CIGARETTE CONSUMPTION

Mariama Soumahoro

O B J E C T I V E

The article presented to us has for purpose to research the cigarette consumption in the United States between 1963 and 1992. The study is made on 1380 observations that are regional. Our goal will be to understand and figure out the relation between all the covariates and understand how each variable contribute to the sales of cigarettes packs. It is also in our interest to find and develop a model to predict how much sales would be made during a period of time.

I N T R O D U C T I O N

Cigarette has been present in the human life for a very long time. The components of most cigarettes are tobacco, chemical additives, a filter, and paper wrapping. Although responsible for most of all tobacco-related disease and death in the United States, about 2,000 teenagers who are under the age of 18 smoke their first cigarette and more than 300 youth under the age of 18 become daily smokers. There are still around 38 million people in the U.S. who smoke cigarettes every day. This is around 15.5 percent of the entire population. Smoking is mainly observed among males than females. Cigarette consumption is a big market in the United States that generates a lot of capital (money) even though it has decreased due to different factors. This is the essence of our research. Indeed, a study was done between 1963 and 1992 in different states to learn more about the sales of the cigarette packs. It will be our duty to figure out the most important factors and if possible, make some predictions.

1.1 Data Collection

The data presented to us has been collected by a panel of 46 states in the United States. The gathering and observation period of the data was from 1963 to 1992. The data frame consists of 9 characteristics that we will be using for our analysis. The objective is to determine the number of sales throughout the period of observation.

1.2. Definitions

The following data contains 1380 observations with 9 variables which are mentioned below:

- state: State abbreviation (numbers)
- year: The year
- price: Price per pack of cigarettes
- pop: Population
- pop16: Population above the age of 16
- cpi: Consumer price index (1983=100)
- ndi: Per capita disposable income
- sales: Cigarette sales in packs per capita
- pimin: Minimum price in adjoining states per pack of cigarettes

2.DATA V I S U A L I S A T I O N A N D E X P L O R A T I O N

With this section, we dive into the data and take a closer look at the information given in order get more familiar. We start by checking the number of observations, whether we are missing any information and so much more.

```
library("openxlsx")
cigar <- read.xlsx("/Users/mariamasoumahoro/Library/Containers/com.microsoft.Excel/Data/Desktop/cigar da
sum(is.na(cigar)) # number of total missing values
```

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

```
nrow(cigar) # sample size
```

```
## [1] 1380
```

```
ncol(cigar) # number of columns
```

```
## [1] 9
```

```
head(cigar)#head of the data(column and rows)
```

```
##   state year price  pop  pop16  cpi      ndi sales pimin
## 1     1   63  28.6 3383 2236.5 30.6 1558.305  93.9  26.1
## 2     1   64  29.8 3431 2276.7 31.0 1684.073  95.4  27.5
## 3     1   65  29.8 3486 2327.5 31.5 1809.842  98.5  28.9
## 4     1   66  31.5 3524 2369.7 32.4 1915.160  96.4  29.5
## 5     1   67  31.6 3533 2393.7 33.4 2023.546  95.5  29.6
## 6     1   68  35.6 3522 2405.2 34.8 2202.486  88.4  32.0
```

We have 1380 observation (sample size), with 9 variables(number of columns). We do not have any missing data. Therefore, no column and row were removed.

str function gives a look at the different data types in the "cigar" dataset.

```
str(cigar)
```

```
## 'data.frame':    1380 obs. of  9 variables:
##  $ state: num  1 1 1 1 1 1 1 1 1 1 ...
##  $ year : num  63 64 65 66 67 68 69 70 71 72 ...
##  $ price: num  28.6 29.8 29.8 31.5 31.6 35.6 36.6 39.6 42.7 42.3 ...
##  $ pop  : num  3383 3431 3486 3524 3533 ...
##  $ pop16: num  2236 2277 2328 2370 2394 ...
##  $ cpi  : num  30.6 31 31.5 32.4 33.4 34.8 36.7 38.8 40.5 41.8 ...
##  $ ndi  : num  1558 1684 1810 1915 2024 ...
##  $ sales: num  93.9 95.4 98.5 96.4 95.5 ...
##  $ pimin: num  26.1 27.5 28.9 29.5 29.6 32 32.8 34.3 35.8 37.4 ...
```

All the variables are numerical as expected.

Now, we fit the whole data using multiple linear regression with `sales` as y and all the other covariates as x to check any possibility of collinearity and multicollinearity in our data set.

```
fit.full <- lm(sales~., data=cigar)
summary(fit.full)
```

```
##
## Call:
## lm(formula = sales ~ ., data = cigar)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -62.766 -15.919  -1.827   9.259 160.766
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 97.7246159 26.3883231   3.703 0.000221 ***
## state       -0.2193838  0.0503077  -4.361 1.39e-05 ***
## year         0.7666884  0.4330772   1.770 0.076895 .
## price       -1.5328148  0.1145436 -13.382  < 2e-16 ***
## pop         -0.0012841  0.0028005  -0.459 0.646635
## pop16        0.0007384  0.0037419   0.197 0.843589
## cpi         -0.0220015  0.1339032  -0.164 0.869512
## ndi          0.0057626  0.0005960   9.668  < 2e-16 ***
## pimin        0.6294724  0.1270127   4.956 8.09e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.77 on 1371 degrees of freedom
## Multiple R-squared:  0.258,  Adjusted R-squared:  0.2537
## F-statistic: 59.59 on 8 and 1371 DF,  p-value: < 2.2e-16
```

```
car::vif(fit.full)
```

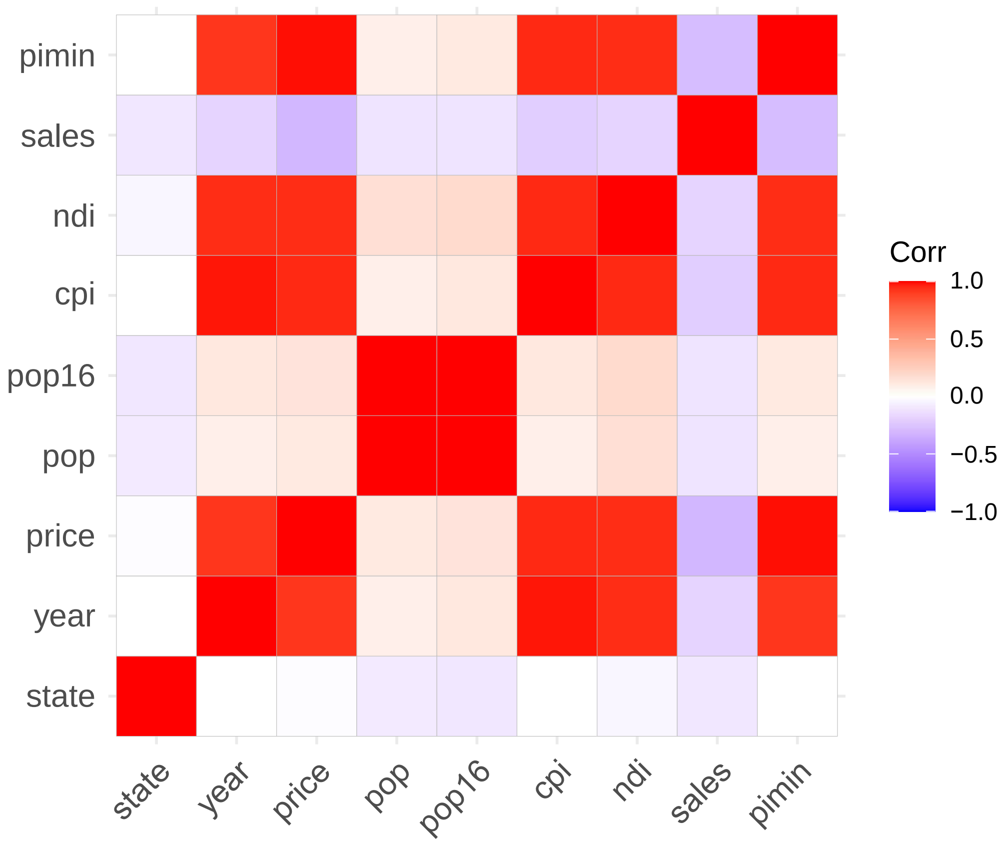
```
##      state       year      price        pop      pop16        cpi        ndi
##   1.020931  27.050832  44.495178 351.802309 357.252387  46.028001  15.406964
##      pimin
##  45.579855
```

Looking at the summary of the above table, we observe that the VIF score of each covariate is very high besides the state's VIF. This is an indication that there is colinearity between the variables. Therefore, we investigate it a little more by checking the correlation between the predictors.

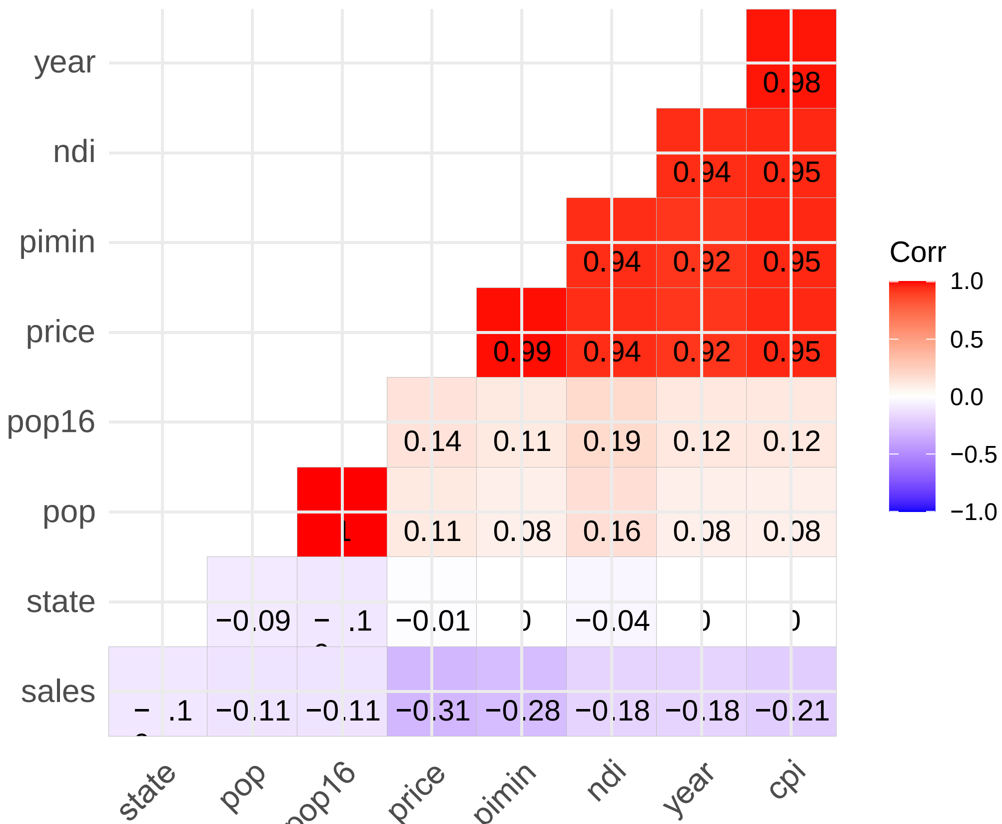
```
# select numeric variables
df <- dplyr::select_if(cigar, is.numeric)

# calulate the correlations
r <- cor(df, use="complete.obs")
round(r,2)
```

```
##       state  year price   pop pop16   cpi   ndi sales pimin
## state  1.00  0.00 -0.01 -0.09 -0.10  0.00 -0.04 -0.10  0.00
## year   0.00  1.00  0.92  0.08  0.12  0.98  0.94 -0.18  0.92
## price -0.01  0.92  1.00  0.11  0.14  0.95  0.94 -0.31  0.99
## pop   -0.09  0.08  0.11  1.00  1.00  0.08  0.16 -0.11  0.08
## pop16 -0.10  0.12  0.14  1.00  1.00  0.12  0.19 -0.11  0.11
## cpi    0.00  0.98  0.95  0.08  0.12  1.00  0.95 -0.21  0.95
## ndi   -0.04  0.94  0.94  0.16  0.19  0.95  1.00 -0.18  0.94
## sales -0.10 -0.18 -0.31 -0.11 -0.11 -0.21 -0.18  1.00 -0.28
## pimin  0.00  0.92  0.99  0.08  0.11  0.95  0.94 -0.28  1.00
```

```
library(ggplot2)
library(ggcorrplot)
ggcorrplot(r)
```



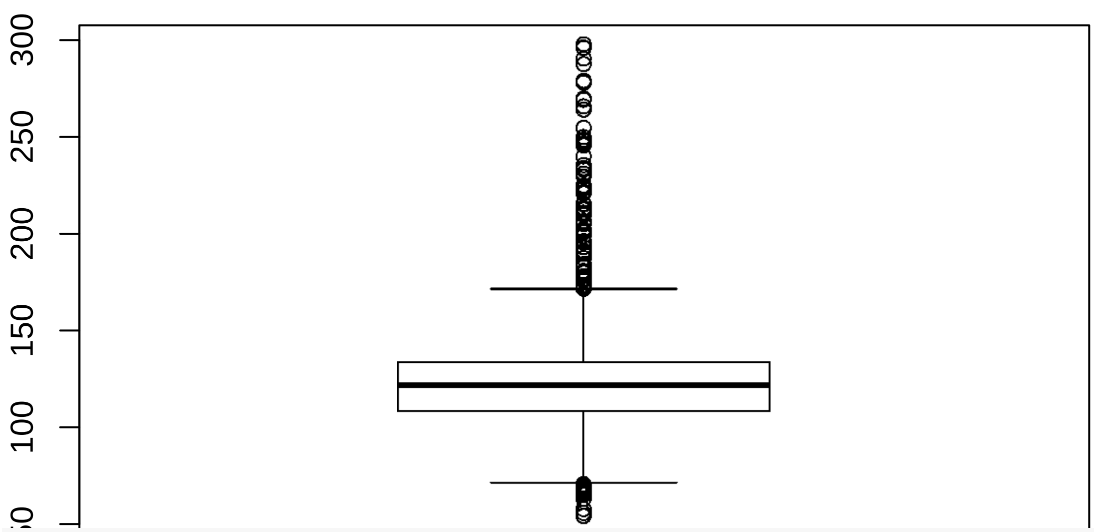

```r
ggcorrplot(r,
           hc.order = TRUE,
           type = "lower",
           lab = TRUE)
```



There is a high correlation between the pair: year-cpi, ndi-cpi, pimin-cpi, price-cpi, ndi-year, pimin-year, price-year, pimin-ndi, price-ndi, price-pimin, and pop-pop16. All those covariates go hand to hand, they are important. Therefore, for the sake of our study, we decide to keep them in the data.

The next step is to check for outliers in the data using the function "boxplot" in R. In case we do encounter outliers, we will be removing them.

```
boxplot(cigar$sales)
```




```
# You can get the actual values of the outliers with this
boxplot(cigar$sales, plot=FALSE)$out
```

```
##   [1]  68.7  67.5 175.5 246.4 235.3 246.5 295.9 249.8 224.3 212.2 200.4 213.0
##  [13] 220.6 209.4 182.7 176.5 173.0 179.4 201.9 212.4 223.0 230.9 229.4 224.7
##  [25] 214.9 215.3 209.7 210.6 201.1 183.2 182.4 179.8 171.2 173.2 171.6 182.5
##  [37] 212.7 206.3 192.7 184.0 175.8 189.7 198.6 189.5 190.5 198.6 201.5 204.7
##  [49] 205.2 201.4 190.8 187.0 183.3 177.7 171.9 221.4 223.9 233.8 287.6 297.9
##  [61] 264.0 248.5 265.7 278.0 296.2 279.0 269.8 269.1 290.5 278.8 269.6 254.6
##  [73] 247.8 245.4 239.8 232.9 215.1 201.1 195.9 195.1 180.4 172.9  69.9  62.3
##  [85]  65.0  65.7  64.3  64.3  65.6  65.5  67.7  69.0  66.3  66.5  64.4  67.7
##  [97]  55.0  57.0  53.4  53.5  55.0
```

We record a total of 101 outliers. We then look at the rows that contain the different outliers and we delete them to better clean the data.

```r
# Now you can assign the outlier values into a vector

outliers <- boxplot(cigar$sales, plot=FALSE)$out

# Check the results

print(outliers)
```

```
##   [1]  68.7  67.5 175.5 246.4 235.3 246.5 295.9 249.8 224.3 212.2 200.4 213.0
##  [13] 220.6 209.4 182.7 176.5 173.0 179.4 201.9 212.4 223.0 230.9 229.4 224.7
##  [25] 214.9 215.3 209.7 210.6 201.1 183.2 182.4 179.8 171.2 173.2 171.6 182.5
##  [37] 212.7 206.3 192.7 184.0 175.8 189.7 198.6 189.5 190.5 198.6 201.5 204.7
##  [49] 205.2 201.4 190.8 187.0 183.3 177.7 171.9 221.4 223.9 233.8 287.6 297.9
##  [61] 264.0 248.5 265.7 278.0 296.2 279.0 269.8 269.1 290.5 278.8 269.6 254.6
##  [73] 247.8 245.4 239.8 232.9 215.1 201.1 195.9 195.1 180.4 172.9  69.9  62.3
##  [85]  65.0  65.7  64.3  64.3  65.6  65.5  67.7  69.0  66.3  66.5  64.4  67.7
##  [97]  55.0  57.0  53.4  53.5  55.0
```

```r
# First you need find in which rows the outliers are
cigar[which(cigar$sales %in% outliers),]
```

```
##     state year price     pop   pop16   cpi       ndi sales pimin
## 119     5   91 186.8 30218.8 22694.0 136.2 17705.000  68.7 151.4
## 120     5   92 201.9 30703.3 22920.0 140.3 18495.000  67.5 165.7
## 157     8   69  32.2   540.0   362.8  36.7  3287.320 175.5  30.7
## 181     9   63  23.4   792.0   563.1  30.6  2733.227 246.4  24.7
## 182     9   64  23.9   795.0   559.3  31.0  2894.005 235.3  25.2
## 183     9   65  24.1   802.0   560.2  31.5  3068.443 246.5  25.1
## 184     9   66  24.1   806.0   559.3  32.4  3215.561 295.9  24.7
## 185     9   67  26.6   808.0   559.3  33.4  3479.321 249.8  26.3
## 186     9   68  26.7   802.0   554.4  34.8  3752.538 224.3  27.1
## 187     9   69  27.2   798.0   561.2  36.7  3907.011 212.2  28.3
## 188     9   70  28.5   756.0   563.1  38.8  4202.296 200.4  28.8
## 189     9   71  32.6   750.0   560.2  40.5  4600.564 213.0  30.2
## 190     9   72  33.7   744.0   558.3  41.8  4964.153 220.6  29.9
## 191     9   73  34.5   734.0   553.4  44.4  5310.929 209.4  30.1
## 192     9   74  36.0   721.0   547.6  49.3  5894.144 182.7  31.3
## 193     9   75  39.4   711.0   543.7  53.8  6587.696 176.5  33.6
## 345    15   77  40.6  5352.0  3960.0  60.6  6012.643 173.0  36.9
## 430    18   72  30.6  3303.0  2360.3  41.8  3239.106 179.4  29.9
## 431    18   73  30.6  3325.0  2396.8  44.4  3653.971 201.9  30.1
## 432    18   74  31.5  3356.0  2440.0  49.3  4052.033 212.4  31.3
## 433    18   75  33.3  3392.0  2484.2  53.8  4380.775 223.0  33.6
```

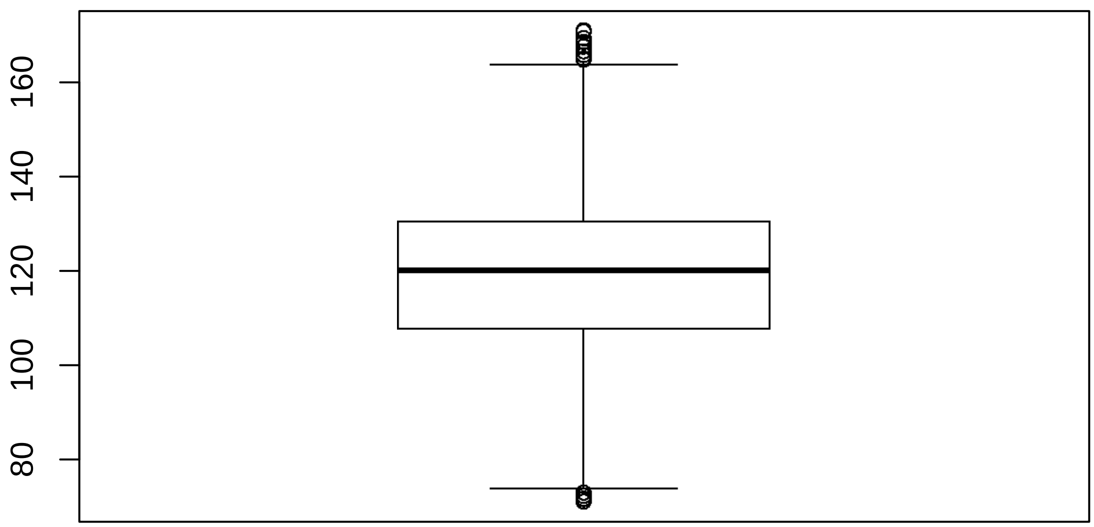
##	434	18	76	36.0	3439.0	2537.0	56.9	4802.993	230.9	37.9
##	435	18	77	36.9	3468.0	2574.5	60.6	5232.563	229.4	38.4
##	436	18	78	41.4	3490.0	2603.3	65.2	5753.508	224.7	42.8
##	437	18	79	43.4	3527.0	2641.7	72.6	6409.942	214.9	45.8
##	438	18	80	46.3	3661.0	2737.8	82.4	6775.444	215.3	48.5
##	439	18	81	49.4	3662.0	2745.5	90.9	7506.448	209.7	51.8
##	440	18	82	56.3	3667.0	2757.0	96.5	7988.533	210.6	56.4
##	441	18	83	66.4	3714.0	2800.2	99.6	8272.113	201.1	68.8
##	442	18	84	75.4	3723.0	2812.7	103.9	9062.984	183.2	76.0
##	443	18	85	79.3	3728.0	2829.1	107.6	9282.495	182.4	83.6
##	444	18	86	85.4	3726.0	2841.6	109.6	9722.568	179.8	91.3
##	445	18	87	90.5	3727.0	2855.0	113.6	10328.587	171.2	94.6
##	446	18	88	94.4	3727.0	2867.0	118.3	11089.000	173.2	102.1
##	447	18	89	103.8	3727.0	2874.0	124.0	11873.000	171.6	109.4
##	448	18	90	115.6	3735.1	2880.3	130.7	12879.000	182.5	128.6
##	751	29	63	29.9	391.0	263.2	30.6	2836.837	212.7	23.9
##	752	29	64	29.5	418.0	280.6	31.0	2919.303	206.3	24.0
##	753	29	65	29.7	434.0	289.3	31.5	3016.063	192.7	24.2
##	754	29	66	29.9	435.0	287.3	32.4	3160.104	184.0	25.5
##	755	29	67	30.2	436.0	287.3	33.4	3332.733	175.8	26.0
##	756	29	68	32.8	449.0	296.0	34.8	3639.508	189.7	31.3
##	757	29	69	33.3	457.0	302.8	36.7	3897.902	198.6	31.9
##	758	29	70	38.9	488.0	341.3	38.8	4377.305	189.5	33.8
##	759	29	71	44.0	514.0	361.6	40.5	4597.215	190.5	33.6
##	760	29	72	40.6	535.0	380.9	41.8	4786.337	198.6	33.7
##	761	29	73	40.3	551.0	395.3	44.4	5256.945	201.5	36.3
##	762	29	74	41.9	573.0	414.6	49.3	5525.235	204.7	37.8
##	763	29	75	44.5	590.0	431.0	53.8	6103.597	205.2	40.3
##	764	29	76	44.9	610.0	450.3	56.9	6572.005	201.4	42.5
##	765	29	77	49.3	634.0	472.5	60.6	7269.119	190.8	44.7
##	766	29	78	54.3	666.0	500.4	65.2	8207.035	187.0	49.5
##	767	29	79	57.1	702.0	530.3	72.6	9016.303	183.3	53.7
##	768	29	80	63.1	799.0	616.2	82.4	9886.046	177.7	56.4
##	769	29	81	63.3	845.0	651.8	90.9	10682.120	171.9	59.2
##	781	30	63	24.2	646.0	446.2	30.6	2320.244	221.4	26.7
##	782	30	64	24.7	659.0	454.9	31.0	2487.419	223.9	26.8
##	783	30	65	24.7	673.0	465.5	31.5	2632.154	233.8	27.2
##	784	30	66	25.9	676.0	468.4	32.4	2845.329	287.6	29.6
##	785	30	67	26.5	691.0	480.0	33.4	3025.967	297.9	30.3
##	786	30	68	29.9	703.0	489.6	34.8	3259.338	264.0	32.0
##	787	30	69	29.9	717.0	505.0	36.7	3493.830	248.5	33.4
##	788	30	70	31.4	737.0	517.6	38.8	3659.883	265.7	37.7
##	789	30	71	34.1	759.0	537.8	40.5	3880.912	278.0	38.8
##	790	30	72	36.1	775.0	554.2	41.8	4114.283	296.2	40.0
##	791	30	73	36.9	793.0	571.5	44.4	4600.098	279.0	39.8
##	792	30	74	37.9	805.0	586.9	49.3	4961.374	269.8	41.3
##	793	30	75	40.8	815.0	599.5	53.8	5317.040	269.1	41.8
##	794	30	76	43.9	829.0	615.9	56.9	5830.905	290.5	47.1
##	795	30	77	45.0	850.0	638.0	60.6	6362.721	278.8	47.0
##	796	30	78	49.7	869.0	658.3	65.2	7041.516	269.6	52.5
##	797	30	79	53.2	887.0	676.6	72.6	7863.923	254.6	54.5
##	798	30	80	55.3	921.0	702.6	82.4	8721.112	247.8	58.9
##	799	30	81	58.4	936.0	719.0	90.9	9604.107	245.4	61.0
##	800	30	82	67.0	951.0	733.4	96.5	10404.075	239.8	66.8

```
## 801      30    83   74.7    959.0    742.1   99.6 11282.581 232.9   77.0
## 802      30    84   90.5    977.0    760.4  103.9 12609.878 215.1   90.6
## 803      30    85   89.2    998.0    778.7  107.6 13589.362 201.1   95.5
## 804      30    86  100.0   1027.0    802.8  109.6 14550.895 195.9  104.9
## 805      30    87  102.0   1057.0    825.0  113.6 15902.875 195.1  113.8
## 806      30    88  113.5   1085.0    843.0  118.3 17201.000 180.4  123.7
## 807      30    89  125.9   1107.0    858.0  124.0 17829.000 172.9  129.7
## 869      32    91  146.9   1572.7   1150.5  136.2 12961.000  69.9  149.1
## 1172     45    64   29.4    977.0    594.8   31.0  2181.171  62.3   24.0
## 1173     45    65   29.7    994.0    609.9   31.5  2282.447  65.0   24.2
## 1174     45    66   30.8   1010.0    624.9   32.4  2365.601  65.7   26.5
## 1175     45    67   31.5   1022.0    637.1   33.4  2454.084  64.3   27.4
## 1176     45    68   32.3   1031.0    649.4   34.8  2573.483  64.3   31.3
## 1177     45    69   33.3   1045.0    644.7   36.7  2706.742  65.6   31.9
## 1178     45    70   34.6   1059.0    686.9   38.8  2975.390  65.5   33.8
## 1179     45    71   36.6   1094.0    716.1   40.5  3192.868  67.7   33.6
## 1191     45    83   82.0   1619.0   1045.9   99.6  8251.351  69.0   71.0
## 1192     45    84   95.3   1652.0   1065.7  103.9  8831.291  66.3   81.7
## 1193     45    85  104.6   1644.0   1081.6  107.6  9230.000  66.5   87.4
## 1194     45    86  103.5   1664.0   1099.5  109.6  9647.898  64.4   97.8
## 1195     45    87  108.6   1680.0   1107.0  113.6 10035.946  67.7  102.7
## 1196     45    88  122.9   1690.0   1116.0  118.3 10650.000  55.0  112.9
## 1197     45    89  135.6   1707.0   1131.0  124.0 11425.000  57.0  118.6
## 1198     45    90  151.9   1724.0   1142.3  130.7 12012.000  53.4  129.5
## 1199     45    91  167.1   1771.0   1178.9  136.2 12492.000  53.5  127.0
## 1200     45    92  170.1   1814.1   1214.5  140.3 13355.000  55.0  155.1
```

```r
# Now you can remove the rows containing the outliers, one possible option is:

cigar.new <- cigar[-which(cigar$sales %in% outliers),]
```

We take a look at the boxplot again to see if we have better results.

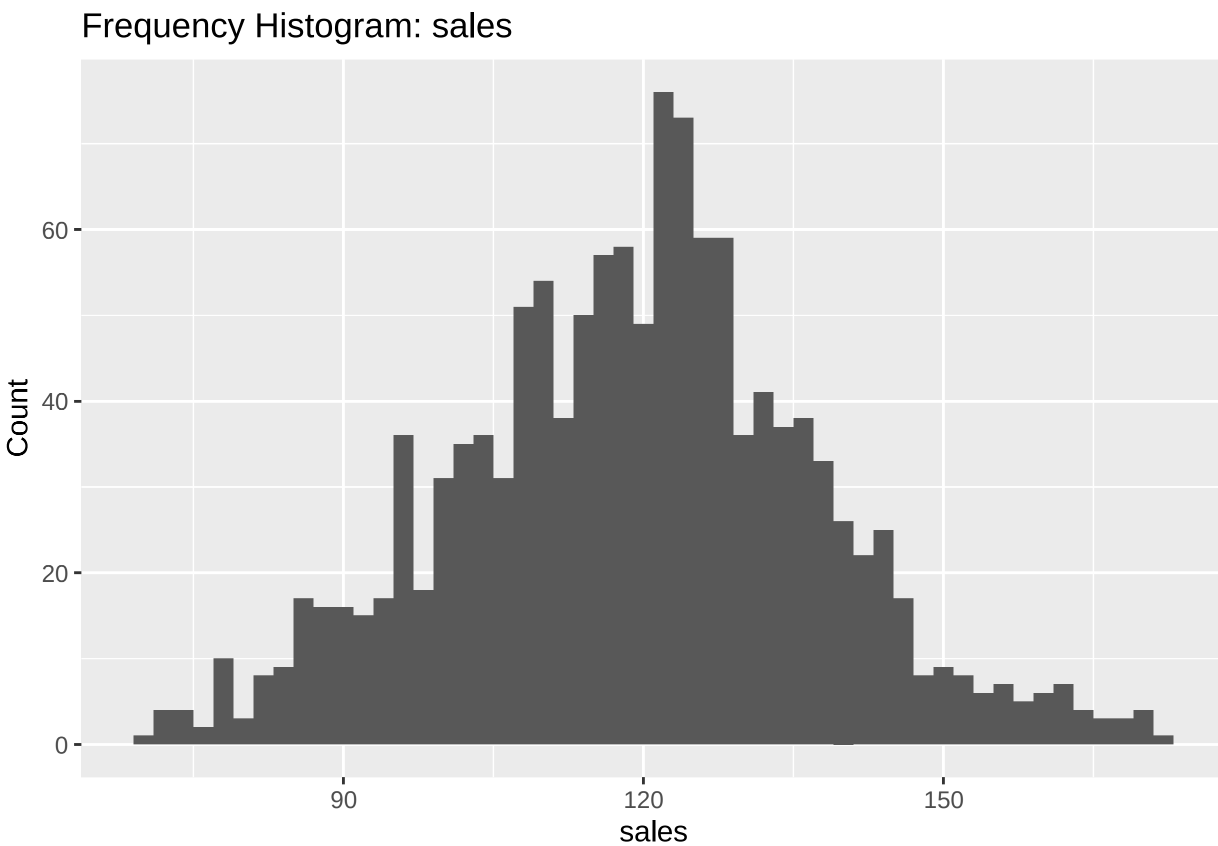
```r
# If you check now with boxplot, you will notice that those pesky outliers are gone

boxplot(cigar.new$sales)
```



Looking at the result, the plot looks better, the remaining points might indicate us that we have leverage points.

In this following section, we look at the distribution for each variable in the dataset by creating histograms using ggplot2's plot function. It will help us begin to identify any relationships between variables that are worth investigating further.

```
# price
library(ggplot2)
qplot(cigar.new$sales, xlab = 'sales', ylab = 'Count', binwidth = 2,
      main='Frequency Histogram: sales')
```

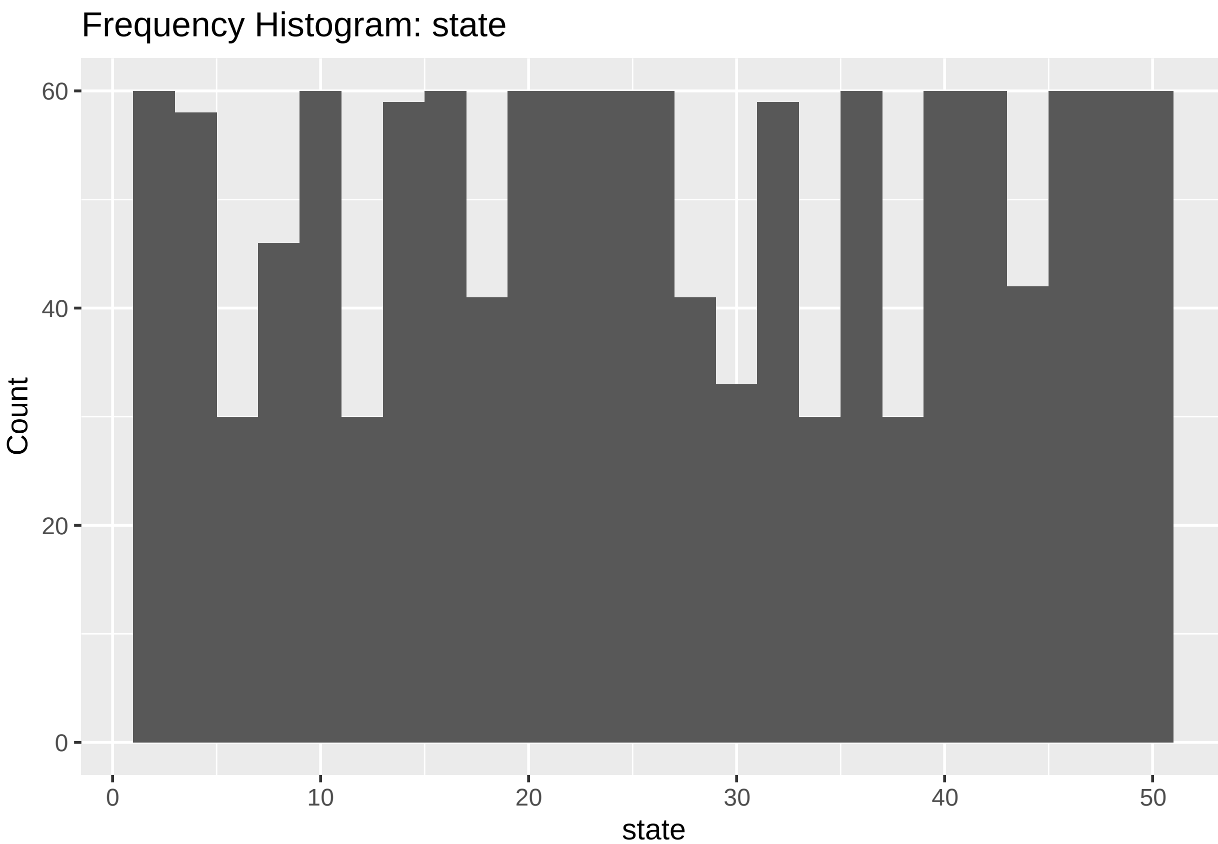



```
summary(cigar.new$sales)
```

```
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     70.8   107.7   120.0   119.1   130.4   171.1
```

Sales are roughly normally distributed, with the mean and median both around $120.0.

```
# year
qplot(cigar.new$state, xlab = 'state', ylab = 'Count', binwidth = 2,
      main='Frequency Histogram: state')
```




```
summary(cigar.new$state)
```

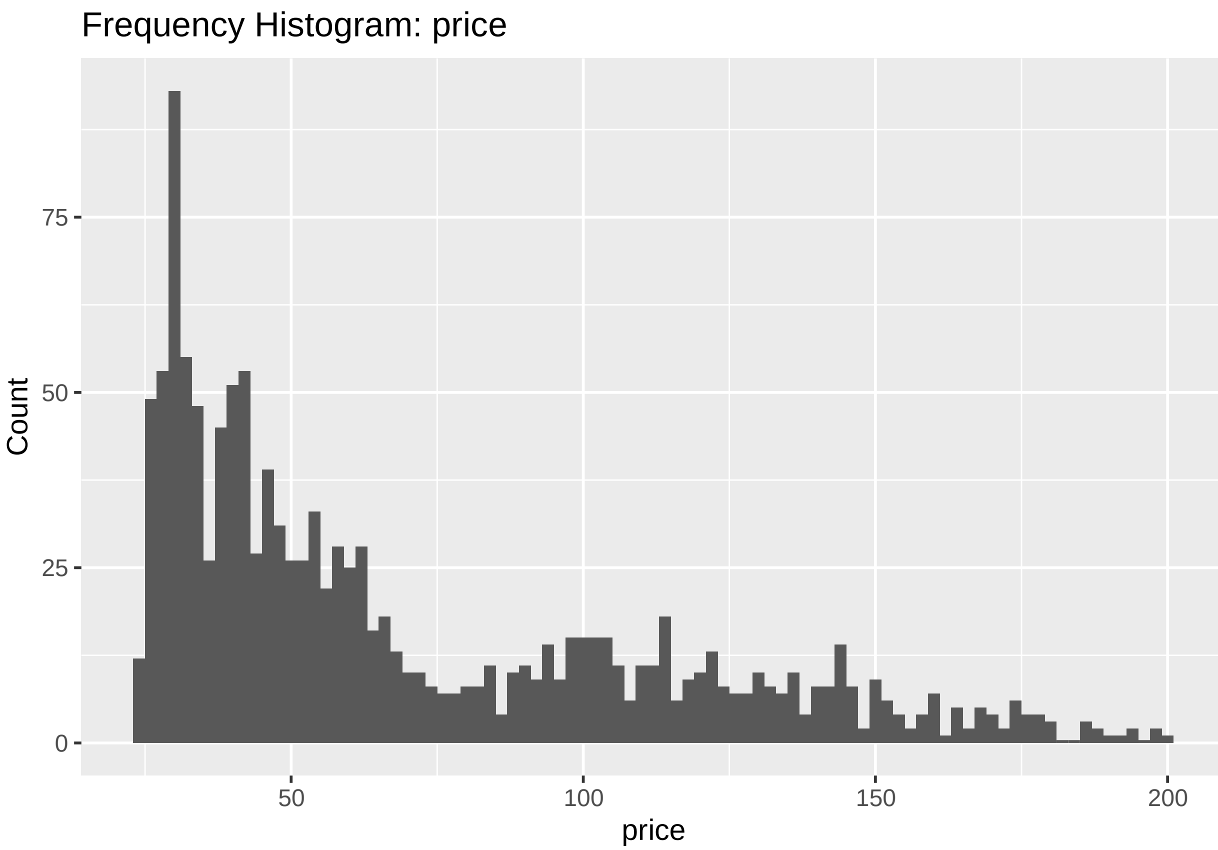
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00   15.00   26.00   26.84   40.00   51.00
```

```
table(cigar.new$state)
```

```
##
##  1  3  4  5  7  8  9 10 11 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
## 30 30 30 28 30 29 17 30 30 30 30 29 30 30 11 30 30 30 30 30 30 30 30 30 30 11
## 30 31 32 33 35 36 37 39 40 41 42 43 44 45 46 47 48 49 50 51
##  3 30 29 30 30 30 30 30 30 30 30 30 30 12 30 30 30 30 30 30
```

All the stats have cigarette counts around 30.

```
# year
qplot(cigar.new$price, xlab = 'price', ylab = 'Count', binwidth = 2,
      main='Frequency Histogram: price')
```

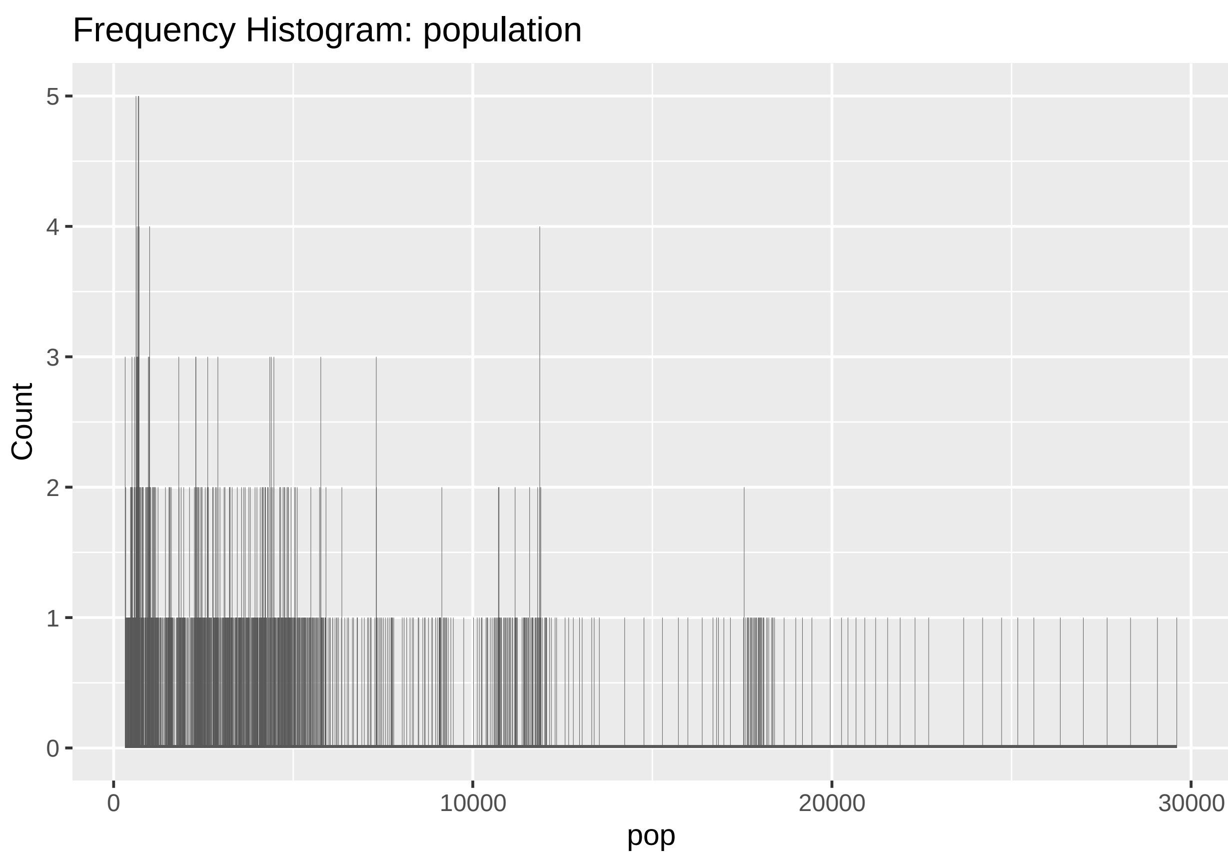



```r
summary(cigar.new$price)
```

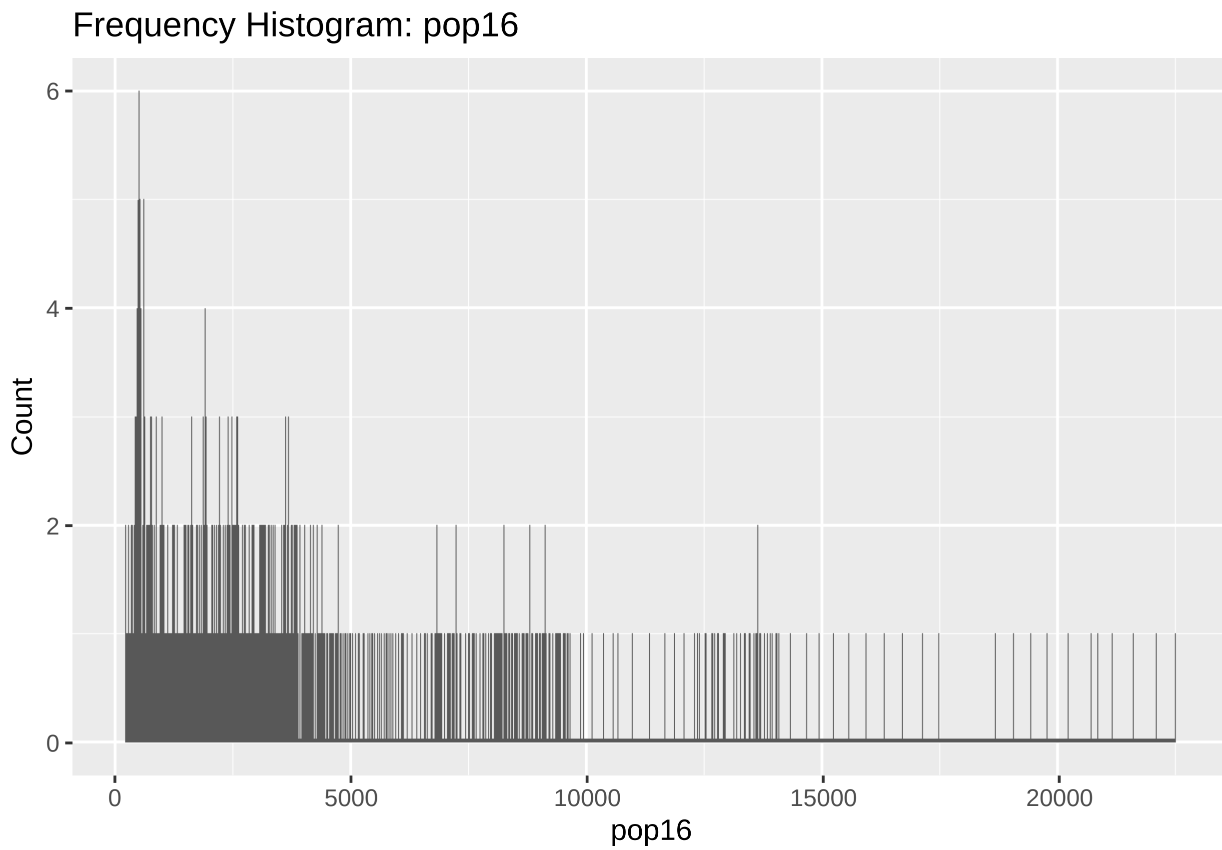
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   23.90   35.90   53.30   69.51   99.25  199.20
```

The distribution for Sales is skewed right with a longer tail toward the higher end of the scale. The median for price is 53.30 and the mean is 69.51.

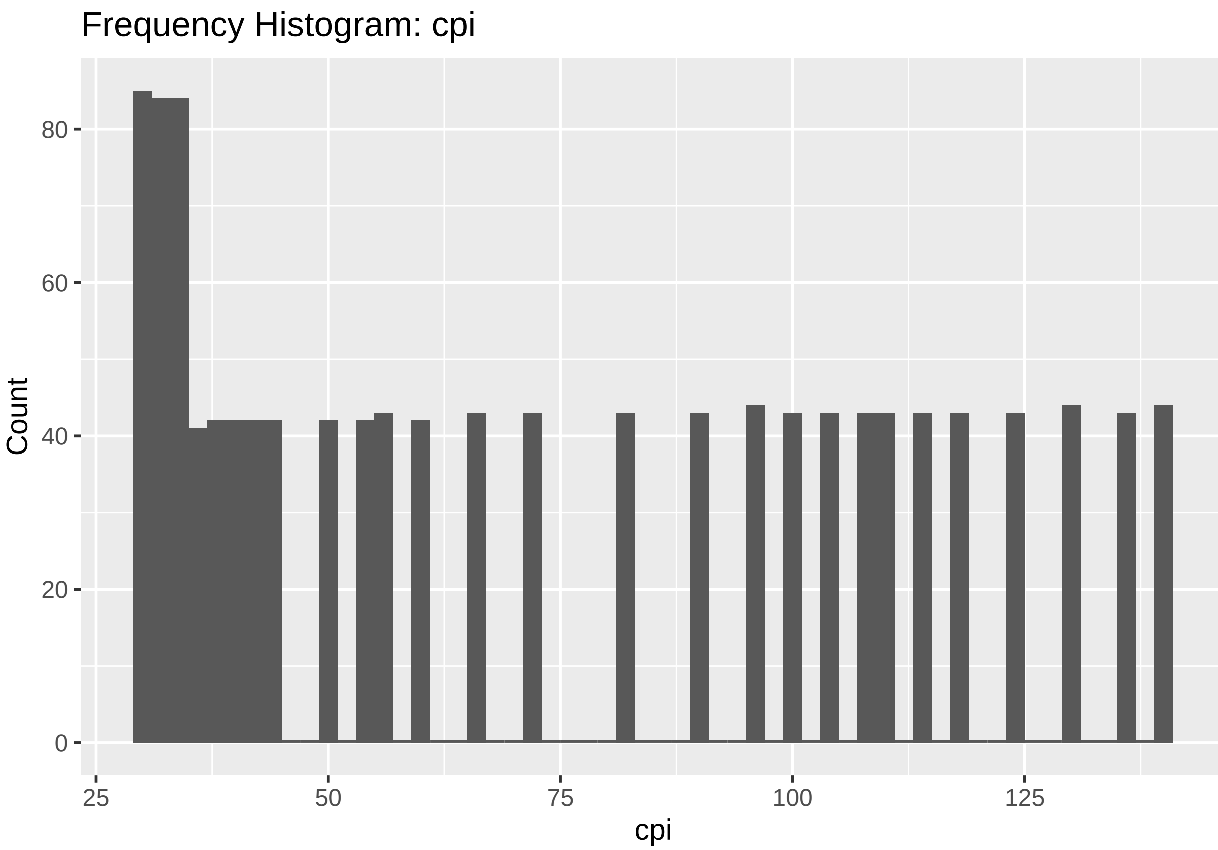
```r
# pop
qplot(cigar.new$pop, xlab = 'pop', ylab = 'Count', binwidth = 2,
      main='Frequency Histogram: population')
```



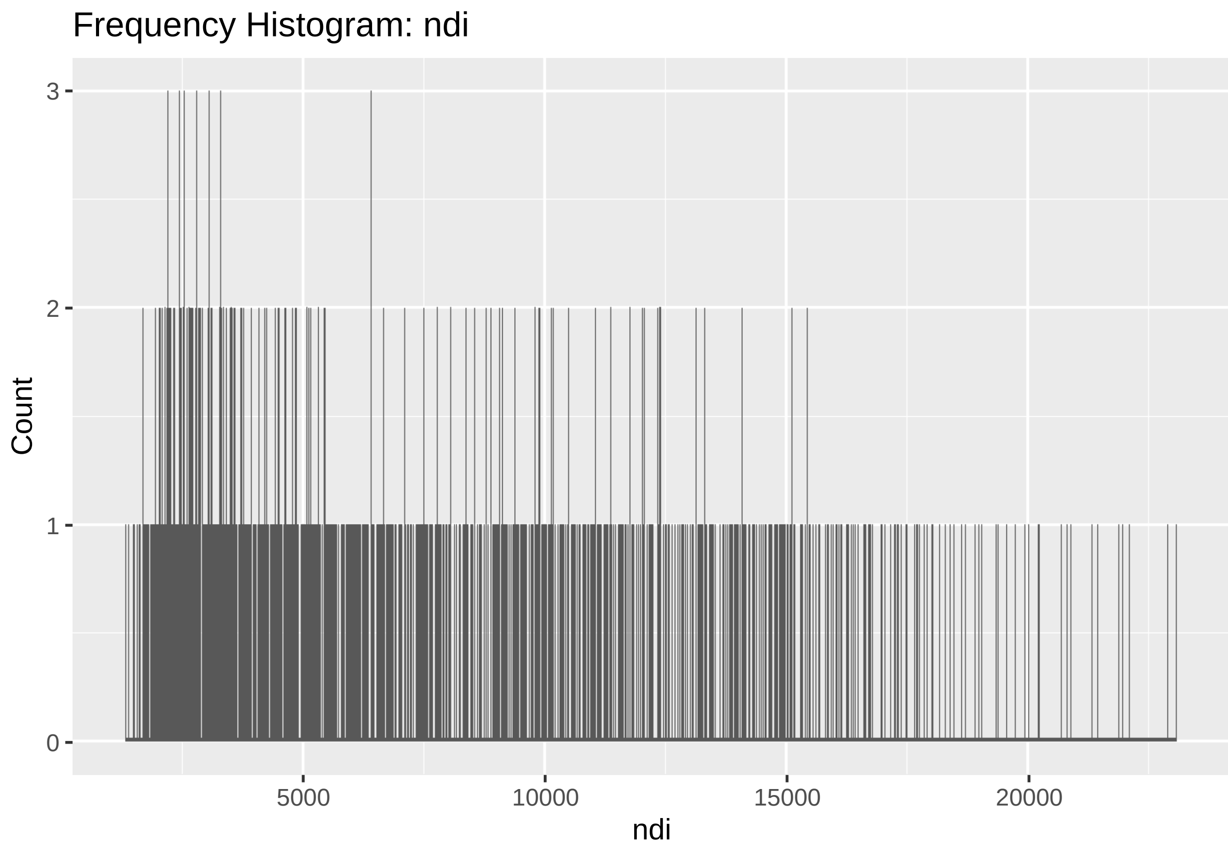

```r
# pop16
qplot(cigar.new$pop16, xlab = 'pop16', ylab = 'Count', binwidth = 2,
      main='Frequency Histogram: pop16')
```



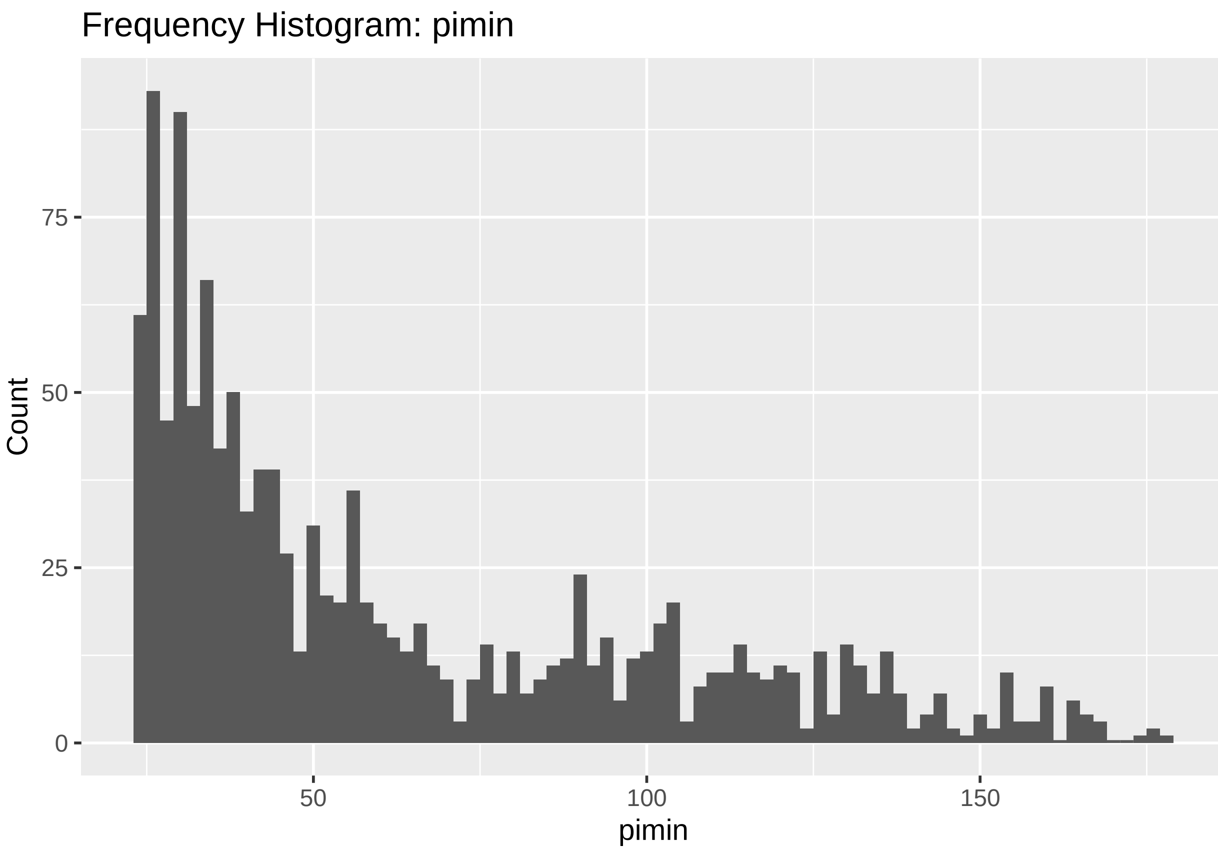

```
# cpi
qplot(cigar.new$cpi, xlab = 'cpi', ylab = 'Count', binwidth = 2,
      main='Frequency Histogram: cpi')
```




```
# ndi
qplot(cigar.new$ndi, xlab = 'ndi', ylab = 'Count', binwidth = 2,
      main='Frequency Histogram: ndi')
```



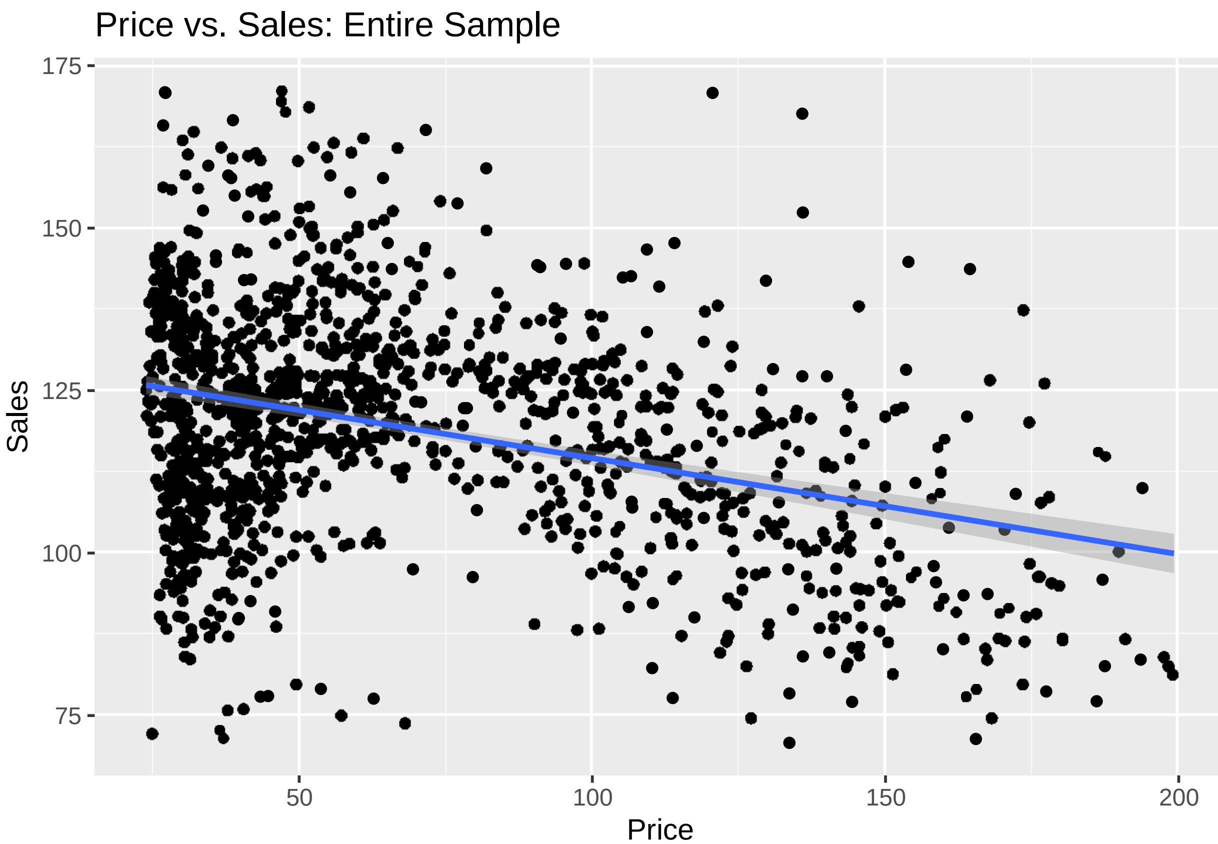

```r
# pimin
qplot(cigar.new$pimin, xlab = 'pimin', ylab = 'Count', binwidth = 2,
      main='Frequency Histogram: pimin')
```



The distributions for population, pop16, cpi, ndi, and pimin are all right skewed which might be caused by the correlations between the variables.

We now use ggplot2 charting techniques to visualize how Price affects the sales by doing a scatter plot with a linear best-fit line.

```
ggplot(data = cigar.new, aes(x = price, y = sales)) +
  geom_point() +
  geom_smooth(method='lm') +
  xlab('Price') +
  ylab('Sales') +
  ggtitle('Price vs. Sales: Entire Sample')
```



The data shows that sales and price are inversely related. In other words, as the price of the cigarette pack increases, the sales decrease.

After the data exploration, the next step will be to dive into the analysis of the data.

3 METHODS

For our analysis, we will be taking in consideration three different methods to find the best model in order to predict the cigarette sales in packs per capita: Multiple Linear Regression (MLR), Neural Network (NN) and Random Forest(RF). The purpose is to compare the performance of the three models while using 10-fold cross validation.

• Multiple linear regression (MLR) is a statistical technique that uses several variables to predict the outcome of a response variable. The goal of multiple linear regression (MLR) is to model the linear relationship between the independent variables (covariates) and response variable (dependent).

• A Neural Network is a series of algorithms which aim to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. Indeed, neural networks refer to systems of neurons. They can adapt to changing input; so, the network generates the best possible result without needing to redesign the output criteria. The concept of neural networks, which has its roots in artificial intelligence, is slowly gaining popularity.

• Random forest is a Supervised Learning algorithm which uses ensemble learning method for classification and regression. Random forest is a bagging technique. Indeed, Bagging regression trees is a technique that can turn a single tree model with high variance and poor predictive power into a fairly accurate prediction function. Unfortunately, bagging regression trees typically suffers from tree correlation, which reduces the overall performance of the model. The trees in random forests are run in parallel. There is no interaction between these trees while building the trees. It operates by constructing a multitude of decision trees at training time and outputting the mean prediction (regression) of the individual trees.

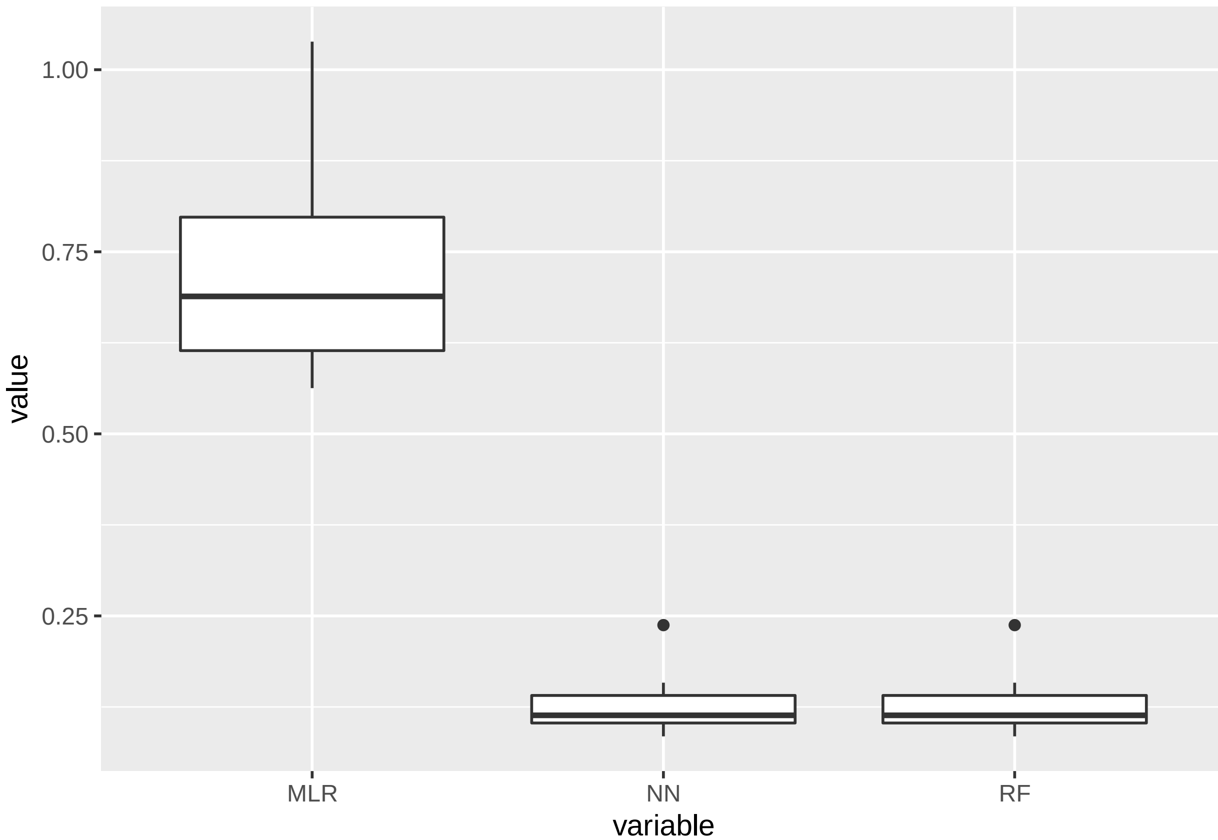
We then proceed by first splitting the data into two parts: training set and validation set which is used to test the model. We fit the different models and we get the results below:

```
library(dplyr)
library(randomForest)
library(nnet)
K = 10
set.seed(123)
fold.assignments =sample(rep(1:K,length=nrow(cigar.new)))

# initialize the matrix to store errors for LM and NN
err.cv = matrix(0,K,3)
colnames(err.cv) = c("MLR","NN","RF")

# outer for loop
for (k in 1:K) {
  # Print out progress
  cat("Fold",k,"... ")
  # Partition into training and test sets
  inds = which(fold.assignments==k)

  train = cigar.new[-inds,]
  test = cigar.new[inds,]

  grid=c(5,10,15,20,25,30,50)

  M= 10
  set.seed(1)
  inner.fold.assignments =sample(rep(1:M,length=nrow(train)))

  NN.err.cv = matrix(0,length(grid),M)
  rownames(NN.err.cv) = grid
  colnames(NN.err.cv) = paste("fold_",1:M,sep="")
  # inner for loop
  for (j in 1:M){
    inner.inds = which(inner.fold.assignments==j)

    inner.train = train[-inner.inds,]

    mean = apply(inner.train,2,mean)
    std = apply(inner.train,2,sd)
    new = scale (train,center=mean,scale=std)

    new.train = new[-inner.inds,]
    new.test =new[inner.inds,]

    for (i in 1:length(grid)){
      num_nodes= grid[i]

      fit = nnet(sales~.,data=new.train,size=num_nodes,
                 linout=TRUE,decay=5e-4,maxit=500)
      pred = predict(fit,newdata=new.test)
      NN.err.cv[i,j] = mean((pred[,1]-new.test[,"sales"])^2)
    }
```

```r
  }
  mse = rowMeans(NN.err.cv)

  # the best number of nodes
  outer.num_nodes = as.numeric(names(mse)[which.min(mse)])

  # next, please normalize the train and test set using info of train      set
  mean = apply(train,2,mean)
  std = apply(train,2,sd)

  new.train = as.data.frame(scale (train,center=mean,scale=std))
  new.test = as.data.frame(scale (test,center=mean,scale=std))

  # then, fit lm() and nnet(), where nnet() must include the argument:    size = outer.num_nodes
  NN.result = nnet(sales~.,data=new.train,size= outer.num_nodes,
                   linout=TRUE,decay=5e-4, maxit=1000)

  lm.result = lm(sales~.,data=as.data.frame(new.train))

  heartrf = randomForest(sales~.,data=as.data.frame(new.train), mtry=3, ntree=275,importance=TRUE)

  # then, predict on test set for both lm() and nnet()
  NN.pred = predict(NN.result,newdata=new.test)
  lm.pred = predict(lm.result,newdata=new.test)
  rf.pred = predict(heartrf,newdata=new.test)
  # calculate the mean square error and store them to the matrix:      err.cv
  err.cv[k,1] = mean((lm.pred-new.test[,"sales"])^2)
  err.cv[k,2] = mean((NN.pred[,1]-new.test[,"sales"])^2)
  err.cv[k,3] = mean((NN.pred[,1]-new.test[,"sales"])^2)
}

err.cv = as.data.frame(err.cv)

# save to local computer
write.csv(err.cv,file="err_cv.csv")
```

After fitting the models, we test them and store the mean square errors in a matrix, and we make the boxplot that shows the different errors. We get the following plot:

```r
# read err.cv back in
err.cv.1 = read.csv(file="err_cv.csv")[-1]
# reshape the data frame
library(reshape2)
err.cv.melt = melt(err.cv.1,id=NULL)

# make boxplot
library(ggplot2)
ggplot(data=err.cv.melt,aes(x=variable,y=value))+
geom_boxplot()
```



As we look at the boxplot, we see that the mean square error of MLR is very high. NN and RF have approximately the same variance which is lower than MLR's. This is an indication that either Neural Network or Random Forest will work to fit the data. We go further by doing the Anova test in order to support our results. The Anova test is used to compare the mean between multiple groups. It can tell if the three groups have similar performances.

```
# perform t test
anova_one_way <- aov(value~variable, data = err.cv.melt)
summary(anova_one_way)
```

```
##              Df Sum Sq Mean Sq F value   Pr(>F)
## variable      2 2.3635  1.1817     138 6.65e-15 ***
## Residuals    27 0.2312  0.0086
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The p-value is lower than the threshold of 0.05. This means that there is a statistical difference between the different groups.

The one-way ANOVA test does not tell us which group has a different mean. Thus, we can perform a Tukey test with the function TukeyHSD().

```
# perform t test
TukeyHSD(anova_one_way)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = value ~ variable, data = err.cv.melt)
##
```

```
## $variable
##                  diff        lwr        upr p adj
## NN-MLR -5.954150e-01 -0.6980155 -0.4928145     0
## RF-MLR -5.954150e-01 -0.6980155 -0.4928145     0
## RF-NN  -1.110223e-16 -0.1026005  0.1026005     1
```

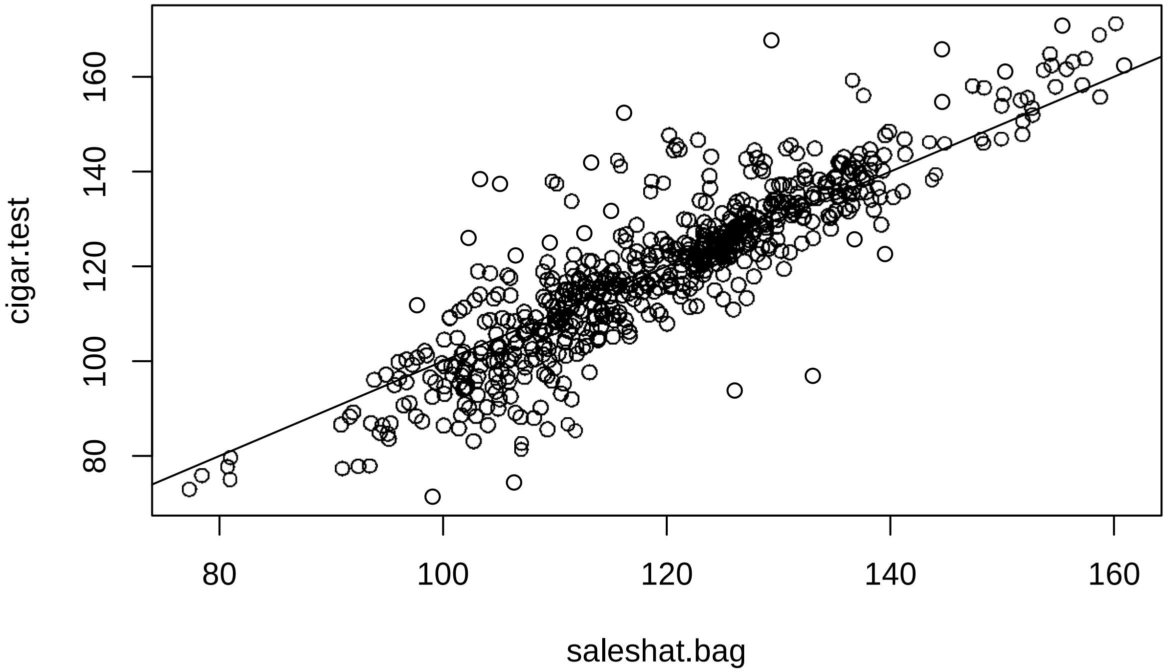
The p-value between RF and NN is 1 which means there is no difference between the two methods. We therefore choose the Random Forest method to fit the whole data.

A random forest model is then fitted on the training data with sales the response variable and the remaining 8 covariates.

```
library(randomForest)
library(MASS)
set.seed(1)
train1 = sample(1:nrow(cigar.new), nrow(cigar.new)/2)
cigarrf <- randomForest(sales~.,data=cigar.new,subset = train1, mtry=8, importance=TRUE)
print(cigarrf)
```

```
##
## Call:
##  randomForest(formula = sales ~ ., data = cigar.new, mtry = 8,      importance = TRUE, subset = trair
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 8
##
##           Mean of squared residuals: 96.54054
##                     % Var explained: 71.05
```

```
cigar.test=cigar.new[-train1,"sales"]
saleshat.bag = predict(cigarrf,newdata=cigar.new[-train1,])

plot(saleshat.bag,cigar.test)
abline(0,1)
```



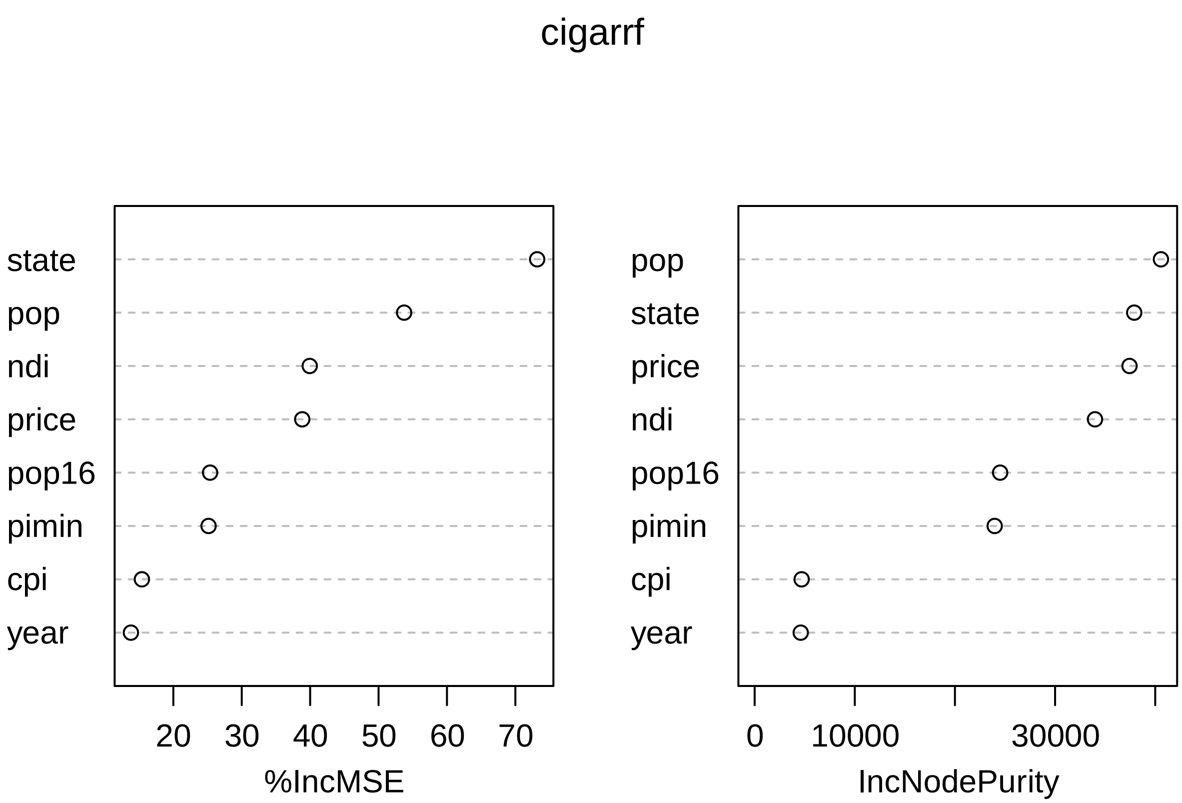
```r
mean((saleshat.bag-cigar.test)^2)
```

```
## [1] 76.52257
```

```r
importance(cigarrf)
```

```
##         %IncMSE IncNodePurity
## state 73.18458     37898.314
## year  13.79472      4598.478
## price 38.83921     37428.708
## pop   53.73755     40564.724
## pop16 25.37012     24510.590
## cpi   15.40093      4688.202
## ndi   39.94028     33980.616
## pimin 25.14662     23966.227
```

```r
varImpPlot(cigarrf)
```



The results above shows that the random forest model built was a regression model. The model grew 500 trees and for each split, 8 variables were tried which is the optimal value. 71.05% of variation in the data has been explained. We then used our fitted model to predict our test data and plot the graphs to be able to visualize the data and reinforced our analysis. We can see that population, state, price and ndi seems are the most important variables in the data. We also make a plot of the training set vs the test set as well. It shows that the performance of the model is good on the test set since all the data is spread closely around the line.

5. LIMITATION

At the end of our study, after deciding on three different methods (Multiple Linear Regression, Neural Network and Random Forest), we were able to determine that Random Forest was the best model to fit the data. Around 71% of variation in the data cigar was explained. This can mean that more predictors should be added or more transformations of the data are required since we were dealing with high correlation in the data. Although Random Forest typically have very good performance, there are also some drawbacks. Indeed, Random Forest can become slow on large data sets and less interpretable. They also have been observed to overfit some datasets with noisy regression tasks.

6. REFERENCES

• Towards data science, Random Forest Regression, https://towardsdatascience.com/random-forest-and-its-implementation-71824ced454f

• LISTEN DATA, Make your data tell a story, A complete Guide to Random Forest in R, https://www.listendata.com/2014/11/random-forest-with-r.html

• R ANOVA Tutorial: One way & Two way (with Examples) https://www.guru99.com/r-anova-tutorial.html

• David Dobor, Trees, Random Forsets, Boosting for Continuous Variable Prediction, http://rstudio-pubs-static.s3.amazonaws.com/156481_80ee6ee3a0414fd38f5d3ad33d14c771.html

• Michy Alice, Fitting a neural network in R; neuralnet package,September 23, 2015, https://www.r-bloggers.com/fitting-a-neural-network-in-r-neuralnet-package/

• R Documentation, Cigarette Consumption, https://vincentarelbundock.github.io/Rdatasets/doc/Ecdat/Cigar.html