Assignment 1

The Old Republic

2023-06-21

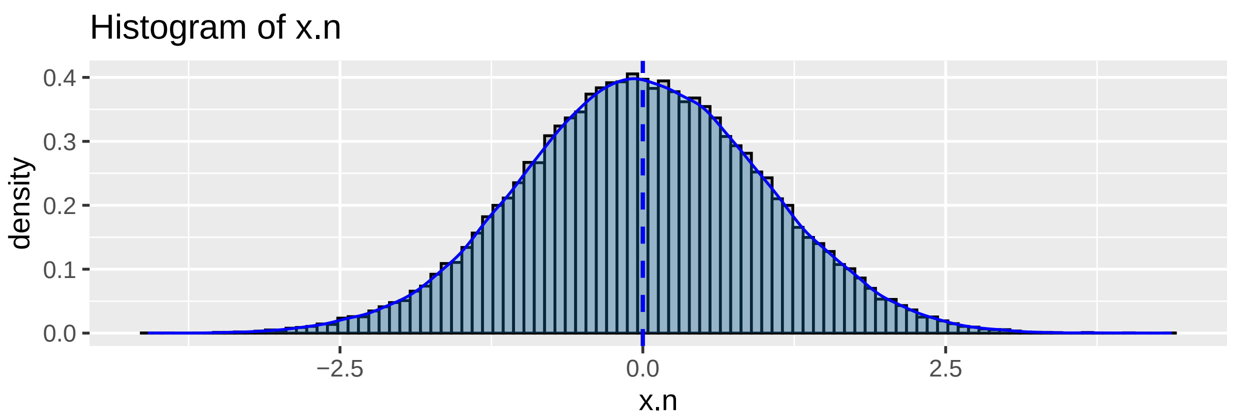
```
library(Pareto)
set.seed(100)
Data = data.frame(x.n = rnorm(50000), x.p = rPareto(50000, t=1, alpha=2))
summary(Data)
```

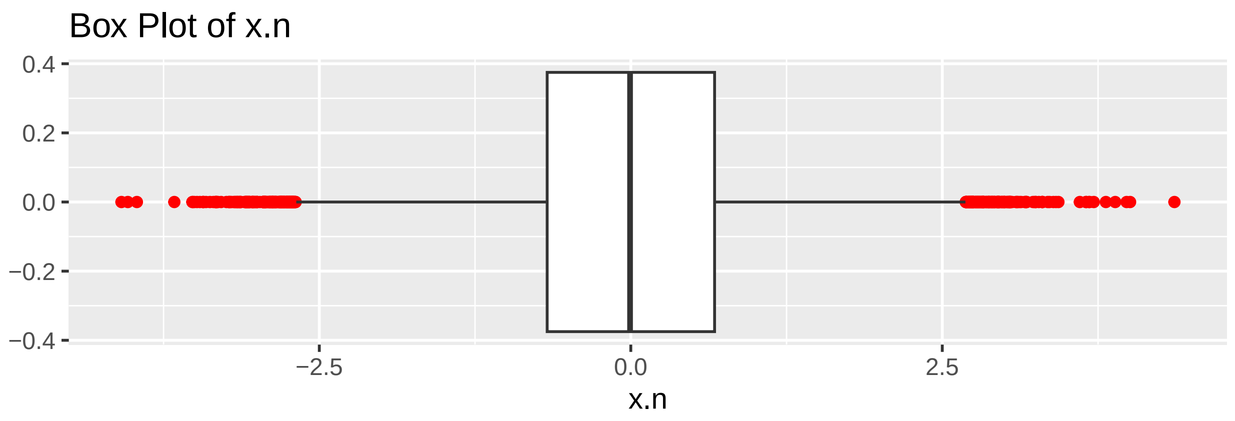
```
##       x.n                  x.p
##  Min.   :-4.087893   Min.   :  1.000
##  1st Qu.:-0.671144   1st Qu.:  1.154
##  Median :-0.005919   Median :  1.412
##  Mean   :-0.000208   Mean   :  1.994
##  3rd Qu.: 0.672466   3rd Qu.:  1.992
##  Max.   : 4.363243   Max.   :159.275
```

Question 1

1. Histogramand Box Plot of the Variable x.n

```
library(ggplot2)
library(grid)
library(gridExtra)
library(dplyr)

x.n.hist = ggplot(data=Data, aes(x=x.n)) +
  geom_histogram(aes(y=after_stat(density)) ,bins=100, fill="grey", color="black") +
  ggtitle("Histogram of x.n") +
  geom_density(lwd=0.5, color="blue", fill=4, alpha=0.25) +
  geom_vline(xintercept=mean(Data$x.n), color="blue", linetype="dashed", lwd=0.75)

x.n.box = ggplot(data=Data, mapping=aes(x=x.n)) +
  geom_boxplot(outlier.color="red") +
  ggtitle("Box Plot of x.n")

grid.arrange(x.n.hist,x.n.box)
```





2. The sample mean, and standard deviation of x.n are $-2.0849558 \times 10^{-4}$ and 0.9989658 respectively rounded to 3 decimal places. It is evident that these parameters are very close to the parameters of a variable Z following a standard normal distribution, namely $Z \sim N(0, 1)$. This result is not surprising since the observations were taken from a normal distribution with $\mu = 0$ and $\sigma^2 = 1$.

3. Judging by the shape of the distribution of the observations of x.n and the values for the mean and standard deviation of these observations, we may conclude that $x.n \sim N(0, 1)$ approximately. This means that it is highly likely that new observations from the variable x.n will have values close to 0 (μ). Although extreme values are possible, there is an equal probability that an extreme value of the same magnitude but of opposite sign will occur, meaning that the expected value of extreme observations will also be close to 0, which further strengthens the claim that new observations will likely take values close to 0. Additionally, we can also predict that approximately 68% of the observations will be between -1 and 1 ($\mu \pm \sigma$), 95% of the observations will be between -2 and 2 ($\mu \pm 2\sigma$) and 99.7% of the observations will be between -3 and 3 ($\mu \pm 3\sigma$).

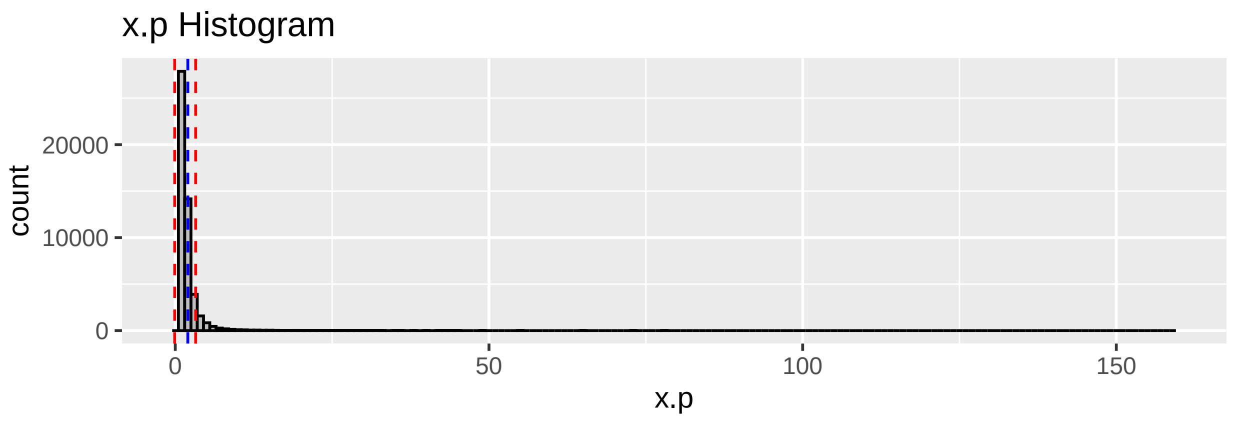
4. Mean and standard deviation of the variable x.p

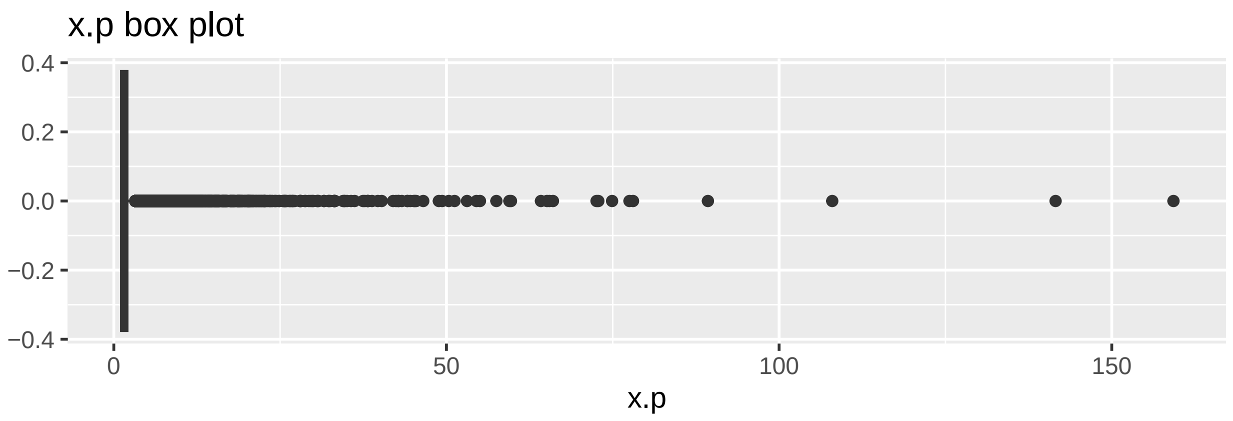
```
x.p.mean = mean(Data$x.p)
x.p.sd = sd(Data$x.p)
x.p.IQR = quantile(Data$x.p, 0.75)-quantile(Data$x.p, 0.25)
```

Histogram and Box Plot of the variable x.p

```
x.p.hist = ggplot(data=Data, mapping=aes(x=x.p)) +
  geom_histogram(binwidth=1, fill="grey", color="black",) +
  ggtitle("x.p Histogram") +
  geom_vline(aes(xintercept=x.p.mean), color="blue", linetype="dashed") +
  geom_vline(aes(xintercept= (quantile(x.p, 0.75) + 1.5*x.p.IQR)), color="red", linetype="dashed")+
  geom_vline(aes(xintercept= (quantile(x.p, 0.25) - 1.5*x.p.IQR)), color="red", linetype="dashed")
```

```r
x.p.box = ggplot(data=Data, mapping=aes(x=x.p)) +
  geom_boxplot() +
  ggtitle("x.p box plot")

grid.arrange(x.p.hist, x.p.box)
```





Create table x.p.count which contains the number of data points in classes of width 0.5 and data frame x.p.df which contains the mean value of the data points that do not belong in a class that has a count less than a specific value.

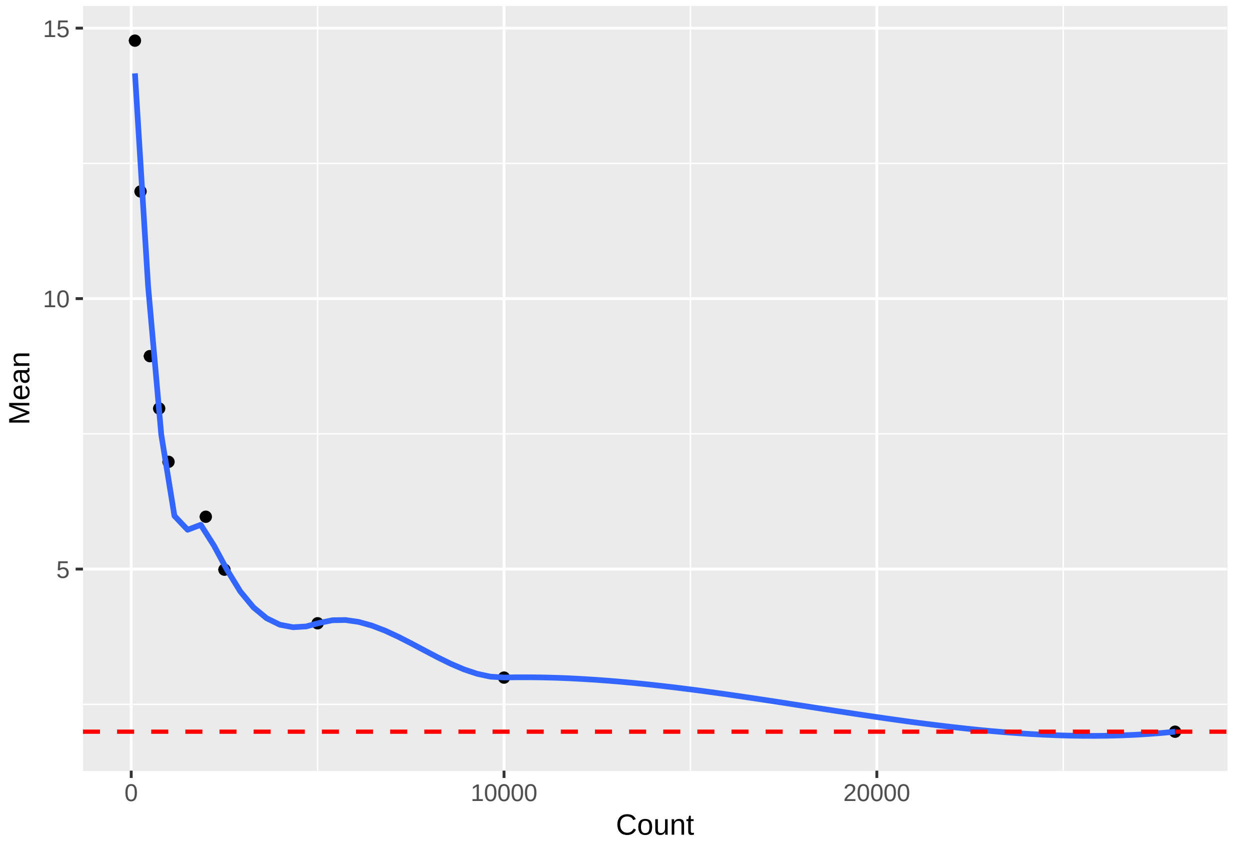
```r
x.p.counts = table(cut(Data$x.p, seq(0, 160, 0.5)))
x.p.counts.lastclass = sum(x.p.counts[45:dim(x.p.counts)])
x.p.counts = x.p.counts[(1:44)]
x.p.counts[">22"] = x.p.counts.lastclass

x.p.df = data.frame(matrix(ncol=2))
colnames(x.p.df) = c("Count", "Mean")

# View x.p.counts
x.p.counts
```

```
##   (0,0.5]   (0.5,1]   (1,1.5]   (1.5,2]
##         0         0     27882      9728
##   (2,2.5]   (2.5,3]   (3,3.5]   (3.5,4]
##      4442      2398      1501       941
##   (4,4.5]   (4.5,5]   (5,5.5]   (5.5,6]
##       640       480       366       251
##   (6,6.5]   (6.5,7]   (7,7.5]   (7.5,8]
##       200       160       111        99
##   (8,8.5]   (8.5,9]   (9,9.5]  (9.5,10]
##        93        60        72        46
## (10,10.5] (10.5,11] (11,11.5] (11.5,12]
##        52        48        36        28
## (12,12.5] (12.5,13] (13,13.5] (13.5,14]
##        25        31        20        15
## (14,14.5] (14.5,15] (15,15.5] (15.5,16]
##        19        18        18        17
## (16,16.5] (16.5,17] (17,17.5] (17.5,18]
##        10        12         4        10
## (18,18.5] (18.5,19] (19,19.5] (19.5,20]
##         4        13         6         6
## (20,20.5] (20.5,21] (21,21.5] (21.5,22]
##        11         9         5         4
##       >22
##       109
```

```
# Choose first class to be for count less than 28000. This occurs for all classes and is thus equivalen
x.p.df[1,1] = 28000
x.p.df[1,2] = mean(Data$x.p)

# Choose next class to be for count less than 10000. This occurs for all classes other than the (1,1.5]
x.p.df[2,1] = 10000
x.p.df[2,2] = mean(filter(Data, x.p<=1 | x.p>1.5)[,2])

# Choose next class to be for count less than 5000. This occurs for all classes other than the (1,2] cl
x.p.df[3,1] = 5000
x.p.df[3,2] = mean(filter(Data, x.p<=1 | x.p>2)[,2])

# Choose next class to be for count less than 2500. This occurs for all classes other than the (1,2.5]
x.p.df[4,1] = 2500
x.p.df[4,2] = mean(filter(Data, x.p<=1 | x.p>2.5)[,2])

# Choose next class to be for count less than 2000. This occurs for all classes other than the (1,3] cl
x.p.df[5,1] = 2000
x.p.df[5,2] = mean(filter(Data, x.p<=1 | x.p>3)[,2])

# Choose next class to be for count less than 1000. This occurs for all classes other than the (1,3.5]
x.p.df[6,1] = 1000
x.p.df[6,2] = mean(filter(Data, x.p<=1 | x.p>3.5)[,2])

# Choose next class to be for count less than 750. This occurs for all classes other than the (1,4] cla
x.p.df[7,1] = 750
x.p.df[7,2] = mean(filter(Data, x.p<=1 | x.p>4)[,2])

# Choose next class to be for count less than 500. This occurs for all classes other than the (1,4.5] c
x.p.df[8,1] = 500
x.p.df[8,2] = mean(filter(Data, x.p<=1 | x.p>4.5)[,2])

# Choose next class to be for count less than 250. This occurs for all classes other than the (1,6] cla
x.p.df[9,1] = 250
x.p.df[9,2] = mean(filter(Data, x.p<=1 | x.p>6)[,2])
```

```r
# Choose next class to be for count less than 100. This occurs for all classes other than the (1,7.5] c
x.p.df[10,1] = 100
x.p.df[10,2] = mean(filter(Data, x.p<=1 | x.p>7.5)[,2])

x.p.df.plot = ggplot(data=x.p.df, mapping=aes(x=Count, y=Mean)) +
  geom_point() +
  geom_smooth(method="auto", se=FALSE) +
  geom_hline(yintercept=mean(Data$x.p), color="red", linetype="dashed", lwd=0.75)
x.p.df.plot
```

```
## `geom_smooth()` using method = 'loess' and
## formula = 'y ~ x'
```



It is evident from the graph above that for the variable x.p there seems to be an inverse relationship between the mean of the observations in classes that have a count less than a specific number and the count corresponding to that specific number. This means that observations in classes that contain fewer values (i.e. classes that likely contain extreme values) will have a higher mean value. As extreme values are likely to be part of any new sample, this implies that using the sample mean (red dashed line) to make predictions about future observations is not optimal since more extreme values will obtain higher average values, and it would thus be erroneous to claim that they would lie close to the sample mean as it does not represent their extreme nature.

Question 2

1. Load the data stored in the cars.csv as a data frame

```r
Data = read.csv("Car_data.csv", na.strings=c("?", "NA"))
head(Data)
```

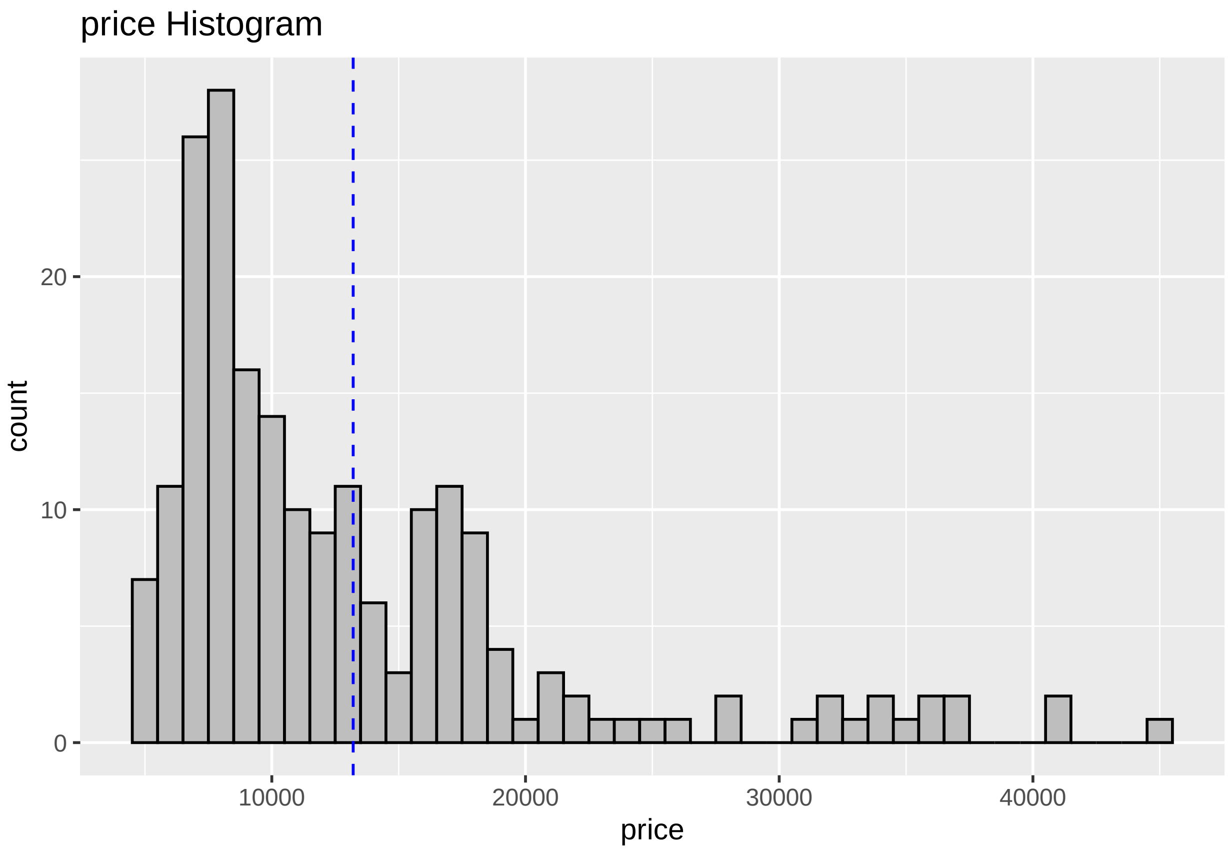
```
##   symboling normalized.losses        make
## 1         3                NA alfa-romero
## 2         3                NA alfa-romero
## 3         1                NA alfa-romero
## 4         2               164        audi
## 5         2               164        audi
## 6         2                NA        audi
##   fuel.type aspiration num.of.doors
## 1       gas        std          two
## 2       gas        std          two
## 3       gas        std          two
## 4       gas        std         four
## 5       gas        std         four
## 6       gas        std          two
##    body.style drive.wheels engine.location
## 1 convertible          rwd           front
## 2 convertible          rwd           front
## 3   hatchback          rwd           front
## 4       sedan          fwd           front
## 5       sedan          4wd           front
## 6       sedan          fwd           front
##   wheel.base length width height curb.weight
## 1       88.6  168.8  64.1   48.8        2548
## 2       88.6  168.8  64.1   48.8        2548
## 3       94.5  171.2  65.5   52.4        2823
## 4       99.8  176.6  66.2   54.3        2337
## 5       99.4  176.6  66.4   54.3        2824
## 6       99.8  177.3  66.3   53.1        2507
##   engine.type num.of.cylinders engine.size
## 1        dohc             four         130
## 2        dohc             four         130
## 3        ohcv              six         152
## 4         ohc             four         109
## 5         ohc             five         136
## 6         ohc             five         136
##   fuel.system bore stroke compression.ratio
## 1        mpfi 3.47   2.68               9.0
## 2        mpfi 3.47   2.68               9.0
## 3        mpfi 2.68   3.47               9.0
## 4        mpfi 3.19   3.40              10.0
## 5        mpfi 3.19   3.40               8.0
## 6        mpfi 3.19   3.40               8.5
##   horsepower peak.rpm city.mpg highway.mpg
## 1        111     5000       21          27
## 2        111     5000       21          27
## 3        154     5000       19          26
## 4        102     5500       24          30
## 5        115     5500       18          22
## 6        110     5500       19          25
##   price
## 1 13495
## 2 16500
## 3 16500
## 4 13950
## 5 17450
## 6 15250
```

Remove all missing values from the variable price

```
Data.drop = Data[-which(is.na(Data$price)),]
```

2. Histogram of the variable price

```
price.hist = ggplot(data=Data.drop, mapping=aes(x=price)) +
  geom_histogram(binwidth=1000, fill="grey", color="black",) +
  ggtitle("price Histogram") +
  geom_vline(aes(xintercept=mean(Data.drop$price)), color="blue", linetype="dashed")

price.hist
```



3. We first create a new data frame called Data.cols that only contains the columns curb.weight, engine.size, horsepower, highway.mpg and price.

```
Data.cols = Data.drop[, c("curb.weight", "engine.size", "horsepower", "highway.mpg", "price")]

# Check if there are any missing values in Data.cols
sum(is.na(Data.cols))
```

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

```
# Check which columns contain these missing values
names(which(colSums(is.na(Data.cols)) > 0))
```

```
## [1] "horsepower"
```

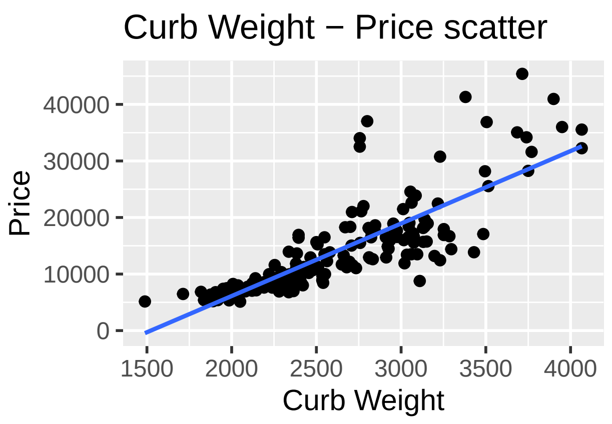
```
# Remove rows containing missing values from the data frame
Data.cols = Data.cols[-which(is.na(Data.cols$horsepower)), ]

# Check that there are no more missing values
sum(is.na(Data.cols))
```

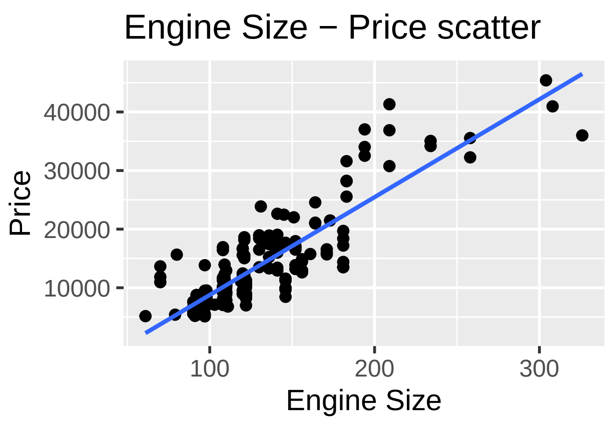
```
## [1] 0
```

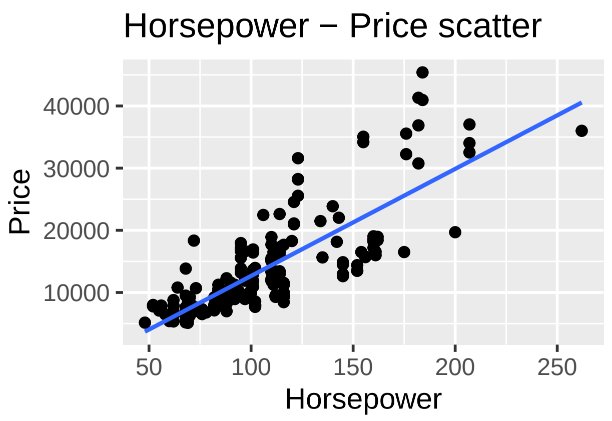
Plot graphs for the relationships between curb.weight, engine.size, horsepower, highway.mpg and price independently

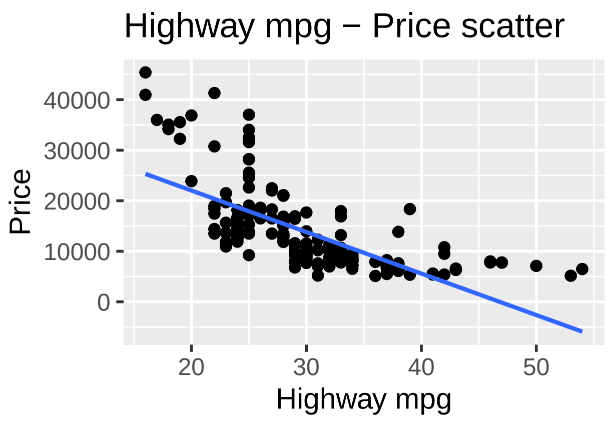
```
# Curb Weight against Price scatter
weight.price = ggplot(data=Data.cols, mapping=aes(x=curb.weight, y=price)) +
  geom_point() +
  labs(x="Curb Weight", y="Price") +
  ggtitle("Curb Weight - Price scatter") +
  geom_smooth(method='lm', se=FALSE, linewidth=0.75)

# Engine Size against Price scatter
esize.price = ggplot(data=Data.cols, mapping=aes(x=engine.size, y=price)) +
  geom_point() +
  labs(x="Engine Size", y="Price") +
  ggtitle("Engine Size - Price scatter") +
  geom_smooth(method='lm', se=FALSE, linewidth=0.75)

# Horsepower against Price scatter
horsepower.price = ggplot(data=Data.cols, mapping=aes(x=horsepower, y=price)) +
  geom_point() +
  labs(x="Horsepower", y="Price") +
  ggtitle("Horsepower - Price scatter") +
  geom_smooth(method='lm', se=FALSE, linewidth=0.75)

# Highway MPG against Price scatter
mpg.price = ggplot(data=Data.cols, mapping=aes(x=highway.mpg, y=price)) +
  geom_point() +
  labs(x="Highway mpg", y="Price") +
  ggtitle("Highway mpg - Price scatter") +
  geom_smooth(method='lm', se=FALSE, linewidth=0.75)


grid.arrange(weight.price, esize.price, horsepower.price, mpg.price)
```

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```









```r
# Calculate the correlation coefficient between each pair of variables
weight.price.cor = cor(Data.cols$curb.weight, Data.cols$price)
esize.price.cor = cor(Data.cols$engine.size, Data.cols$price)
horsepower.price.cor = cor(Data.cols$horsepower, Data.cols$price)
mpg.price.cor = cor(Data.cols$highway.mpg, Data.cols$price)

c(weight.price.cor, esize.price.cor, horsepower.price.cor, mpg.price.cor)
```

```
## [1]  0.8350904  0.8738870  0.8105331 -0.7052299
```

As can be seen from both the graphs above and the correlation coefficients, strong relationships exist between each pair of variables. More specifically, there seems to be a very strong positive relationship between the weight of the car and its price, the engine size of the car and its price, and the horsepower of the car and its price. This means that cars that weigh more will tend to cost more, cars that have a large engine will tend to cost more, and cars that have high horsepower also tend to cost more. Conversely, there seems to be a negative relationship between highway mpg and price. This means that cars that have a high mpg will cost less but those with low mpg will cost more.

4. Firstly create a new data frame called Data.zscores that will contain the standardised scores of the data stored in the curb.weight, engine.size, horsepower, and highway.mpg columns of the Data.cols data frame

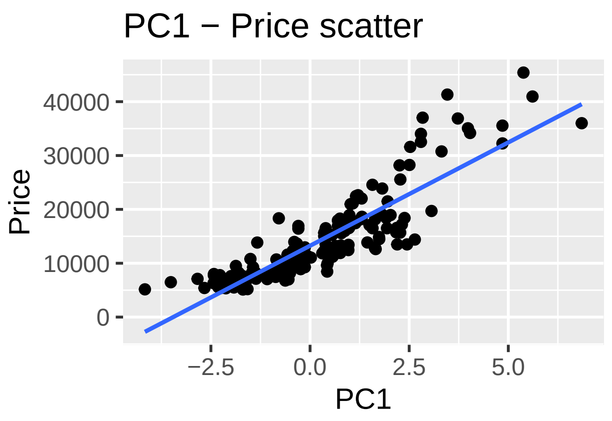
```r
# Create the data frame
Data.zscores = Data.cols[,-5]

# Standardise data stored in each column
Data.zscores$curb.weight = (Data.zscores$curb.weight - mean(Data.zscores$curb.weight)) / sd(Data.zscores
Data.zscores$engine.size = (Data.zscores$engine.size - mean(Data.zscores$engine.size)) / sd(Data.zscores
Data.zscores$horsepower = (Data.zscores$horsepower - mean(Data.zscores$horsepower)) / sd(Data.zscores$h
Data.zscores$highway.mpg = (Data.zscores$highway.mpg - mean(Data.zscores$highway.mpg)) / sd(Data.zscores

# Perform PCA
Data.PCA = prcomp(Data.zscores, center=FALSE)
Data.PCA$rotation
```

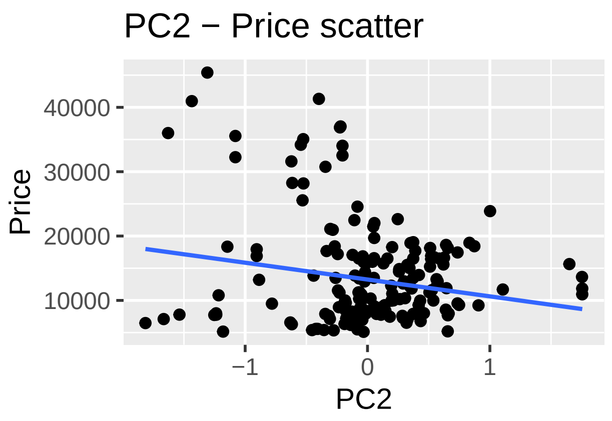
```
##                    PC1         PC2        PC3
## curb.weight  0.5073415 -0.18580359 -0.6571533
## engine.size  0.4999754 -0.63617369  0.1031988
## horsepower   0.5045853  0.09956758  0.7262252
## highway.mpg -0.4878759 -0.74219024  0.1734832
##                    PC4
## curb.weight -0.5255770
## engine.size  0.5784960
## horsepower  -0.4561544
## highway.mpg -0.4254813
```

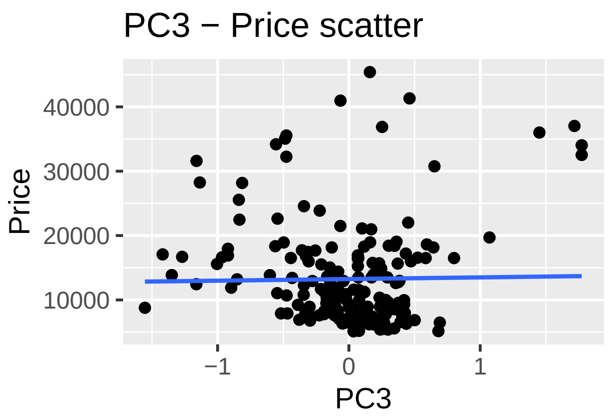
5. To investigate the relationship between each of the principal components and price, we can create scatter plots for each of the principal components and price. To do this, we firstly create a new data frame called Data.PCA.price, which includes all the data contained in the Data.PCA data frame as well as the data contained in the price column of the Data.cols data frame.

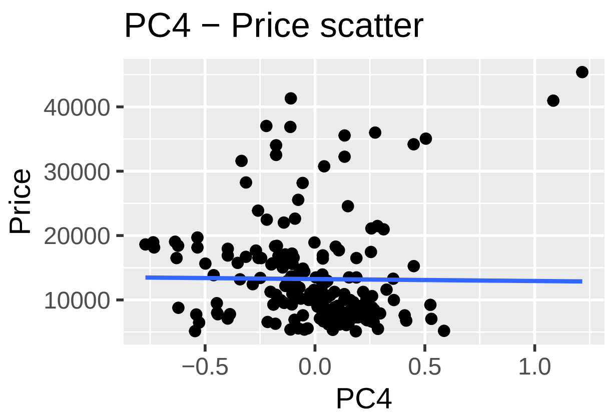
```
# Create new data frame and change column names
Data.PCA.price = data.frame(Data.PCA$x, Data.cols$price)
colnames(Data.PCA.price) <- c("PC1", "PC2", "PC3", "PC4", "price")

# Create scatter plot of PC1 against price
PC1.price = ggplot(data=Data.PCA.price, mapping=aes(x=PC1, y=price)) +
  geom_point() +
  labs(x="PC1", y="Price") +
  ggtitle("PC1 - Price scatter") +
  geom_smooth(method='lm', se=FALSE, linewidth=0.75)

# Create scatter plot of PC2 against price
PC2.price = ggplot(data=Data.PCA.price, mapping=aes(x=PC2, y=price)) +
  geom_point() +
  labs(x="PC2", y="Price") +
  ggtitle("PC2 - Price scatter") +
  geom_smooth(method='lm', se=FALSE, linewidth=0.75)

# Create scatter plot of PC3 against price
PC3.price = ggplot(data=Data.PCA.price, mapping=aes(x=PC3, y=price)) +
  geom_point() +
  labs(x="PC3", y="Price") +
  ggtitle("PC3 - Price scatter") +
  geom_smooth(method='lm', se=FALSE, linewidth=0.75)

# Create scatter plot of PC4 against price
PC4.price = ggplot(data=Data.PCA.price, mapping=aes(x=PC4, y=price)) +
  geom_point() +
  labs(x="PC4", y="Price") +
  ggtitle("PC4 - Price scatter") +
  geom_smooth(method='lm', se=FALSE, linewidth=0.75)

grid.arrange(PC1.price, PC2.price, PC3.price, PC4.price)
```

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```


PC1 − Price scatter
Price
PC1


PC2 − Price scatter
Price
PC2


PC3 − Price scatter
Price
PC3


PC4 − Price scatter
Price
PC4