high priced houses. Excluding this bit, the rest can be considered as part of the gamma distribution.

Part c. Compute the mean price

```
## Mean price of the house: 1080.473
```

Sub Question 2:

Part a. Distribution of mean price using bootstrap



```
## Mean house price of bootstrapped data: 1080.853
```

As seen, the mean price hasn't changed much with the bootstrapping process. But the distribution has changed. It has become more of a normal distribution. I think this is expected as this follows the Central Limit Theorem.

Part b. Bootstrap bias-correction and variance of the mean

```
## Bootstrap bias-correction: 1080.092

##
## Bootstrap variance of mean: 1272.836
```

Part c. 95% CI for the mean price using bootstrap percentile, bootstrap BCa, and first order normal approximation

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_res, type = c("perc", "bca", "norm"))
##
## Intervals :
## Level      Normal             Percentile            BCa
## 95%   (1010, 1150 )   (1014, 1150 )   (1016, 1160 )
## Calculations and Intervals on Original Scale
```

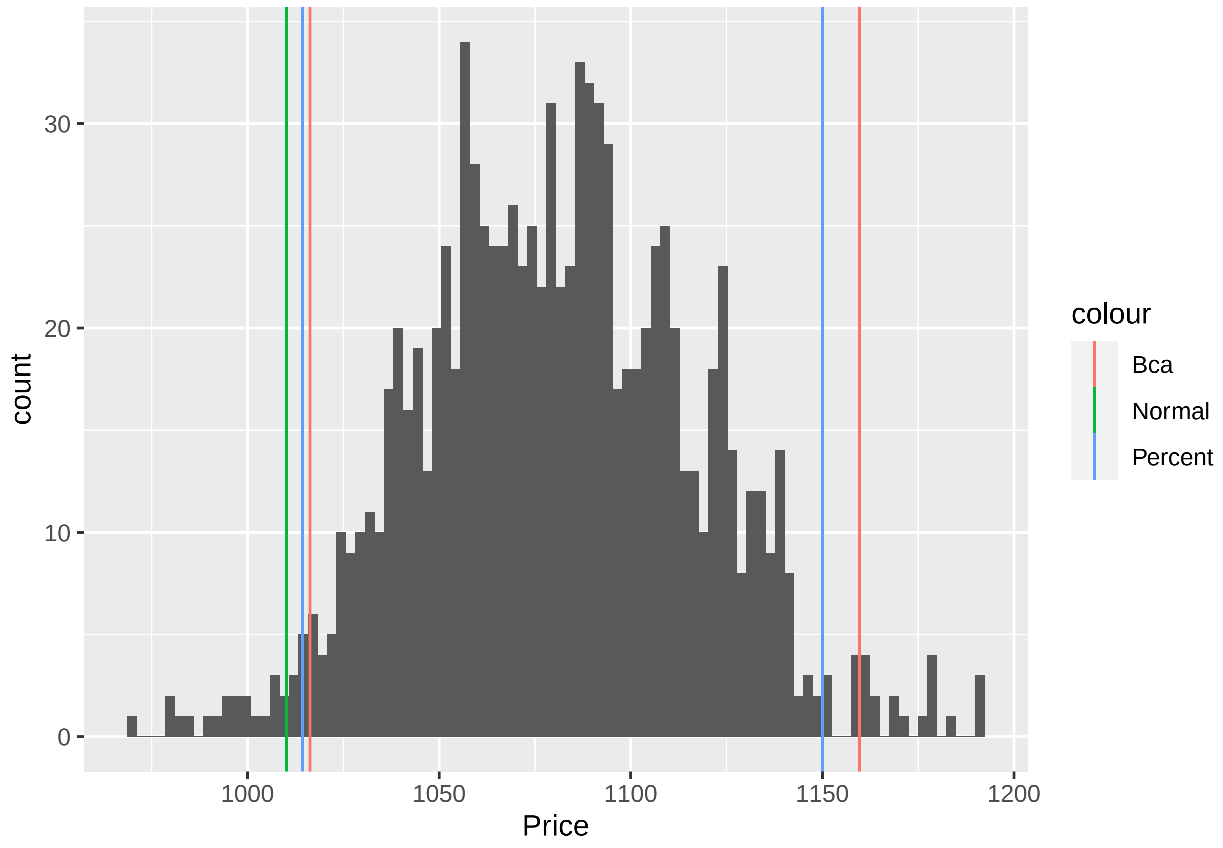
Sub Question 3:

Part a. Variance of the mean using jackknife and compare it with bootstrap estimate

```
## Variance of the mean price using Jackknife:  1320.911
```

The variance as compared the bootstrap is a bit higher. It could be interpreted that the variance is less in bootstrap as it uses a lot more samples than jackknife, and also could be more precise.

Sub Question 4: Compare the confidence intervals obtained with respect to their length and the location of the estimated mean in these intervals.



There are 3 CIs in the above graph: CIs calculated using Normal approximation, Bootstrap Percentile(Percentile) and Adjusted Bootstrap Percentile(BCa). All 3 considers 95% CIs.

Normal Vs Percentile: The upper bound for both are the same, however, the lower bound for Normal is slightly less than the Percentile one. This maybe because the 2.5 percentile may be after the lower-bound of the Normal CI. Normal CIs are calculated using the mean and standard deviation, while Bootstrap Percentile CIs are calculated based on the percentile values.

BCa vs Percentile: The BCa accounts for the bias and corrects the CIs and this is the reason why we see the BCa interval slighly moved to the right in comparison to the Percentile CIs. As we saw in the initial graph of house prices, we saw that there is indeed a skewness in house prices; there were a high frequency of high priced houses. The BCa factors this in and that's the reason for the slight move to the right.

Appendix

```r
library(ggplot2)
library(boot)
library(dplyr)
knitr::opts_chunk$set(echo = TRUE)
lottery_data <- read.csv(file = 'lottery.csv', header = TRUE, sep = ';')

# Scatter Plot of Y vs. X

plot1 <- ggplot(lottery_data, aes(x = Day_of_year, y = Draft_No)) +
  geom_point()+
  ggtitle(label = 'Day of the Year vs. Draft Number')+
  xlab(label = 'Day of the Year')+
  ylab(label = 'Draft Number')
plot1
# Fitting a loess smoother to compute y_hat
loess_smoother <- loess(Draft_No ~ Day_of_year, lottery_data)
y_hat <- predict(loess_smoother, newdata = lottery_data$Day_of_year)
lottery_data$Loess_Y_Hat <- y_hat
ggplot(lottery_data, aes(x = Day_of_year, y = Draft_No)) +
  geom_point() +
  geom_line(aes(y = Loess_Y_Hat))+
  ggtitle(label = 'Day of the Year vs. Draft Number with Loess Curve')+
  xlab(label = 'Day of the Year')+
  ylab(label = 'Draft Number')
# T-statistic function:

boot_fun <- function(data, index){
  data <- data[index,]#Create bootstrapped data
  loess_smoother <- loess(Draft_No ~ Day_of_year, data)
  y_hat <- loess_smoother$fitted
  x_b <- data$Day_of_year[which.max(y_hat)]
  x_a <- data$Day_of_year[which.min(y_hat)]
  y_hat_b <- max(y_hat)
  y_hat_a <- min(y_hat)
  t <- (y_hat_b - y_hat_a)/(x_b - x_a)
  return(t)
}
set.seed(12345)
t_actual <- boot_fun(data = lottery_data)
cat('The T-statistic value for the data is ',t_actual)
# Non parametric bootstrap:

B <- 2000
set.seed(12345)
boot_res <- boot(lottery_data, boot_fun, R = B)
plot_T <- ggplot(data.frame(T_Val = boot_res$t), aes(x = T_Val)) +
  geom_histogram(bins = 150) +
  geom_vline(xintercept = mean(boot_res$t))+
  annotate("text", x=(mean(boot_res$t)+0.02), y=20, label="Mean", angle = 90)+
  geom_vline(xintercept = t_actual, colour = 'red')+
  annotate("text", x=t_actual-0.031, y=20, label="T Stat Actual", angle = 90)+
  ggtitle(label = 'Distribution of T-statistic')
plot_T
quant <- quantile(boot_res$t, probs = seq(0.005, 1, 0.001))
p_quant <- names(quant[quant< 0 & quant> -0.14])
cat('The p-quantile at T=0 using the quantile function is estimated to be',
    p_quant)
p_quantile <- (sum(boot_res$t < 0)/length(boot_res$t))*100
cat('The p-quantile at T = 0 can be calculated in an alternative method
    and estimated to be', p_quantile)
plot_T+
  geom_vline(xintercept = quant[quant< 0 & quant> -0.14], colour = 'blue')+
  annotate("text", x=-0.165, y=40, label="Pth Quantile", angle = 90)+
  ggtitle(label = 'Distribution of T-statistic')
# Permutation test:

permute_fn <- function(data, B){
  t_stat <- c()
  perm_data <- data
  for (i in 1:B) {
    perm_data$Draft_No <- sample(perm_data$Draft_No,
                                 size = length(perm_data$Draft_No),
                                 replace = FALSE)
    t_stat <- c(t_stat, boot_fun(data = perm_data))
  }

  permute_p_val <- mean(abs(t_stat) >= abs(boot_fun(data = data)))
  return(permute_p_val)
}
permute_p_val <- permute_fn(data = lottery_data, B = 2000)
cat('The p-value of this test is', permute_p_val)
# Data frame to store p-values for given Alpha:

p_val_df <- as.data.frame(matrix(nrow = 0, ncol = 2))
colnames(p_val_df) <- c('Alpha', 'P_Val')

# For one single value of alpha:

alpha_v <- 0.01
p_val <- c()

for (alph in 1:length(alpha_v)) {
  x <- lottery_data$Day_of_year
  y <- sapply(x, function(x){
    bet <- rnorm(1, mean = 183, sd = 10)
    max(0, min((alpha_v[alph]*x + bet), 366))
  })
  new_data <- data.frame(Day_of_year = x, Draft_No = y)
  p_val <- c(p_val, permute_fn(data = new_data, B = 200))
  p_val_df <- rbind(p_val_df, data.frame('Alpha' = alpha_v[alph],
                                         'P_Val' = p_val[alph]))
}
p_val_df
p_val_df <- as.data.frame(matrix(nrow = 0, ncol = 2))
```

```r
colnames(p_val_df) <- c('Alpha', 'P_Val')
alpha_v <- seq(from = 0.01, to = 1, by = 0.01)
p_val <- c()

for (alph in 1:length(alpha_v)) {
  x <- lottery_data$Day_of_year
  y <- sapply(x, function(x){
    bet <- rnorm(1, mean = 183, sd = 10)
    max(0, min((alpha_v[alph]*x + bet), 366))
  })
  new_data <- data.frame(Day_of_year = x, Draft_No = y)
  p_val <- c(p_val, permute_fn(data = new_data, B = 200))
  p_val_df <- rbind(p_val_df, data.frame('Alpha' = alpha_v[alph],
                                         'P_Val' = p_val[alph]))
}
ggplot(data = p_val_df, aes(x = Alpha, y = P_Val)) +
  geom_line() +
  geom_hline(yintercept = 0.05, colour = 'red')+
  annotate("text", x=0.25, y=0.052, label="P-Value = 0.05", angle = 0)+
  ggtitle(label = 'Alpha vs. P-values')+
  xlab(label = 'Alpha')+
  ylab(label = 'P-Values')
p_val_df <- cbind(p_val_df,
                  data.frame('Accept/Reject' = ifelse(p_val_df$P_Val <= 0.05,
                                                      'Reject',
                                                      'Accept')))

power_test <- length(which(p_val_df[,3] == 'Reject'))/100

cat('The power of the test is calculated to be ', power_test)
data <- read.csv(file = 'prices1.csv', header = TRUE, sep = ';')
ggplot(data, aes(x = Price)) +
  geom_histogram(bins = 90)
mean_price <- mean(data$Price)
cat('Mean price of the house:', mean_price)
boot_fun <- function(Price, index){
  return(mean(Price[index]))
}
B <- 1000
set.seed(12345)
boot_res <- boot(data$Price, boot_fun, R = B)
boot_plot <- ggplot(data.frame(Price = boot_res$t), aes(x = Price)) +
  geom_histogram(bins = 90)
boot_plot
cat('Mean house price of bootstrapped data:', mean(boot_res$t))
#Determine the bootstrap bias-correction. See slide 21.
boot_bias_corr_t <- 2 * boot_res$t0 - mean(boot_res$t) #1079.812
cat('Bootstrap bias-correction:', boot_bias_corr_t)

#Determine the variance of the mean. See slide 19.
boot_var_t <- 1/(B-1) * sum((boot_res$t - mean(boot_res$t))^2)
cat('\nBootstrap variance of mean:', boot_var_t)
boot_ci <- boot.ci(boot_res, type = c('perc', 'bca', 'norm'))
```

```r
print(boot_ci)
T_i <- rep(0,nrow(data))
n <- nrow(data)
for (i in 1:nrow(data)) {
  T_i[i] <- (n * mean(data$Price)) - ((n-1) * mean(data$Price[-i]))
}
J_T <- (1/n) * sum(T_i)
jack_var_t <- (1/( n*(n-1) )) * sum((T_i - J_T)^2)
cat('Variance of the mean price using Jackknife: ',jack_var_t)
boot_plot +
  geom_vline(aes(xintercept = boot_ci$normal[1,2], color = 'Normal'))+
  geom_vline(aes(xintercept = boot_ci$normal[1,3], color = 'Normal'))+
  geom_vline(aes(xintercept = boot_ci$percent[1,4], color = 'Percent'))+
  geom_vline(aes(xintercept = boot_ci$percent[1,5], color = 'Percent'))+
  geom_vline(aes(xintercept = boot_ci$bca[1,4], color = 'Bca'))+
  geom_vline(aes(xintercept = boot_ci$bca[1,5], color = 'Bca'))
```