FinalProject

Group 3

Section 1 - Loading the data and Exploring the data

We have selected Bike dataset to perform exploration of the data. We start by loading the data into the R Markdown file

```
bikeData <- read_csv("Cleaned_bike_data.csv",show_col_types = FALSE)
```

```
summary(bikeData)
```

```
##   model_name          model_year     kms_driven         owner          
##  Length:5062        Min.   :1970   Min.   :      0   Length:5062       
##  Class :character   1st Qu.:2014   1st Qu.:   9782   Class :character  
##  Mode  :character   Median :2016   Median :  18000   Mode  :character  
##                     Mean   :2015   Mean   :  24079                     
##                     3rd Qu.:2018   3rd Qu.:  30708                     
##                     Max.   :2021   Max.   :1000000                     
##    location            mileage         power            price        
##  Length:5062        Min.   : 5.0   Min.   :  6.15   Min.   :      0  
##  Class :character   1st Qu.:35.0   1st Qu.: 14.00   1st Qu.:  44925  
##  Mode  :character   Median :38.0   Median : 19.80   Median :  80000  
##                     Mean   :41.8   Mean   : 21.93   Mean   : 115391  
##                     3rd Qu.:53.0   3rd Qu.: 24.60   3rd Qu.: 133375  
##                     Max.   :95.0   Max.   :197.30   Max.   :1900000
```

> We can see that there are different columns in the table which are Model_name, Model_year, Kms_driven, Owner, Location, Mileage, Power and Price.

To determine the different types of the datatypes, we can use the str command in R. This command tells us the different types of data which is present in the datasheet.

```
str(bikeData)
```

```
## spec_tbl_df [5,062 x 8] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ model_name: chr [1:5062] "Bajaj Avenger Cruise" "Royal Enfield
Classic" "Hyosung GT250R 2012" "KTM Duke" ...
##  $ model_year: num [1:5062] 2017 2016 2012 2012 2016 ...
##  $ kms_driven: num [1:5062] 17000 50000 14795 24561 19718 ...
##  $ owner     : chr [1:5062] "first owner" "first owner" "first owner"
"third owner" ...
##  $ location  : chr [1:5062] "hyderabad" "hyderabad" "hyderabad"
"bangalore" ...
##  $ mileage   : num [1:5062] 35 35 30 35 65 25 35 32 40 35 ...
##  $ power     : num [1:5062] 19 19.8 28 25 17 42.9 19.8 24.5 19.8
19.8 ...
##  $ price     : num [1:5062] 63500 115000 300000 63400 55000 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   model_name = col_character(),
##   ..   model_year = col_double(),
##   ..   kms_driven = col_double(),
##   ..   owner = col_character(),
##   ..   location = col_character(),
##   ..   mileage = col_double(),
##   ..   power = col_double(),
##   ..   price = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

> The different types of datatypes which are present in the dataset are numeric and charachter.

To determine the null values in the table, we can use the built in method in R.

```
null_values <- sum(is.na(bikeData))
null_values
```

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

> With the above method, we can see that there are 0 missing/Null values in the dataset.

Using built in methods to check for duplicates

```
duplicate_values_old <- sum(duplicated(bikeData))
duplicate_values_old
```

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

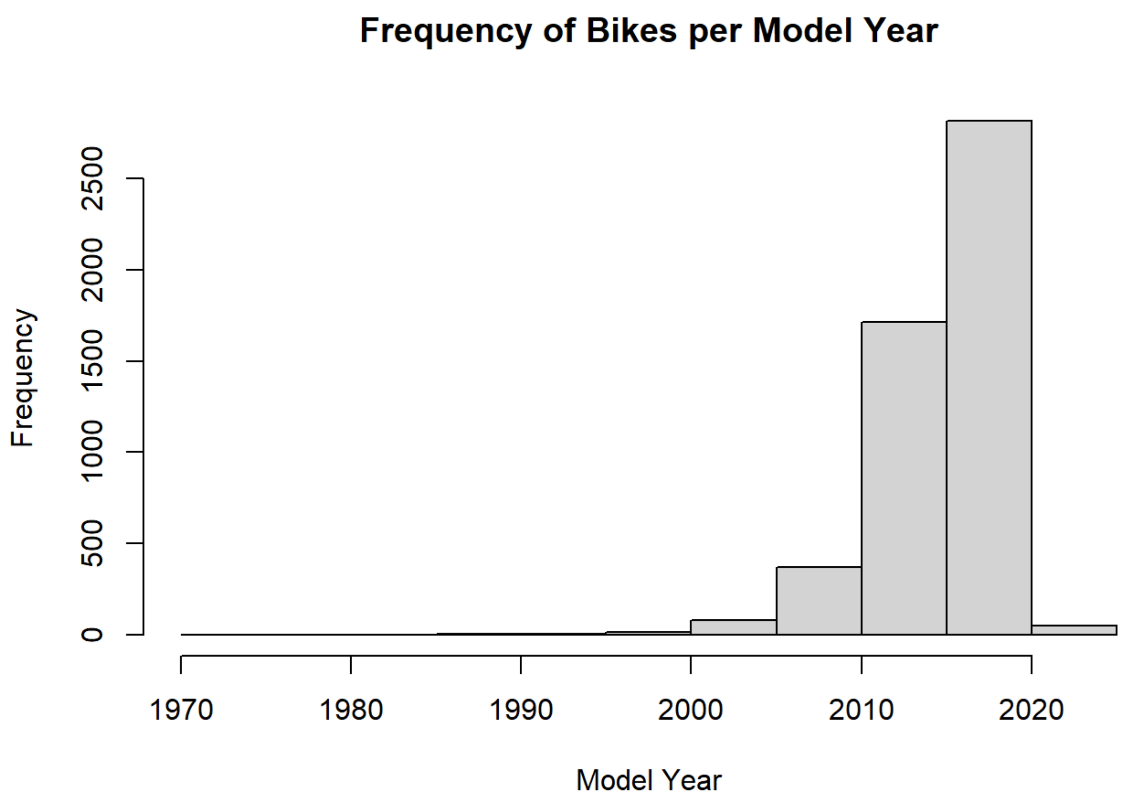
> There are 2 duplicate values in this dataset, that we will remove.

Removing duplicates

```
bike_data <- bikeData[!duplicated(bikeData),]
duplicate_values_new <- sum(duplicated(bike_data))
duplicate_values_new
```

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

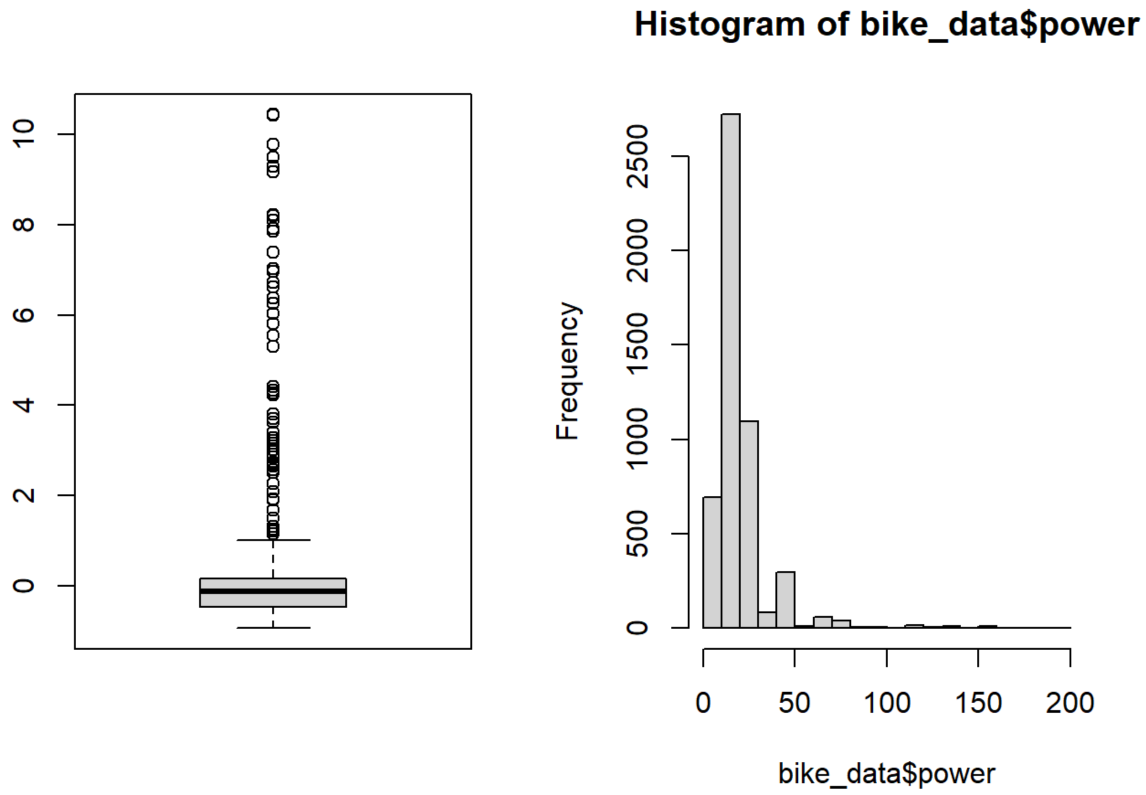
Part 2 - Graphical Overview



> This graph indicates the frequency of bikes by their model year. While most of the motorcycles were manufactured in from 2010-2019, there are some from 2000-2009, and 2020 and onwards as well. There are also some preceding the year 2000, and while we cannot ignore them, those numbers are incredibly minute and non-discernible.

Standardizing some values and checking for outliers:

```
standard_bikePower <- scale(bike_data$power)
par(mfrow = c(1,2))
boxplot(standard_bikePower)
hist(bike_data$power)
```



> These graphs show us that outliers do indeed exist in this data. The boxplot (of the scaled data) indicates that all of the outliers are high and exist beyond Q3, and also shows that the interquartile range is small, and that Q1 and Q3 are as well. The histogram (not scaled) indicates the true values of the power of these bikes, and we can immediately see that the majority of them have a power of under 50, with over 2500 falling into the 20-29 range alone. However, there are still some small numbers of bikes that exceed that, some with a power of close to 200.

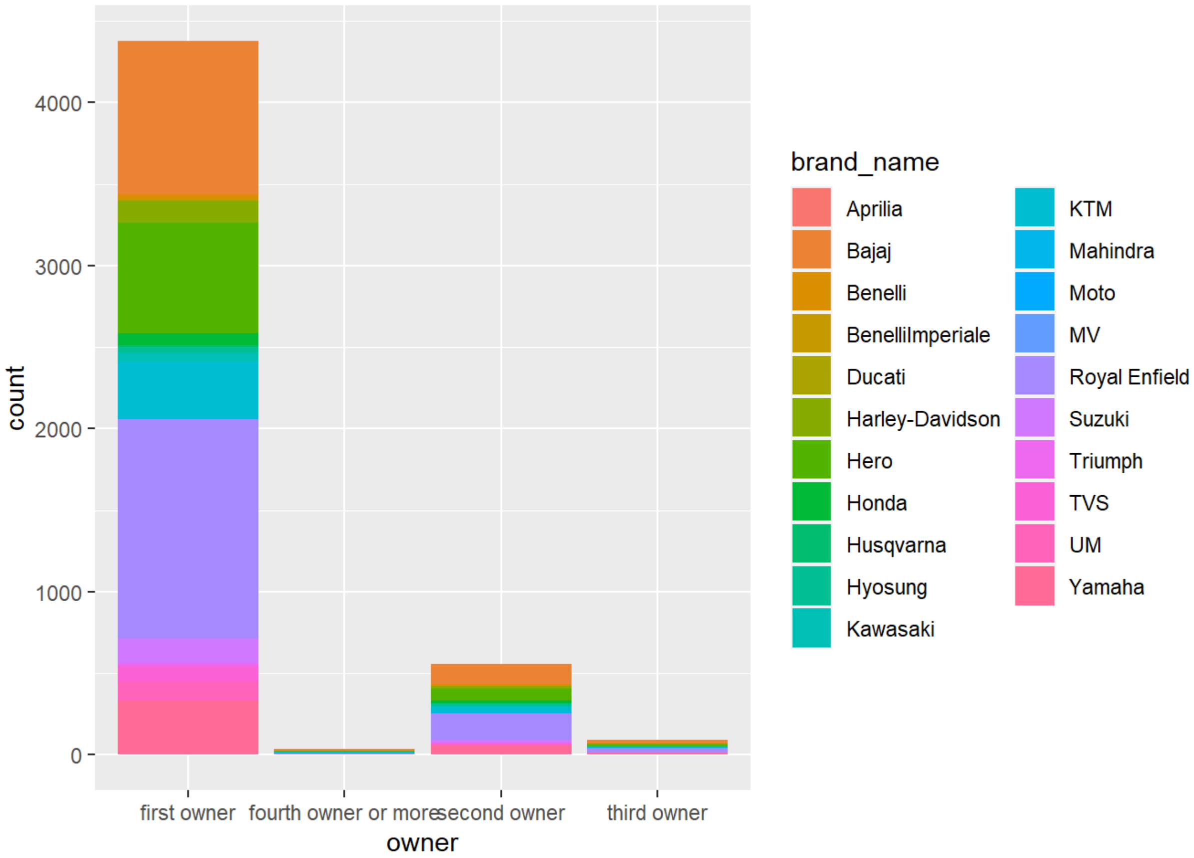
Extracting brand names from model names column to be used in later graphs

```
brand_model <- bike_data$model_name
testing <- str_split_fixed(brand_model, " ", 2)
brand_name <- testing[,1]
brand_name = str_replace_all(brand_name, "Royal", "Royal Enfield")
brand_df <- data.frame(brand_name)

bike_data_new <- cbind(bike_data, brand_df)
```

Bar plot showing relationship between brands and number of owners

```
ggplot(data = bike_data_new) +
  geom_bar(mapping = aes(fill = brand_name, x = owner))
```



> “Owner” is a discrete category found in bike_data and “brand_name” is a discrete category that has been appended to the data by extracting existing values. This barplot illustrates the relationship between the brand/make of the bikes and how many owners they’ve had. While the categories for “third owner” and “fourth owner or more” are too small to be discernible, this plot shows us that Royal Enfield, Bajaj, Honda and Yamaha are the most popular brands for bikes for both first and second owners, which could potentially be attributed to their longevity.

Part 3 - Hypothesis Testing

This part we gonna discuss about Hypothesis testing in order to see if it has meaningful results. We did three hypothesis testing. All of these focus on bike’s price.

```
data <- read.csv("Cleaned_bike_data.csv")
str(data)
```

```
## 'data.frame':    5062 obs. of  8 variables:
##  $ model_name: chr  "Bajaj Avenger Cruise" "Royal Enfield Classic"
"Hyosung GT250R 2012" "KTM Duke" ...
##  $ model_year: int  2017 2016 2012 2012 2016 2018 2018 2016 2017
2019 ...
##  $ kms_driven: int  17000 50000 14795 24561 19718 1350 25000 26240
18866 12634 ...
##  $ owner     : chr  "first owner" "first owner" "first owner" "third
owner" ...
##  $ location  : chr  "hyderabad" "hyderabad" "hyderabad" "bangalore" ...
##  $ mileage   : num  35 35 30 35 65 25 35 32 40 35 ...
##  $ power     : num  19 19.8 28 25 17 42.9 19.8 24.5 19.8 19.8 ...
##  $ price     : int  63500 115000 300000 63400 55000 198000 136900
112000 110000 160000 ...
```

First hand owner price higher the Second hand owner (One-side test)

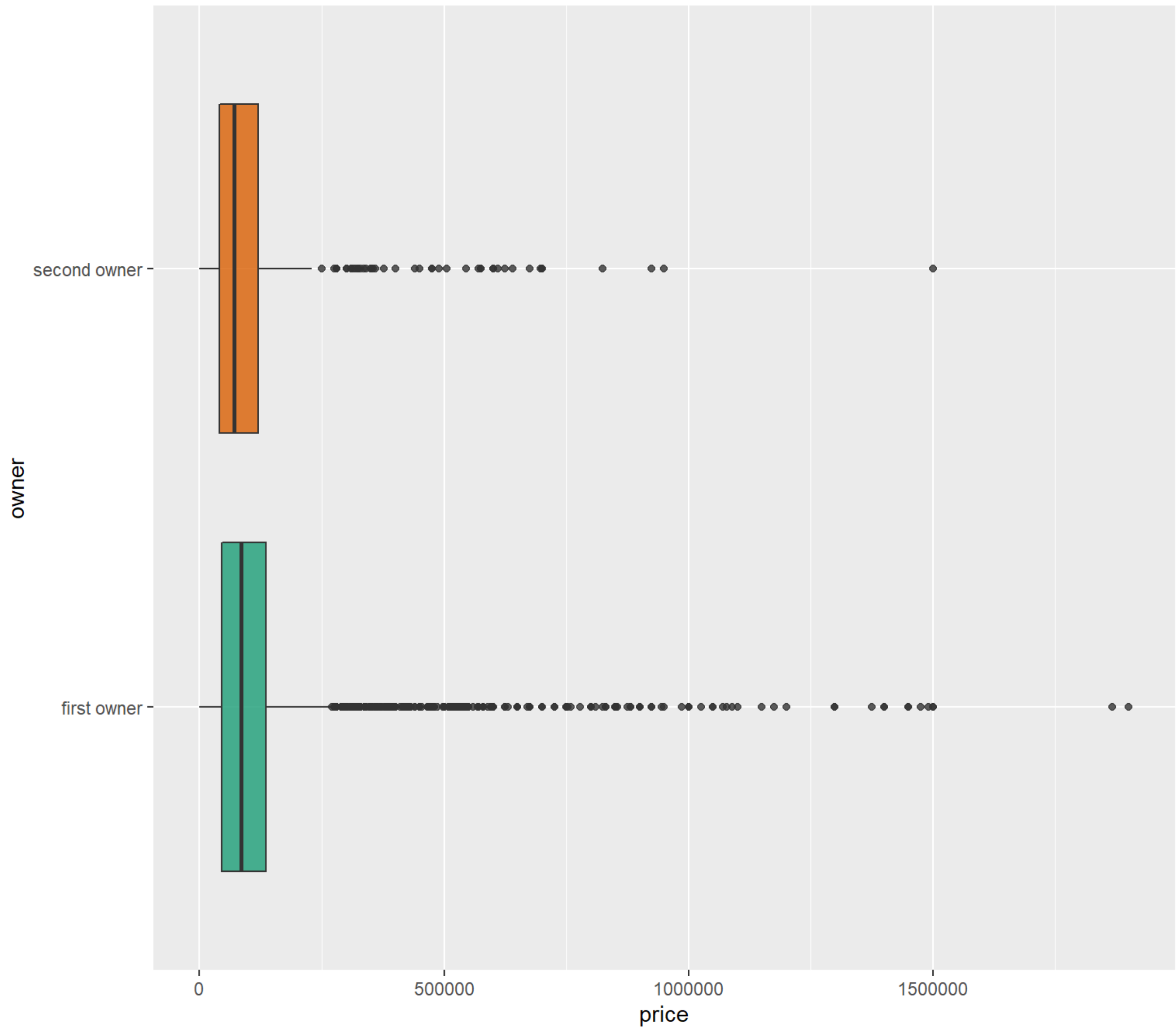
Let’s begin with testing First hand owner price and the Second hand owner, the hypothesis is the first hand price should higher. Therefore, H0 is the first owner price and second owner price are equal. The alternative is first owner price will higher that the second hand price.

\[\begin{align*} H_0: \mu_{first_owner_price} & = \mu_{second_owner_price}\\ H_1: \mu_{first_owner_price} & > \mu_{second_owner_price} \end{align*}\]

```
#split first hand owner and second hand owner
first_hand_owner <- subset(data, owner=="first owner")
second_hand_owner <- subset(data, owner=="second owner")


first_second_owner <- rbind(first_hand_owner, second_hand_owner)


ggplot(first_second_owner, aes(x=owner, y=price, fill=owner)) +
    geom_boxplot(alpha=0.8) +
    theme(legend.position="none") +
    scale_fill_brewer(palette="Dark2") +
    coord_flip()
```



> Check boxplot and we can see that there are many extreme price for first hand owners and the both median price look no different.

Don't know variance. So, use T-testing but need to check variance whether or not equal.
Check variance equal or not

$$\begin{align*} H_0: \sigma_{first\_owner\_price}^2 = \sigma_{second\_owner\_price}^2 \\ H_1: \sigma_{first\_owner\_price}^2 \ne \sigma_{second\_owner\_price}^2 \end{align*}$$

```
var.test(first_hand_owner$price, second_hand_owner$price, alternative =
"two.sided")
```

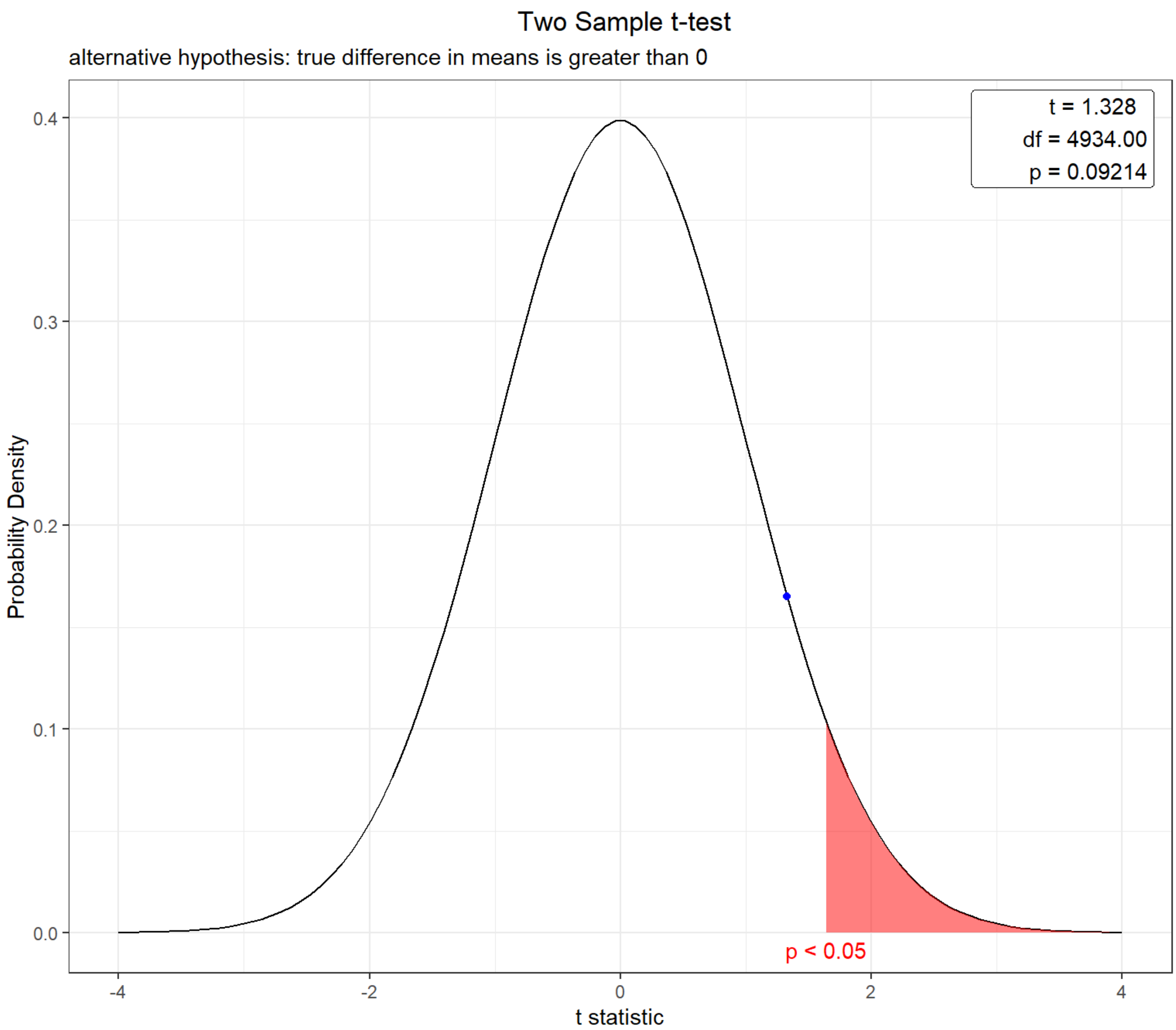
```
##
##  F test to compare two variances
##
## data:  first_hand_owner$price and second_hand_owner$price
## F = 1.0795, num df = 4377, denom df = 557, p-value = 0.2404
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.9499365 1.2192487
## sample estimates:
## ratio of variances
##           1.079493
```

```
#Not reject H0 => variance equal

#t-testing
t.test(first_hand_owner$price,
second_hand_owner$price,alternative="greater", var.equal = TRUE,
conf.level = 0.95)
```

```
##
##  Two Sample t-test
##
## data:  first_hand_owner$price and second_hand_owner$price
## t = 1.3279, df = 4934, p-value = 0.09214
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  -2058.878       Inf
## sample estimates:
## mean of x mean of y
##  117156.0  108539.3
```

```
plot(t.test(first_hand_owner$price,
second_hand_owner$price,alternative="greater", var.equal = TRUE,
conf.level = 0.95))
```



>Due to we don’t have variance, we have to choose T-testing. However, we need to check the variance between two groups first whether it equal or not. P value greater than .05 So, we should not reject H0. It means the variance of two groups are equal. Then do one-side t testing, the result of the testing P value greater than .05. It means that we should not reject H0. It indicates that we can accept that first hand owner price not different from the second hand owner price.

Price of second hand bike not be different from third and later price (Two-side test)

The second testing, we will check whether or not Price of Second hand bike not be different from third and later hand.

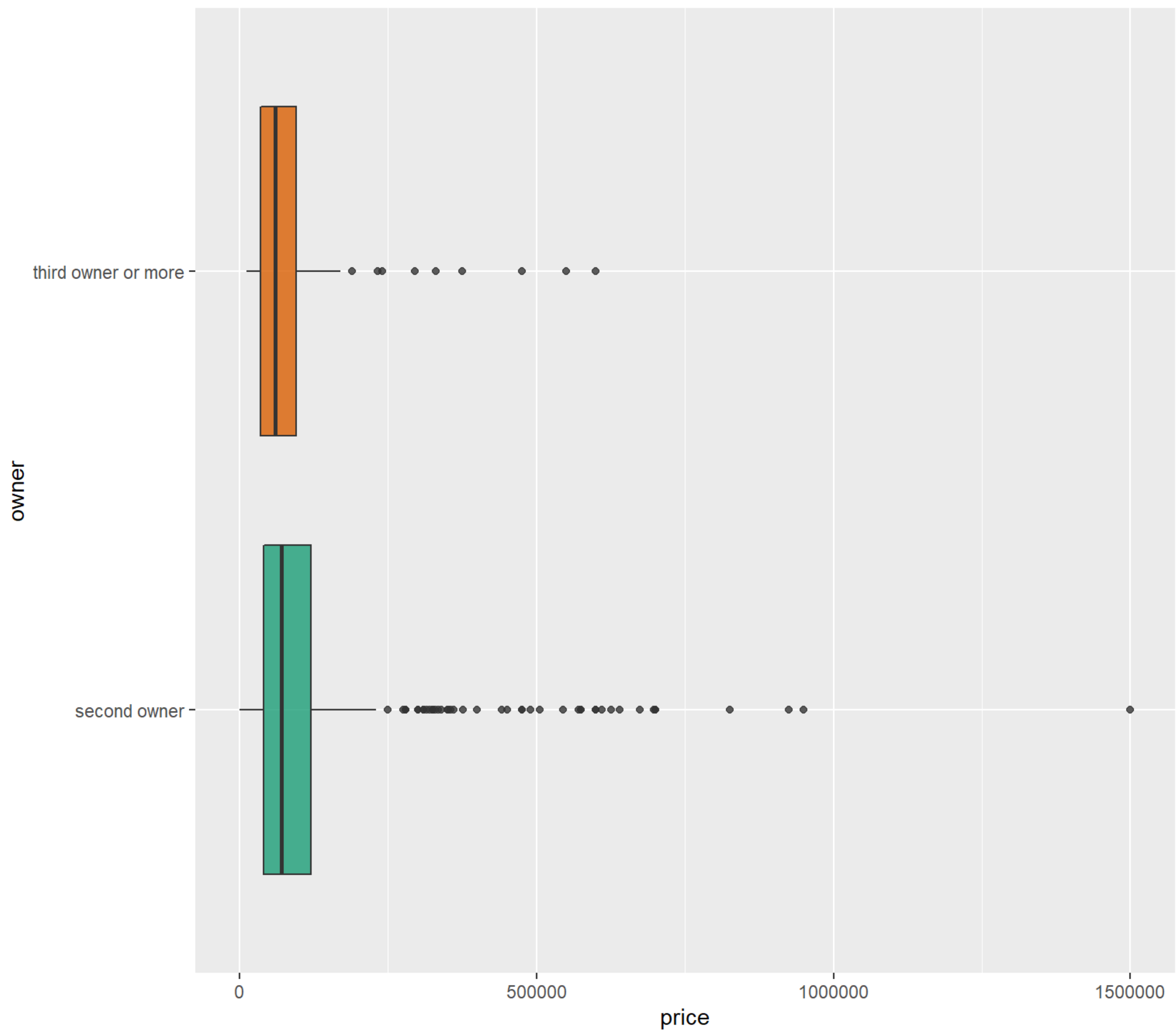
\[\begin{align*} H_0: \mu_{second_owner_price} & = \mu_{later_second_owner_price} \\ H_1: \mu_{second_owner_price} & \ne \mu_{later_second_owner_price} \end{align*}\]

```
second_hand_owner <- subset(data, owner=="second owner")
later_second_hand_owner <- subset(data, owner =="third owner" | owner ==
"fourth owner or more")

later_second_hand_owner$owner[later_second_hand_owner$owner == "third
owner" | later_second_hand_owner$owner == "fourth owner or more"]<- "third
owner or more"

second_later_owner <- rbind(second_hand_owner, later_second_hand_owner)


ggplot(second_later_owner, aes(x=owner, y=price, fill=owner)) +
    geom_boxplot(alpha=0.8) +
    theme(legend.position="none") +
    scale_fill_brewer(palette="Dark2") +
    coord_flip()
```



After split the data into two group, we created a boxplot. We can see that the second owners has more outliers and the the range larger.

Don’t know variance. So, use T-testing but need to check variance whether or not equal.
Check variance equal or not
\[\begin{align*} H_0: \sigma_{second_owner_price}^2 = \sigma_{later_second_owner_price}^2 \\ H_1: \sigma_{second_owner_price}^2 \ne \sigma_{later_second_owner_price}^2 \end{align*}\]

```
var.test(second_hand_owner$price, later_second_hand_owner$price,
alternative = "two.sided")
```

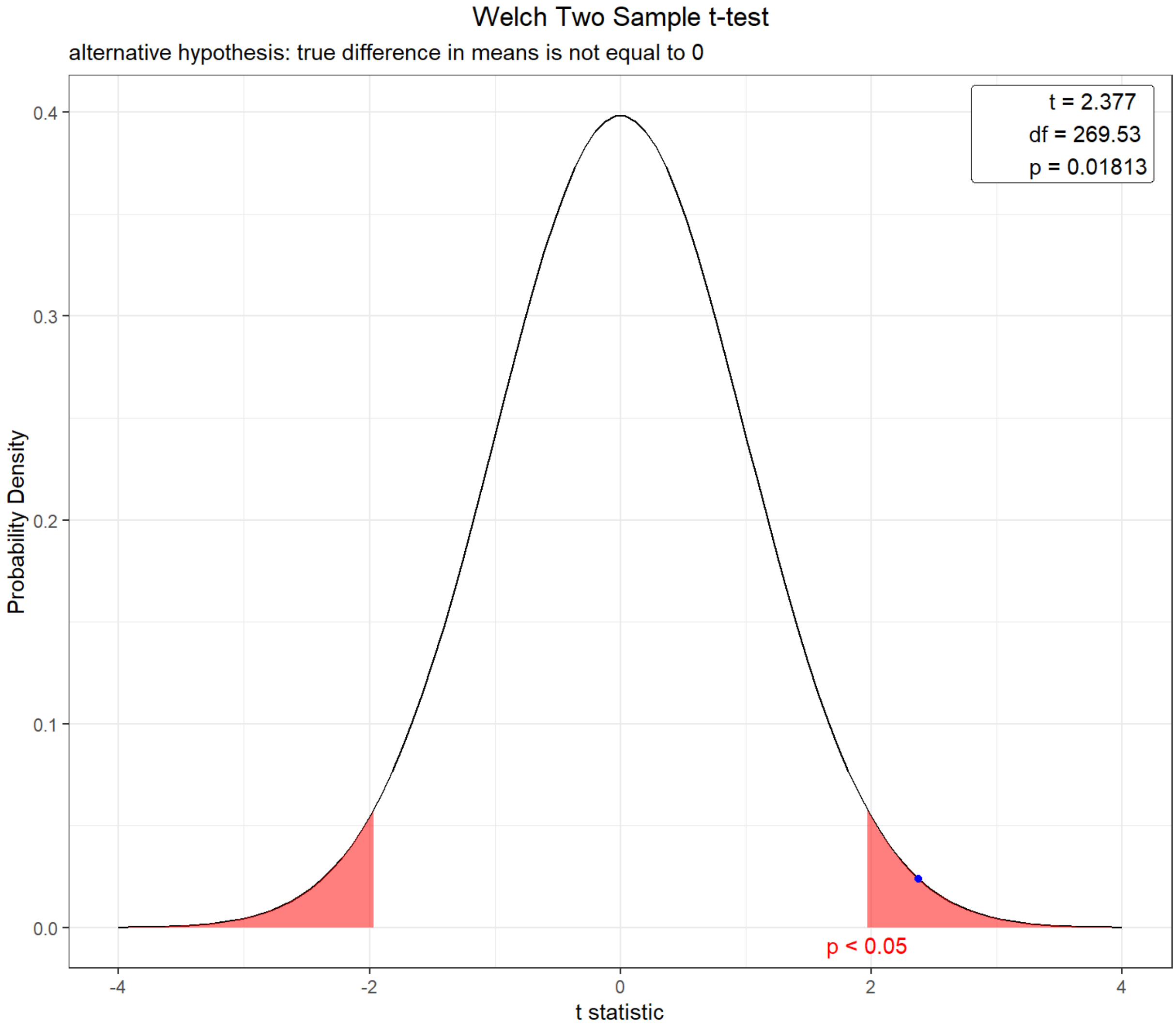
```
##
##  F test to compare two variances
##
## data:  second_hand_owner$price and later_second_hand_owner$price
## F = 2.2621, num df = 557, denom df = 125, p-value = 1.152e-07
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  1.696984 2.943189
## sample estimates:
## ratio of variances
##           2.262141
```

```
#reject H0 => variance not equal

#t-testing
t.test(second_hand_owner$price,
later_second_hand_owner$price,alternative="two.sided", var.equal = FALSE,
conf.level = 0.95)
```

```
##
##  Welch Two Sample t-test
##
## data:  second_hand_owner$price and later_second_hand_owner$price
## t = 2.3775, df = 269.53, p-value = 0.01813
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   4151.481 44150.365
## sample estimates:
## mean of x mean of y
## 108539.26  84388.33
```

```
plot(t.test(second_hand_owner$price,
later_second_hand_owner$price,alternative="two.sided", var.equal = FALSE,
conf.level = 0.95))
```



> Due to we don’t have variance, we have to choose T-testing. However, we need to check the variance between two groups first whether it equal or not. P value less than .05 So, we should reject H0. It means the variance of two groups are not equal. Then do two-side t testing, the result of the testing P value less than .05. It means that we should reject H0. It indicates that we can accept that second hand owner price is different from the later owner price.

Higher kilometers driven price cheaper than lower kilometers driven price (One side test)

The last hypothesis testing, we will test on the hypothesis about the price of two group which was splited by the median of kilometers. The group that has higher kilometers driven will cheaper than the group that has lower kilometers driven.

\[\begin{align*} H_0: \mu_{greater_kms_driven_price} & = \mu_{less_kms_driven_price} \\ H_1: \mu_{greater_kms_driven_price} & < \mu_{less_kms_driven_price} \end{align*}\]

```
median(data$kms_driven)
```

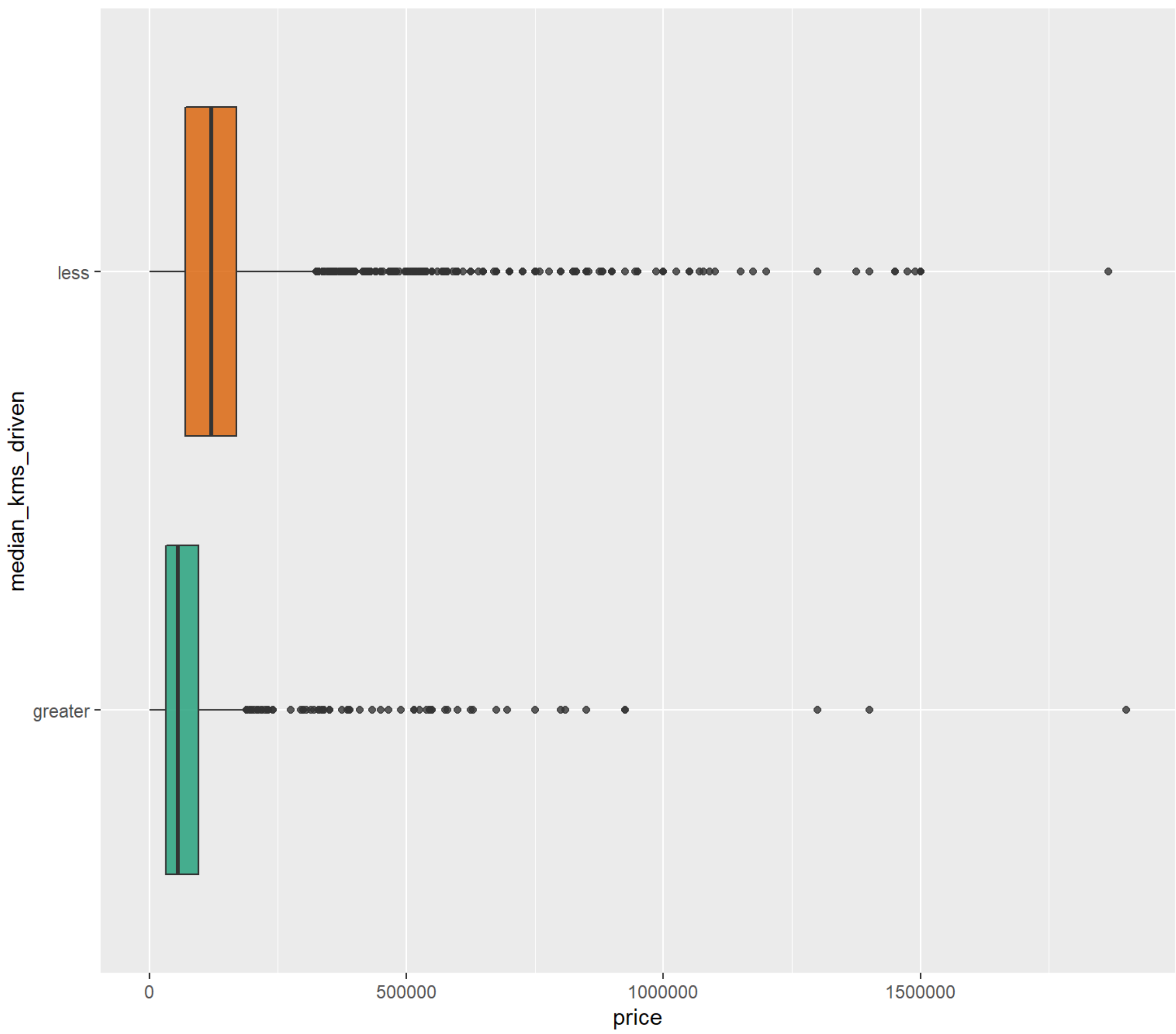
```
## [1] 18000
```

```
greater_median_kms_driven <- subset(data, kms_driven > 18000)
less_median_kms_driven <- subset(data, kms_driven <= 18000)

greater_median_kms_driven["median_kms_driven"] <- "greater"
less_median_kms_driven["median_kms_driven"] <- "less"

all_kms_driven <- rbind(greater_median_kms_driven, less_median_kms_driven)


ggplot(all_kms_driven, aes(x=median_kms_driven, y=price,
fill=median_kms_driven)) +
    geom_boxplot(alpha=0.8) +
    theme(legend.position="none") +
    scale_fill_brewer(palette="Dark2") +
    coord_flip()
```



> The boxplot shows that the group that has less kms driven has higher price.

Don’t know variance. So, use T-testing but need to check variance whether or not equal. Check variance equal or not \[\begin{align*} H_0: \sigma_{greater_median_kms_driven_price}^2 = \sigma_{less_median_kms_driven_price}^2 \\ H_1: \sigma_{greater_median_kms_driven_price}^2 \ne \sigma_{less_median_kms_driven_price}^2 \end{align*}\]

```
var.test(greater_median_kms_driven$price, less_median_kms_driven$price ,
alternative = "two.sided")
```

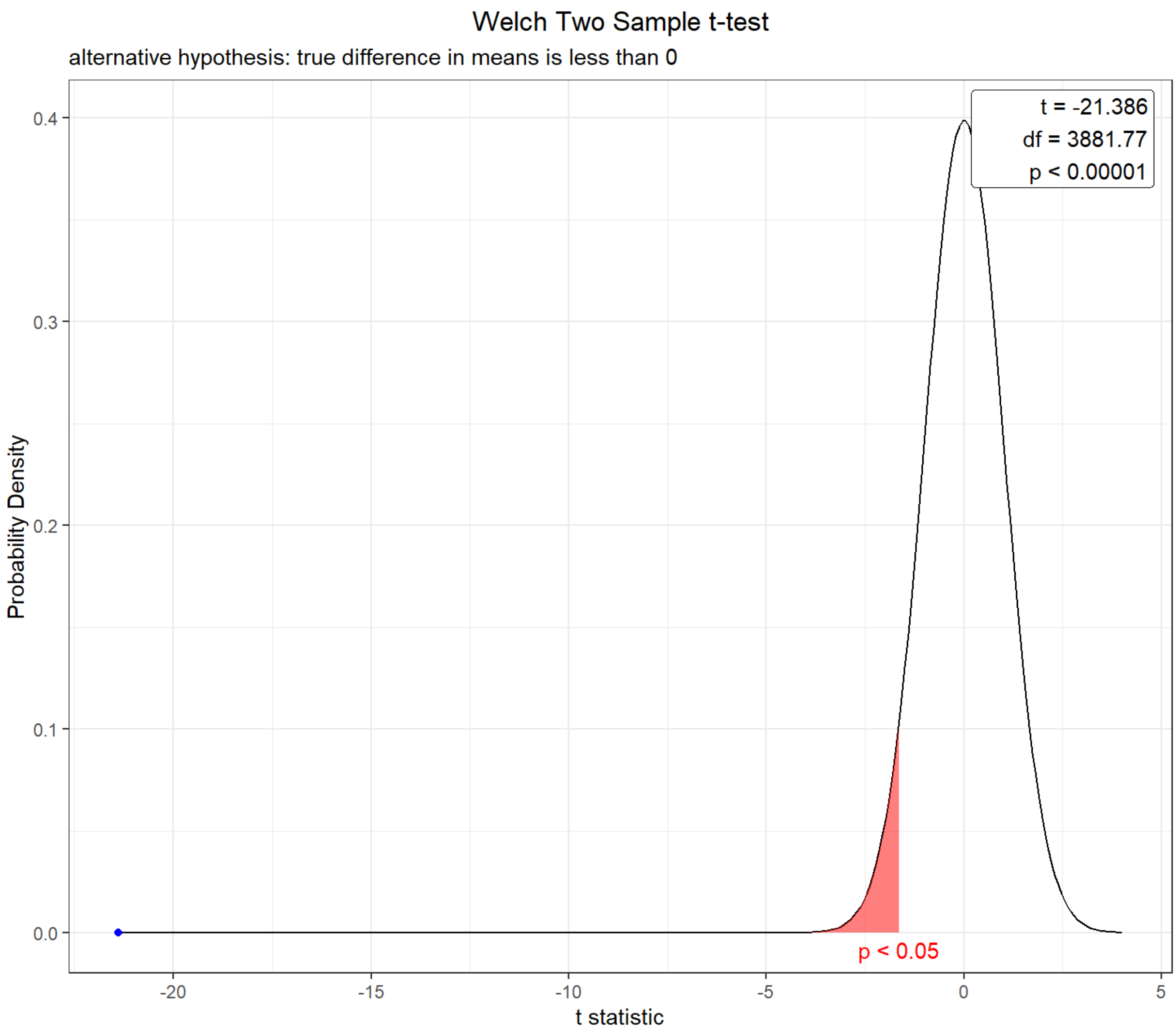
```
##
##  F test to compare two variances
##
## data:  greater_median_kms_driven$price and less_median_kms_driven$price
## F = 0.28238, num df = 2517, denom df = 2543, p-value < 2.2e-16
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.2612031 0.3052711
## sample estimates:
## ratio of variances
##          0.2823765
```

```
# reject H0 => variance not equal

#t-testing
t.test(greater_median_kms_driven$price,
less_median_kms_driven$price,alternative="less", var.equal = FALSE,
conf.level = 0.95)
```

```
##
##  Welch Two Sample t-test
##
## data:  greater_median_kms_driven$price and less_median_kms_driven$price
## t = -21.386, df = 3881.8, p-value < 2.2e-16
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
##       -Inf -76010.11
## sample estimates:
## mean of x mean of y
##  74006.54 156351.61
```

```
plot(t.test(greater_median_kms_driven$price,
less_median_kms_driven$price,alternative="less", var.equal = FALSE,
conf.level = 0.95))
```



> Due to we don't have variance, we have to choose T-testing. However, we need to check the variance between two groups first whether it equal or not. P value less than .05 So, we should reject H0. It means the variance of two groups are not equal. Then do one-side t testing, the result of the testing P value much less than .05. It means that we should reject H0. It would mean that we can accept that the group that has greater kilometer driven will cheaper than less kilometer driven.

Part 4 - Linear Regression

This section is for Linear Regression.

Including the necessary packages for regression

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(MLmetrics)
```

```
## Warning: package 'MLmetrics' was built under R version 4.1.3
```

```
##
## Attaching package: 'MLmetrics'
```

```
## The following object is masked from 'package:base':
##
##     Recall
```

Setting the seed to store the data and keep it same.

```
set.seed(101)
```

Splitting the Dataset into Training and Testing Dataset randomly. We have splitted the data in 70% and 30%.

```
i = sample(2, nrow(bike_data), replace=TRUE, prob=c(0.7, 0.3))
bikeTraining <- bike_data[i==1,]
bikeTesting <- bike_data[i==2,]
```

The Training Dataset consists of 3554 entries and the Testing dataset consists of 1508 entries.

Model 1 - Forward Propogation Model.

We start with constructing the intercept model. The intercept is used to form a linear regression model with a constant variable. The full model is used to select all the attributes which are seen in the data table. We then use the stepAIC function to travel step by step and select all the elements with the highest AIC until the occurrence of the null variable.

```
intercept_model <- lm(price ~ 1, data = bikeTraining[,1:8])
full_model <- lm(price ~ .-model_name, data = bikeTraining[,1:8])
forward_model <- stepAIC(intercept_model, direction = "forward",scope =
formula(full_model))
```

```
## Start:  AIC=83305.72
## price ~ 1
##
##               Df  Sum of Sq        RSS   AIC
## + power        1 5.3384e+13 1.6455e+13 78229
## + mileage      1 1.5112e+13 5.4727e+13 82451
## + model_year   1 4.3751e+12 6.5464e+13 83080
## + kms_driven   1 2.8466e+12 6.6992e+13 83162
## + owner        3 1.2014e+11 6.9719e+13 83306
## <none>                      6.9839e+13 83306
## + location   363 9.6233e+12 6.0216e+13 83511
##
## Step:  AIC=78229.46
## price ~ power
##
##               Df  Sum of Sq        RSS   AIC
## + model_year   1 9.9644e+11 1.5459e+13 78012
## + owner        3 4.6492e+11 1.5990e+13 78135
## + kms_driven   1 3.5124e+11 1.6104e+13 78156
## + mileage      1 1.4293e+10 1.6441e+13 78228
## <none>                      1.6455e+13 78229
## + location   363 2.4532e+12 1.4002e+13 78388
##
## Step:  AIC=78012.01
## price ~ power + model_year
##
##               Df  Sum of Sq        RSS   AIC
## + owner        3 3.3695e+11 1.5122e+13 77941
## + kms_driven   1 9.3287e+10 1.5365e+13 77993
## + mileage      1 3.0729e+10 1.5428e+13 78007
## <none>                      1.5459e+13 78012
## + location   363 2.1832e+12 1.3275e+13 78203
##
## Step:  AIC=77940.6
## price ~ power + model_year + owner
##
##               Df  Sum of Sq        RSS   AIC
## + kms_driven   1 4.7746e+10 1.5074e+13 77931
## + mileage      1 2.9021e+10 1.5093e+13 77936
## <none>                      1.5122e+13 77941
## + location   363 2.2017e+12 1.2920e+13 78114
##
## Step:  AIC=77931.49
## price ~ power + model_year + owner + kms_driven
##
##             Df  Sum of Sq        RSS   AIC
## + mileage    1 3.5453e+10 1.5038e+13 77925
## <none>                    1.5074e+13 77931
## + location 363 2.1844e+12 1.2889e+13 78108
##
## Step:  AIC=77925.21
```

```
## price ~ power + model_year + owner + kms_driven + mileage
##
##              Df  Sum of Sq        RSS   AIC
## <none>                     1.5038e+13 77925
## + location 363 2.1849e+12 1.2854e+13 78100
```

As we can see that the full model is constructed without the use of the model name. The reason for not including the model name is that it is not revelant to the price.

```
summary(forward_model)
```

```
##
## Call:
## lm(formula = price ~ power + model_year + owner + kms_driven +
##     mileage, data = bikeTraining[, 1:8])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -655790  -25360   -1305   20147  850709
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -8.341e+06  6.506e+05 -12.821  < 2e-16 ***
## power                      7.532e+03  8.148e+01  92.446  < 2e-16 ***
## model_year                 4.112e+03  3.227e+02  12.744  < 2e-16 ***
## ownerfourth owner or more -9.862e+04  1.387e+04  -7.108 1.42e-12 ***
## ownersecond owner         -1.408e+04  3.589e+03  -3.924 8.87e-05 ***
## ownerthird owner          -1.710e+04  8.331e+03  -2.053 0.040150 *
## kms_driven                -1.440e-01  4.051e-02  -3.553 0.000385 ***
## mileage                    2.385e+02  8.295e+01   2.875 0.004070 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 65500 on 3505 degrees of freedom
## Multiple R-squared:  0.7847, Adjusted R-squared:  0.7842
## F-statistic:  1825 on 7 and 3505 DF,  p-value: < 2.2e-16
```

```
forward_model$anova
```

```
## Stepwise Model Path
## Analysis of Deviance Table
##
## Initial Model:
## price ~ 1
##
## Final Model:
## price ~ power + model_year + owner + kms_driven + mileage
##
##
##            Step Df     Deviance Resid. Df   Resid. Dev      AIC
## 1                                    3512 6.983886e+13 83305.72
## 2       + power  1 5.338388e+13      3511 1.645498e+13 78229.46
## 3  + model_year  1 9.964350e+11      3510 1.545854e+13 78012.01
## 4       + owner  3 3.369470e+11      3507 1.512160e+13 77940.60
## 5  + kms_driven  1 4.774630e+10      3506 1.507385e+13 77931.49
## 6     + mileage  1 3.545320e+10      3505 1.503840e+13 77925.21
```

We can calculate the MAE and MSE for both the backward model as follows:

```
forward_pred <-predict(object = forward_model, newdata = bikeTesting[,1:8])
MAE(y_pred = forward_pred, y_true = bikeTesting$price)
```

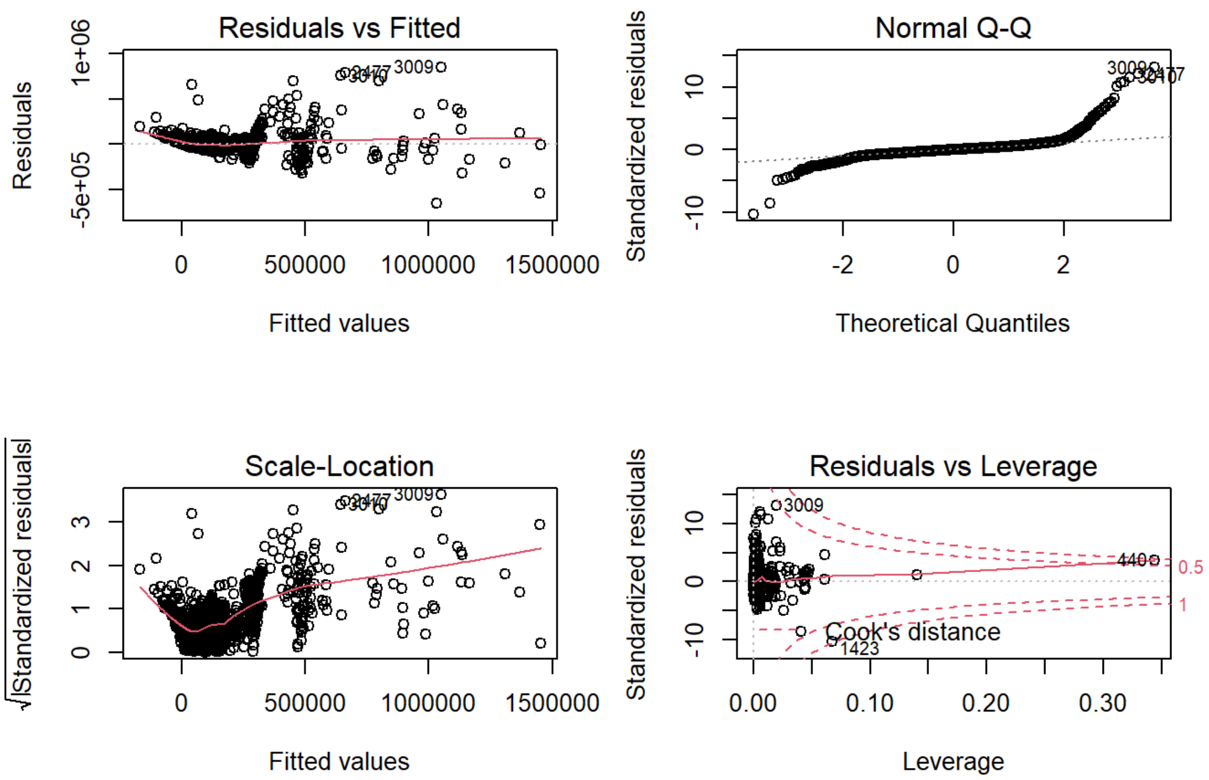
```
## [1] 37452.99
```

```
MSE(y_pred = forward_pred, y_true = bikeTesting$price)
```

```
## [1] 5406660098
```

Plotting the Forward Propogation Model

```
par(mfrow=c(2,2))
plot(forward_model)
```



Model 2 - Backward Propogation

```
backward <- stepAIC(full_model, direction = "backward")
```

```
## Start:  AIC=78099.72
## price ~ (model_name + model_year + kms_driven + owner + location +
##     mileage + power) - model_name
##
##               Df  Sum of Sq        RSS   AIC
## - location   363 2.1849e+12 1.5038e+13 77925
## <none>                      1.2854e+13 78100
## - kms_driven   1 3.4881e+10 1.2888e+13 78107
## - mileage      1 3.5869e+10 1.2889e+13 78108
## - owner        3 3.0626e+11 1.3160e+13 78176
## - model_year   1 5.0847e+11 1.3362e+13 78234
## - power        1 3.3260e+13 4.6113e+13 82586
##
## Step:  AIC=77925.21
## price ~ model_year + kms_driven + owner + mileage + power
##
##              Df  Sum of Sq        RSS   AIC
## <none>                     1.5038e+13 77925
## - mileage     1 3.5453e+10 1.5074e+13 77931
## - kms_driven  1 5.4178e+10 1.5093e+13 77936
## - owner       3 2.8714e+11 1.5326e+13 77986
## - model_year  1 6.9686e+11 1.5735e+13 78082
## - power       1 3.6668e+13 5.1706e+13 82262
```

```
summary(backward)
```

```
##
## Call:
## lm(formula = price ~ model_year + kms_driven + owner + mileage +
##     power, data = bikeTraining[, 1:8])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -655790  -25360   -1305   20147  850709
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -8.341e+06  6.506e+05 -12.821  < 2e-16 ***
## model_year                 4.112e+03  3.227e+02  12.744  < 2e-16 ***
## kms_driven                -1.440e-01  4.051e-02  -3.553 0.000385 ***
## ownerfourth owner or more -9.862e+04  1.387e+04  -7.108 1.42e-12 ***
## ownersecond owner         -1.408e+04  3.589e+03  -3.924 8.87e-05 ***
## ownerthird owner          -1.710e+04  8.331e+03  -2.053 0.040150 *
## mileage                    2.385e+02  8.295e+01   2.875 0.004070 **
## power                      7.532e+03  8.148e+01  92.446  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 65500 on 3505 degrees of freedom
## Multiple R-squared:  0.7847, Adjusted R-squared:  0.7842
## F-statistic:  1825 on 7 and 3505 DF,  p-value: < 2.2e-16
```

> We see that the accuracy of the backward model is similar to that of the forward model.

Calculation of the MAE and MSE for the backward model

```
backward_pred <-predict(object = backward, newdata = bikeTesting[,1:8])
MAE(y_pred = backward_pred, y_true = bikeTesting$price)
```

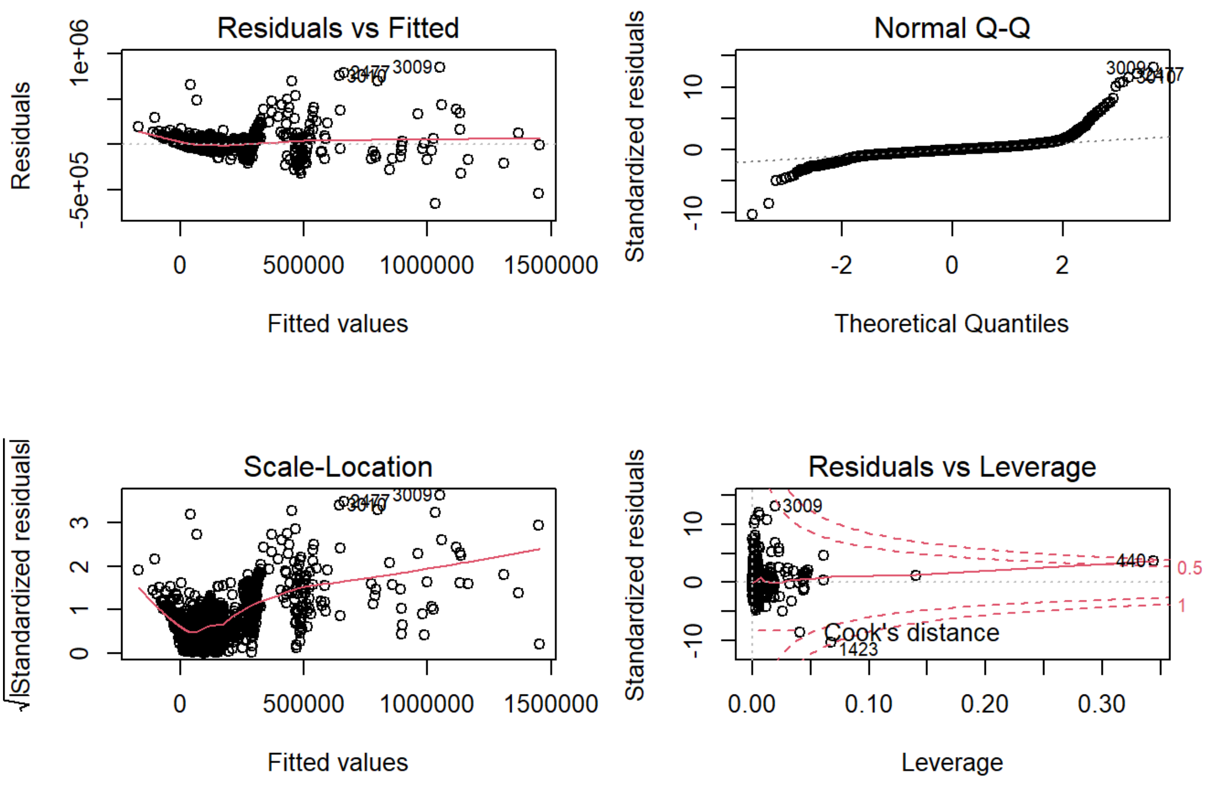
```
## [1] 37452.99
```

```
MSE(y_pred = backward_pred, y_true = bikeTesting$price)
```

```
## [1] 5406660098
```

Plotting the Backward Propogation model

```
par(mfrow = c(2,2))
plot(backward)
```



We can see that both the models have the same accuracy and same mean square error and Mean absolute error. The accuracy for both the models is 77.02%.