Intro_To_Linear_Statistical_Models_Final

Aman Goswami & Abdulaziz Alshalfan

12/15/2020

General Information : Aziz and I collaborated through Github on the final given the timezone difference. We uploaded our codes at a repository with details here - https://github.com/ag77in/ISLM. We both did all 6 questions together for a dry run and then proceeded to discuss to see how we can improve the answers and the final results. We then jointly created this document.

Let us load libraries

```
# clear environment
rm(list = ls())

# defining libraries

library(ggplot2)
library(dplyr)
library(PerformanceAnalytics)
library(data.table)
library(sqldf)
library(nortest)
library(MASS)
library(rpart)
library(class)
library(ISLR)
library(scales)
library(ClustOfVar)
library(GGally)
library(reticulate)
library(ggthemes)
library(RColorBrewer)
library(gridExtra)
library(kableExtra)
library(Hmisc)
library(corrplot)
library(energy)
library(nnet)
library(Hotelling)
library(car)
library(devtools)
library(ggbiplot)
library(factoextra)
library(rgl)
```

```r
library(FactoMineR)
library(psych)
library(nFactors)
library(scatterplot3d)
library(lmtest)
library(mctest)
library(aod)
library(InformationValue)
library(pROC)
library(tidyverse)
library(caret)
library(Information)
library(reshape)
library(olsrr)
library(faraway)
library(readxl)
library(tidyverse)
library(lubridate)
library(tsutils)
library(seastests)
library(emmeans)
library(forecast)
library(tseries)
library(tidyquant)
library(modelr)
library(grid)
library(aTSA)
library(fpp2)
library(MLmetrics)
```

1) Fit a model to explain price in terms of the predictors. Perform regression diagnostics to answer the following questions. Display any plots that are relevant. Suggest improvements if any.

Let us load the data and summarise the information

```r
# reading data
stockdata <- read.csv('/Users/mac/Downloads/final-2020-canvas/datasets/stockdata.csv')
str(stockdata)
```

```
## 'data.frame':    100 obs. of  8 variables:
##  $ X             : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ cap.to.gdp    : num  0.0694 0.8178 0.9426 0.2694 0.1693 ...
##  $ q.ratio       : num  0.483 0.841 0.455 0.861 0.676 ...
##  $ gaap          : num  0.973 0.217 0.521 0.828 0.964 ...
##  $ trailing.pe   : num  0.6 0.831 0.74 0.564 0.364 ...
##  $ avg.allocation: num  0.00141 0.27665 0.47984 0.51034 0.90312 ...
##  $ price         : num  1.14 1.16 1.16 1.16 1.17 ...
##  $ vol           : num  0.92 0.22 0.02 0.37 0.93 0.42 0.78 0.4 0.09 0.23 ...
```

```r
#summary
summary(stockdata)
```

```
##        X           cap.to.gdp         q.ratio             gaap
##  Min.   :  1.00   Min.   :0.008325   Min.   :0.01581   Min.   :0.002853
##  1st Qu.: 25.75   1st Qu.:0.257293   1st Qu.:0.29298   1st Qu.:0.184894
##  Median : 50.50   Median :0.436999   Median :0.54062   Median :0.473457
##  Mean   : 50.50   Mean   :0.479715   Mean   :0.52838   Mean   :0.484240
##  3rd Qu.: 75.25   3rd Qu.:0.715682   3rd Qu.:0.77409   3rd Qu.:0.765638
##  Max.   :100.00   Max.   :0.980869   Max.   :0.99876   Max.   :0.991848
##   trailing.pe      avg.allocation        price            vol
##  Min.   :0.01565   Min.   :0.001411   Min.   :1.123   Min.   :0.0100
##  1st Qu.:0.24465   1st Qu.:0.264229   1st Qu.:1.145   1st Qu.:0.2575
##  Median :0.45378   Median :0.486100   Median :1.156   Median :0.5050
##  Mean   :0.48442   Mean   :0.502492   Mean   :1.154   Mean   :0.5050
##  3rd Qu.:0.75006   3rd Qu.:0.726507   3rd Qu.:1.165   3rd Qu.:0.7525
##  Max.   :0.99022   Max.   :0.964295   Max.   :1.178   Max.   :1.0000
```

Key observations -

1. cap.to.gdp., trailing.pe, gaap, avg.allocation are marginally positively skewed while q.ratio is marginally negatively skewed.

2. Volatility looks normal however we will confirm this later.

3. We don't see any evidence of outliers but we will check the plots before commenting on this.

Any missing values ?

```r
data <-na.omit(stockdata)
str(data)
```

```
## 'data.frame':    100 obs. of  8 variables:
##  $ X             : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ cap.to.gdp    : num  0.0694 0.8178 0.9426 0.2694 0.1693 ...
##  $ q.ratio       : num  0.483 0.841 0.455 0.861 0.676 ...
##  $ gaap          : num  0.973 0.217 0.521 0.828 0.964 ...
##  $ trailing.pe   : num  0.6 0.831 0.74 0.564 0.364 ...
##  $ avg.allocation: num  0.00141 0.27665 0.47984 0.51034 0.90312 ...
##  $ price         : num  1.14 1.16 1.16 1.16 1.17 ...
##  $ vol           : num  0.92 0.22 0.02 0.37 0.93 0.42 0.78 0.4 0.09 0.23 ...
```

We did not find any missing values in the data.

Correlation plot -

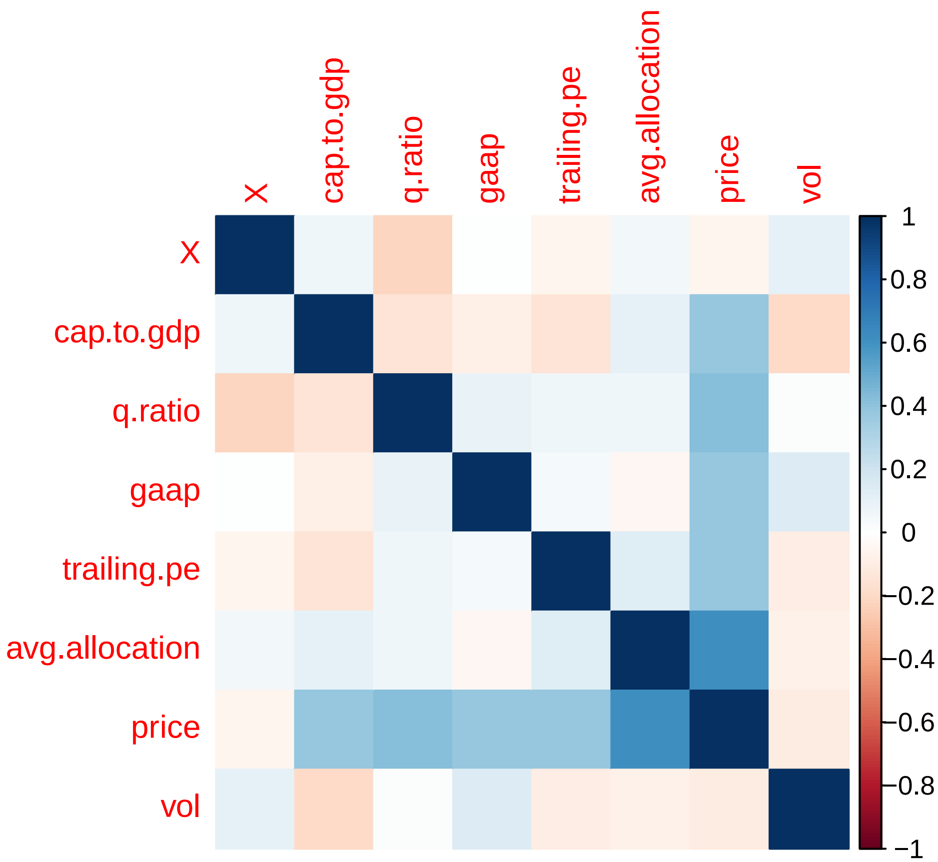
We check correlation before moving towards modeling exercise.

```r
M<-cor(stockdata)
head(round(M,2))
```

```
##                    X cap.to.gdp q.ratio  gaap trailing.pe avg.allocation
## X               1.00       0.07   -0.22  0.01       -0.06           0.05
## cap.to.gdp      0.07       1.00   -0.14 -0.08       -0.14           0.11
## q.ratio        -0.22      -0.14    1.00  0.10        0.07           0.06
```

```
## gaap             0.01       -0.08     0.10  1.00         0.04           -0.05
## trailing.pe     -0.06       -0.14     0.07  0.04         1.00            0.13
## avg.allocation   0.05        0.11     0.06 -0.05         0.13            1.00
##                 price   vol
## X              -0.05  0.11
## cap.to.gdp      0.39 -0.19
## q.ratio         0.42  0.02
## gaap            0.39  0.15
## trailing.pe     0.39 -0.10
## avg.allocation  0.62 -0.08
```

```r
corrplot(M, method="color")
```



Key observations -

1. We see price is correlated strongly with avg.allocation (+0.62).

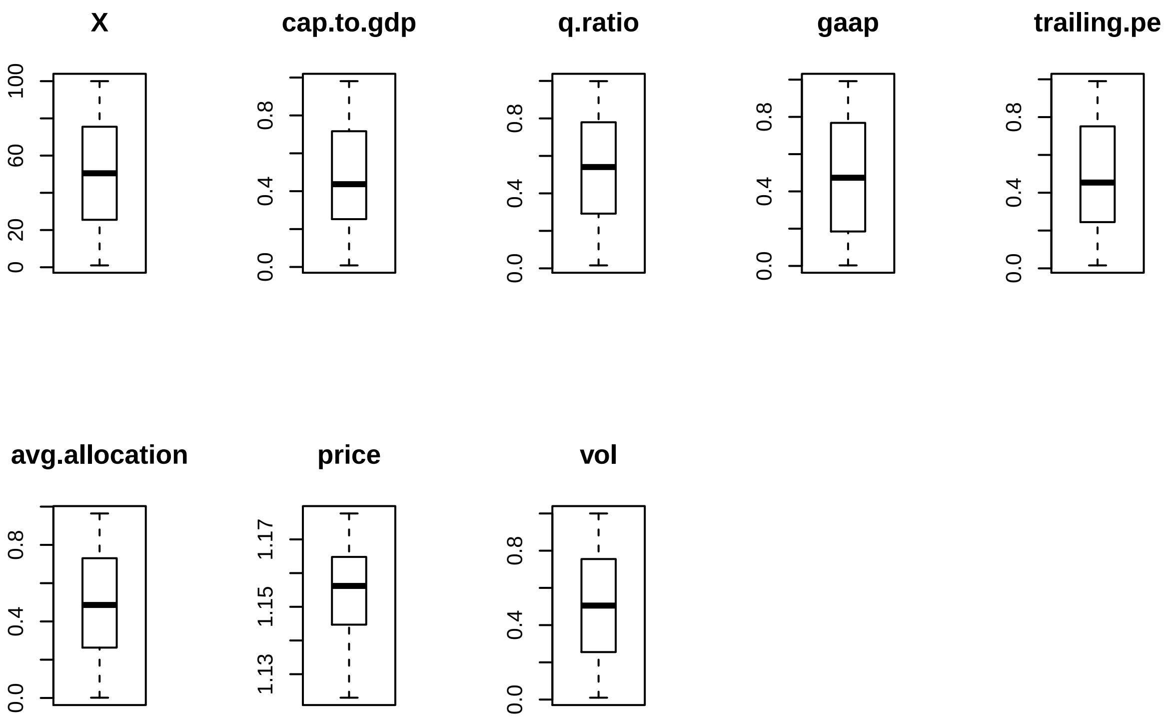
2. We also see price being correlated with q.ratio, cap.to.gdp, gaap and trailing.pe in similar range (+0.39-+0.42).

3. We see very little correlation between most of the independent variables however we note the -ve correlations of cap.to.gdp with q.ratio, trailing.pe and gaap albeit small.

Outlier/ Univariate checks -

We confirm our earlier hypothesis of no outliers by looking at some univariate and outlier checks.

```
par(mfrow=c(2,5))
for (i in 1:length(stockdata)) {
        boxplot(stockdata[,i], main=names(stockdata[i]), type="l")
}
```



We see no evidence of any outliers in the univariate form.

We now proceed to modeling exercise.

1a) Fit a model to explain price in terms of the predictors. Which variables are important, can any of the variables be removed ? Please use F-tests to justify.

We use lm() function for this regression

```
fit <- lm(price~cap.to.gdp+ q.ratio+gaap+trailing.pe+avg.allocation+vol, data=stockdata)
summary(fit)
```

```
##
## Call:
## lm(formula = price ~ cap.to.gdp + q.ratio + gaap + trailing.pe +
##     avg.allocation + vol, data = stockdata)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0067787 -0.0015687  0.0002342  0.0019888  0.0075661
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.1087154  0.0012722 871.527   <2e-16 ***
## cap.to.gdp      0.0209002  0.0010535  19.839   <2e-16 ***
## q.ratio         0.0181111  0.0010414  17.391   <2e-16 ***
## gaap            0.0163251  0.0009298  17.557   <2e-16 ***
## trailing.pe     0.0143780  0.0009750  14.747   <2e-16 ***
## avg.allocation  0.0225869  0.0009978  22.637   <2e-16 ***
## vol            -0.0005667  0.0009918  -0.571    0.569
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.002758 on 93 degrees of freedom
## Multiple R-squared:  0.9535, Adjusted R-squared:  0.9505
## F-statistic:   318 on 6 and 93 DF,  p-value: < 2.2e-16
```

We see significant result in p-value for all variables except for Volatility which is not significant even at 10% level of signifance hence this variable can be removed from the model. We get F value of 318 with probability <0.5 indicating the joint hypothesis of this model being better than null model. Every other variable seems important to prediction.

The above model has a good R-square of 95.35%.

We now confirm this with joint hypothesis test using F-statistic. We use both anova and linearHypothesis() for this.

Model without vol

```
fit2 <- lm(price~cap.to.gdp+ q.ratio+gaap+trailing.pe+avg.allocation, data=stockdata)
summary(fit2)
```

```
##
## Call:
## lm(formula = price ~ cap.to.gdp + q.ratio + gaap + trailing.pe +
##     avg.allocation, data = stockdata)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
```

```
## -0.0066732 -0.0015245  0.0003056  0.0019045  0.0073869
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.1083616  0.0011074 1000.89   <2e-16 ***
## cap.to.gdp     0.0210170  0.0010297   20.41   <2e-16 ***
## q.ratio        0.0181196  0.0010375   17.46   <2e-16 ***
## gaap           0.0162510  0.0009174   17.71   <2e-16 ***
## trailing.pe    0.0144476  0.0009639   14.99   <2e-16 ***
## avg.allocation 0.0226051  0.0009937   22.75   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.002748 on 94 degrees of freedom
## Multiple R-squared:  0.9534, Adjusted R-squared:  0.9509
## F-statistic: 384.3 on 5 and 94 DF,  p-value: < 2.2e-16
```

We now look at Anova

```
anova(fit2, fit)
```

```
## Analysis of Variance Table
##
## Model 1: price ~ cap.to.gdp + q.ratio + gaap + trailing.pe + avg.allocation
## Model 2: price ~ cap.to.gdp + q.ratio + gaap + trailing.pe + avg.allocation +
##     vol
##   Res.Df        RSS Df  Sum of Sq      F Pr(>F)
## 1     94 0.00070986
## 2     93 0.00070738  1 2.4832e-06 0.3265 0.5691
```

Since p-value of 0.5691 is > 0.5 we conclude that the model with vol is not significantly better than the model without vol.

We can see this with individual F tests for each variable using linearhypothesis() now as well.

```
linearHypothesis(fit, c("cap.to.gdp=0"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## cap.to.gdp = 0
##
## Model 1: restricted model
## Model 2: price ~ cap.to.gdp + q.ratio + gaap + trailing.pe + avg.allocation +
##     vol
##
##   Res.Df       RSS Df Sum of Sq     F    Pr(>F)
## 1     94 0.0037012
## 2     93 0.0007074  1 0.0029938 393.6 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
linearHypothesis(fit, c("q.ratio =0"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## q.ratio = 0
```

```
##
## Model 1: restricted model
## Model 2: price ~ cap.to.gdp + q.ratio + gaap + trailing.pe + avg.allocation +
##     vol
##
##   Res.Df        RSS Df Sum of Sq      F    Pr(>F)
## 1     94 0.00300793
## 2     93 0.00070738  1 0.0023006 302.46 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
linearHypothesis(fit, c("gaap=0"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## gaap = 0
##
## Model 1: restricted model
## Model 2: price ~ cap.to.gdp + q.ratio + gaap + trailing.pe + avg.allocation +
##     vol
##
##   Res.Df        RSS Df Sum of Sq      F    Pr(>F)
## 1     94 0.00305206
## 2     93 0.00070738  1 0.0023447 308.26 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
linearHypothesis(fit, c("trailing.pe=0"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## trailing.pe = 0
##
## Model 1: restricted model
## Model 2: price ~ cap.to.gdp + q.ratio + gaap + trailing.pe + avg.allocation +
##     vol
##
##   Res.Df        RSS Df Sum of Sq      F    Pr(>F)
## 1     94 0.00236157
## 2     93 0.00070738  1 0.0016542 217.48 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
linearHypothesis(fit, c("avg.allocation=0"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## avg.allocation = 0
##
## Model 1: restricted model
## Model 2: price ~ cap.to.gdp + q.ratio + gaap + trailing.pe + avg.allocation +
##     vol
##
```

```
##   Res.Df       RSS Df Sum of Sq      F    Pr(>F)
## 1     94 0.0046051
## 2     93 0.0007074  1 0.0038977 512.43 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
linearHypothesis(fit, c("vol=0"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## vol = 0
##
## Model 1: restricted model
## Model 2: price ~ cap.to.gdp + q.ratio + gaap + trailing.pe + avg.allocation +
##     vol
##
##   Res.Df        RSS Df  Sum of Sq      F Pr(>F)
## 1     94 0.00070986
## 2     93 0.00070738  1 2.4832e-06 0.3265 0.5691
```

We ran the F-tests for the hypothesis of each individual variable and in only one case is the F-score is very low (0.3265) in the 'vol'=0 hypothesis. Hence, this confirms from our linear regression model again that given the Pr(>F) is 0.56, we find volatility to not influence the price and hence we can remove this variable from our model.

Sometimes, however it is important to run the heteroskedastic robust version of the F-test as well. We do this as follows -

```
linearHypothesis(fit, c("cap.to.gdp=0"),white.adjust = "hc1")
```

```
## Linear hypothesis test
##
## Hypothesis:
## cap.to.gdp = 0
##
## Model 1: restricted model
## Model 2: price ~ cap.to.gdp + q.ratio + gaap + trailing.pe + avg.allocation +
##     vol
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F    Pr(>F)
## 1     94
## 2     93  1 375.34 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
linearHypothesis(fit, c("q.ratio =0"),white.adjust = "hc1")
```

```
## Linear hypothesis test
##
## Hypothesis:
## q.ratio = 0
##
## Model 1: restricted model
## Model 2: price ~ cap.to.gdp + q.ratio + gaap + trailing.pe + avg.allocation +
```

```
##     vol
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F    Pr(>F)
## 1     94
## 2     93  1 250.75 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
linearHypothesis(fit, c("gaap=0"),white.adjust = "hc1")
```

```
## Linear hypothesis test
##
## Hypothesis:
## gaap = 0
##
## Model 1: restricted model
## Model 2: price ~ cap.to.gdp + q.ratio + gaap + trailing.pe + avg.allocation +
##     vol
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F    Pr(>F)
## 1     94
## 2     93  1 338.81 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
linearHypothesis(fit, c("trailing.pe=0"),white.adjust = "hc1")
```

```
## Linear hypothesis test
##
## Hypothesis:
## trailing.pe = 0
##
## Model 1: restricted model
## Model 2: price ~ cap.to.gdp + q.ratio + gaap + trailing.pe + avg.allocation +
##     vol
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F    Pr(>F)
## 1     94
## 2     93  1 248.47 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
linearHypothesis(fit, c("avg.allocation=0"),white.adjust = "hc1")
```

```
## Linear hypothesis test
##
## Hypothesis:
## avg.allocation = 0
##
## Model 1: restricted model
```

```
## Model 2: price ~ cap.to.gdp + q.ratio + gaap + trailing.pe + avg.allocation +
##     vol
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F    Pr(>F)
## 1     94
## 2     93  1 565.09 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
linearHypothesis(fit, c("vol=0"),white.adjust = "hc1")
```

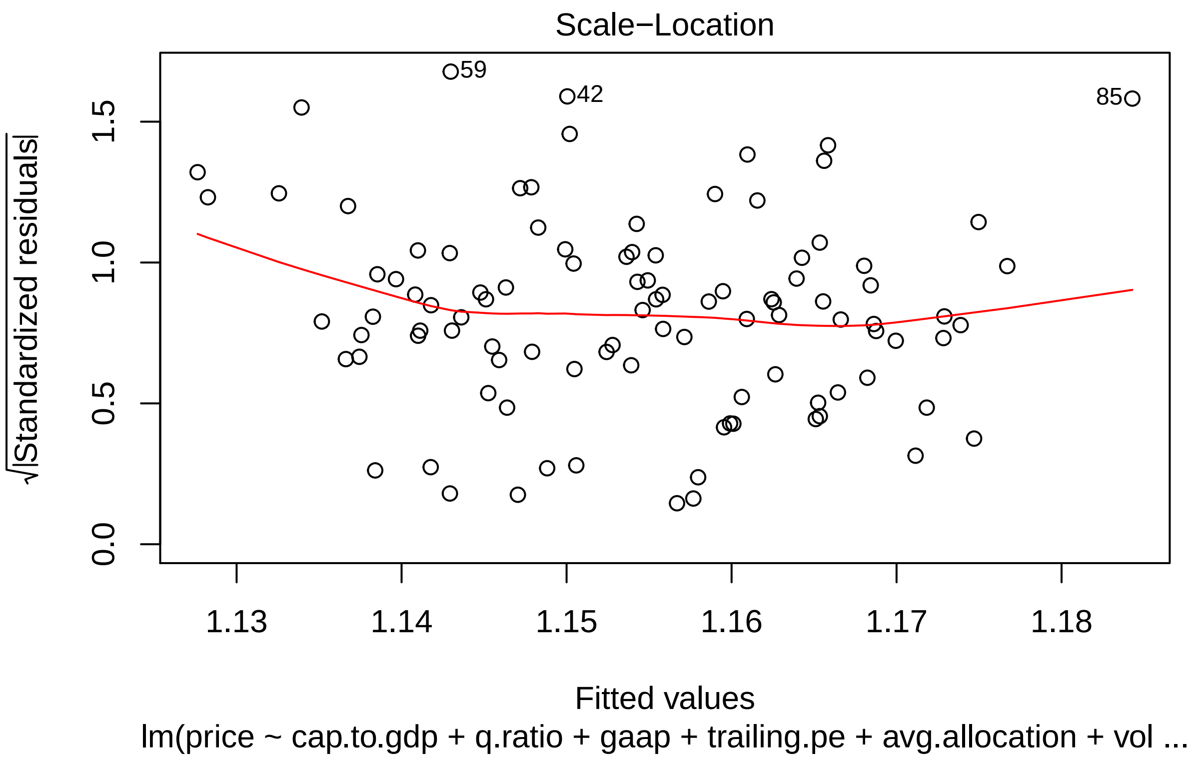
```
## Linear hypothesis test
##
## Hypothesis:
## vol = 0
##
## Model 1: restricted model
## Model 2: price ~ cap.to.gdp + q.ratio + gaap + trailing.pe + avg.allocation +
##     vol
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F Pr(>F)
## 1     94
## 2     93  1 0.3095 0.5793
```

We see no change in results although the Pr(>F) for vol=0 hypothesis is slightly higher than before. We still only confirm that volatility can be removed from the model and state that fit2 is our best model for now.

1b) Check the constant variance assumption for the errors.

We do this in 3 ways - we look at scale-location plot of the regression, then perform ncvTest() and finally Breusch-pagan test for homoskedasticity.

```r
plot(fit,which=3)
```



We mostly see as they say "stars in the sky" expression in the above graph. While the line visually is not completely horizontal, we do not in particular see any pattern and it would be hard to deny that the errors are homoskedastic. However, we perform further tests to confirm our suspicion.

ncvTest() For Homoscedasticity

```
ncvTest(fit)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 0.8141361, Df = 1, p = 0.3669
```

We see a p-value > .05, indicating homoscedasticity.

Breusch-Pagan Test For Homoscedasticity

```
bptest(fit)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  fit
## BP = 2.3166, df = 6, p-value = 0.8884
```

We once again see a p-value > .05, indicating homoscedasticity.

1c) Check the independentness of the errors assumption.

We can do this with the durbin watson statistic. The Durbin Watson examines whether the errors are autocorrelated with themselves. The null states that they are not autocorrelated.

```
durbinWatsonTest(fit)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1      0.07666919      1.839674   0.416
##  Alternative hypothesis: rho != 0
```

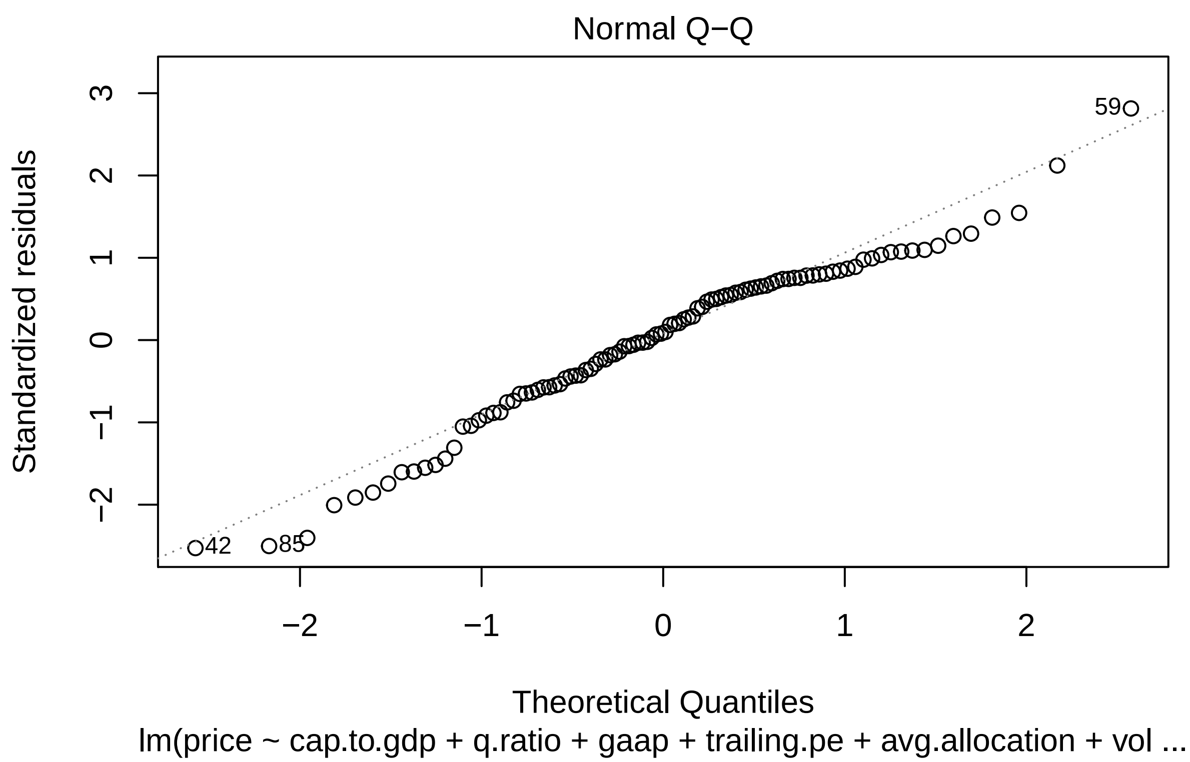
We see that p-value > 0.05 so the errors are not autocorrelated.

1d) Check the normality assumption.

We again do this in 2 ways - we look at QQ plot and perform the Shapiro Wilk normality test.

The normal probability plot of residuals should approximately follow a straight line.

```
plot(fit,which=2)
```



We see points falling mostly along reference line however we also see some falling outside on both sides of the quantile-spectrum so we dig deeper with a statistical test.

Shapiro-Wilk Normality Test

```
resid <- studres(fit)
shapiro.test(resid)
```

```
##
##  Shapiro-Wilk normality test
```

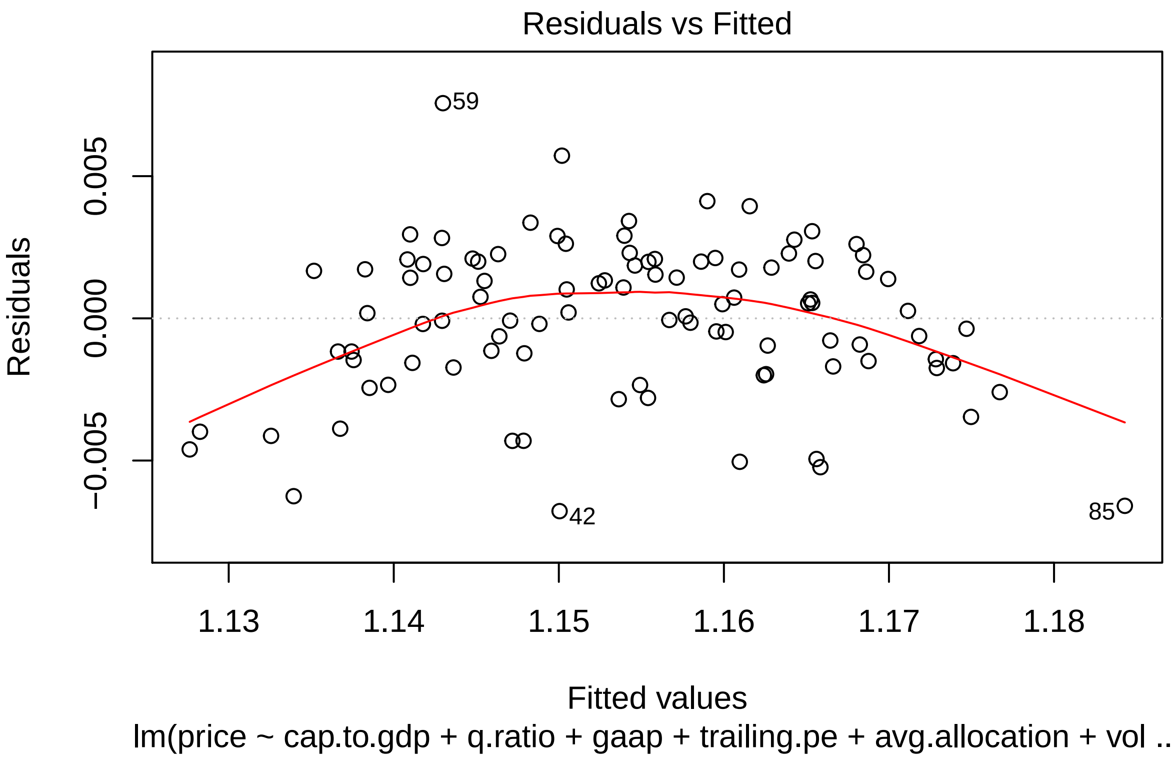
```
##
## data:  resid
## W = 0.97064, p-value = 0.02474
```

From the p-value = 0.02474 < 0.05, we can see that the residuals are not normally distributed

1e) Is non-linearity a problem ?

The linearity assumption can be checked by inspecting the Residuals vs Fitted plot.

```
plot(fit,which=1)
```



This is very interesting since the residuals take both +ve and -ve values. However, we see an inverted U shaped curve above. This suggests that the fit of the model can be improved by taking the square of some explanatory variable.

1f) Check for outliers, compute and plot the cook's distance.

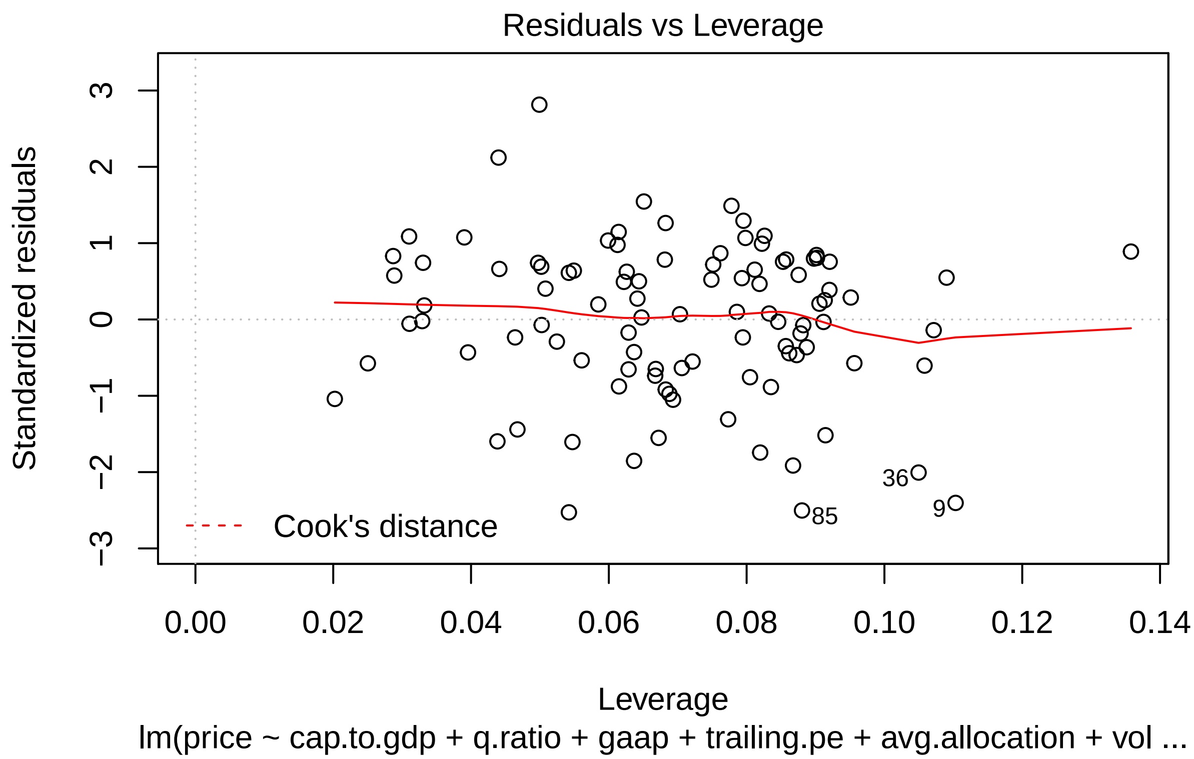
A standard way to check for outliers is to look at residuals above a certain threshold. An example would be as follows -

```
rstandard(fit)[abs(rstandard(fit)) > 2]
```

```
##         9        29        36        42        59        85
## -2.403777  2.121101 -2.005918 -2.527321  2.814529 -2.502884
```

Here, we see points 9, 29, 36, 42, 59 and 85 with large residuals but note that not all of them or maybe none of them could be outliers. So we now look at the model plot of Residuals vs leverage.

```r
plot(fit, which=5)
```

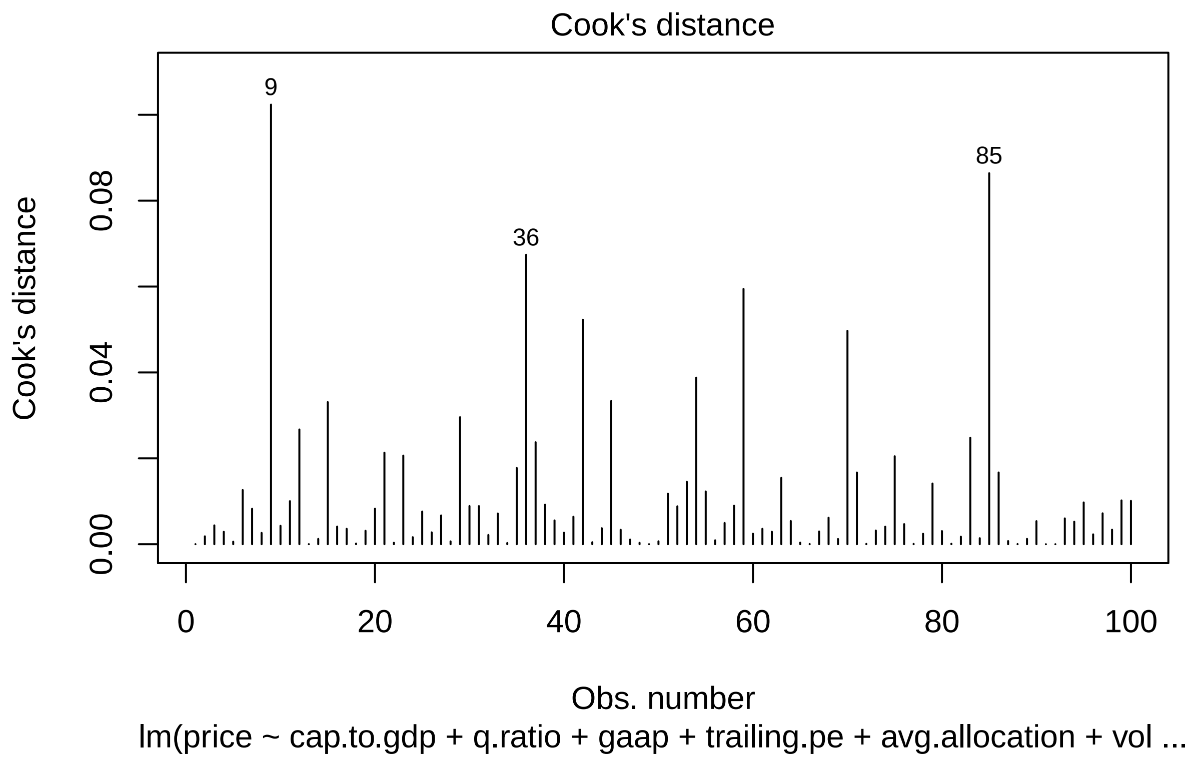


Leverage statistic is defined as -
$\hat{L} = \dfrac{2(p+1)}{n}$ where p is number of predictors and n is number of observations

In the above graph we see all points fall under the dashed lines of the cook's distance (missing) which tells us there are no outliers in the data but still some influential points do exist.

We can plot the cook's distance with the below command -

```r
#Cook's distance
plot(fit, 4)
```



We see that apart from three points - 9, 36 & 85, everyone else's cook's distance is below 0.06

We also compute the cook's distance for each observation as follows -

```r
cooks.distance(fit)
```

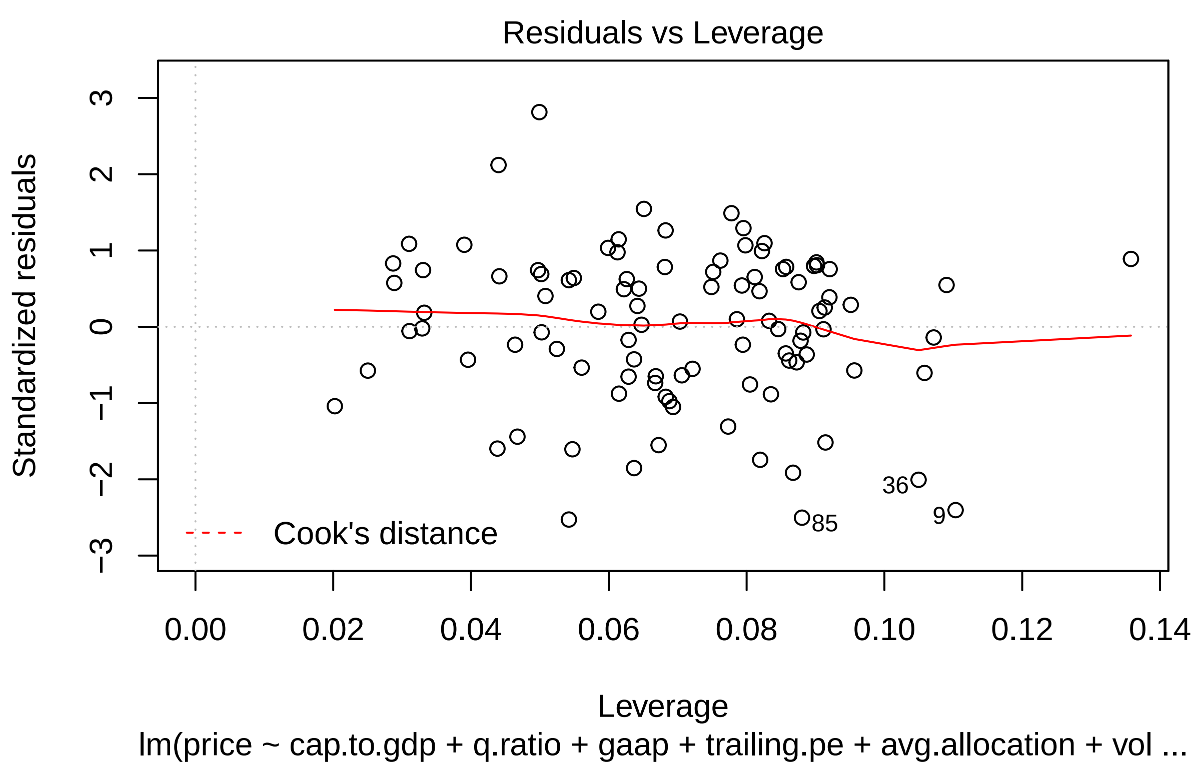
```
##            1            2            3            4            5
## 1.506514e-05 1.841114e-03 4.392873e-03 2.887821e-03 6.071909e-04
##            6            7            8            9           10
## 1.261316e-02 8.260664e-03 2.641588e-03 1.023677e-01 4.304524e-03
##           11           12           13           14           15
## 1.002734e-02 2.672218e-02 1.462150e-05 1.242961e-03 3.308526e-02
##           16           17           18           19           20
## 4.116641e-03 3.605215e-03 1.658446e-04 3.147045e-03 8.279441e-03
##           21           22           23           24           25
## 2.132377e-02 3.458243e-04 2.064649e-02 1.633809e-03 7.615960e-03
##           26           27           28           29           30
## 2.766711e-03 6.705494e-03 6.662909e-04 2.957698e-02 8.898526e-03
##           31           32           33           34           35
## 8.869588e-03 2.166015e-03 7.159917e-03 2.844827e-04 1.775694e-02
##           36           37           38           39           40
## 6.740666e-02 2.375397e-02 9.216673e-03 5.546929e-03 2.693527e-03
##           41           42           43           44           45
## 6.414643e-03 5.229512e-02 4.604372e-04 3.728839e-03 3.335245e-02
##           46           47           48           49           50
## 3.378819e-03 1.097739e-03 3.388588e-04 1.255405e-05 6.816257e-04
##           51           52           53           54           55
## 1.177120e-02 8.826426e-03 1.454570e-02 3.879785e-02 1.229223e-02
##           56           57           58           59           60
## 9.137980e-04 4.958389e-03 8.977201e-03 5.945721e-02 2.458277e-03
##           61           62           63           64           65
```

```
## 3.603885e-03 2.912661e-03 1.545356e-02 5.408925e-03 3.841784e-04
##           66           67           68           69           70
## 5.089891e-05 2.973666e-03 6.187951e-03 1.212264e-03 4.969724e-02
##           71           72           73           74           75
## 1.670104e-02 7.743198e-05 3.198812e-03 4.099079e-03 2.048703e-02
##           76           77           78           79           80
## 4.676067e-03 7.967441e-05 2.434098e-03 1.413915e-02 3.057224e-03
##           81           82           83           84           85
## 1.188117e-04 1.775396e-03 2.478882e-02 1.403672e-03 8.639160e-02
##           86           87           88           89           90
## 1.670982e-02 7.304436e-04 3.999237e-05 1.245046e-03 5.371759e-03
##           91           92           93           94           95
## 2.175562e-06 6.847372e-06 6.016824e-03 5.245686e-03 9.734318e-03
##           96           97           98           99          100
## 2.298364e-03 7.199278e-03 3.392819e-03 1.020753e-02 1.009369e-02
```

1g) Check for influential points.

This was partially done above itself, but neverthless we can check for influential points through the plot itself.

```
# High leverage points
plot(fit, which=5)
```



So from this plot again, we see points 9, 36, 85 as values of extreme nature and we see these as influential points. We also see some other point to the extreme right with high leverage but low residual. Either ways, we check for more robust solution through below.

A rule of thumb is that an observation has high influence if Cook's distance exceeds $\frac{4}{(n-p-1)}$

```r
cooks.distance(fit) > 4 / length(cooks.distance(fit))
```

```
##     1     2     3     4     5     6     7     8     9    10    11    12
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
##    13    14    15    16    17    18    19    20    21    22    23    24
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##    25    26    27    28    29    30    31    32    33    34    35    36
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
##    37    38    39    40    41    42    43    44    45    46    47    48
## FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
##    49    50    51    52    53    54    55    56    57    58    59    60
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
##    61    62    63    64    65    66    67    68    69    70    71    72
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
##    73    74    75    76    77    78    79    80    81    82    83    84
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##    85    86    87    88    89    90    91    92    93    94    95    96
##  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##    97    98    99   100
## FALSE FALSE FALSE FALSE
```

We see points 9, 36, 42, 59, 70, 85 as influential points just going by cooks distance.

We also however check for hatvalues in the data.

```r
hatvalues(fit) > 0.1
```

```
##     1     2     3     4     5     6     7     8     9    10    11    12
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
##    13    14    15    16    17    18    19    20    21    22    23    24
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##    25    26    27    28    29    30    31    32    33    34    35    36
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
##    37    38    39    40    41    42    43    44    45    46    47    48
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
##    49    50    51    52    53    54    55    56    57    58    59    60
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##    61    62    63    64    65    66    67    68    69    70    71    72
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
##    73    74    75    76    77    78    79    80    81    82    83    84
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##    85    86    87    88    89    90    91    92    93    94    95    96
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
##    97    98    99   100
## FALSE FALSE FALSE FALSE
```

We see only few observations with hatvalues above 0.1. These are points 9, 35, 36, 48, 68 & 94.

Now, we combine the above two results of cooks distance and hatvalues which is exactly what influence.measures does for us.

```r
summary(influence.measures(fit))
```

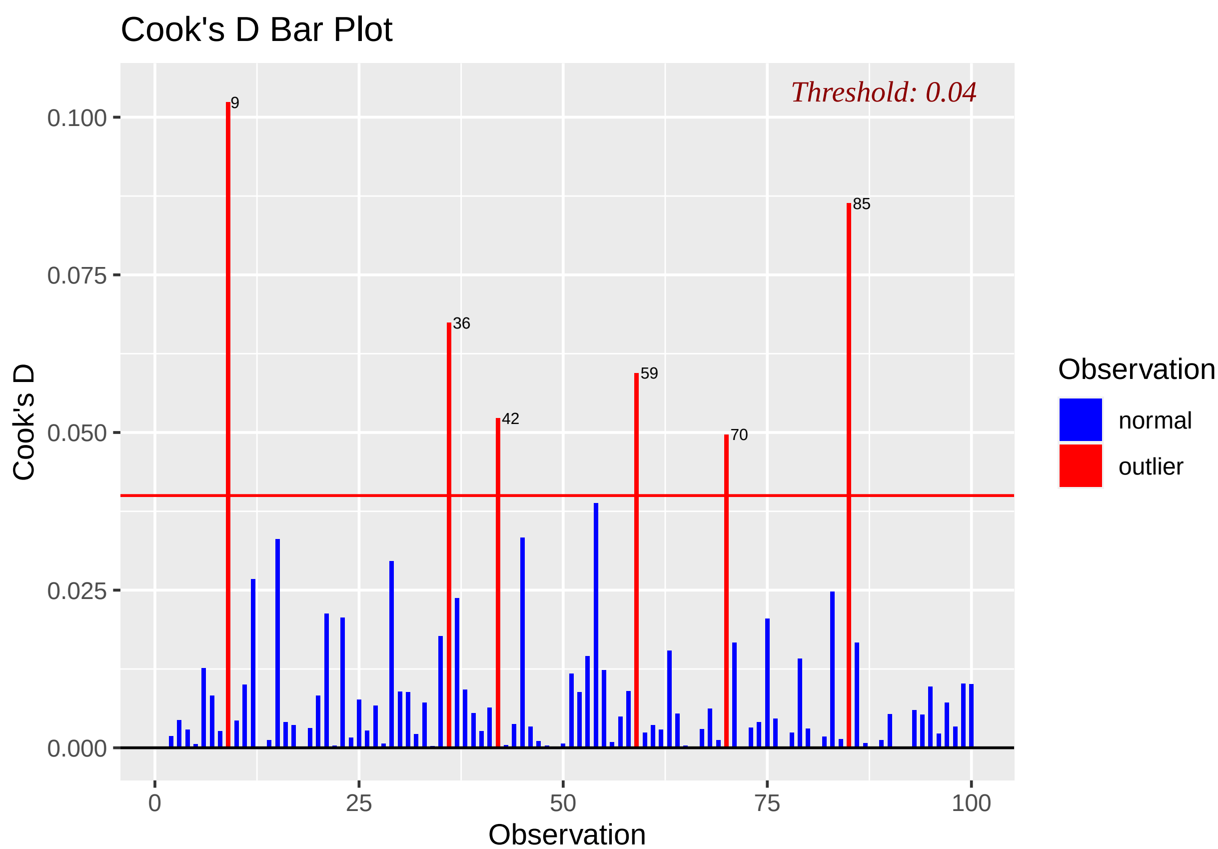
```
## Potentially influential observations of
##   lm(formula = price ~ cap.to.gdp + q.ratio + gaap + trailing.pe +      avg.allocation + vol, data =
##
##    dfb.1_ dfb.cp.. dfb.q.rt dfb.gaap dfb.trl. dfb.avg. dfb.vol dffit
```

```
## 9  -0.67   0.52     0.43    -0.10    -0.09     0.26     0.48   -0.87_*
## 42 -0.19  -0.03    -0.36     0.36     0.11     0.16     0.18   -0.62
## 59  0.07   0.24    -0.04    -0.07    -0.14    -0.37     0.34    0.67
## 85  0.46  -0.31    -0.25    -0.40    -0.42    -0.13     0.16   -0.80
##     cov.r   cook.d hat
## 9    0.77_*  0.10   0.11
## 42   0.69_*  0.05   0.05
## 59   0.61_*  0.06   0.05
## 85   0.73_*  0.09   0.09
```

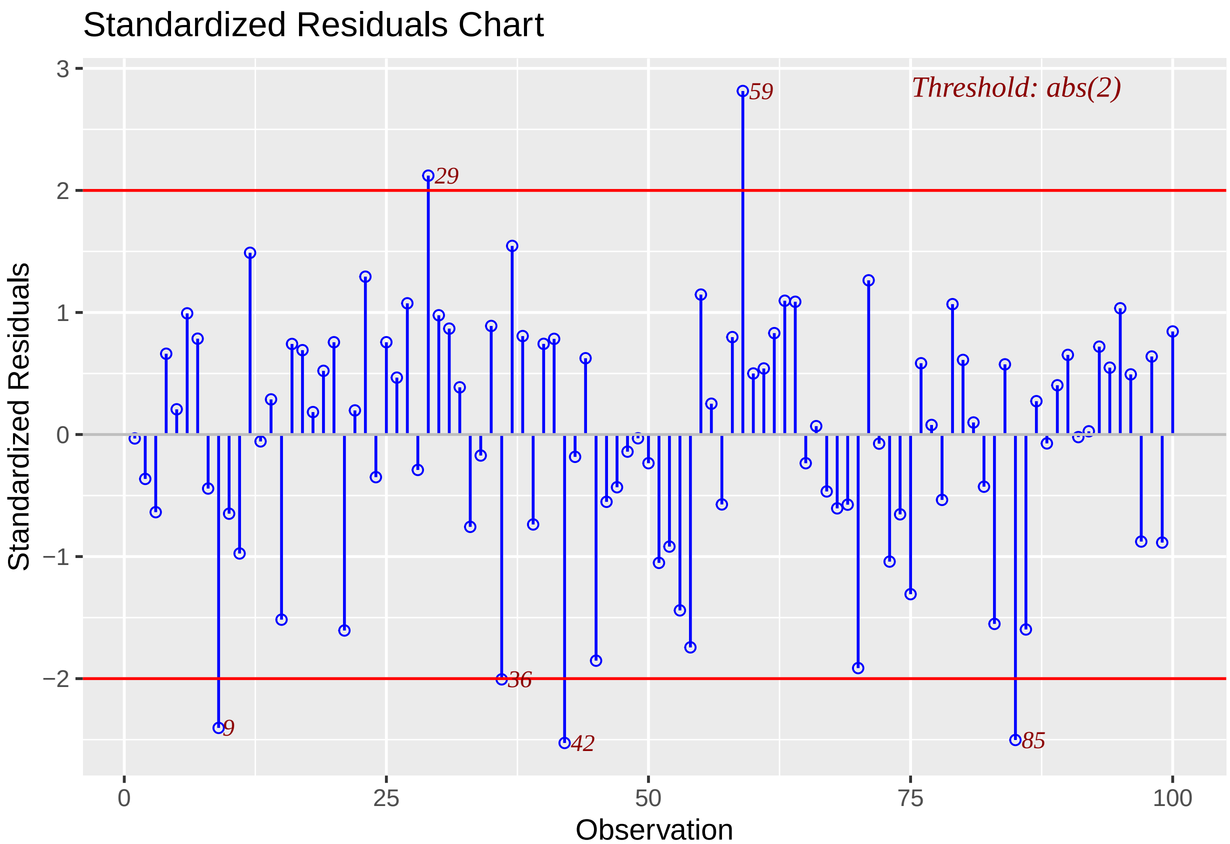
The last two columns give the cooks distance and the hatvalues. We see points 9,42,59 and 85 as influential observations given cooks distance > 0.5 and hat value also > 0.5.

We plot the results slightly better now -

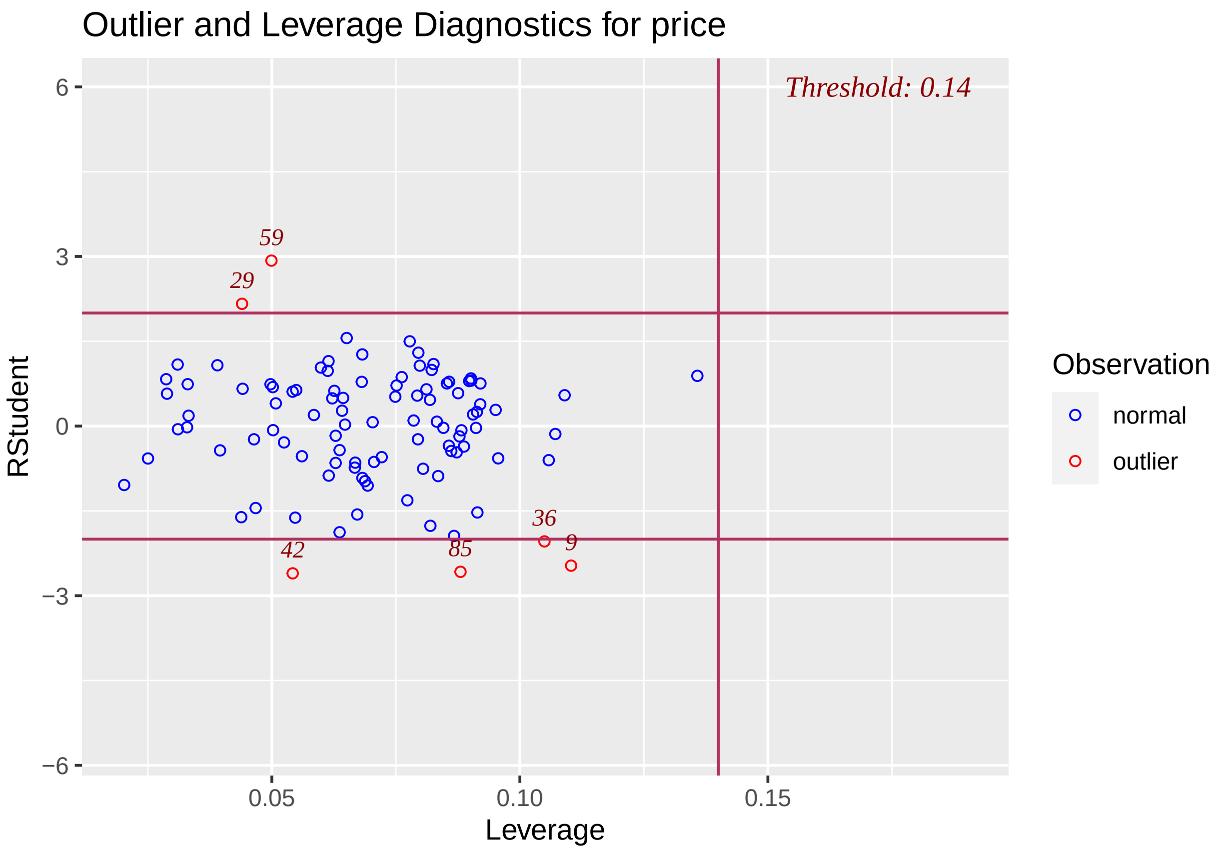
```
ols_plot_cooksd_bar(fit)
```




```
ols_plot_resid_stand(fit)
```




```r
ols_plot_resid_lev(fit)
```



Above we plotted the cooks d bar plot, the standarized residual plot and the rstudent vs leverage plot.

The first plot tells us that points 9,85, 36, 59, 42 and 70 have larger cooks distance than other points The second plot tells us that points 59, 85, 42, 9, 29 and 36 have larger residuals. And the third plot tells us that if we use a leverage threshold of 0.14 we do not get any outlier but we see how close points 9, 36, 85, 42, 29 and 59 are in the range we create and these are clearly the most influential points in the data.

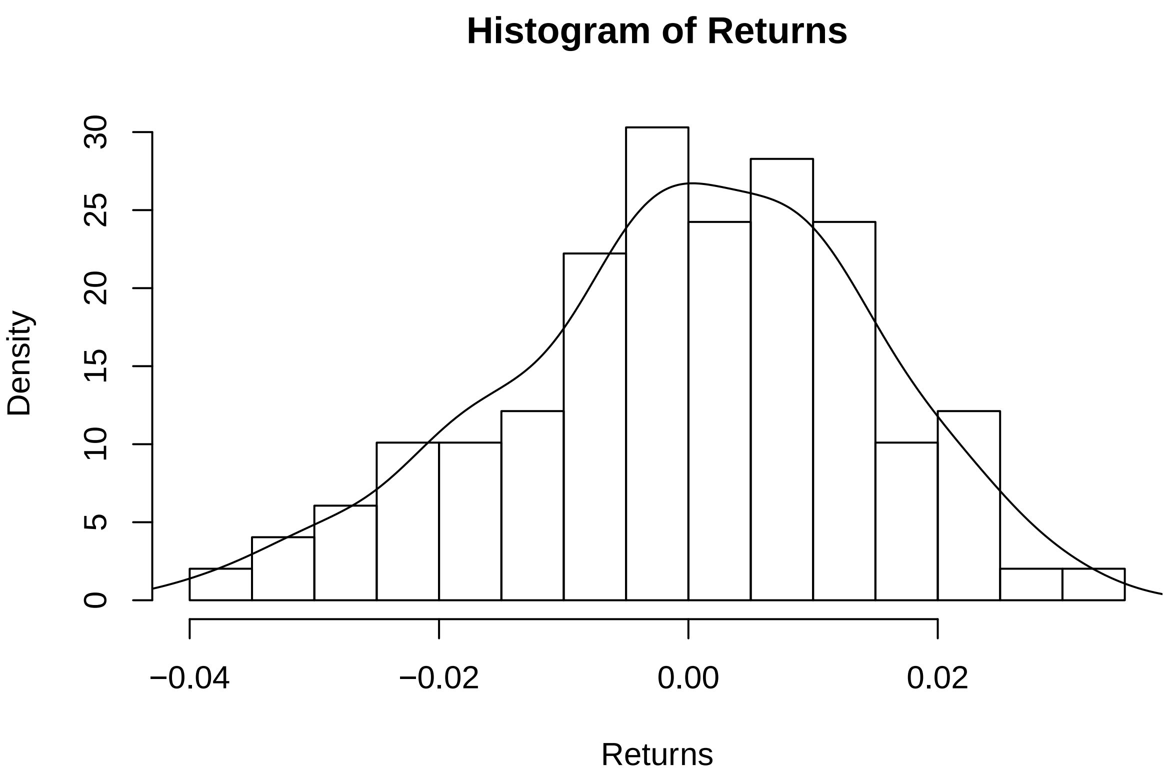
1h) The return at time t is defined as r(t)=[p(t+1)/p(t)]-1 where p is price data for day t. Are returns normally distributed ? Please justify using qq plot and normality test.

First, we create a 'return' variable in the data using the above expression.

```
# create the lag of price
stockdata$price_lag  <- lag(stockdata$price)
stockdata$return <- (stockdata$price_lag/stockdata$price) - 1

# We ignore the first observation as it is NA
return_data <- na.omit(stockdata)
```

We now plot the histogram of returns -

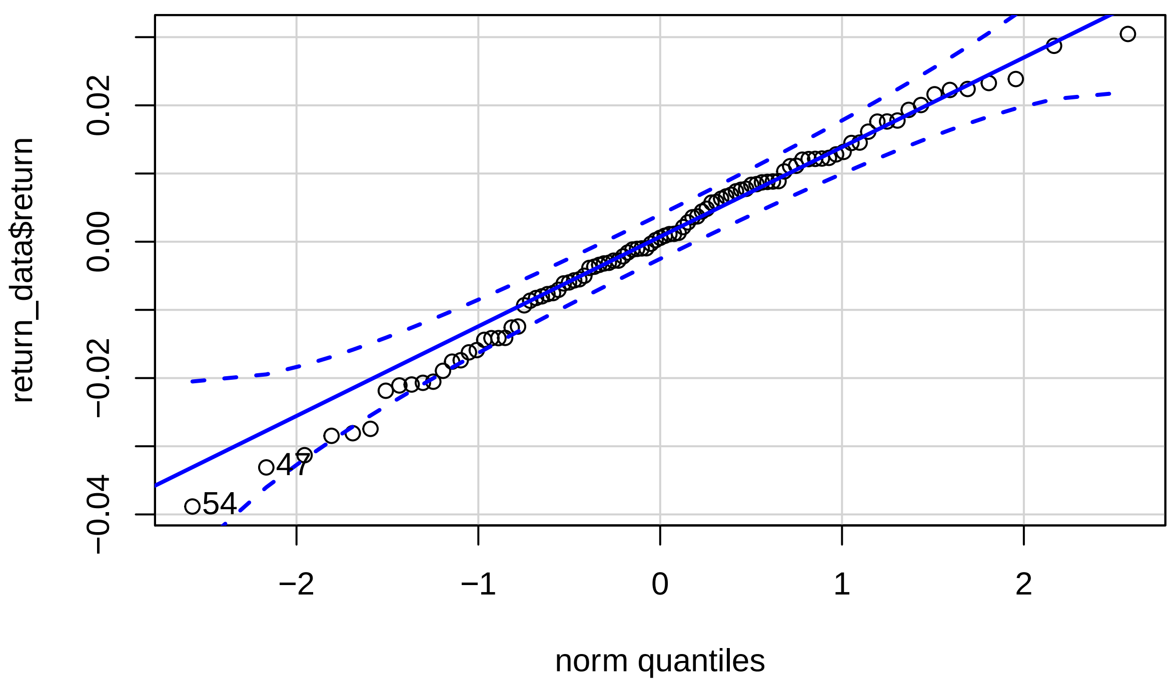
```
hist(x = return_data$return,
     breaks = 20,
     probability = TRUE,
     main = "Histogram of Returns",
     xlab = "Returns")

lines(density(return_data$return))
```



The returns seem pretty normal from above.

We now check the qq plot -

```
qqPlot(return_data$return)
```




```
## [1] 54 47
```

The QQ plot shows that returns look pretty normally distributed.

We also perform the Shapiro wilk normality test.

```r
shapiro.test(return_data$return)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  return_data$return
## W = 0.98716, p-value = 0.4559
```

From the p-value $= 0.4559 > 0.05$, we can see that the returns are normally distributed.

This makes sense to us as in most stock price modeling, one does not model the price but rather models the return.

2) Repeat question 1 from a to i on cheddar dataset from the book by fitting a model with taste as the response and the other three variables as predictors. Answer the questions posed in the first problem.

Let us load the data and summarise the information

```
# reading data
data(cheddar)
str(cheddar)
```

```
## 'data.frame':    30 obs. of  4 variables:
##  $ taste : num  12.3 20.9 39 47.9 5.6 25.9 37.3 21.9 18.1 21 ...
##  $ Acetic: num  4.54 5.16 5.37 5.76 4.66 ...
##  $ H2S   : num  3.13 5.04 5.44 7.5 3.81 ...
##  $ Lactic: num  0.86 1.53 1.57 1.81 0.99 1.09 1.29 1.78 1.29 1.58 ...
```

```
#summary
summary(cheddar)
```

```
##      taste           Acetic           H2S             Lactic
##  Min.   : 0.70   Min.   :4.477   Min.   : 2.996   Min.   :0.860
##  1st Qu.:13.55   1st Qu.:5.237   1st Qu.: 3.978   1st Qu.:1.250
##  Median :20.95   Median :5.425   Median : 5.329   Median :1.450
##  Mean   :24.53   Mean   :5.498   Mean   : 5.942   Mean   :1.442
##  3rd Qu.:36.70   3rd Qu.:5.883   3rd Qu.: 7.575   3rd Qu.:1.667
##  Max.   :57.20   Max.   :6.458   Max.   :10.199   Max.   :2.010
```

Key observations -

1. taste is +vely skewed, Acetic and H2S are marginally +vely skewed while only Lactic is negatively skewed.
2. We don't see any evidence of outliers but we will check the plots before commenting on this

Any missing values ?

```
data <-na.omit(cheddar)
str(data)
```

```
## 'data.frame':    30 obs. of  4 variables:
##  $ taste : num  12.3 20.9 39 47.9 5.6 25.9 37.3 21.9 18.1 21 ...
##  $ Acetic: num  4.54 5.16 5.37 5.76 4.66 ...
##  $ H2S   : num  3.13 5.04 5.44 7.5 3.81 ...
##  $ Lactic: num  0.86 1.53 1.57 1.81 0.99 1.09 1.29 1.78 1.29 1.58 ...
```

We did not find any missing values in the data.

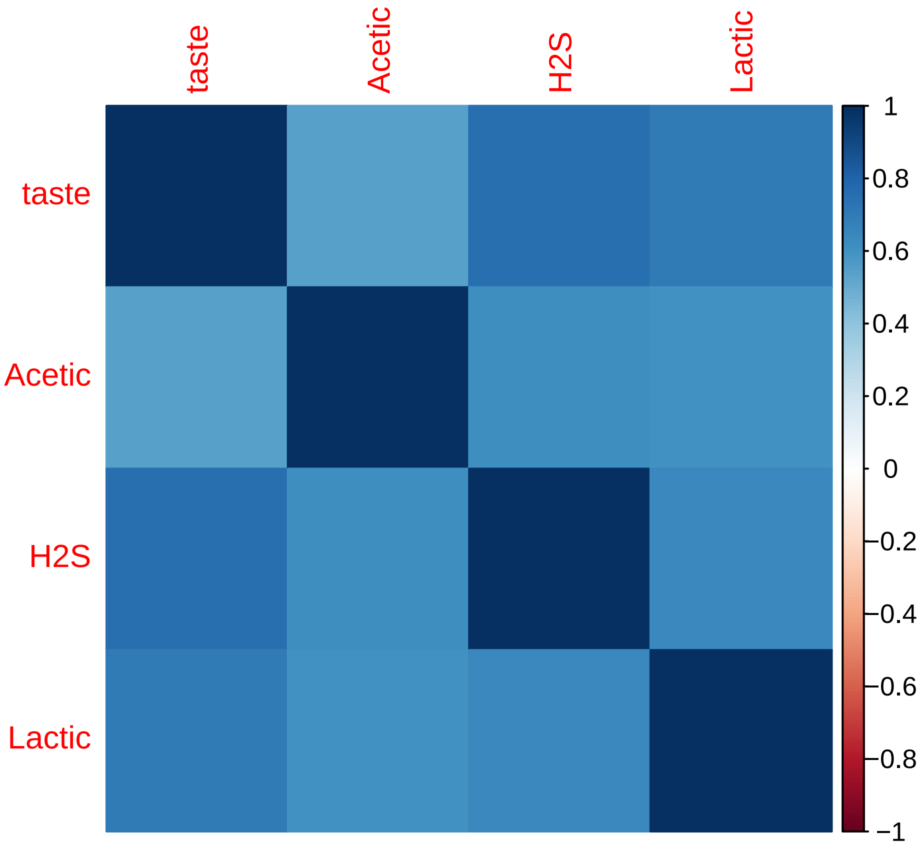
Correlation plot -

We check correlation before moving towards modeling exercise.

```
M<-cor(cheddar)
head(round(M,2))
```

```
##         taste Acetic  H2S Lactic
## taste   1.00   0.55 0.76   0.70
## Acetic  0.55   1.00 0.62   0.60
## H2S     0.76   0.62 1.00   0.64
## Lactic  0.70   0.60 0.64   1.00
```

```
corrplot(M, method="color")
```



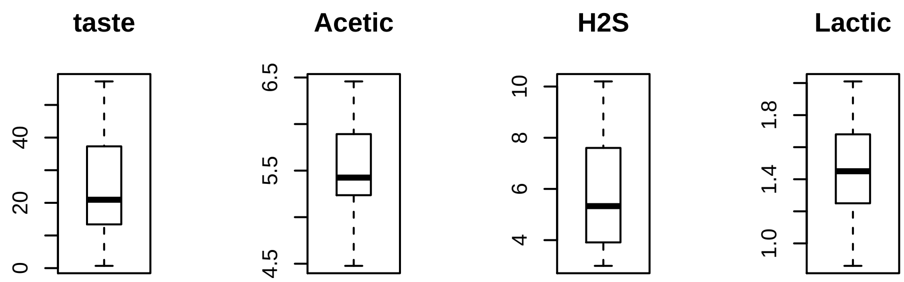
Key observations -

1. We see high correlation between taste and H2S, Lactic (>0.7). We also see Acetic is correlated well with taste (>0.5).
2. We also see strong correlations between the independent variables as well (>0.6).

Outlier/ Univariate checks -

We confirm our earlier hypothesis of no outliers by looking at some univariate and outlier checks.

```
par(mfrow=c(2,5))
for (i in 1:length(cheddar)) {
        boxplot(cheddar[,i], main=names(cheddar[i]), type="l")
}
```



We see no evidence of any outliers in the univariate form.

We now proceed to modeling exercise.

2a) Fit a model to explain taste in terms of the predictors. Which variables are important, can any of the variables be removed ? Please use F-tests to justify.

We use lm() function for this regression

```
fit <- lm(taste~Acetic+H2S+Lactic, data=cheddar)
summary(fit)
```

```
##
## Call:
## lm(formula = taste ~ Acetic + H2S + Lactic, data = cheddar)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.390  -6.612  -1.009   4.908  25.449
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -28.8768    19.7354  -1.463  0.15540
## Acetic        0.3277     4.4598   0.073  0.94198
## H2S           3.9118     1.2484   3.133  0.00425 **
## Lactic       19.6705     8.6291   2.280  0.03108 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.13 on 26 degrees of freedom
## Multiple R-squared:  0.6518, Adjusted R-squared:  0.6116
## F-statistic: 16.22 on 3 and 26 DF,  p-value: 3.81e-06
```

We see significant result in p-value for both H2S and Lactic at 5% however we do not see Acetic acid coming as significant in the model hence this variable can be removed from the model. We get F value of 16.2 with probability <0.5 indicating the joint hypothesis of this model being better than null model.

The above model has a good R-square of 65.35%.

We now confirm this with joint hypothesis test using F-statistic. We use both anova and linearHypothesis() for this.

Model without Acetic

```
fit2 <- lm(taste~+H2S+Lactic, data=cheddar)
summary(fit2)
```

```
##
## Call:
## lm(formula = taste ~ +H2S + Lactic, data = cheddar)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.343  -6.530  -1.164   4.844  25.618
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -27.592      8.982  -3.072  0.00481 **
## H2S            3.946      1.136   3.475  0.00174 **
```

```
## Lactic        19.887      7.959   2.499  0.01885 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.942 on 27 degrees of freedom
## Multiple R-squared:  0.6517, Adjusted R-squared:  0.6259
## F-statistic: 25.26 on 2 and 27 DF,  p-value: 6.551e-07
```

We now look at Anova

```r
anova(fit2, fit)
```

```
## Analysis of Variance Table
##
## Model 1: taste ~ +H2S + Lactic
## Model 2: taste ~ Acetic + H2S + Lactic
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     27 2669.0
## 2     26 2668.4  1   0.55427 0.0054  0.942
```

Since p-value of 0.942 is > 0.5 we conclude that the model with Acetic is not significantly better than the model without Acetic

We can see this with individual F tests for each variable using linearhypothesis() now as well.

```r
linearHypothesis(fit, c("Acetic=0"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## Acetic = 0
##
## Model 1: restricted model
## Model 2: taste ~ Acetic + H2S + Lactic
##
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     27 2669.0
## 2     26 2668.4  1   0.55427 0.0054  0.942
```

```r
linearHypothesis(fit, c("H2S =0"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## H2S = 0
##
## Model 1: restricted model
## Model 2: taste ~ Acetic + H2S + Lactic
##
##   Res.Df    RSS Df Sum of Sq      F   Pr(>F)
## 1     27 3676.1
## 2     26 2668.4  1    1007.7 9.8182 0.004247 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
linearHypothesis(fit, c("Lactic=0"))
```

```
## Linear hypothesis test
```

```
##
## Hypothesis:
## Lactic = 0
##
## Model 1: restricted model
## Model 2: taste ~ Acetic + H2S + Lactic
##
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1     27 3201.7
## 2     26 2668.4  1    533.32 5.1964 0.03108 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We ran the F-tests for the hypothesis of each individual variable and in only one case is the F-score is very low (0.0054) in the 'Acetic'=0 hypothesis. Hence, this confirms from our linear regression model again that given the Pr(>F) is 0.94, we find concentration of acetic acid to not influence the taste of cheese and hence we can remove this variable from our model.

Sometimes, however it is important to run the heteroskedastic robust version of the F-test as well. We do this as follows -

```r
linearHypothesis(fit, c("Acetic=0"),white.adjust = "hc1")
```

```
## Linear hypothesis test
##
## Hypothesis:
## Acetic = 0
##
## Model 1: restricted model
## Model 2: taste ~ Acetic + H2S + Lactic
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F Pr(>F)
## 1     27
## 2     26  1 0.0052 0.9429
```

```r
linearHypothesis(fit, c("H2S =0"),white.adjust = "hc1")
```

```
## Linear hypothesis test
##
## Hypothesis:
## H2S = 0
##
## Model 1: restricted model
## Model 2: taste ~ Acetic + H2S + Lactic
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F    Pr(>F)
## 1     27
## 2     26  1 25.327 3.083e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
linearHypothesis(fit, c("Lactic=0"),white.adjust = "hc1")
```

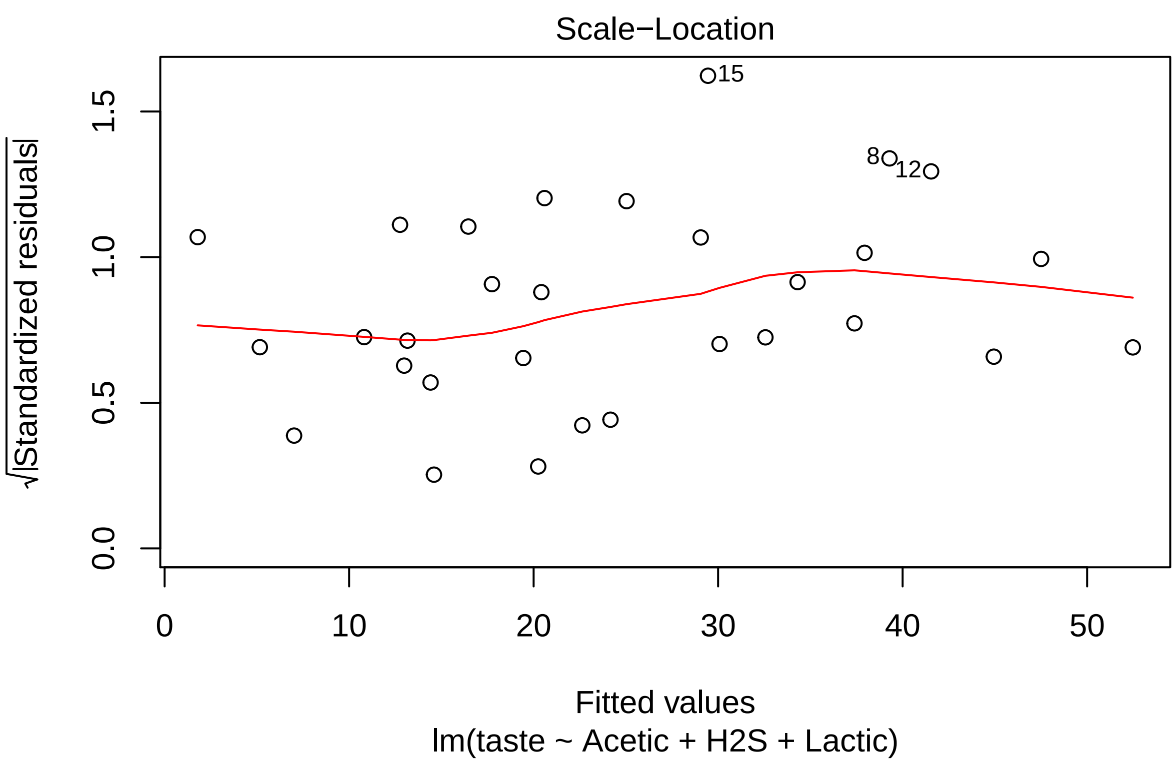
```
## Linear hypothesis test
##
## Hypothesis:
## Lactic = 0
##
## Model 1: restricted model
## Model 2: taste ~ Acetic + H2S + Lactic
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F   Pr(>F)
## 1     27
## 2     26  1 8.1261 0.008431 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see no change in results although the Pr(>F) for Acetic=0 hypothesis is slightly higher than before. We still only confirm that Acetic can be removed from the model.

2b) Check the constant variance assumption for the errors.

We do this in 3 ways - we look at scale-location plot of the regression, then perform ncvTest() and finally Breusch-pagan test for homoskedasticity.

```r
plot(fit,which=3)
```



We mostly see as they say “stars in the sky” expression in the above graph. While the line visually is not completely horizontal, we do not in particular see any pattern and it would be hard to deny that the errors are homoskedastic. However, we perform further tests to confirm our suspicion.

ncvTest() For Homoscedasticity

```r
ncvTest(fit)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 1.157465, Df = 1, p = 0.28199
```

We see a p-value > .05, indicating homoscedasticity.

Breusch-Pagan Test For Homoscedasticity

```r
bptest(fit)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  fit
## BP = 4.2193, df = 3, p-value = 0.2387
```

We once again see a p-value > .05, indicating homoscedasticity.

2c) Check the independentness of the errors assumption.

We can do this with the durbin watson statistic. The Durbin Watson examines whether the errors are autocorrelated with themselves. The null states that they are not autocorrelated.

```r
durbinWatsonTest(fit)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.1692325       1.57513   0.188
##  Alternative hypothesis: rho != 0
```

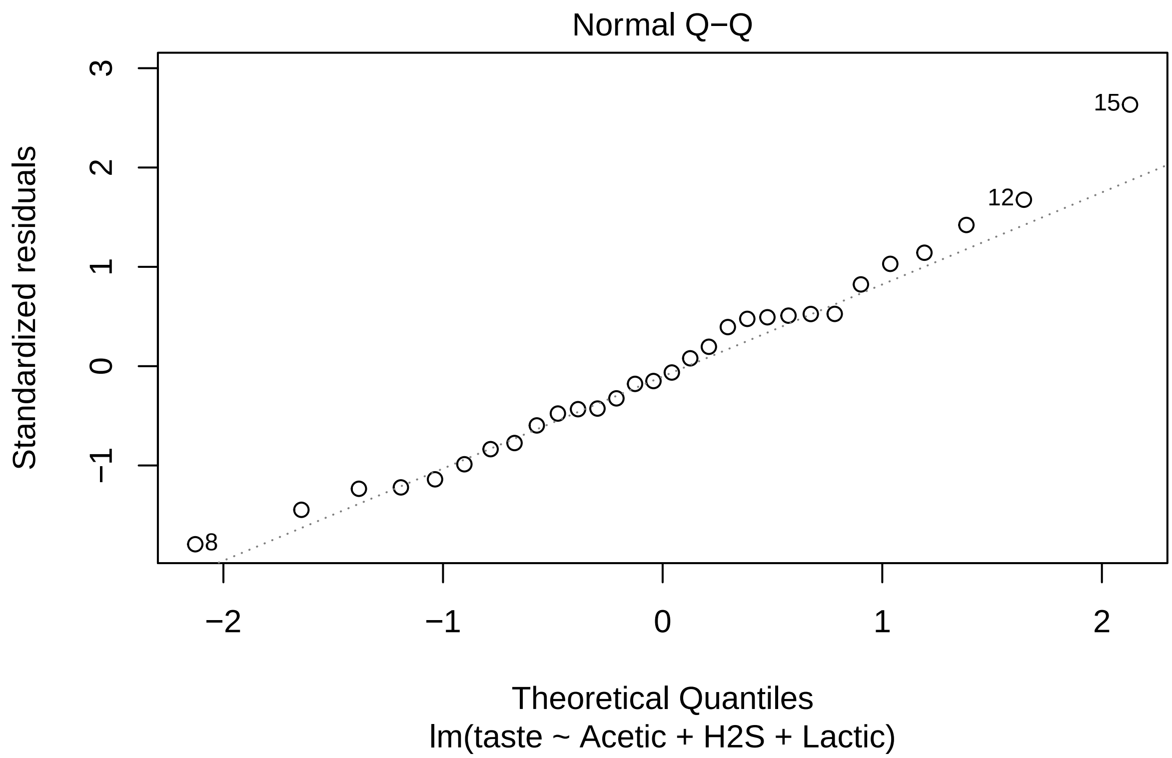
We see that p-value > 0.05 so the errors are not autocorrelated.

1d) Check the normality assumption.

We again do this in 2 ways - we look at QQ plot and perform the Shapiro Wilk normality test.

The normal probability plot of residuals should approximately follow a straight line.

```r
plot(fit,which=2)
```



We see points falling mostly along reference line. We see some minor observations falling outside range but still we feel that the residuals are normal. We confirm our suspicions with a statistical test.

Shapiro-Wilk Normality Test

```
resid <- studres(fit)
shapiro.test(resid)
```

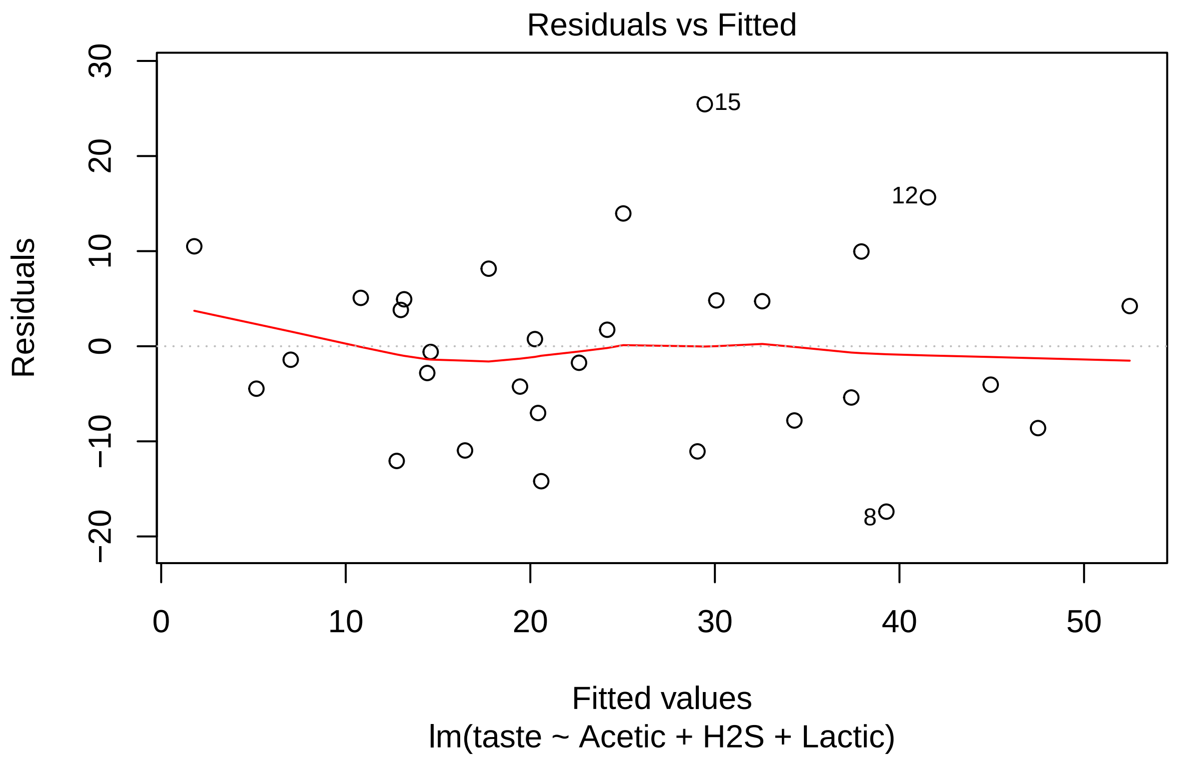
```
##
##  Shapiro-Wilk normality test
##
## data:  resid
## W = 0.97019, p-value = 0.5444
```

From the p-value = $0.5444 > 0.05$, we can see that the residuals are normal which satisfies the linear regression assumption.

1e) Is non-linearity a problem ?

The linearity assumption can be checked by inspecting the Residuals vs Fitted plot.

```
plot(fit,which=1)
```



We see the linearity relationship mostly holds in the data indicating non-linearity isn't an issue. This might be because the acid levels are already in a log scale.

2f) Check for outliers, compute and plot the cook's distance.

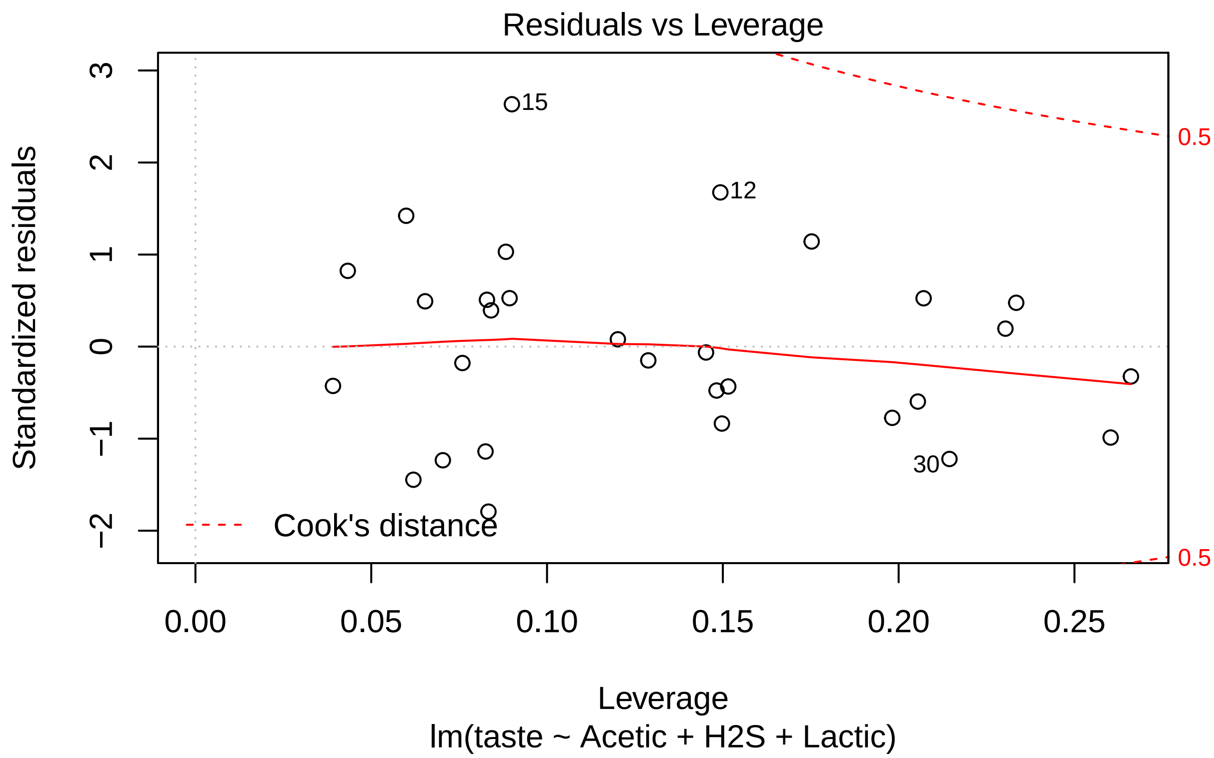
A standard way to check for outliers is to look at residuals above a certain threshold. An example would be as follows -

```r
rstandard(fit)[abs(rstandard(fit)) > 1.5]
```

```
##         8        12        15
## -1.792952  1.675450  2.633351
```

Here, we see points 8,12 and 15 with large residuals but note that not all of them or maybe none of them could be outliers. So we now look at the model plot of Residuals vs leverage.

```r
plot(fit, which=5)
```

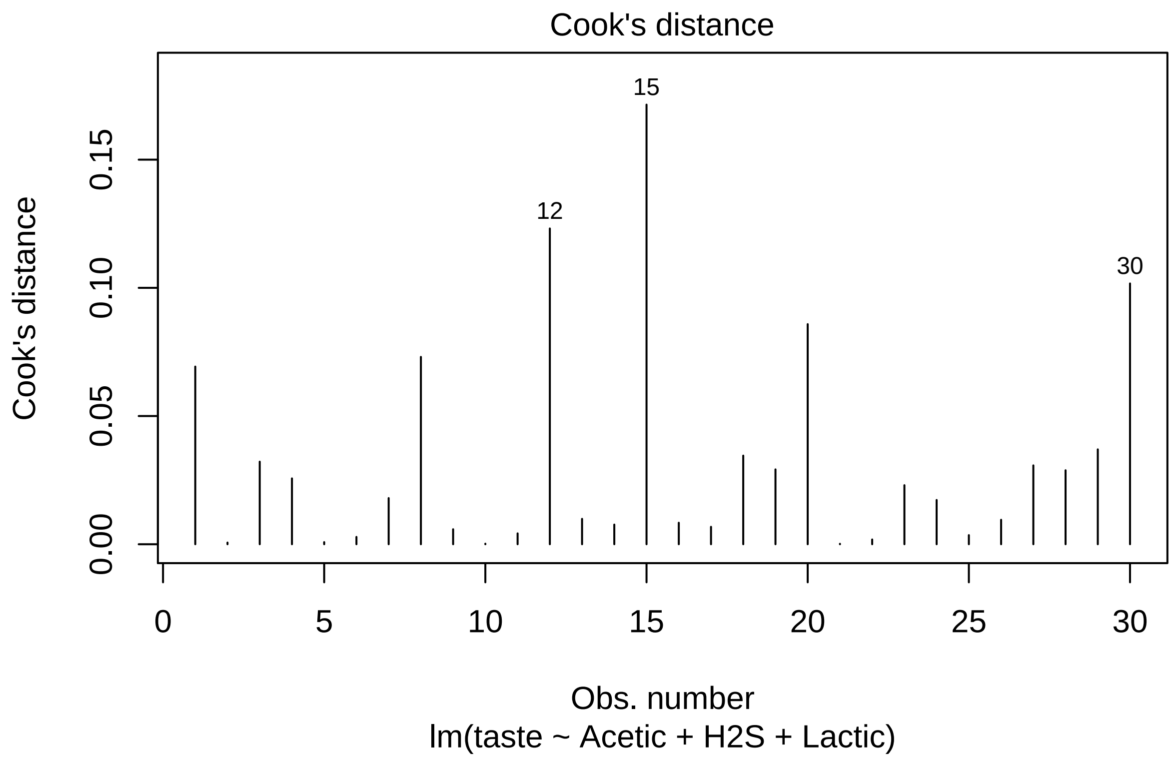


Leverage statistic is defined as -
$\hat{L} = \dfrac{2(p+1)}{n}$ where p is number of predictors and n is number of observations

In the above graph we see all points fall under the dashed lines of the cook's distance (missing) which tells us there are no outliers in the data but still some influential points do exist.

We can plot the cook's distance with the below command -

```
#Cook's distance
plot(fit, 4)
```



We see that apart from three points - 15, 12 and 30, everyone else's cook's distance is below 0.1

We also compute the cook's distance for each observation as follows -

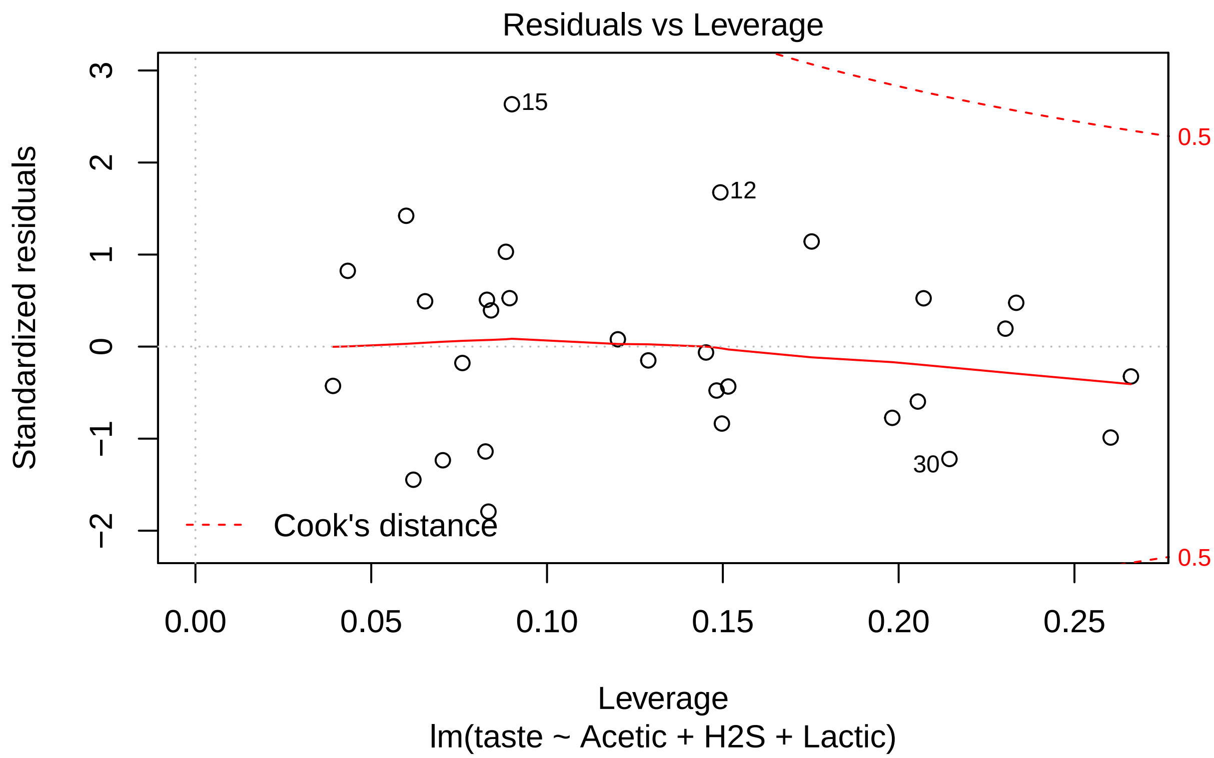
```r
cooks.distance(fit)
```

```
##            1            2            3            4            5
## 0.0692954594 0.0006538367 0.0322134155 0.0256812534 0.0008307915
##            6            7            8            9           10
## 0.0028512031 0.0179979270 0.0730650866 0.0058503793 0.0002133561
##           11           12           13           14           15
## 0.0042356476 0.1231598979 0.0099034331 0.0076724201 0.1714653442
##           16           17           18           19           20
## 0.0083806741 0.0067909340 0.0345647920 0.0291892000 0.0858382548
##           21           22           23           24           25
## 0.0001742073 0.0018567690 0.0230305489 0.0172589455 0.0035559153
##           26           27           28           29           30
## 0.0095340749 0.0307525936 0.0288491545 0.0370008615 0.1017165104
```

2g) Check for influential points.

This was partially done above itself, but neverthless we can check for influential points through the plot itself.

```r
# High leverage points
plot(fit, which=5)
```



So from this plot again, we see points 15, 12 and 30 as values of extreme nature and we see these as influential points.

A rule of thumb is that an observation has high influence if Cook's distance exceeds $\frac{4}{(n-p-1)}$

```
cooks.distance(fit) > 4 / length(cooks.distance(fit))
```

```
##     1     2     3     4     5     6     7     8     9    10    11    12
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##    13    14    15    16    17    18    19    20    21    22    23    24
## FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##    25    26    27    28    29    30
## FALSE FALSE FALSE FALSE FALSE FALSE
```

We see only point 15 as influential point just going by cooks distance.

We also however check for hatvalues in the data.

```
hatvalues(fit) > 0.25
```

```
##     1     2     3     4     5     6     7     8     9    10    11    12
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##    13    14    15    16    17    18    19    20    21    22    23    24
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
##    25    26    27    28    29    30
## FALSE  TRUE FALSE FALSE FALSE FALSE
```

We see only few observations with hatvalues above 0.25. These are points 20 and 26.

Now, we combine the above two results of cooks distance and hatvalues which is exactly what influence.measures does for us.

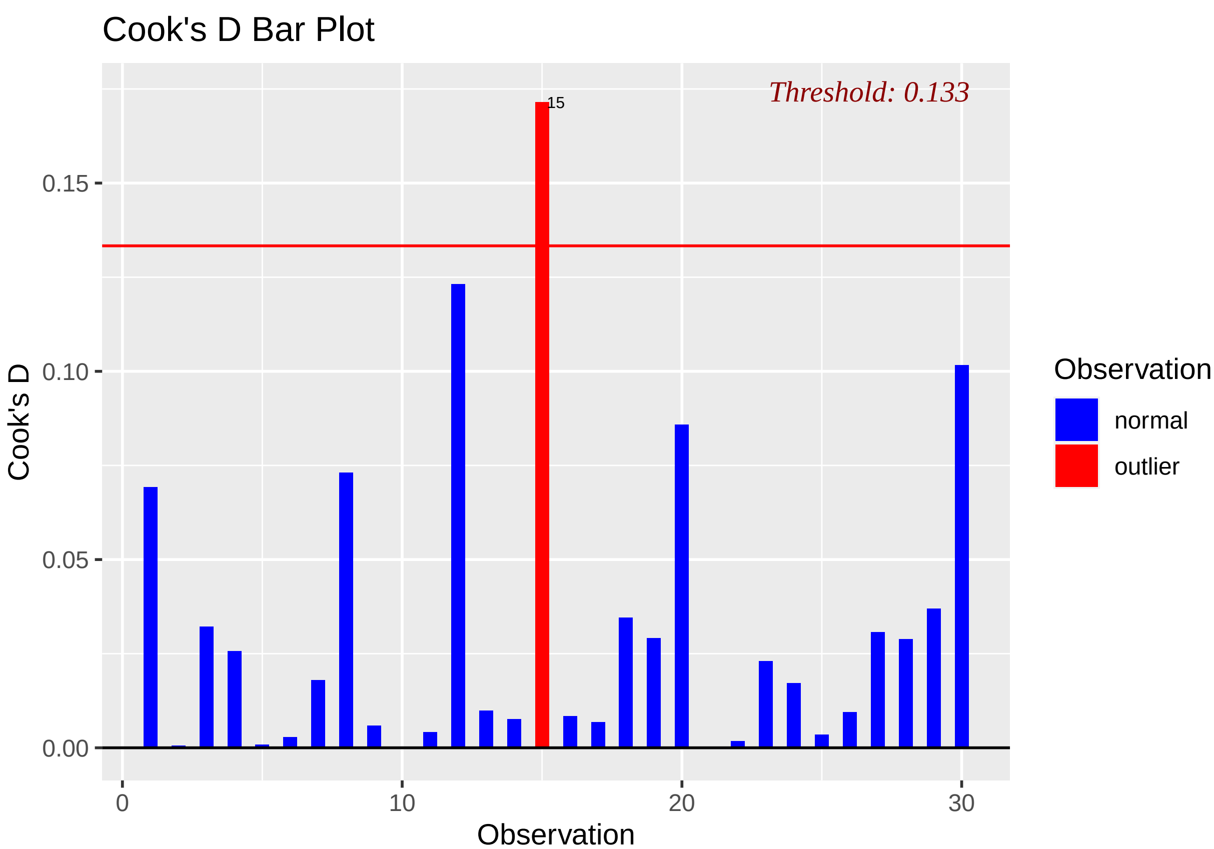
```
summary(influence.measures(fit))
```

```
## Potentially influential observations of
##   lm(formula = taste ~ Acetic + H2S + Lactic, data = cheddar) :
##
##    dfb.1_ dfb.Actc dfb.H2S dfb.Lctc dffit cov.r   cook.d hat
## 6   0.01   0.02     0.07   -0.09     0.10  1.51_*  0.00   0.23
## 15 -0.65   0.72    -0.12   -0.24     0.95  0.37_*  0.17   0.09
## 24  0.03  -0.11     0.14    0.11     0.26  1.47_*  0.02   0.23
## 26  0.12  -0.13     0.16   -0.03    -0.19  1.57_*  0.01   0.27
```

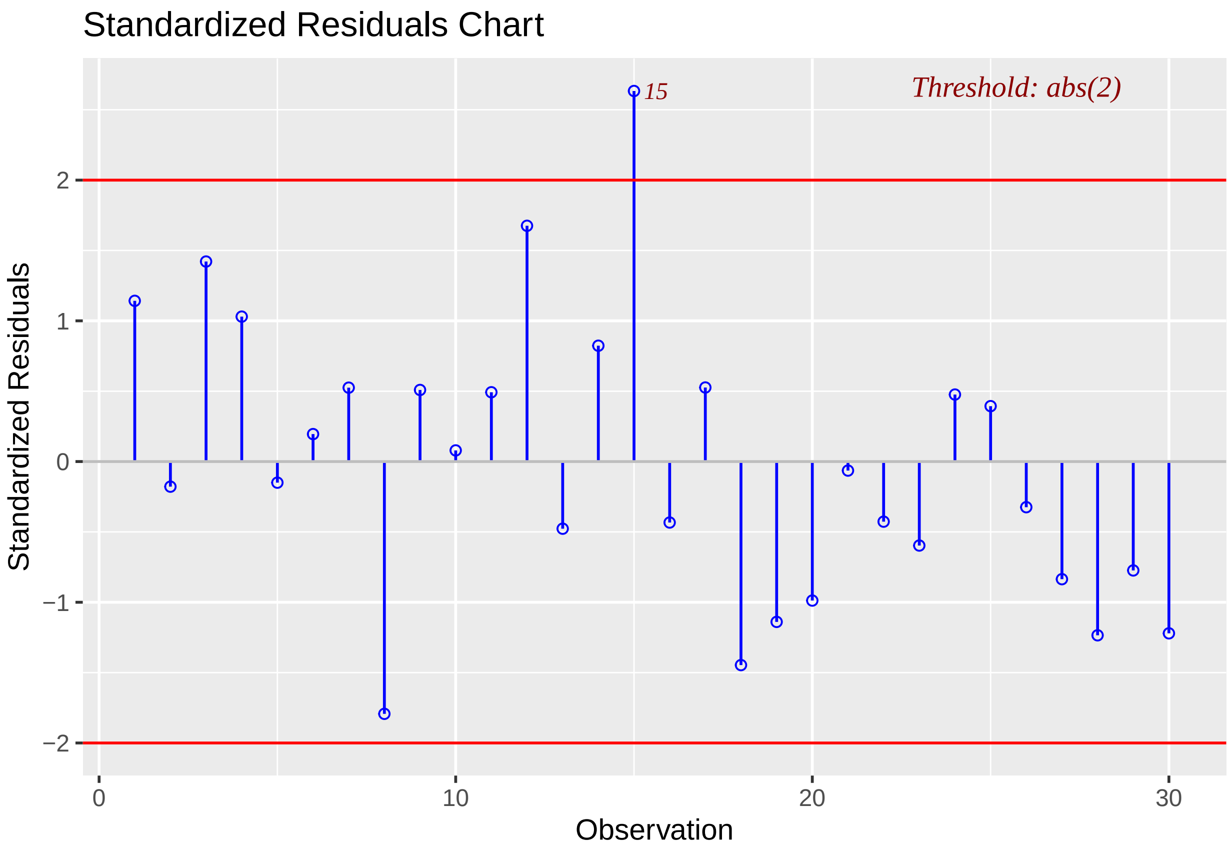
The last two columns give the cooks distance and the hatvalues. We see points 6, 15, 24 and 26 as influential observations however points 6,24 and 26 have high hat value but low cooksd whereas point 15 is definitely the most influential observation in the data.

We plot the results slightly better now -

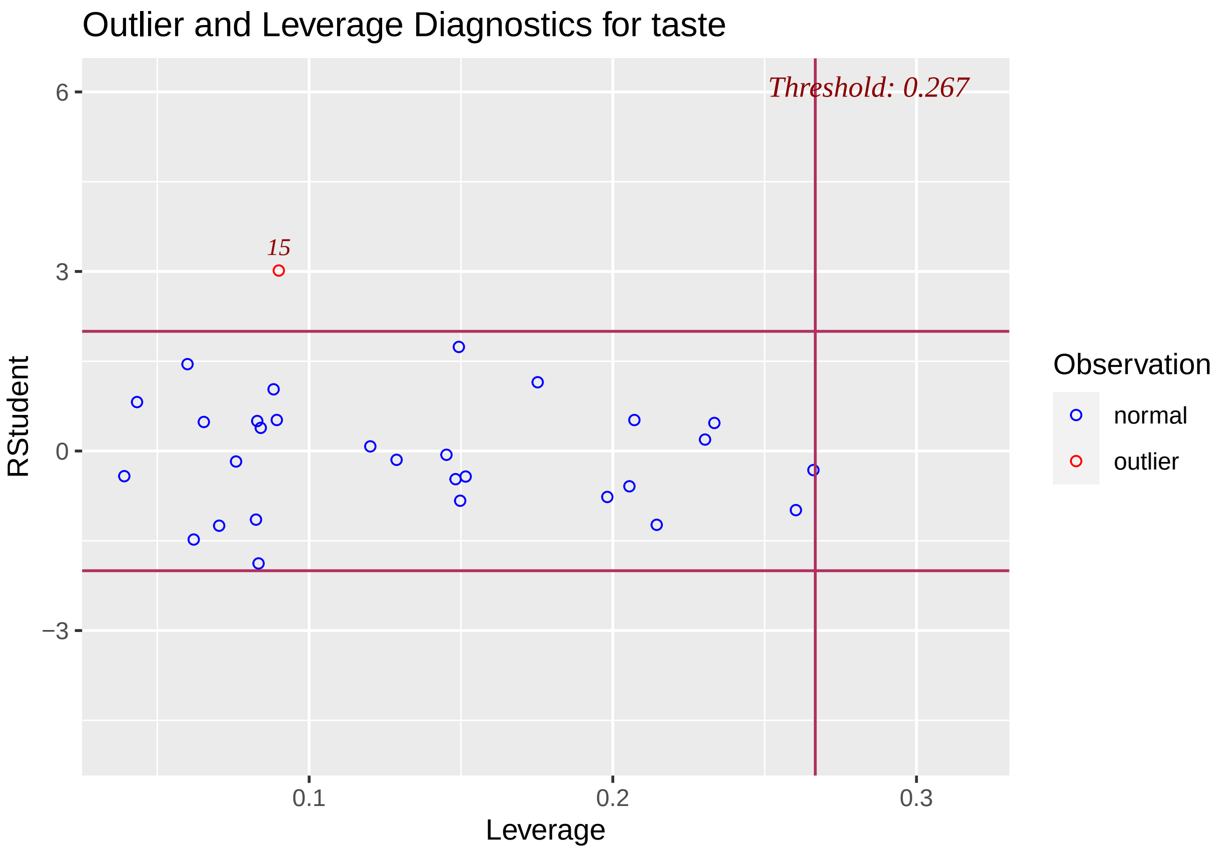
```
ols_plot_cooksd_bar(fit)
```




```
ols_plot_resid_stand(fit)
```




```r
ols_plot_resid_lev(fit)
```



Above we plotted the cooks d bar plot, the standarized residual plot and the rstudent vs leverage plot.

The first plot tells us that point 15 has larger cooks distance than other points The second plot tells us that points 15 again has larger residuals And the third plot tells us that if we use a leverage threshold of 0.267 we do not get any outlier but we see 15 as highly influential given the large residual.

3) The problem is to discover relation between US new house construction starts data (HOUST) and macro economic indicators : GDP, CPI, and POP.

a) Combine all data into an R dataframe object and construct dummy or factor variable for 4 quarters. First model is HOUST~GDP+ CPI + quarter

We first load all the files and look at their structures-

Loading files

```
# in the cpi file, the data starts from row 55 onwards so we skip the first 54 rows
cpi <- read_excel('/Users/mac/Downloads/final-2020-canvas/datasets/House/CPI.xls',skip=54)
# in the gdp file, the data starts from row 19 onwards so we skip the first 18 rows
gdp <- read_excel('/Users/mac/Downloads/final-2020-canvas/datasets/House/GDP.xls',skip=18)
# in the pop file, the data starts from row 11 onwards so we skip the first 10 rows
pop <- read_excel('/Users/mac/Downloads/final-2020-canvas/datasets/House/POP.xls',skip=10)
# in the houst file, the data starts from row 11 onwards so we skip the first 10 rows
houst <- read_excel('/Users/mac/Downloads/final-2020-canvas/datasets/House/HOUST.xls',skip=10)
```

```
str(cpi)
```

```
## tibble [161 x 2] (S3: tbl_df/tbl/data.frame)
##  $ DATE : POSIXct[1:161], format: "1976-01-01" "1976-04-01" ...
##  $ VALUE: num [1:161] 0.633 0.5 0.9 0.833 1.067 ...
```

```
str(gdp)
```

```
## tibble [163 x 2] (S3: tbl_df/tbl/data.frame)
##  $ DATE : POSIXct[1:163], format: "1976-01-01" "1976-04-01" ...
##  $ VALUE: num [1:163] 58.6 32.4 33.6 47.9 54.1 ...
```

```
str(pop)
```

```
## tibble [160 x 2] (S3: tbl_df/tbl/data.frame)
##  $ observation_date   : POSIXct[1:160], format: "1976-01-01" "1976-04-01" ...
##  $ B230RC0Q173SBEA_CHG: num [1:160] 462 562 579 510 529 617 628 518 562 652 ...
```

```
str(houst)
```

```
## tibble [161 x 2] (S3: tbl_df/tbl/data.frame)
##  $ observation_date: POSIXct[1:161], format: "1975-10-01" "1976-01-01" ...
##  $ HOUST           : num [1:161] 297 281 439 434 383 ...
```

Key observations -

1. We see that cpi has 161 observations from 1976-01-01 till 2016-01-01
2. We see that gdp has 163 observations from 1976-01-01 till 2016-07-01
3. We see that pop has 160 observations from 1976-01-01 till 2015-10-01
4. We see that houst has 161 observations from 1975-10-01 till 2015-10-01.

We use a nested merge to merge the 4 files. Do note, we do not include some observations that only have a GDP value or only a CPI value. We take the data from 1976 1st quarter till 2015 last quarter this way.

Merging the 4 datasets

We merge the four datasets and convert the DATE variable from POSIXct to date format.

```
data  <- merge(merge(merge(houst,cpi, by.x='observation_date', by.y='DATE')
               ,gdp, by.x='observation_date', by.y='DATE')
               ,pop, by.x='observation_date', by.y='observation_date')
colnames(data)[1] <- "DATE"
colnames(data)[2] <- "HOUST"
colnames(data)[3] <- "CPI"
colnames(data)[4] <- "GDP"
colnames(data)[5] <- "POP"
# we also convert the date variable from posixct to date format
data$DATE <- as.Date(data$DATE)
str(data)
```

```
## 'data.frame':    160 obs. of  5 variables:
##  $ DATE : Date, format: "1976-01-01" "1976-04-01" ...
##  $ HOUST: num  281 439 434 383 367 ...
##  $ CPI  : num  0.633 0.5 0.9 0.833 1.067 ...
##  $ GDP  : num  58.6 32.4 33.6 47.9 54.1 ...
##  $ POP  : num  462 562 579 510 529 617 628 518 562 652 ...
```

Constructing the dummy variable for each quarter as 1-4

```
data$quarter = as.factor(substr(as.yearqtr(data$DATE),7,8))
str(data)
```

```
## 'data.frame':    160 obs. of  6 variables:
##  $ DATE   : Date, format: "1976-01-01" "1976-04-01" ...
##  $ HOUST  : num  281 439 434 383 367 ...
##  $ CPI    : num  0.633 0.5 0.9 0.833 1.067 ...
##  $ GDP    : num  58.6 32.4 33.6 47.9 54.1 ...
##  $ POP    : num  462 562 579 510 529 617 628 518 562 652 ...
##  $ quarter: Factor w/ 4 levels "1","2","3","4": 1 2 3 4 1 2 3 4 1 2 ...
```

We can also create individual factor variables from below and add them as binary variables to the data

```
quarter_data <- data.table::dcast(data, DATE ~ paste0("Q", lubridate::quarter(data$DATE)), length,
                value.var = "DATE")
houst_data <- merge(data,quarter_data,by=c("DATE"))
houst_data$Q1 <- as.factor(houst_data$Q1)
houst_data$Q2 <- as.factor(houst_data$Q2)
houst_data$Q3 <- as.factor(houst_data$Q3)
houst_data$Q4 <- as.factor(houst_data$Q4)
str(houst_data)
```

```
## 'data.frame':    160 obs. of  10 variables:
##  $ DATE   : Date, format: "1976-01-01" "1976-04-01" ...
##  $ HOUST  : num  281 439 434 383 367 ...
##  $ CPI    : num  0.633 0.5 0.9 0.833 1.067 ...
##  $ GDP    : num  58.6 32.4 33.6 47.9 54.1 ...
```

```
##  $ POP    : num  462 562 579 510 529 617 628 518 562 652 ...
##  $ quarter: Factor w/ 4 levels "1","2","3","4": 1 2 3 4 1 2 3 4 1 2 ...
##  $ Q1     : Factor w/ 2 levels "0","1": 2 1 1 1 2 1 1 1 2 1 ...
##  $ Q2     : Factor w/ 2 levels "0","1": 1 2 1 1 1 2 1 1 1 2 ...
##  $ Q3     : Factor w/ 2 levels "0","1": 1 1 2 1 1 1 2 1 1 1 ...
##  $ Q4     : Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 2 1 1 ...
```

We now fit the first model as in question

```r
fit <- lm(HOUST~GDP+CPI+quarter, data=houst_data)
summary(fit)
```

```
##
## Call:
## lm(formula = HOUST ~ GDP + CPI + quarter, data = houst_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -266.68  -69.19   12.62   71.28  217.21
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 271.0686    20.9148  12.961  < 2e-16 ***
## GDP           0.2208     0.1207   1.829 0.069328 .
## CPI           1.8468     9.8302   0.188 0.851224
## quarter2    105.3363    23.5381   4.475 1.48e-05 ***
## quarter3     88.2852    23.4548   3.764 0.000237 ***
## quarter4     30.4973    23.4202   1.302 0.194801
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 104.4 on 154 degrees of freedom
## Multiple R-squared:  0.1782, Adjusted R-squared:  0.1515
## F-statistic: 6.677 on 5 and 154 DF,  p-value: 1.173e-05
```

We see a model with R-sq of 17.8%. We notice how the intercept term is significant. The GDP is very close as well (0.06 p-value) however the significant terms in the model are quarter 2 and quarter 3. CPI is not significant here.

It doesn't matter if we use the binary variables as well. We get the same R-square and model. Do note we don't put all 4 quarter variables due to collinearity issues but choose only Q2-Q4 here.

```r
fit <- lm(HOUST~GDP+CPI+Q2+Q3+Q4, data=houst_data)
summary(fit)
```

```
##
## Call:
## lm(formula = HOUST ~ GDP + CPI + Q2 + Q3 + Q4, data = houst_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -266.68  -69.19   12.62   71.28  217.21
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 271.0686    20.9148  12.961  < 2e-16 ***
## GDP           0.2208     0.1207   1.829 0.069328 .
```

```
## CPI            1.8468     9.8302   0.188 0.851224
## Q21          105.3363    23.5381   4.475 1.48e-05 ***
## Q31           88.2852    23.4548   3.764 0.000237 ***
## Q41           30.4973    23.4202   1.302 0.194801
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 104.4 on 154 degrees of freedom
## Multiple R-squared:  0.1782, Adjusted R-squared:  0.1515
## F-statistic: 6.677 on 5 and 154 DF,  p-value: 1.173e-05
```

3b) Do you think the data needs some cleaning. If so, clean the data.

We start by checking the summary of data.

```r
summary(houst_data)
```

```
##       DATE                HOUST            CPI               GDP
##  Min.   :1976-01-01   Min.   :114.4   Min.   :-5.012   Min.   :-293.10
##  1st Qu.:1985-12-09   1st Qu.:274.5   1st Qu.: 0.833   1st Qu.:  62.27
##  Median :1995-11-16   Median :357.4   Median : 1.125   Median : 101.20
##  Mean   :1995-11-15   Mean   :351.9   Mean   : 1.143   Mean   : 102.86
##  3rd Qu.:2005-10-24   3rd Qu.:440.4   3rd Qu.: 1.500   3rd Qu.: 140.07
##  Max.   :2015-10-01   Max.   :624.5   Max.   : 3.323   Max.   : 283.80
##       POP          quarter Q1       Q2       Q3       Q4
##  Min.   :441.0   1:40    0:120   0:120   0:120   0:120
##  1st Qu.:574.0   2:40    1: 40   1: 40   1: 40   1: 40
##  Median :650.5   3:40
##  Mean   :662.0   4:40
##  3rd Qu.:746.8
##  Max.   :947.0
```

Key observations -

1. We see that CPI and GDP have some negative values. In theory this is possible during negative inflation and recession in the economy can make the GDP growth negative.

Let's check their scale.

```r
which(houst_data$CPI < 0)
```

```
##  [1]  42 104 110 124 132 133 138 150 156 157
```

```r
which(houst_data$GDP < 0)
```

```
## [1]  25  60 129 132 133 134
```

We see 10 observations with negative CPI (~6.25%)
We see 6 observations with negative GDP (~3.75%)
We see 2 observations with both -ve CPI and -ve GDP

Let's delete the data and refit the model.

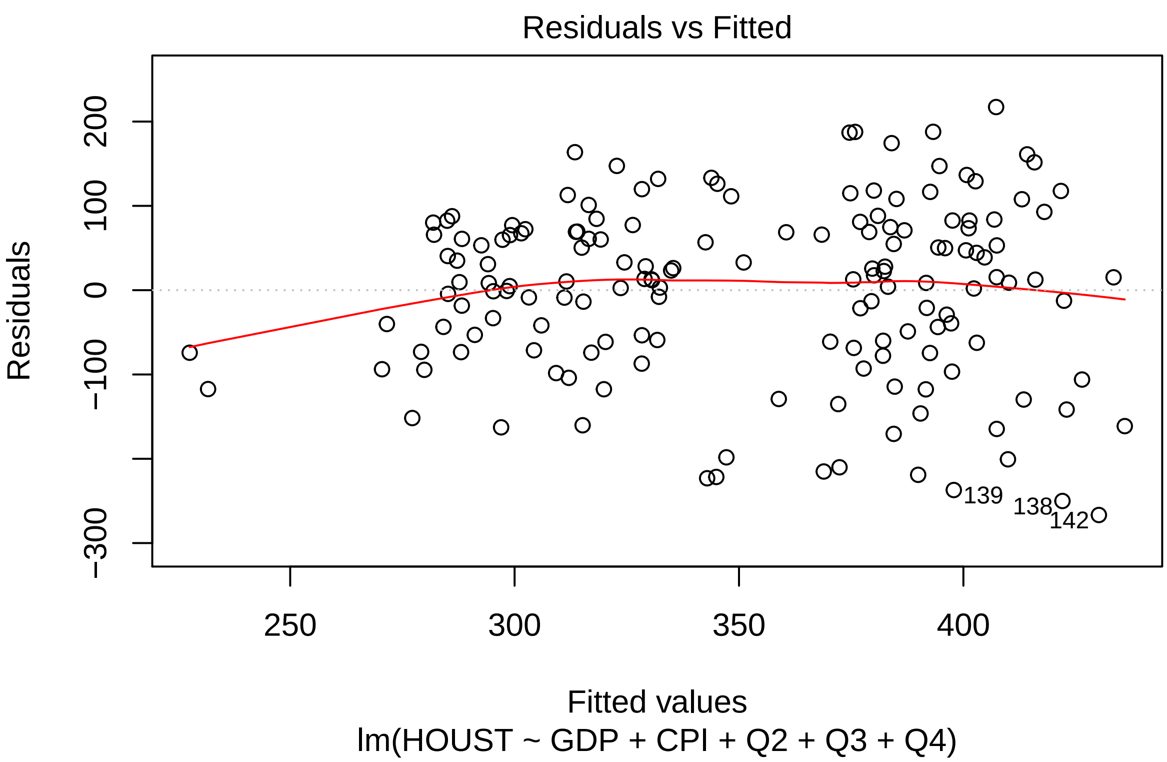
```r
houst_data_clean <- houst_data[houst_data$CPI >= 0, ]
houst_data_clean <- houst_data_clean[houst_data_clean$GDP >= 0, ]
str(houst_data_clean)
```

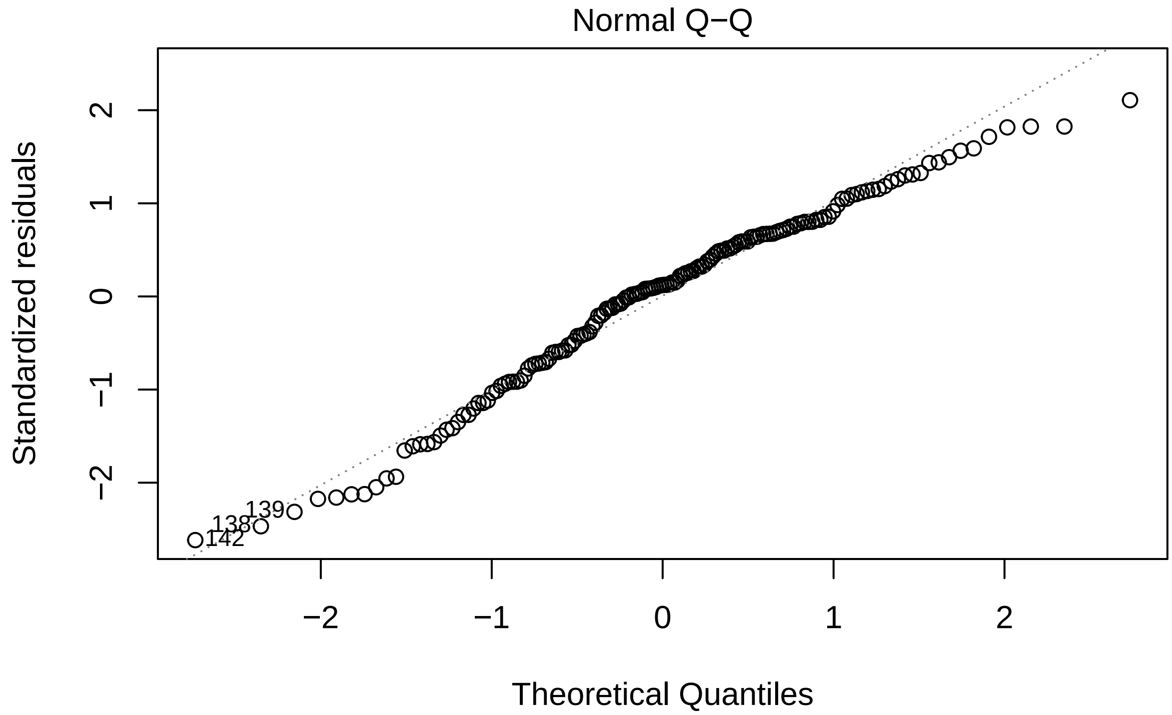
```
## 'data.frame':    146 obs. of  10 variables:
##  $ DATE   : Date, format: "1976-01-01" "1976-04-01" ...
##  $ HOUST  : num  281 439 434 383 367 ...
##  $ CPI    : num  0.633 0.5 0.9 0.833 1.067 ...
##  $ GDP    : num  58.6 32.4 33.6 47.9 54.1 ...
##  $ POP    : num  462 562 579 510 529 617 628 518 562 652 ...
##  $ quarter: Factor w/ 4 levels "1","2","3","4": 1 2 3 4 1 2 3 4 1 2 ...
##  $ Q1     : Factor w/ 2 levels "0","1": 2 1 1 1 2 1 1 1 2 1 ...
##  $ Q2     : Factor w/ 2 levels "0","1": 1 2 1 1 1 2 1 1 1 2 ...
##  $ Q3     : Factor w/ 2 levels "0","1": 1 1 2 1 1 1 2 1 1 1 ...
##  $ Q4     : Factor w/ 2 levels "0","1": 1 1 1 2 1 1 1 2 1 1 ...
```

We removed 14 observations.

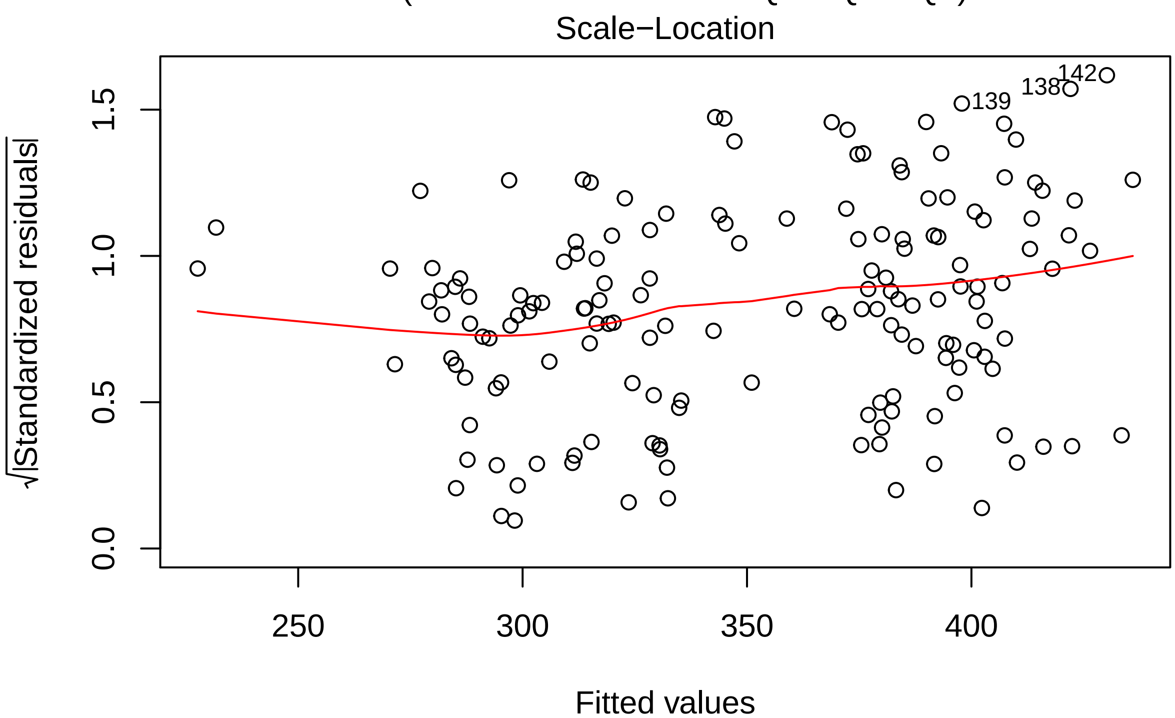
We now look at the model plots.

```r
plot(fit)
```

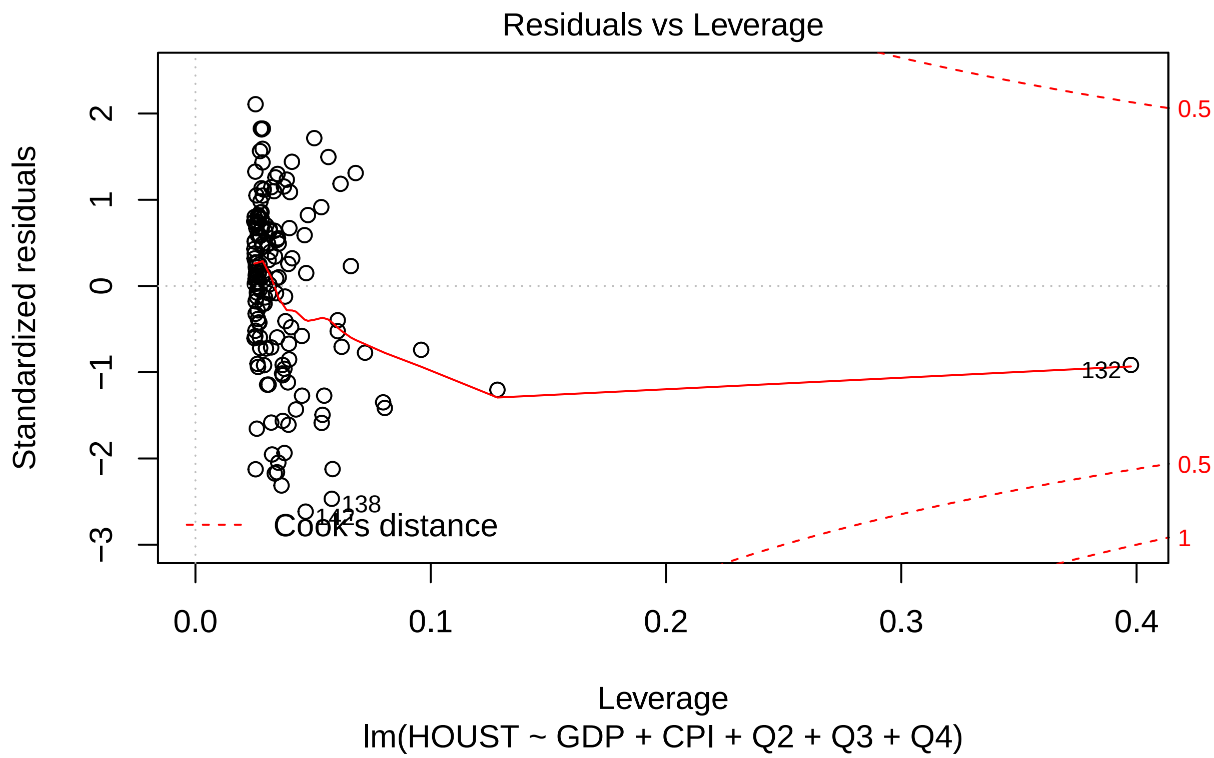




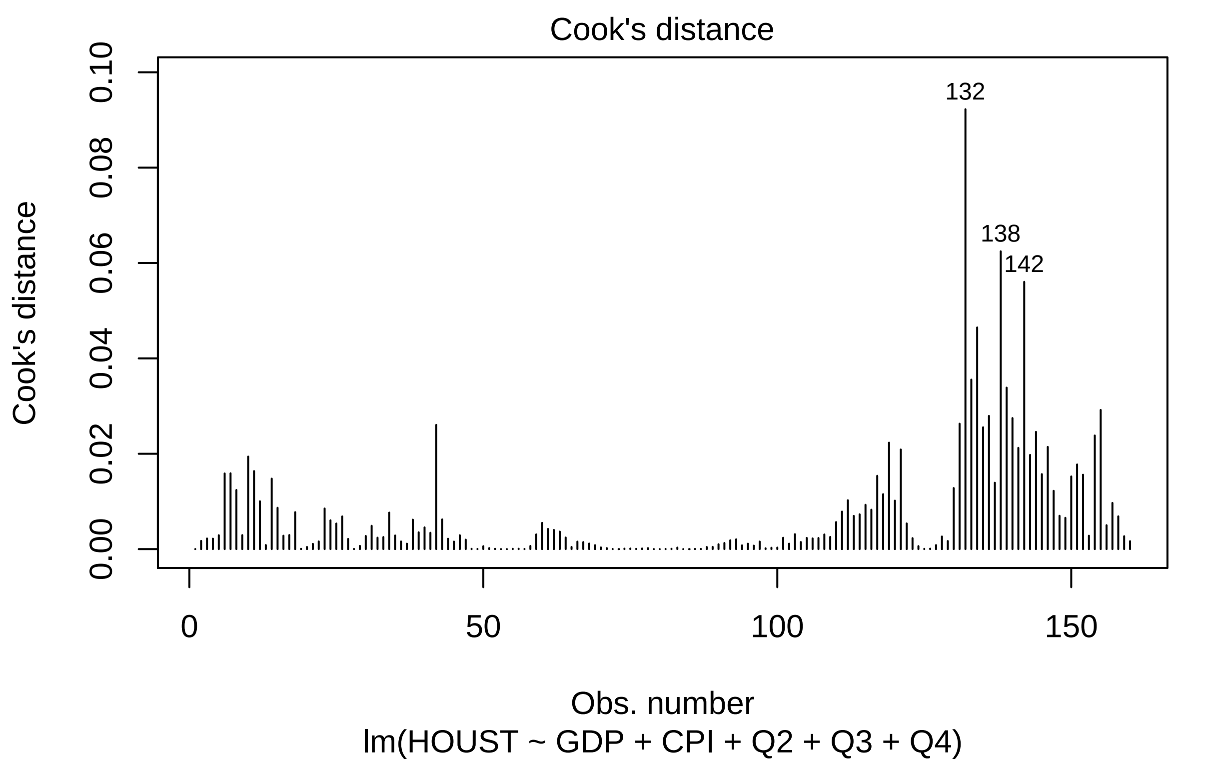
Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(HOUST ~ GDP + CPI + Q2 + Q3 + Q4)



Scale−Location
√|Standardized residuals|
Fitted values
lm(HOUST ~ GDP + CPI + Q2 + Q3 + Q4)




```r
plot(fit,which=4)
```

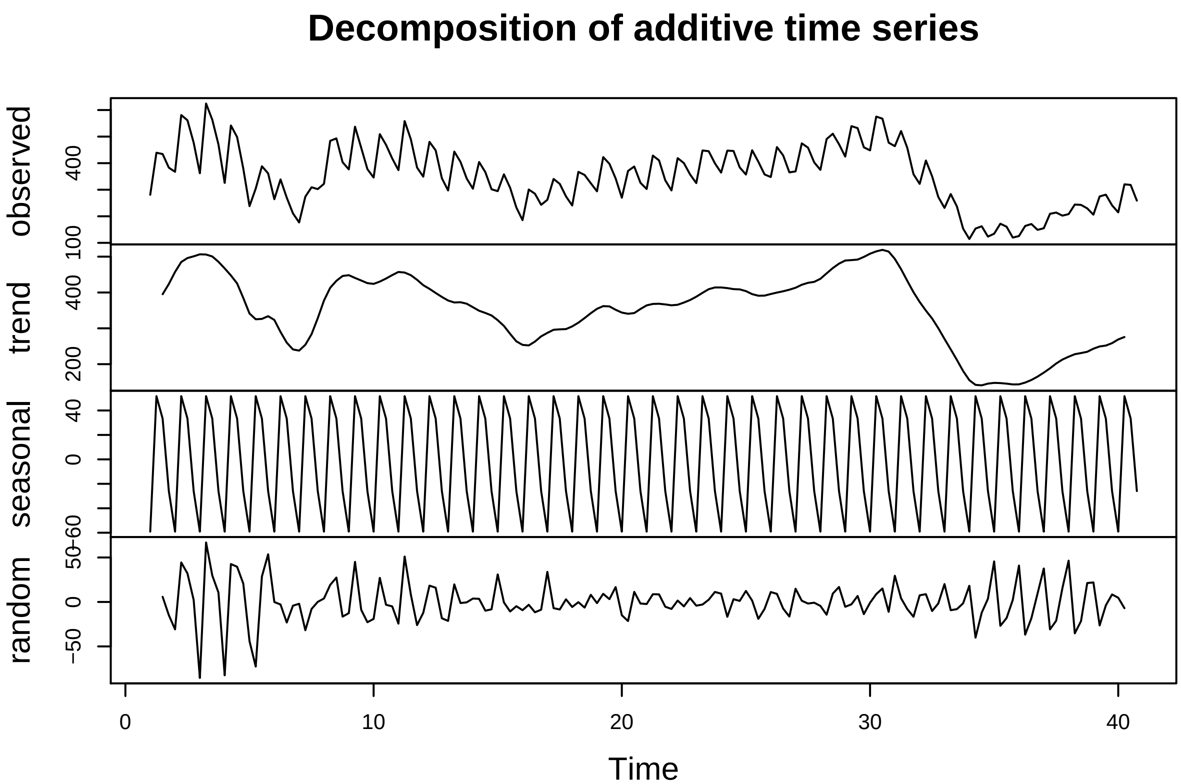


Although we see some influential observations like 132, 138 and 142, we do not see any outlier significant enough to be removed from the data.

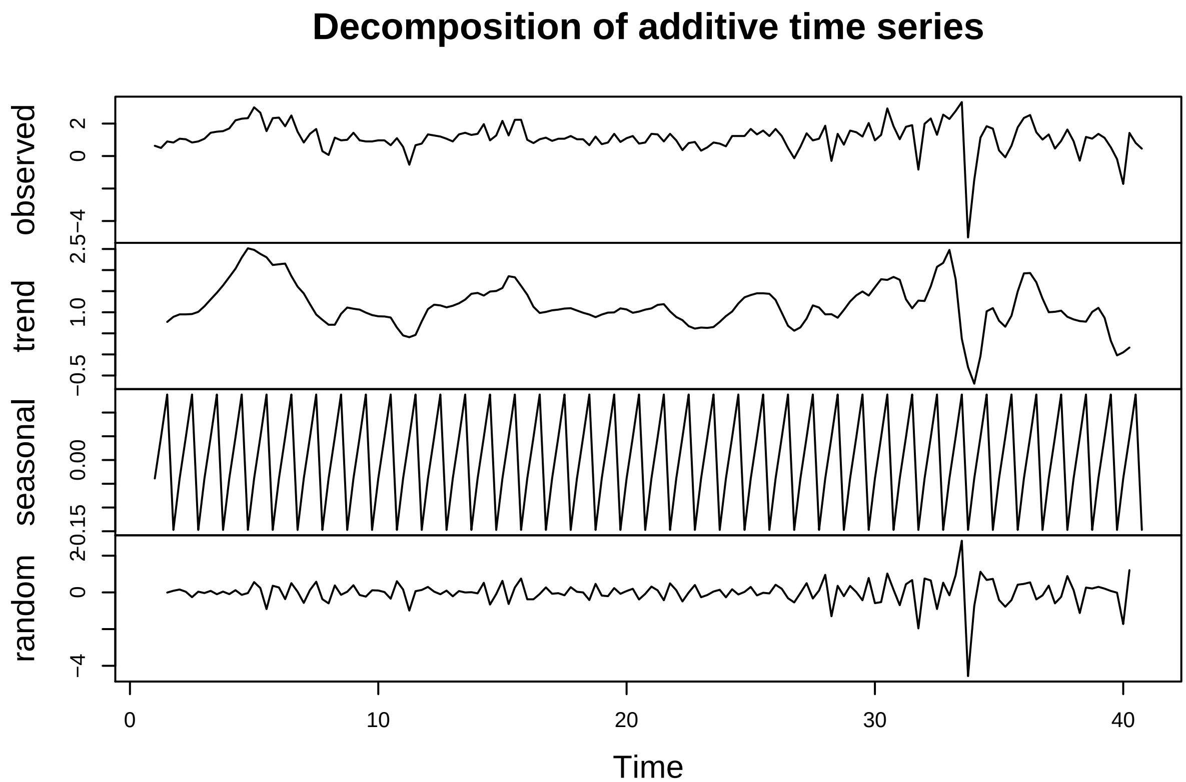
3c) Is there a seasonal effect you observe in the data ? Show necessary steps and explanation.

We convert our variables to a time series in R first and then use decompose() function to decompose the time series into random, trend, seasonal and observed components. We can do this only for the Houst variable but we do it for all explanatory variables as well.

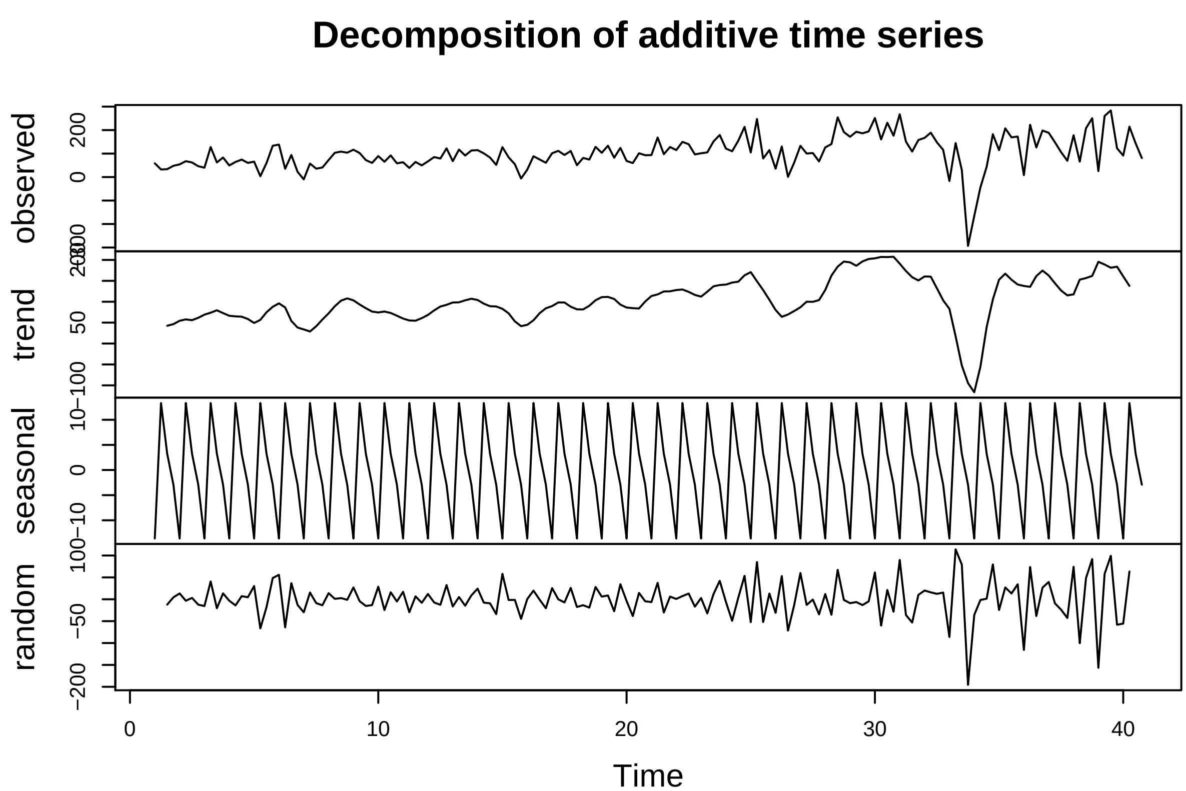
```
houst_ts <- ts(houst_data$HOUST,frequency = 4)
houst_components <- decompose(houst_ts)
plot(houst_components)
```



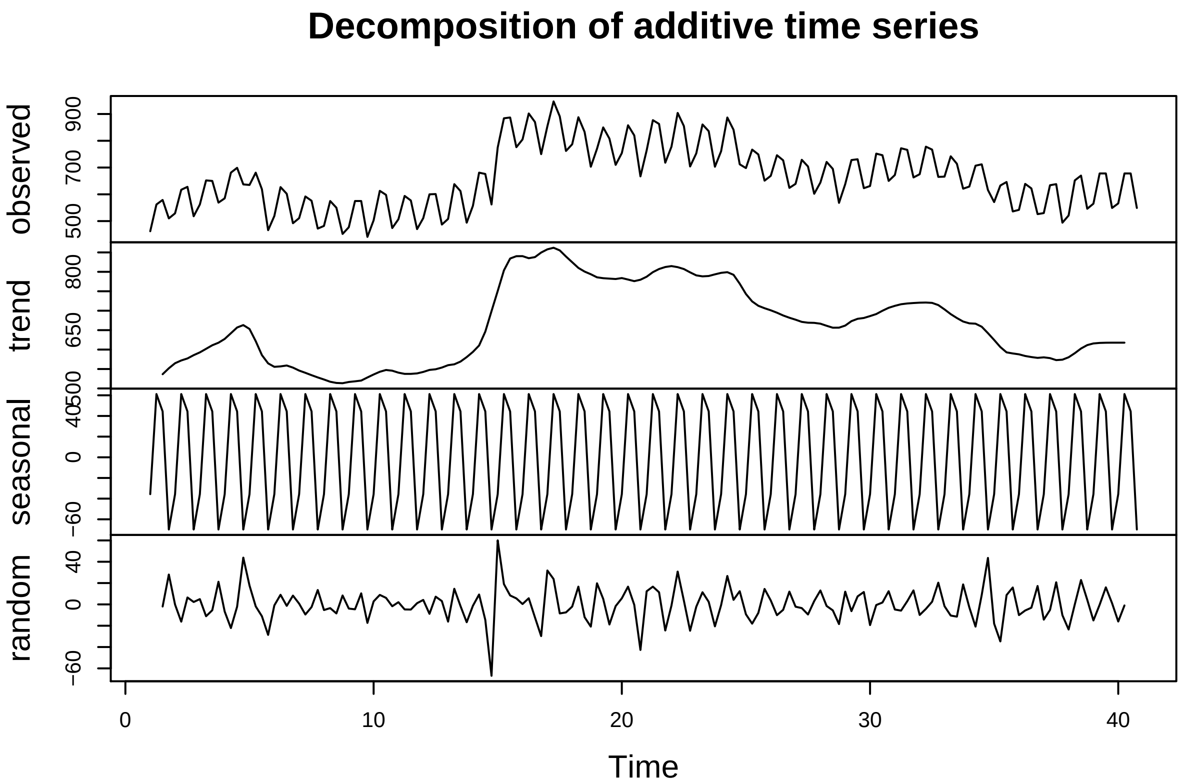

```
cpi_ts <- ts(houst_data$CPI,frequency = 4)
cpi_components <- decompose(cpi_ts)
plot(cpi_components)
```




```r
gdp_ts <- ts(houst_data$GDP,frequency = 4)
gdp_components <- decompose(gdp_ts)
plot(gdp_components)
```

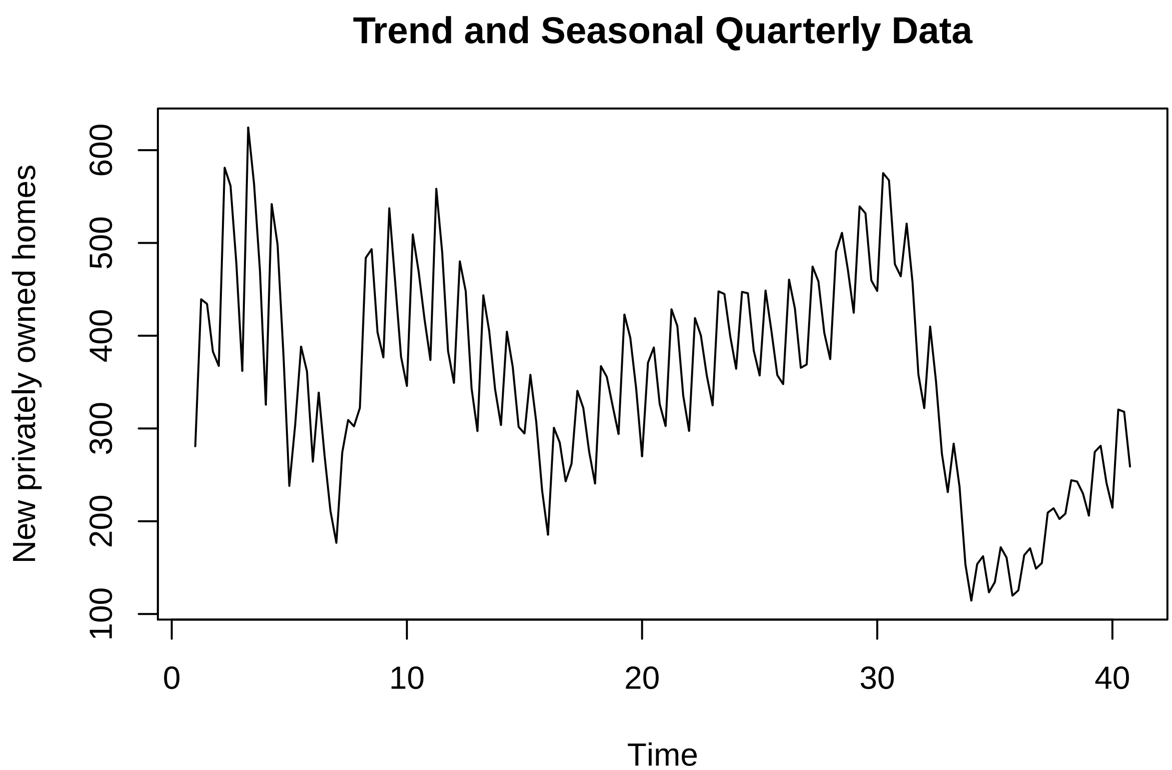


```r
pop_ts <- ts(houst_data$POP,frequency = 4)
pop_components <- decompose(pop_ts)
plot(pop_components)
```

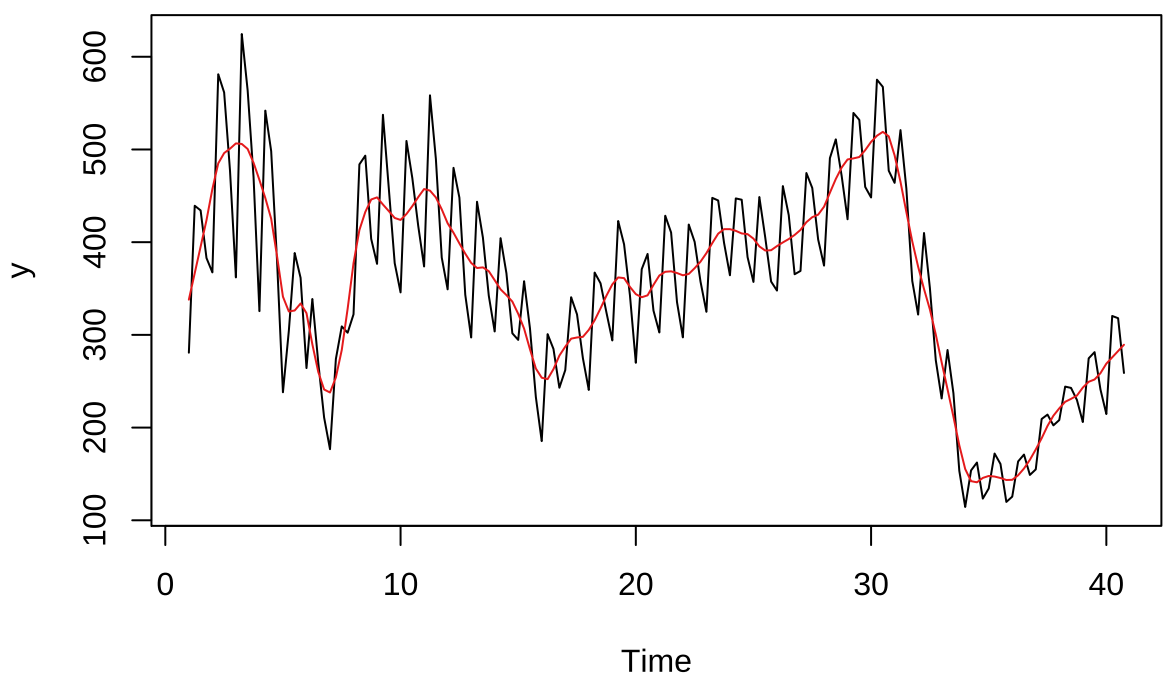


We analyze the houst graph above and see major spikes at period 2 and minor spike in period 3 with a dip followed in period 4. Clearly, this indicates spike at Q2 and Q3 for new privately owned homes.

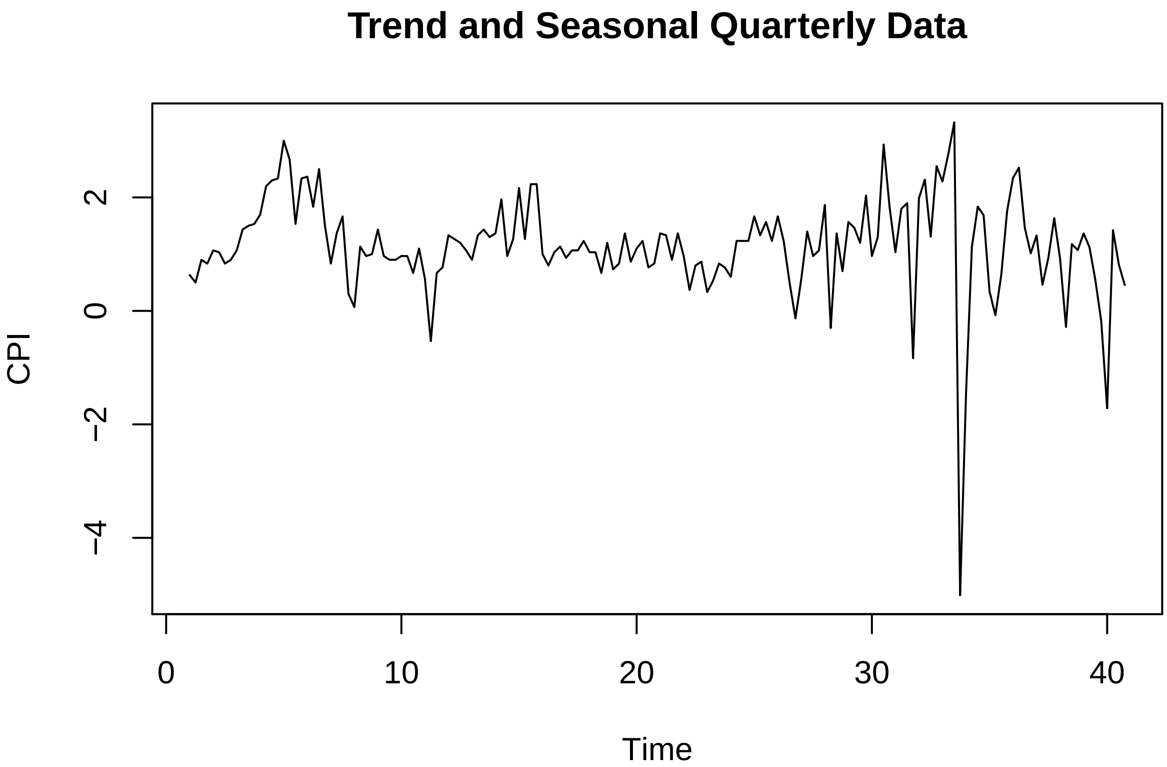
```r
plot(houst_ts, main="Trend and Seasonal Quarterly Data", ylab="New privately owned homes")
```



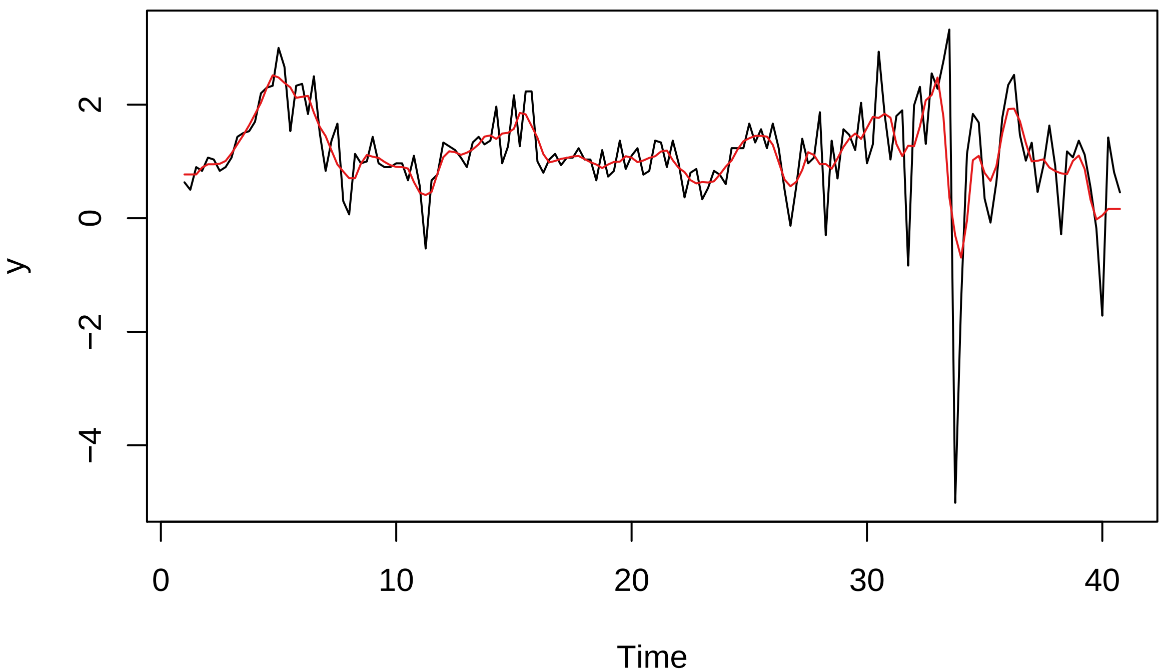

```r
cma_houst <- cmav(houst_ts, outplot=1)
```



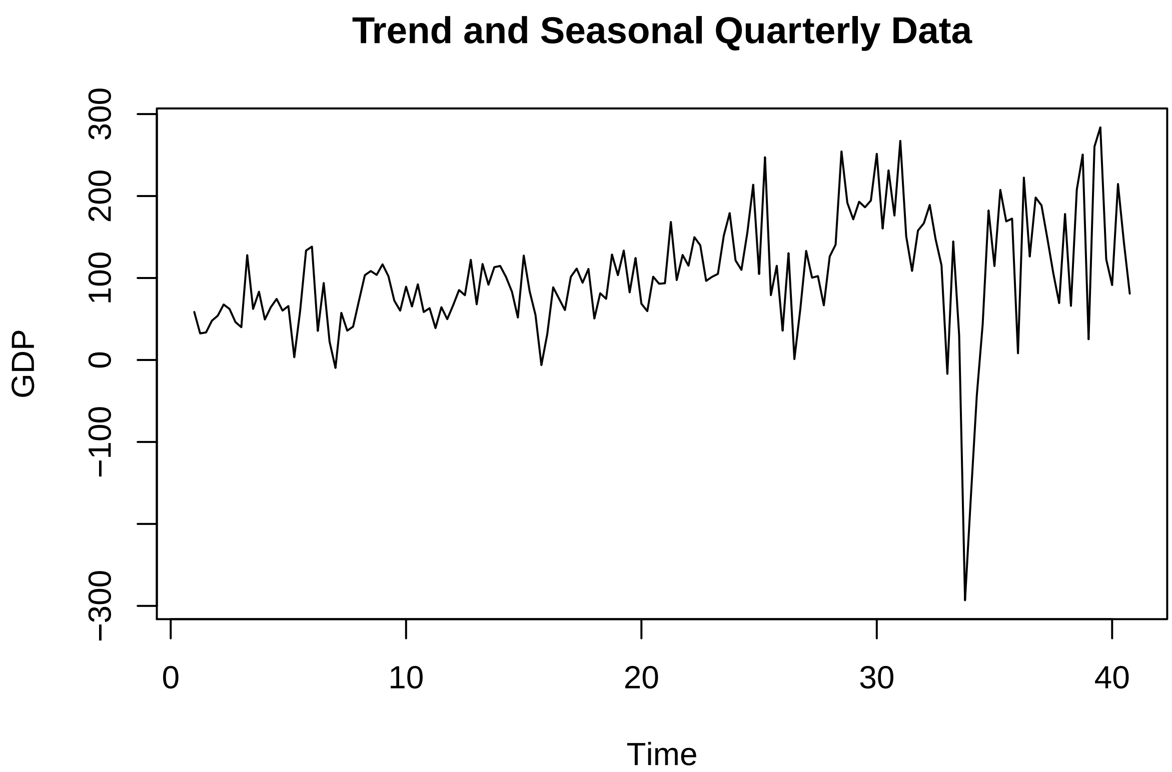

```r
plot(cpi_ts, main="Trend and Seasonal Quarterly Data", ylab="CPI")
```



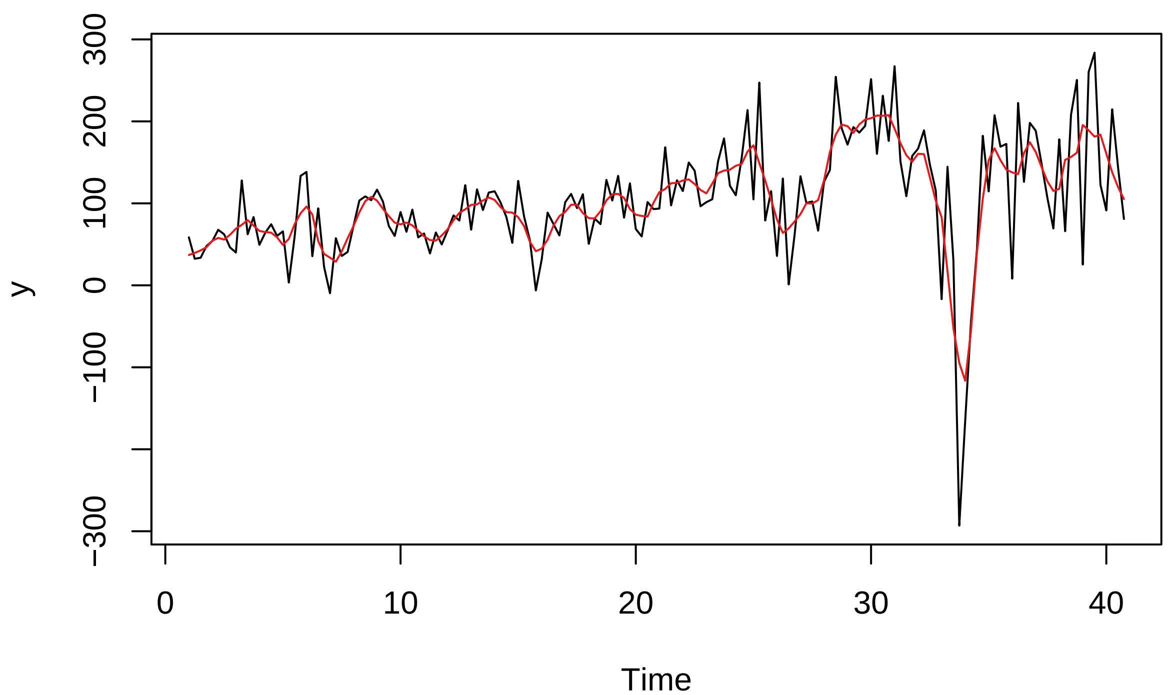

```r
cma_cpi <- cmav(cpi_ts, outplot=1)
```



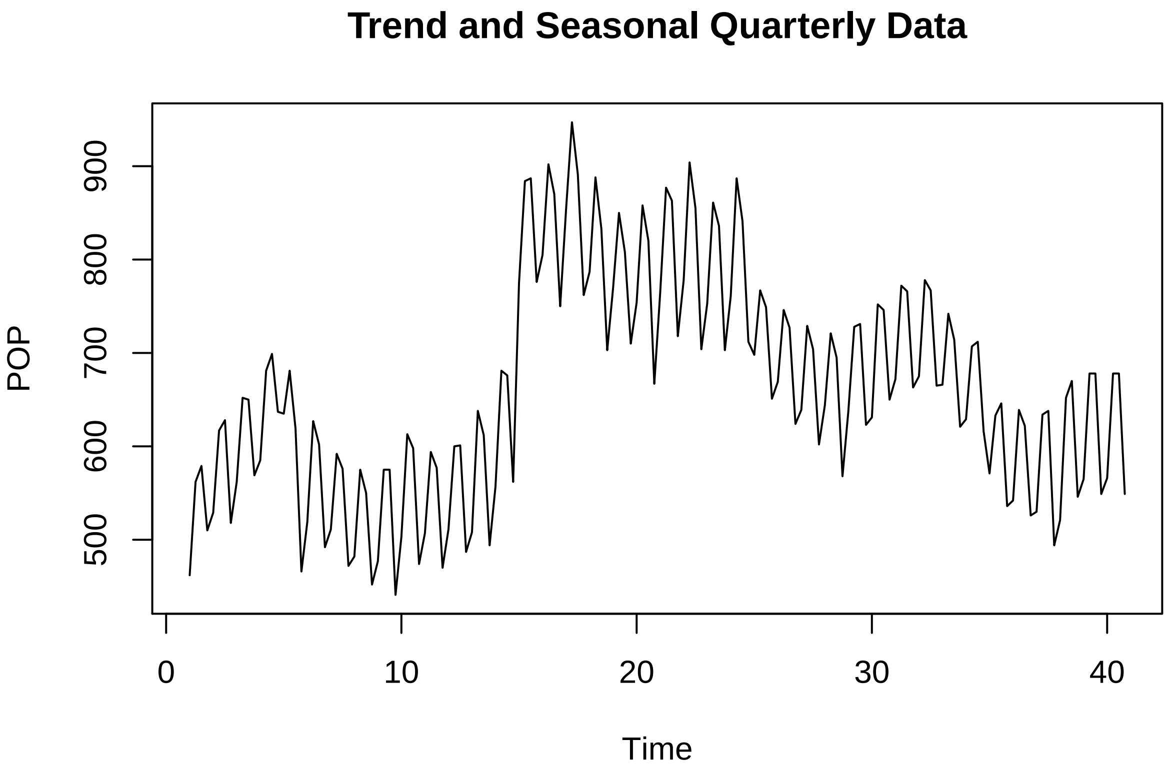

```r
plot(gdp_ts, main="Trend and Seasonal Quarterly Data", ylab="GDP")
```



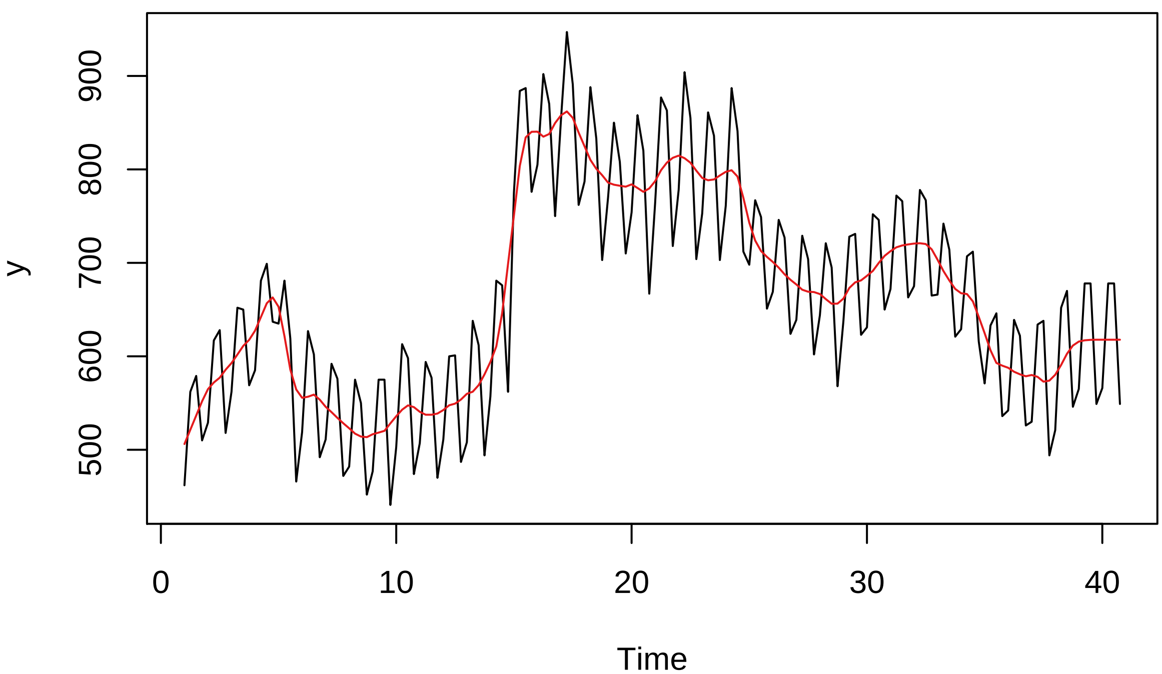

```
cma_gdp <- cmav(gdp_ts, outplot=1)
```




```
plot(pop_ts, main="Trend and Seasonal Quarterly Data", ylab="POP")
```




```
cma_pop <- cmav(pop_ts, outplot=1)
```

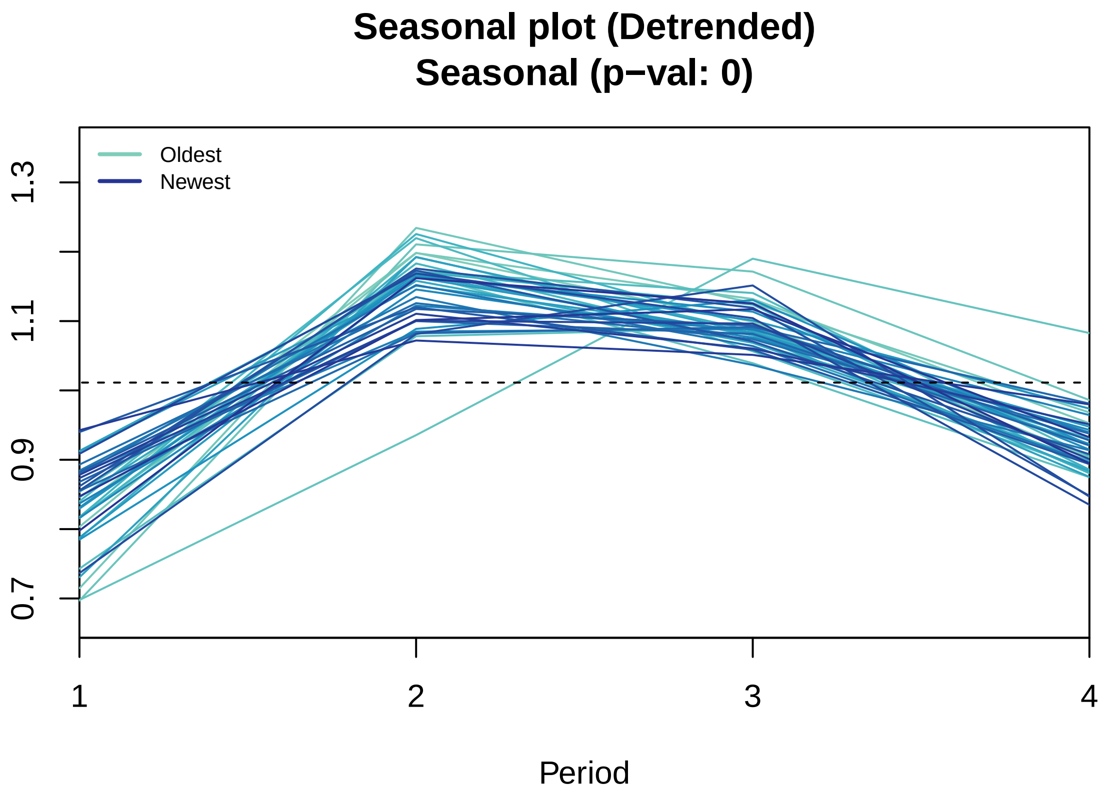


###

We can test for seasonality visually from the plot as well

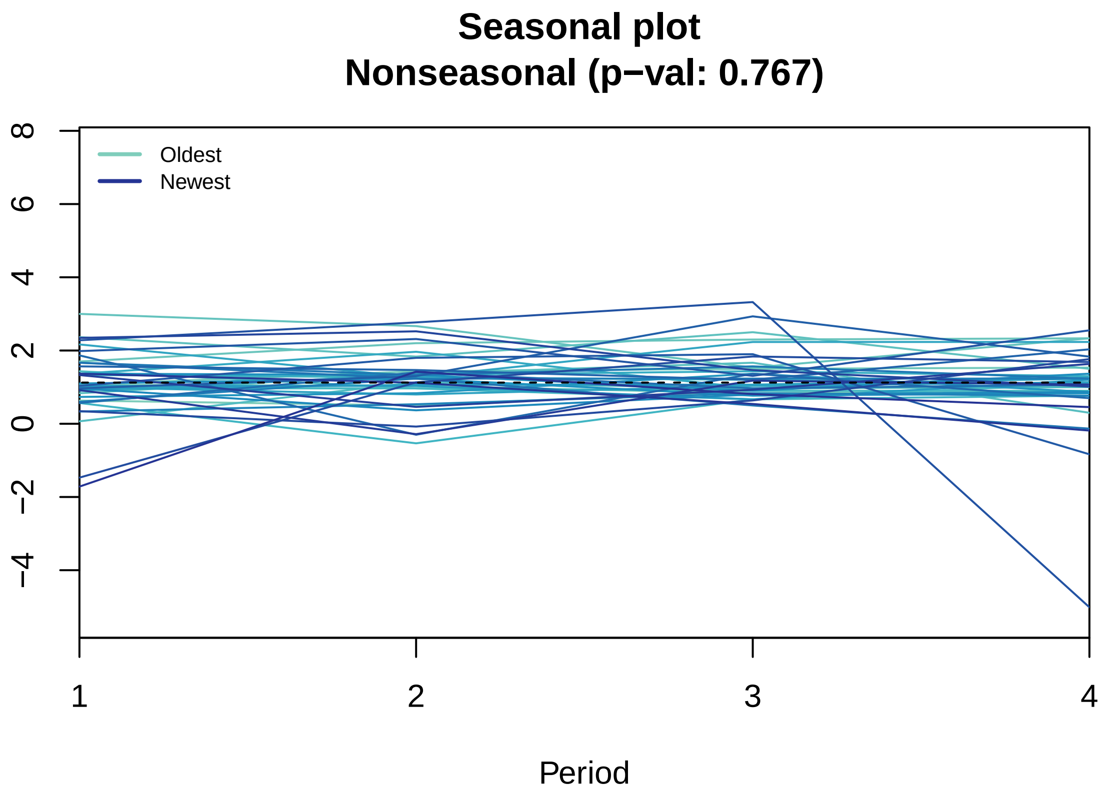
We now use seasplot(). The function shows both trend and seasonality test in the data.

```
seasplot(houst_ts)
```



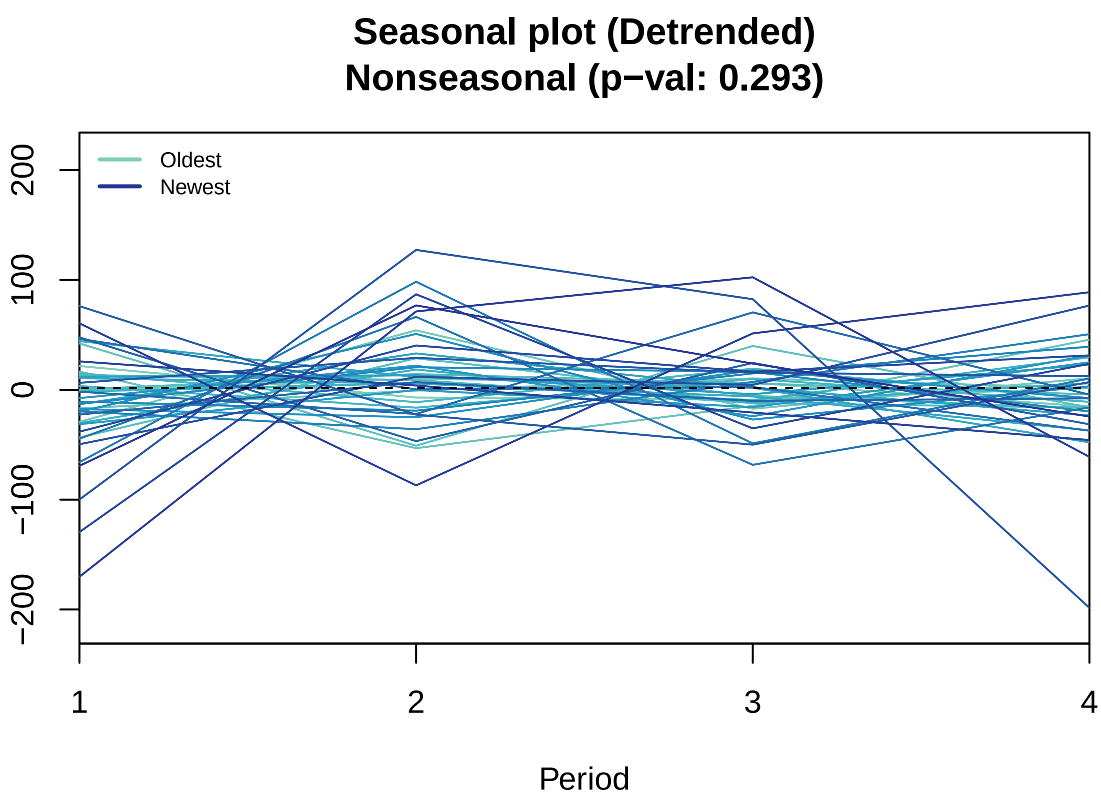

```
## Results of statistical testing
## Evidence of trend: TRUE  (pval: 0.009)
## Evidence of seasonality: TRUE  (pval: 0)
```

```
seasplot(cpi_ts)
```



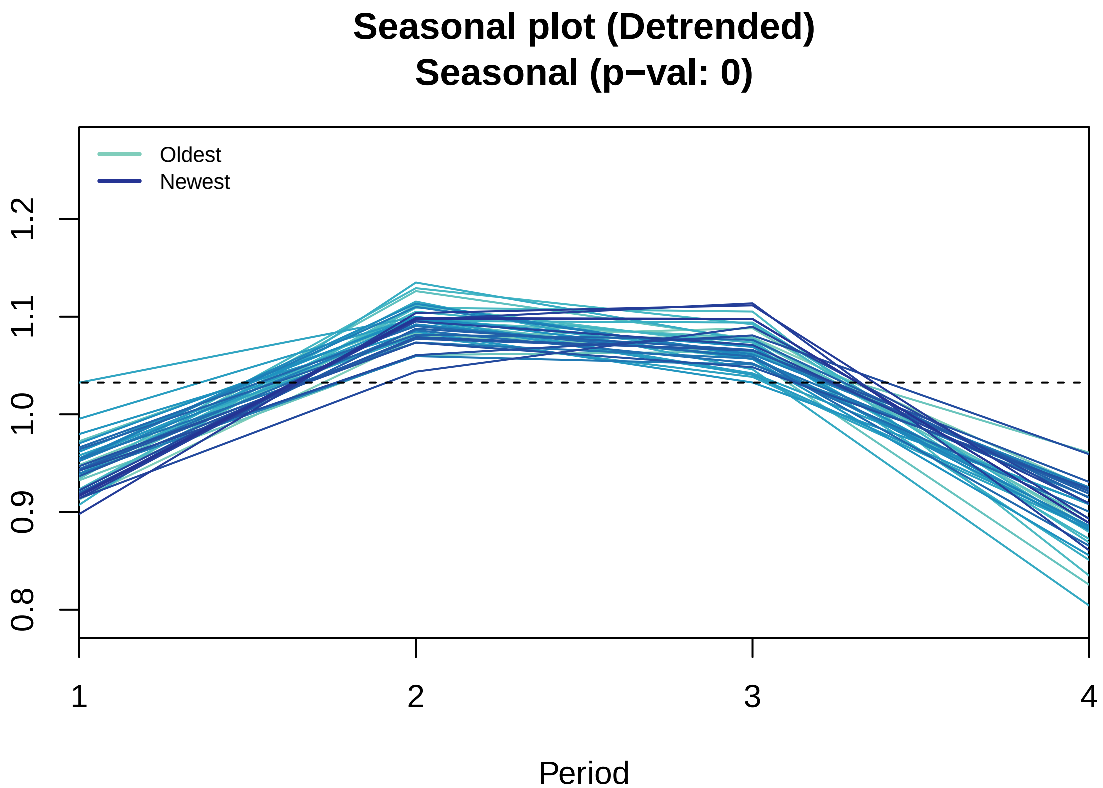

```
## Results of statistical testing
## Evidence of trend: FALSE  (pval: 0.046)
## Evidence of seasonality: FALSE  (pval: 0.767)
```

```r
seasplot(gdp_ts)
```




```
## Results of statistical testing
## Evidence of trend: TRUE  (pval: 0)
## Evidence of seasonality: FALSE  (pval: 0.293)
```

```r
seasplot(pop_ts)
```



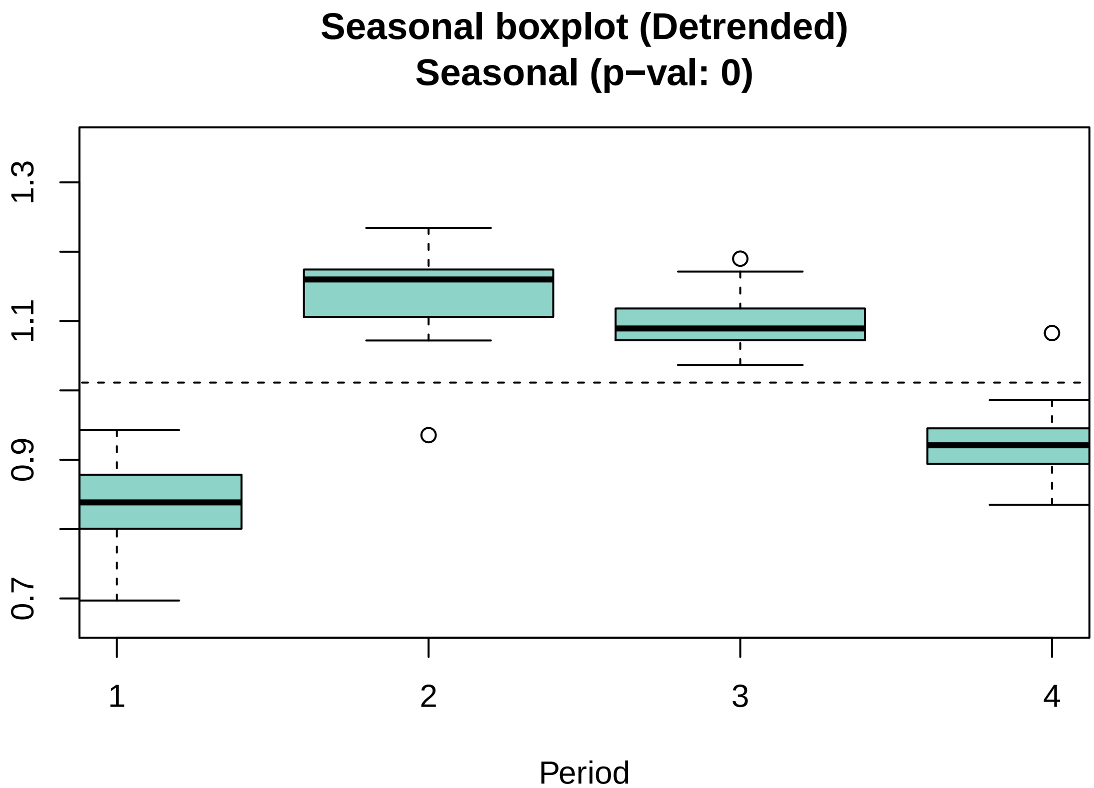

```
## Results of statistical testing
## Evidence of trend: TRUE  (pval: 0.001)
## Evidence of seasonality: TRUE  (pval: 0)
```

Key observations -

1. We find evidence for trend and seasonality in new privately owned homes (HOUST).
2. We find no evidence for trend or seasonality in CPI.
3. We find evidence for trend however none for seasonality in GDP
4. We find evidence for trend and seasonality in POP.

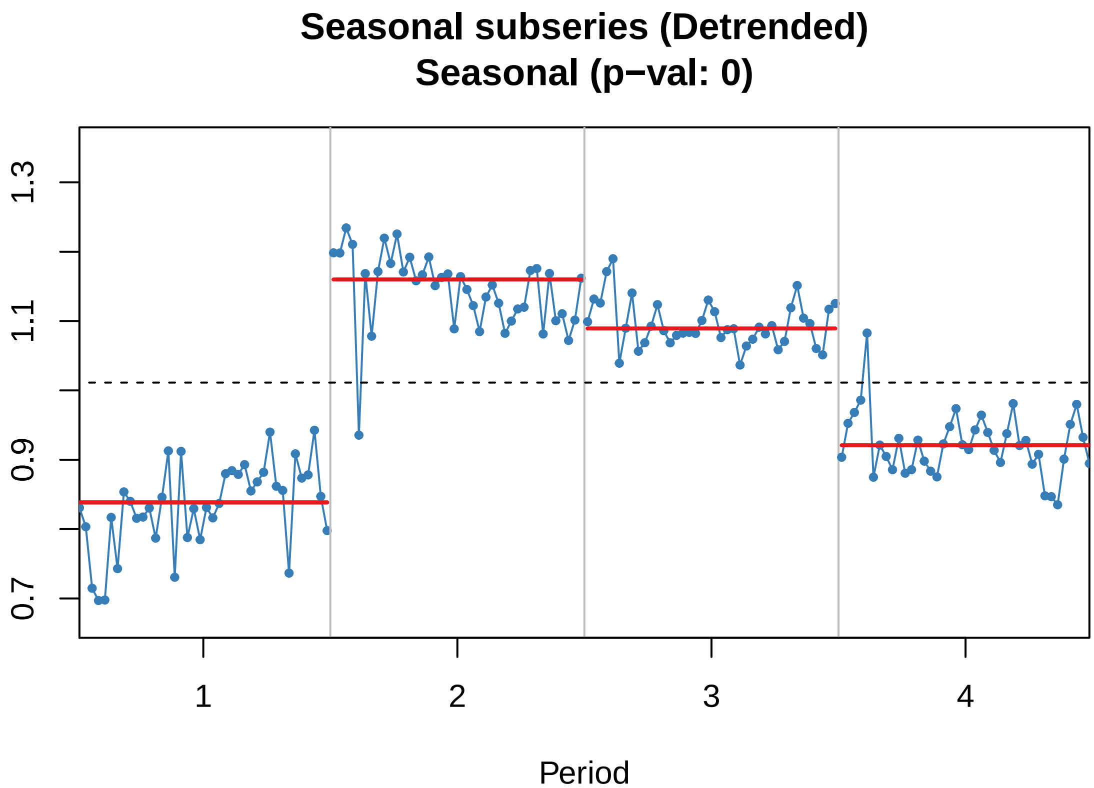
We also check for other seasonality plots for one of the above (Houst).

```
seasplot(houst_ts, outplot =2)
```



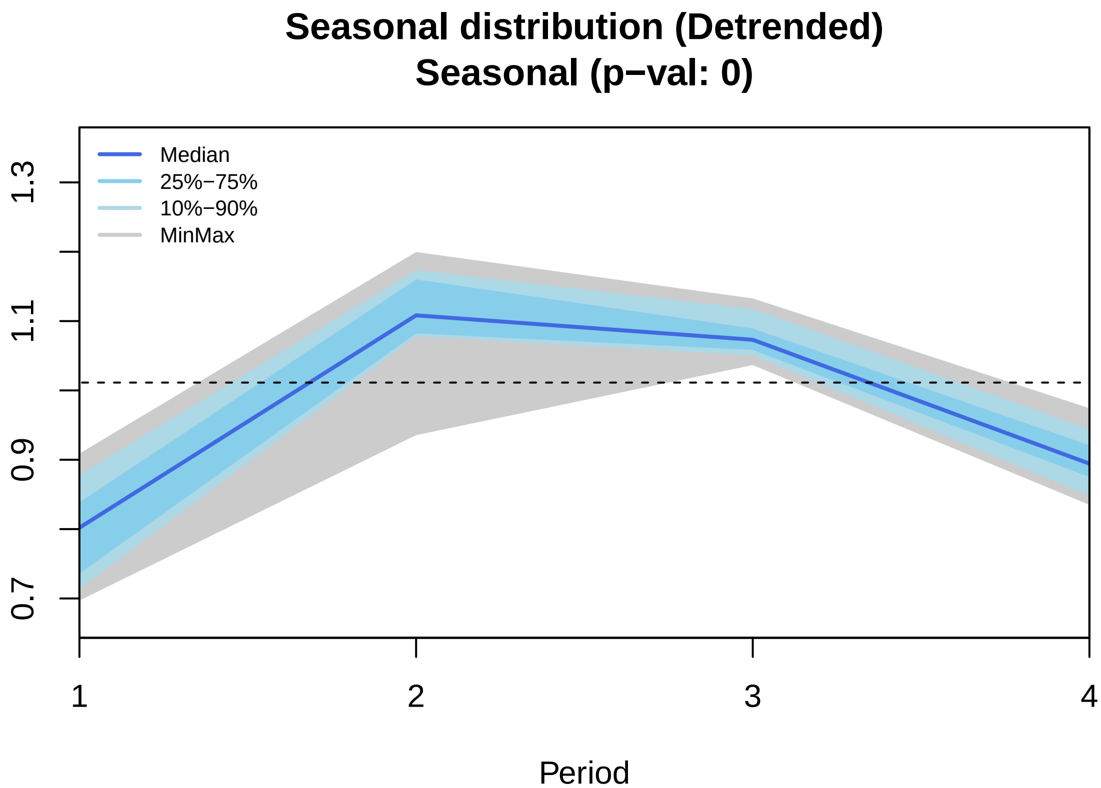

```
## Results of statistical testing
## Evidence of trend: TRUE  (pval: 0.009)
## Evidence of seasonality: TRUE  (pval: 0)
```

```r
seasplot(houst_ts, outplot =3)
```




```
## Results of statistical testing
## Evidence of trend: TRUE  (pval: 0.009)
## Evidence of seasonality: TRUE  (pval: 0)
```

```r
seasplot(houst_ts, outplot =4)
```

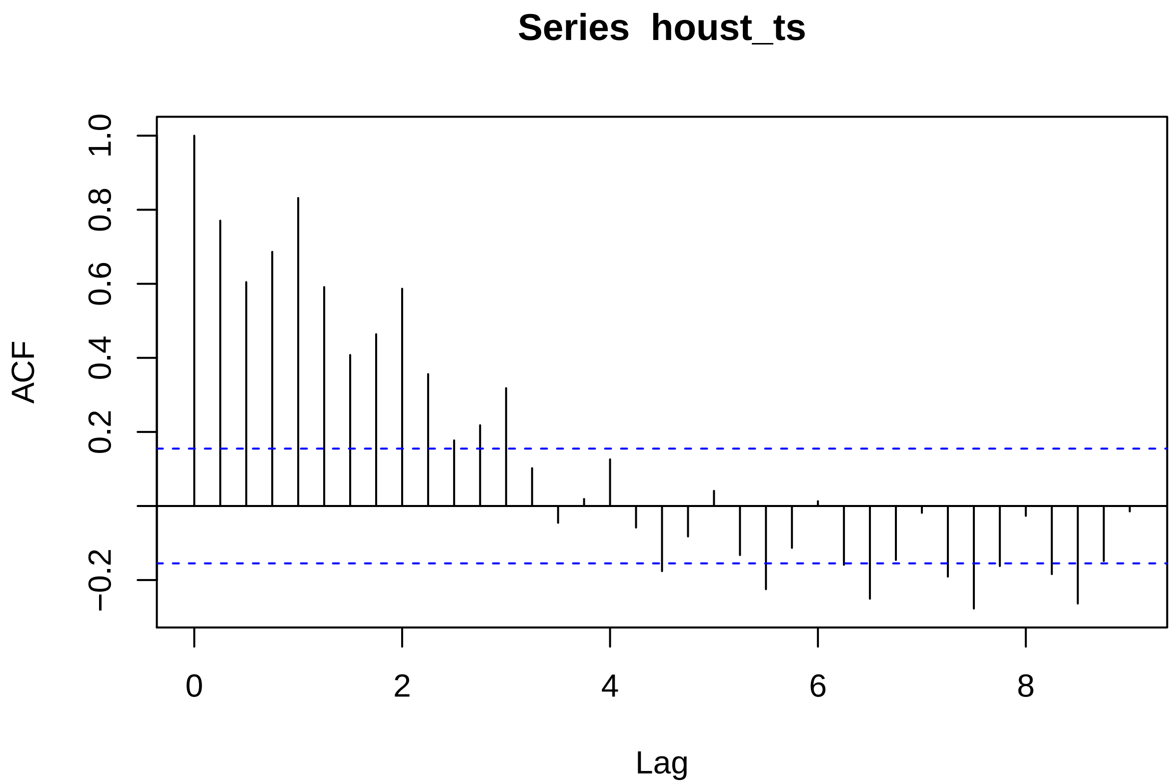



```
## Results of statistical testing
## Evidence of trend: TRUE  (pval: 0.009)
## Evidence of seasonality: TRUE  (pval: 0)
```

We plot the seasonal box plot, sub series, and distribution above. Notice how we get outlier point in Q2, Q3 and Q4 since we haven't used the clean data for this plot. We also see the -ve skew in Q2 due to a sharp dip as seen in the subseries.

Another way to test for seasonality is the ACF plot.

```
acf(houst_ts, lag.max = 36)
```



Generally, a time series with a seasonality component tend to start at a large value and decrease over time. We see how the above series for Houst follows this.

Finally, we now perform statistical tests to detect seasonality. Both Student t-test and Wilcoxon Signed Rank test are considered good tests for detection. We will however use something very recent - WO-test, i.e. the overall seasonality test developed in Webel and Ollech (2019).

```
summary(wo(houst_ts))
```

```
## Test used:  WO
##
## Test statistic:  1
## P-value:  0 0 0
##
## The WO - test identifies seasonality
```

```
summary(wo(gdp_ts))
```

```
## Test used:  WO
##
## Test statistic:  0
## P-value:  1 1 0.2255651
##
## The WO - test does not identify  seasonality
```

```
summary(wo(cpi_ts))
```

```
## Test used:  WO
##
## Test statistic:  0
## P-value:  1 1 0.8590326
##
```

```
## The WO - test does not identify  seasonality
```

```r
summary(wo(pop_ts))
```

```
## Test used:  WO
##
## Test statistic:  1
## P-value:  0 0 0
##
## The WO - test identifies seasonality
```

This test confirms our earlier results that there is clear seasonality in the data of New privately owned homes and population but not for CPI and GDP.

3d) Do a pair-wise comparison for different quarters. Which quarter do you think is the best to buy a house ? Show necessary steps and explanation.

We check anova to see if there is a significant difference between number of construction starts between various quarters of the year.

```r
anova_mod <- aov(formula = HOUST ~ quarter,
                   data = houst_data)
summary(anova_mod)
```

```
##              Df  Sum Sq Mean Sq F value   Pr(>F)    
## quarter       3  320981  106994   9.696 6.63e-06 ***
## Residuals   156 1721498   11035                     
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Since p-value is < 0.5 we conclude that there is a significant difference between number of construction starts between any two quarters of the year.

WE also show that even with other variables, quarter is signifcant using ANOVA.

```r
anova_res <- aov(HOUST~GDP+CPI+quarter, data=houst_data)
summary(anova_res)
```

```
##              Df  Sum Sq Mean Sq F value   Pr(>F)    
## GDP           1   76745   76745   7.041   0.0088 ** 
## CPI           1    2025    2025   0.186   0.6671    
## quarter       3  285122   95041   8.719 2.23e-05 ***
## Residuals   154 1678588   10900                     
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now we compute pair-wise comparisons for different quarters - we can do this using TukeyHSD or pair-wise.t.test.

```r
TukeyHSD(anova_mod, conf.level = 0.95)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = HOUST ~ quarter, data = houst_data)
##
## $quarter
##          diff       lwr        upr     p adj
```

```
## 2-1 111.2400   50.2389 172.241102 0.0000287
## 3-1  92.3625   31.3614 153.363602 0.0007236
## 4-1  32.5150  -28.4861  93.516102 0.5111122
## 3-2 -18.8775  -79.8786  42.123602 0.8526299
## 4-2 -78.7250 -139.7261 -17.723898 0.0055150
## 4-3 -59.8475 -120.8486   1.153602 0.0566456
```

```r
pairwise.t.test(houst_data$HOUST, houst_data$quarter, p.adj = "none")
```

```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  houst_data$HOUST and houst_data$quarter
##
##   1       2       3
## 2 4.9e-06 -       -
## 3 0.00013 0.42282 -
## 4 0.16827 0.00101 0.01181
##
## P value adjustment method: none
```

We see interestingly that Q1-Q2, Q1-Q3, Q2-Q4 & Q3-Q4 are significant in pair-wise comparisons. In fact, the only non-significant pairs are Q1-Q4 and Q2-Q3.

```r
fit <- lm(HOUST~GDP+CPI+quarter, data=houst_data)
summary(fit)
```

```
##
## Call:
## lm(formula = HOUST ~ GDP + CPI + quarter, data = houst_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -266.68  -69.19   12.62   71.28  217.21
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 271.0686    20.9148  12.961  < 2e-16 ***
## GDP           0.2208     0.1207   1.829 0.069328 .
## CPI           1.8468     9.8302   0.188 0.851224
## quarter2    105.3363    23.5381   4.475 1.48e-05 ***
## quarter3     88.2852    23.4548   3.764 0.000237 ***
## quarter4     30.4973    23.4202   1.302 0.194801
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 104.4 on 154 degrees of freedom
## Multiple R-squared:  0.1782, Adjusted R-squared:  0.1515
## F-statistic: 6.677 on 5 and 154 DF,  p-value: 1.173e-05
```

Recall our model results now. We saw the coefficient of Q2 being the largest followed by Q3 and then Q4. It seems that more people buy homes in Q2 and Q3 than in Q1 and Q4. It makes sense as well since most people buy properties in Spring or Summer and not the winter months. Given the boxplot spread as well since most construction starts happen in Q2, prices will be lower and we can recommend this for a person not looking for ready to move in investment.

Although it is hard to say what is the best time to buy a month based on this data given we don't have any indication of demand. For instance, people may choose to buy during months they feel the prices will be lower, or demand will be lesser however this data only provides us with an indication of the economy strength and the number of privately owned homes in each quarter. A customer knowing that demand is high in Q2 may choose to buy in Q4 when demand is low and prices are lower compared to Spring-Summer months.

However, we take the former answer for now and recommend that the best time to buy a house given the strength of the economy would be in Q2 (the quarter with the highest coefficient in indicating when the number of homes goes up).

3e) Add population to the first model. Do steps b and c again.

We add population to the model now.

```
fit <- lm(HOUST~GDP+CPI+POP+quarter, data=houst_data)
summary(fit)
```

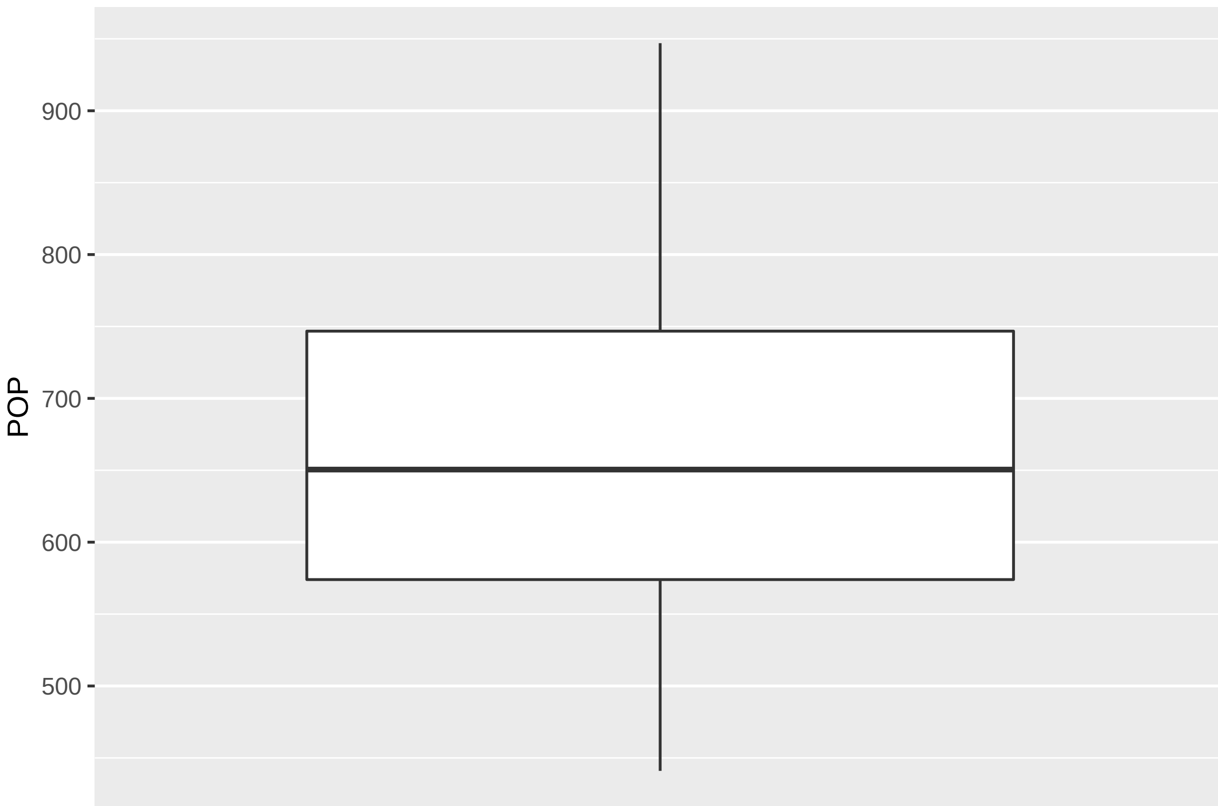
```
##
## Call:
## lm(formula = HOUST ~ GDP + CPI + POP + quarter, data = houst_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -271.25  -64.11   15.78   70.93  213.90
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 299.00784   53.40285   5.599 9.73e-08 ***
## GDP           0.22720    0.12149   1.870  0.06337 .
## CPI           1.88789    9.85210   0.192  0.84829
## POP          -0.04569    0.08032  -0.569  0.57031
## quarter2    109.61624   24.76083   4.427 1.81e-05 ***
## quarter3     91.91961   24.35938   3.773  0.00023 ***
## quarter4     28.98273   23.62236   1.227  0.22174
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 104.6 on 153 degrees of freedom
## Multiple R-squared:  0.1799, Adjusted R-squared:  0.1477
## F-statistic: 5.594 on 6 and 153 DF,  p-value: 2.852e-05
```

We see R-square marginally improves to 17.99% however we do not see population as a significant predictor of HOUST.

Steps for b - data cleaning in pop ?

Recall we did not see any data cleaning requirements in population variable earlier itself. We can however check the distribution by a boxplot for population and see if there are any outliers.

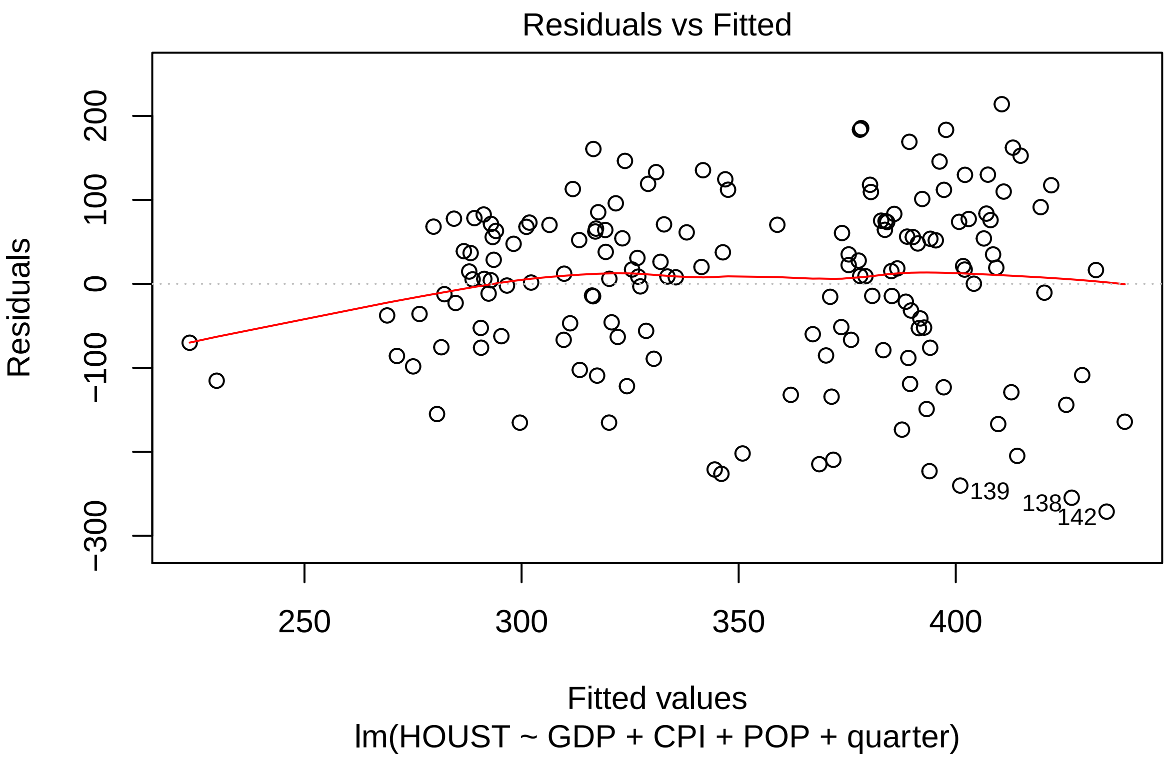
```
ggplot(houst_data, aes(x = factor(0), y = POP)) + geom_boxplot() + xlab("") +
  scale_x_discrete(breaks = NULL)
```

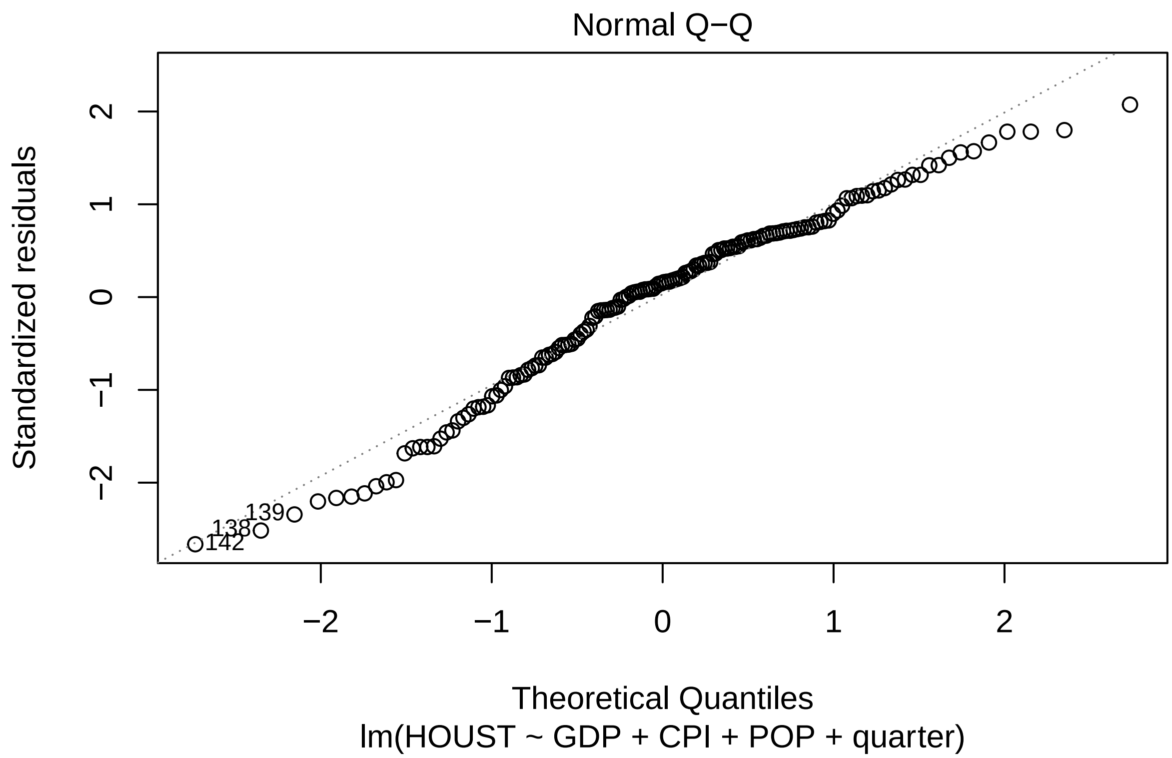


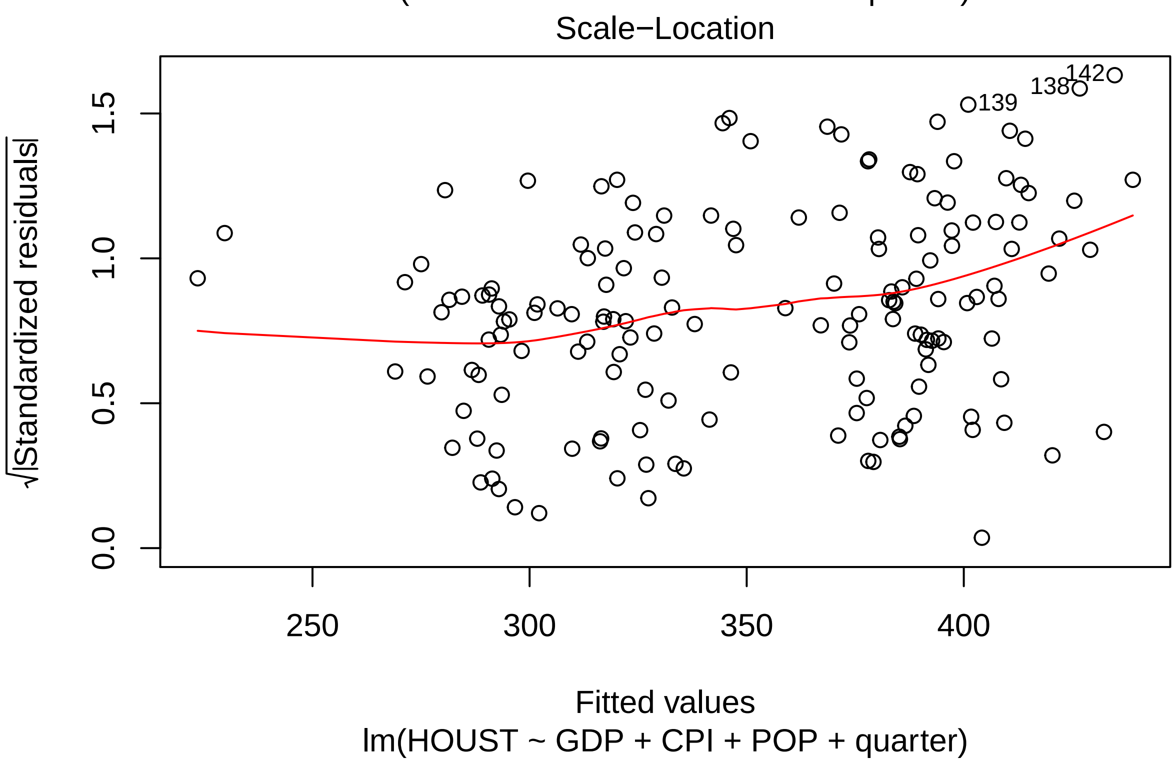
We do not observe any outliers in the variable.

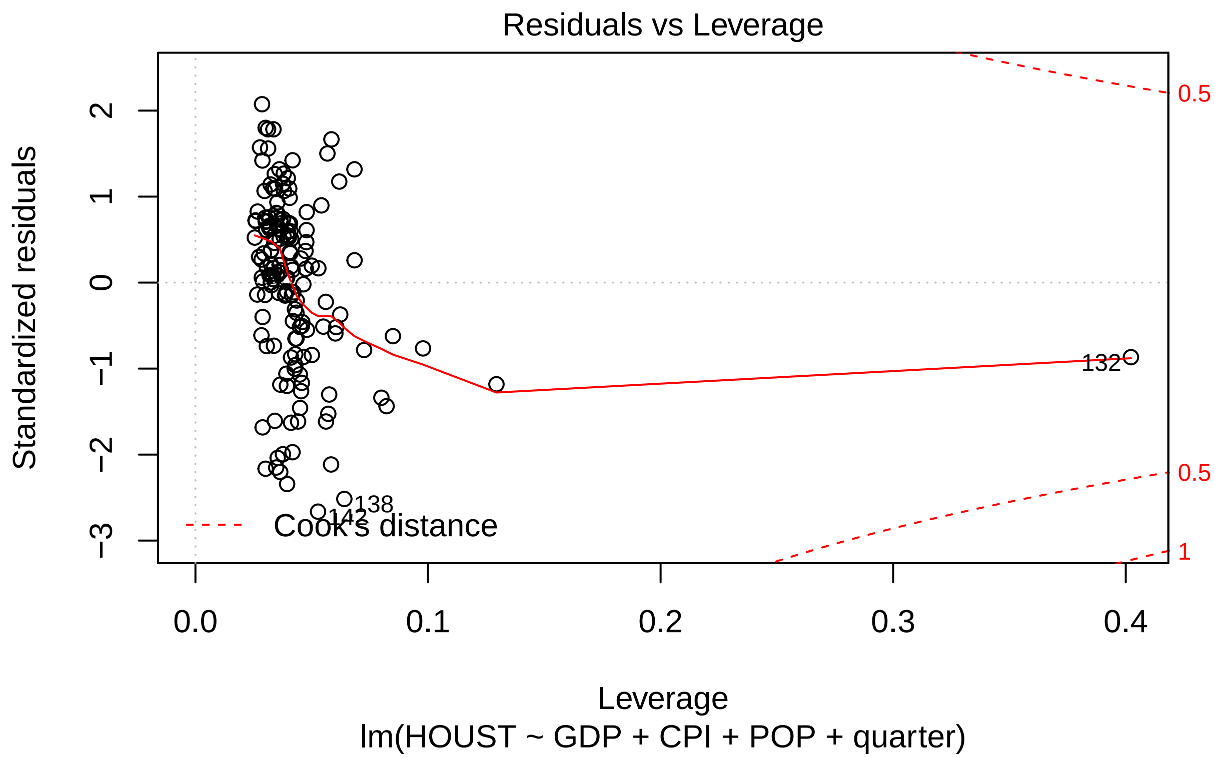
Now let's check the plots of the model.

```r
plot(fit)
```

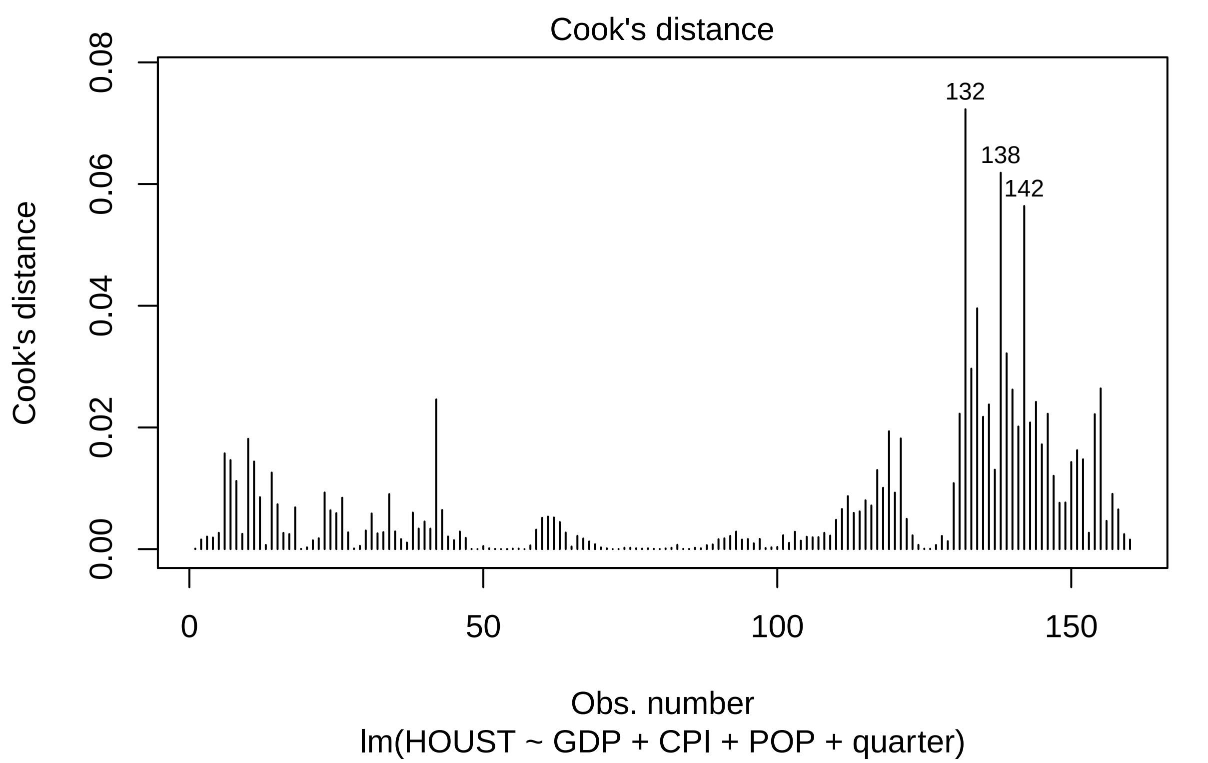



Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(HOUST ~ GDP + CPI + POP + quarter)
139
138
142


Scale−Location
√|Standardized residuals|
Fitted values
lm(HOUST ~ GDP + CPI + POP + quarter)
139
138
142




```r
plot(fit,which=4)
```

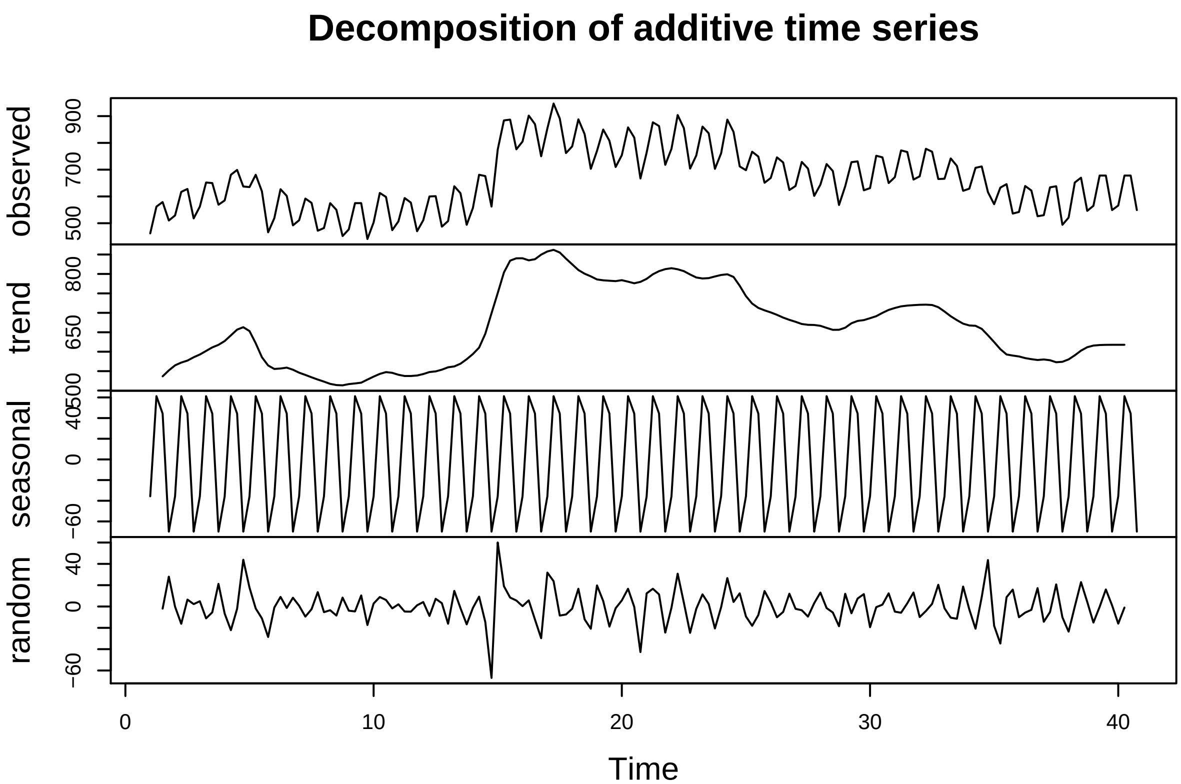


We again see no outliers hence no need for cleaning. We also already noted the variable isn't significant so we do not need any further cleaning.

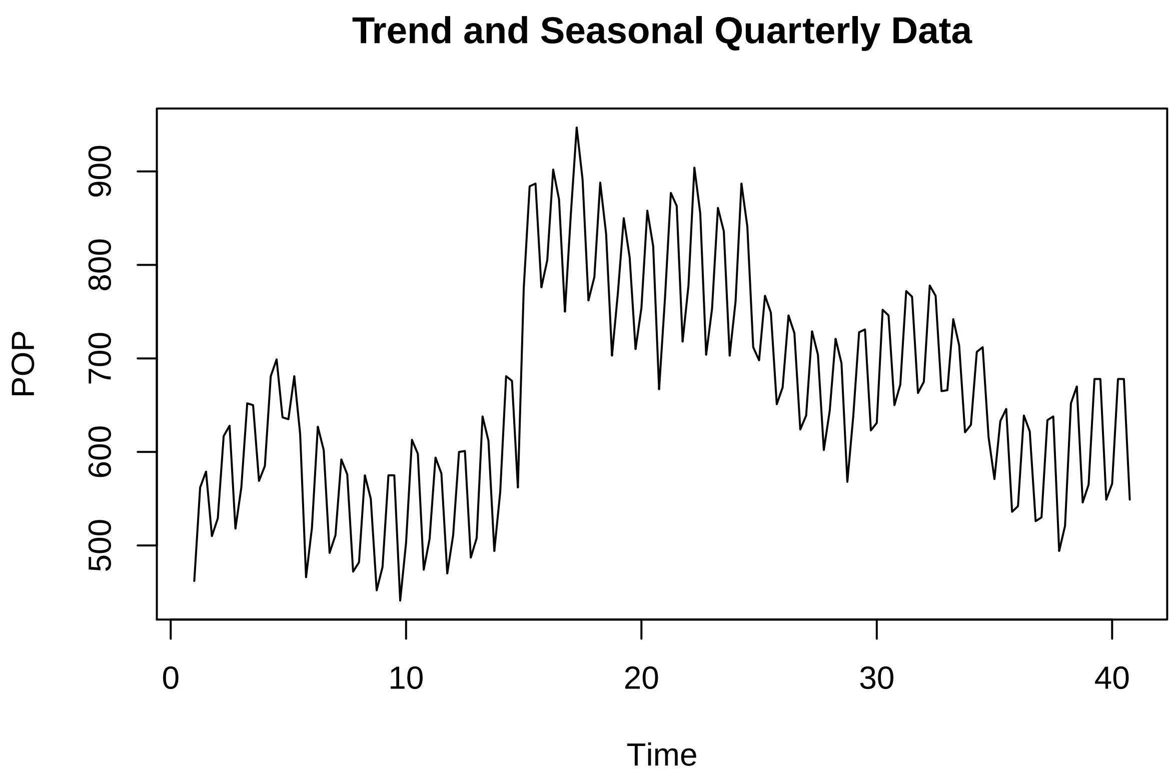
Steps for c - Seasonality in pop.

Recall we have already shown earlier the seasonality trend in population when we were solving 3c). We in fact detected Seasonality for all explanatory variables there. Here's a brief summary of what we did for poopulation.

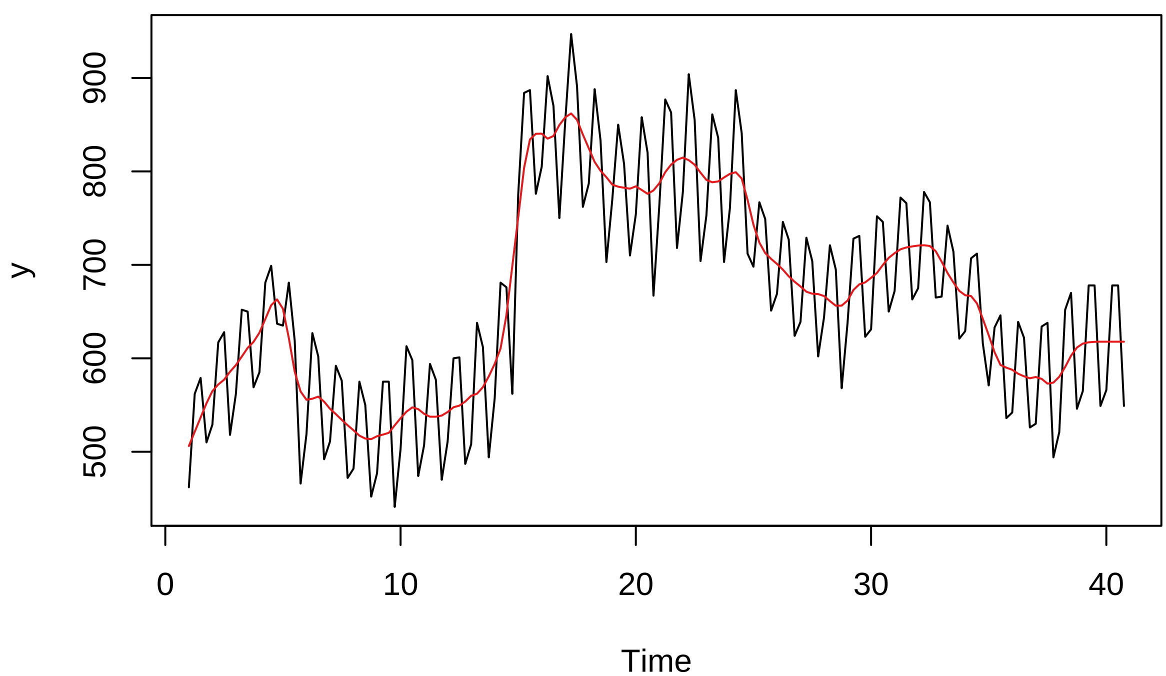
```
pop_ts <- ts(houst_data$POP,frequency = 4)
pop_components <- decompose(pop_ts)
plot(pop_components)
```



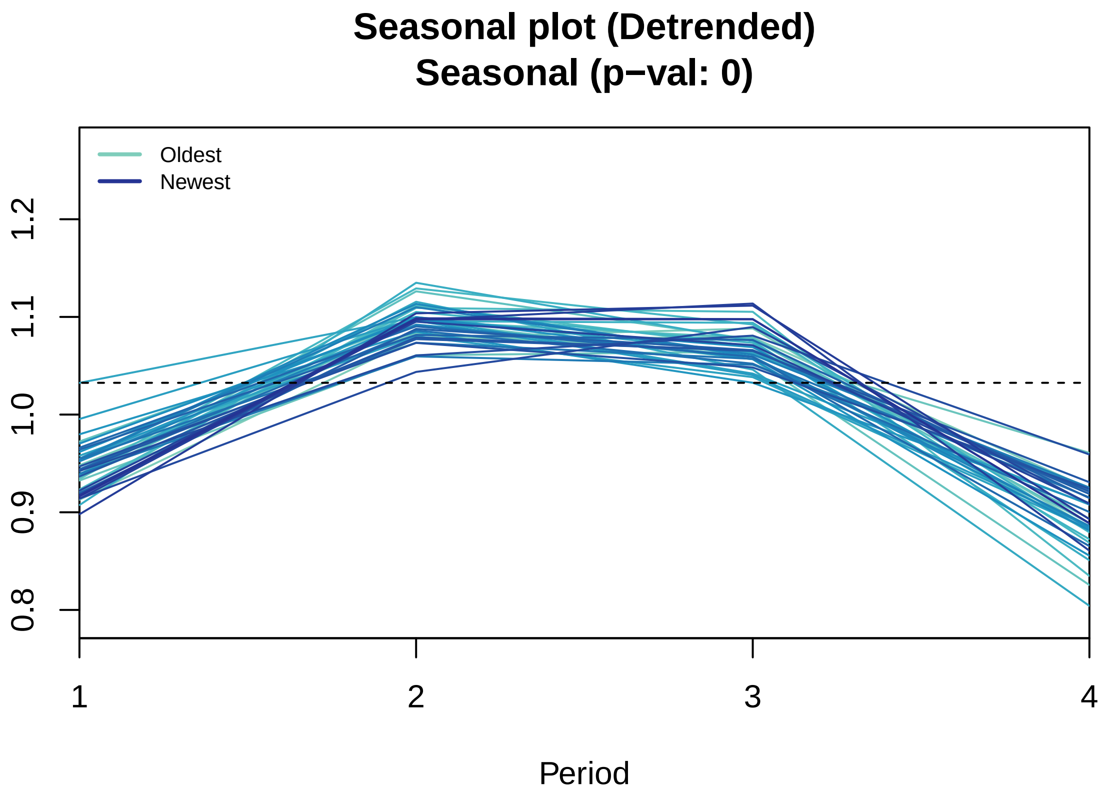

```
plot(pop_ts, main="Trend and Seasonal Quarterly Data", ylab="POP")
```




```
cma_pop <- cmav(pop_ts, outplot=1)
```

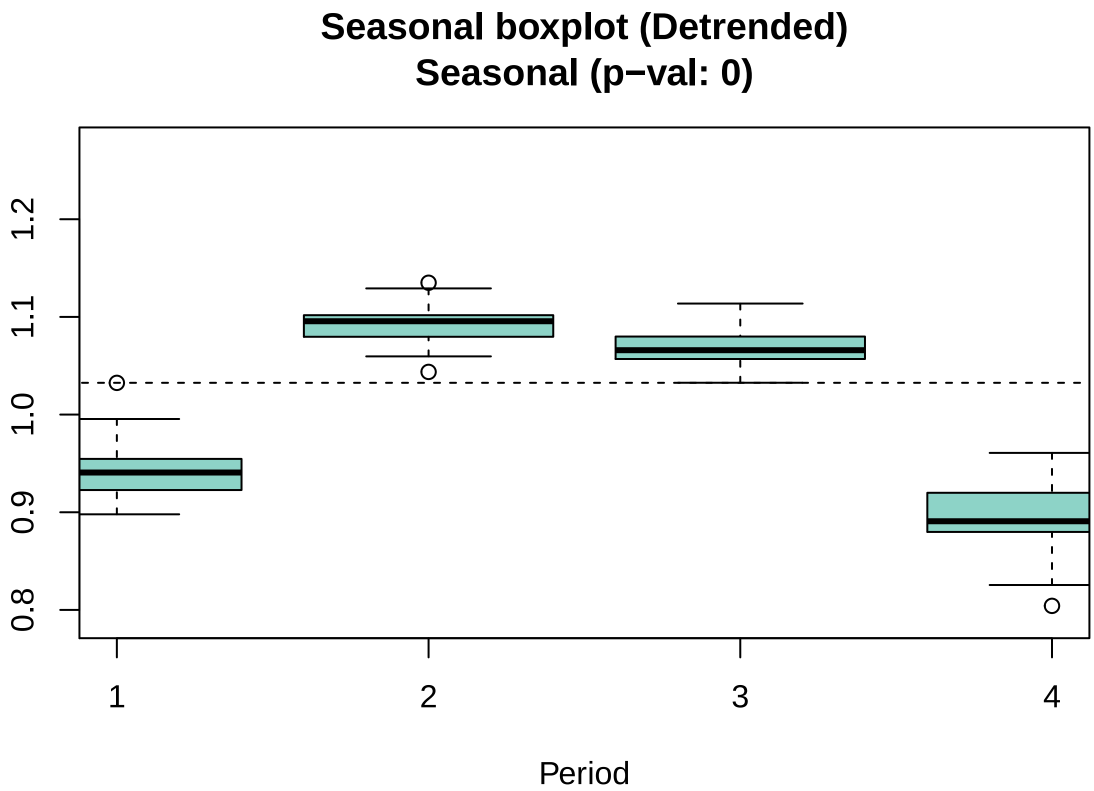



```
seasplot(pop_ts)
```



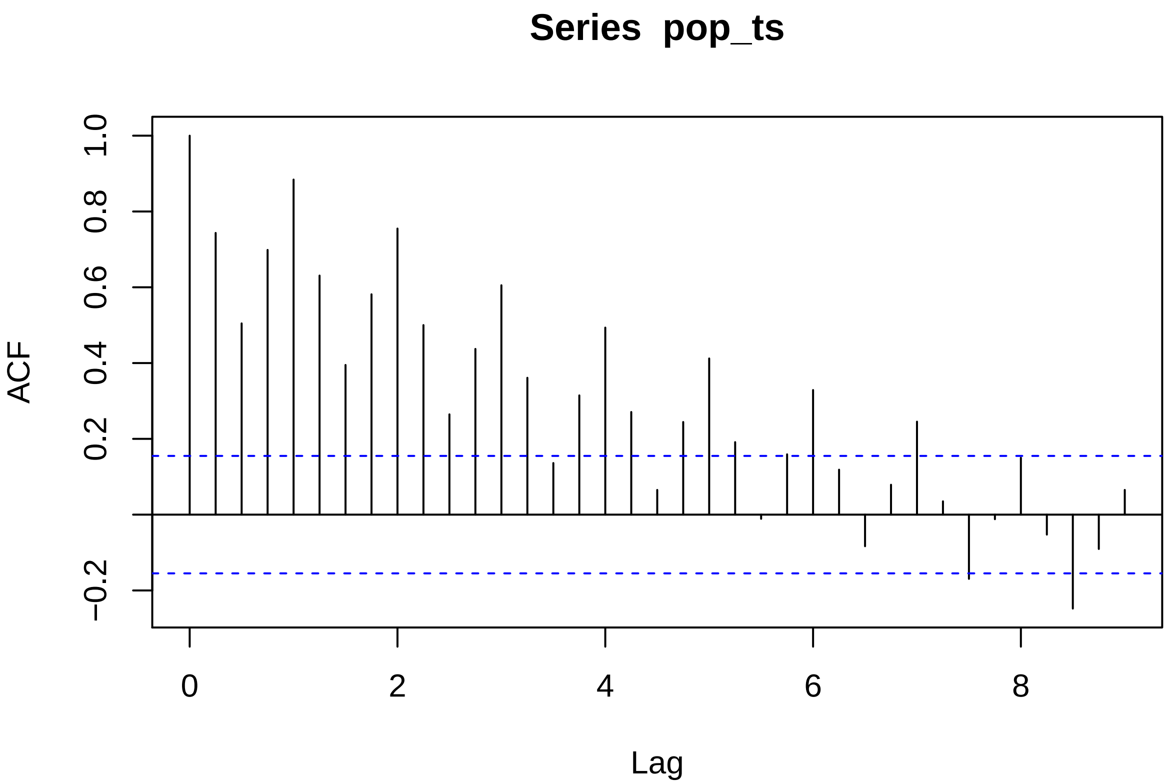

```
## Results of statistical testing
## Evidence of trend: TRUE  (pval: 0.001)
## Evidence of seasonality: TRUE  (pval: 0)
```

```r
seasplot(pop_ts, outplot =2)
```




```
## Results of statistical testing
## Evidence of trend: TRUE  (pval: 0.001)
## Evidence of seasonality: TRUE  (pval: 0)
```

```r
acf(pop_ts, lag.max = 36)
```




```r
summary(wo(pop_ts))
```

```
## Test used:  WO
##
## Test statistic:  1
## P-value:  0 0 0
##
## The WO - test identifies seasonality
```

We detected seasonality in population from the above. In fact, this is very similar to the HOUST variables as we see spikes in Q2-Q3 and dip in Q1 and Q4.

4. Read the train.csv and test.csv files in R which contain training and test information on 10,000 customers. Aim is to predict wihch customers will default on their credit card debt. These datasets contain the following information/ variables - default, student, balance, income.

Let us load the data and summarise the information

```
# reading data
train_data <- read.csv('/Users/mac/Downloads/final-2020-canvas/datasets/train-default.csv')
test_data <- read.csv('/Users/mac/Downloads/final-2020-canvas/datasets/test-default.csv')
str(train_data)
```

```
## 'data.frame':    6047 obs. of  5 variables:
##  $ X      : int  1 3 6 7 9 10 12 14 15 17 ...
##  $ default: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ student: Factor w/ 2 levels "No","Yes": 1 1 2 1 1 1 2 1 1 1 ...
##  $ balance: num  730 1074 920 826 1161 ...
##  $ income : num  44362 31767 7492 24905 37469 ...
```

```
str(test_data)
```

```
## 'data.frame':    3953 obs. of  5 variables:
##  $ X      : int  2 4 5 8 11 13 16 20 21 22 ...
##  $ default: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ student: Factor w/ 2 levels "No","Yes": 2 1 1 2 2 1 1 1 1 1 ...
##  $ balance: num  817 529 786 809 0 ...
##  $ income : num  12106 35704 38463 17600 21871 ...
```

```
#summary
summary(train_data)
```

```
##        X         default    student       balance           income     
##  Min.   :   1   No :5852   No :4240   Min.   :   0.0   Min.   :  772  
##  1st Qu.:2486   Yes: 195   Yes:1807   1st Qu.: 472.7   1st Qu.:21103  
##  Median :5027                         Median : 825.6   Median :34368  
##  Mean   :5008                         Mean   : 836.3   Mean   :33379  
##  3rd Qu.:7520                         3rd Qu.:1171.0   3rd Qu.:43657  
##  Max.   :9999                         Max.   :2654.3   Max.   :73554  
```

```
summary(test_data)
```

```
##        X          default    student       balance           income     
##  Min.   :    2   No :3815   No :2816   Min.   :   0.0   Min.   : 1498  
##  1st Qu.: 2509   Yes: 138   Yes:1137   1st Qu.: 491.0   1st Qu.:21663  
##  Median : 4971                         Median : 819.1   Median :34891  
##  Mean   : 4989                         Mean   : 833.9   Mean   :33728  
##  3rd Qu.: 7478                         3rd Qu.:1153.3   3rd Qu.:44030  
##  Max.   :10000                         Max.   :2461.5   Max.   :70701  
```

Key observations -

1. Defaulters in the training data are quite low (3.22%)
2. Students are about 29.8% in the training data
3. We see balace is marginally +vely skewed while income is marginally -vely skewed
4. We don't see any evidence of outliers but we will check the plots before commenting on this

Any missing values ?

```
training_data <-na.omit(train_data)
str(training_data)
```

```
## 'data.frame':    6047 obs. of  5 variables:
##  $ X      : int  1 3 6 7 9 10 12 14 15 17 ...
##  $ default: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ student: Factor w/ 2 levels "No","Yes": 1 1 2 1 1 1 2 1 1 1 ...
##  $ balance: num  730 1074 920 826 1161 ...
##  $ income : num  44362 31767 7492 24905 37469 ...
```

```
testing_data <-na.omit(test_data)
str(testing_data)
```

```
## 'data.frame':    3953 obs. of  5 variables:
##  $ X      : int  2 4 5 8 11 13 16 20 21 22 ...
##  $ default: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ student: Factor w/ 2 levels "No","Yes": 2 1 1 2 2 1 1 1 1 1 ...
##  $ balance: num  817 529 786 809 0 ...
##  $ income : num  12106 35704 38463 17600 21871 ...
```

We did not find any missing values in the data.

Let us convert the No-Yes in default and student columns to 0-1.

```
train_data <- train_data %>%
      mutate(default = ifelse(default == "No",0,1)) %>%
      mutate(student = ifelse(student == "No",0,1))
test_data <- test_data %>%
      mutate(default = ifelse(default == "No",0,1)) %>%
      mutate(student = ifelse(student == "No",0,1))
str(train_data)
```

```
## 'data.frame':    6047 obs. of  5 variables:
##  $ X      : int  1 3 6 7 9 10 12 14 15 17 ...
##  $ default: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ student: num  0 0 1 0 0 0 1 0 0 0 ...
##  $ balance: num  730 1074 920 826 1161 ...
##  $ income : num  44362 31767 7492 24905 37469 ...
```

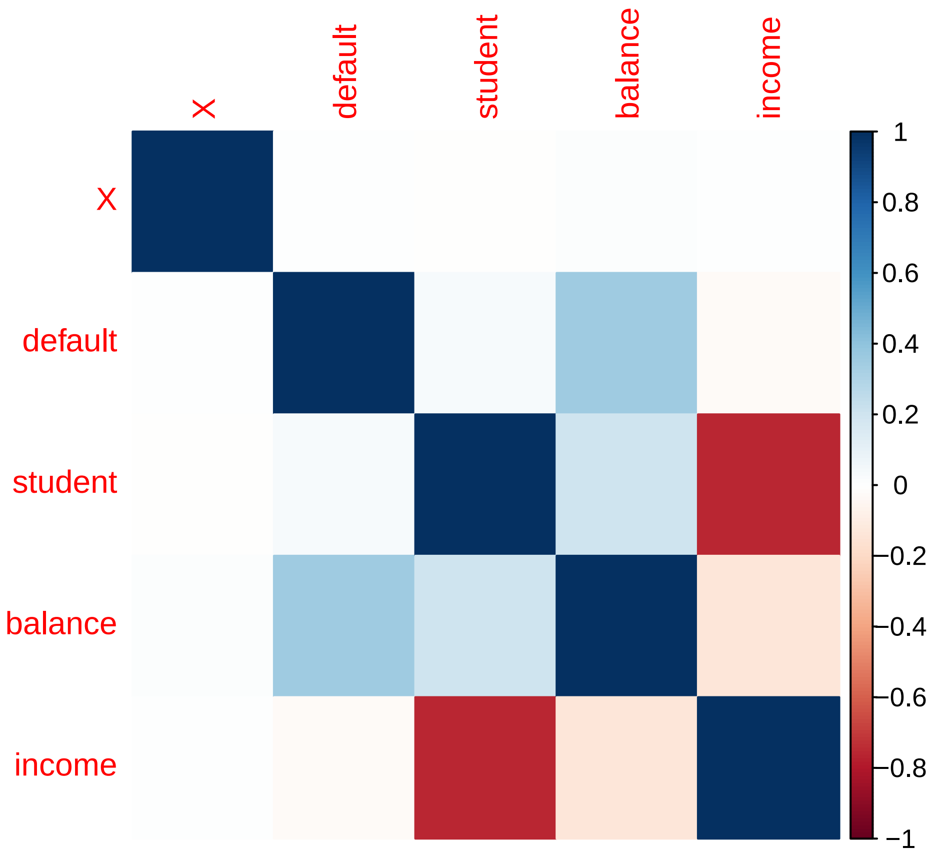
Correlation plot -

We check correlation before moving towards modeling exercise.

```
M<-cor(train_data)
head(round(M,2))
```

```
##             X default student balance income
## X        1.00    0.01   -0.01    0.02   0.01
## default  0.01    1.00    0.04    0.35  -0.02
## student -0.01    0.04    1.00    0.20  -0.75
## balance  0.02    0.35    0.20    1.00  -0.14
## income   0.01   -0.02   -0.75   -0.14   1.00
```

```r
corrplot(M, method="color")
```



Key observations -

1. We see default is +vely correlated with balance although the relation isn't very strong (0.35)

2. We also see default being marginally -vely correlated to income as individuals with higher income may have lower default.

3. We see strong negative correlation (-0.75) between student and income which is understandable.

4. We also see weak positive correlation between student and balance and weak negative relation between income and balance.

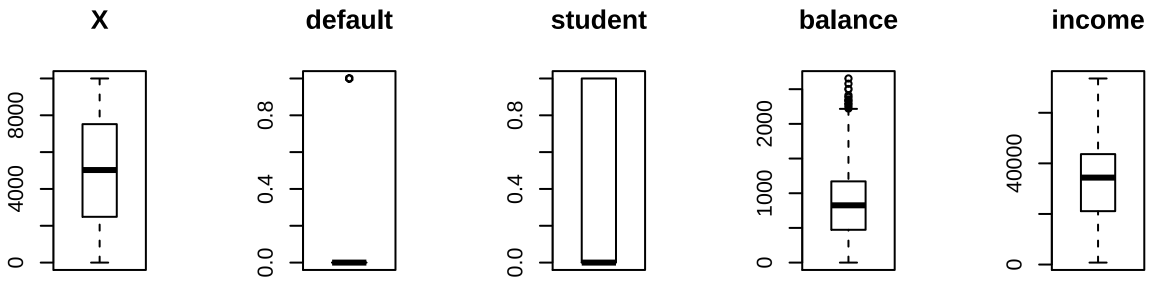
5. A theme that comes out here is that students with low to no income may have higher likelihood of defaulting. On the contrary, we note that students credit card debt if taken care of by parents would depend on their income. We can construct such hypothesis before proceeding to modeling exercise.

Outlier/ Univariate checks -

We create a quick boxplot of all variables.

```r
par(mfrow=c(2,5))
for (i in 1:length(train_data)) {
```

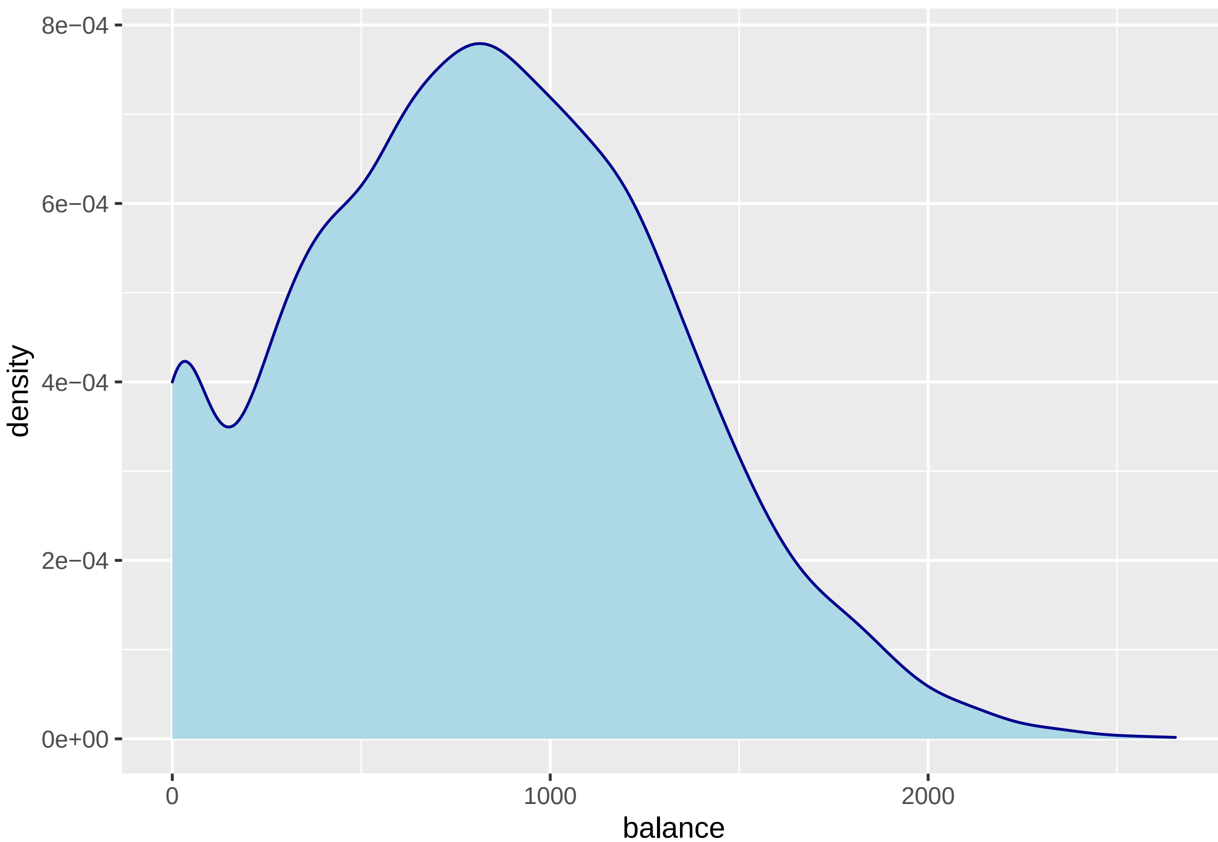
```
        boxplot(train_data[,i], main=names(train_data[i]), type="l")
}
```



We see some evidence of outliers in balance. Let us look closely to understand if this is data issue or actual high balance of customers.

Plotting balance

```
ggplot(train_data, aes(x=balance)) +
  geom_density(color="darkblue", fill="lightblue")
```



We see how the balance plot tapers down above a balance of 2000.There is not much reason to think that these balances are outliers at the moment.

Armed with the EDA above, We now proceed to modeling exercise.

4a) Fit logistic regression with default as the response and other variables balance and income as the predictor. Make sure the variables in your model are significant. Perform regression diagnostics. Display any relevant plots.

We convert default and student back to factor for modeling.

```
train_data$default <- factor(train_data$default)
train_data$student <- factor(train_data$student)

test_data$default <- factor(test_data$default)
test_data$student <- factor(test_data$student)

str(train_data)
```

```
## 'data.frame':    6047 obs. of  5 variables:
##  $ X      : int  1 3 6 7 9 10 12 14 15 17 ...
##  $ default: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ student: Factor w/ 2 levels "0","1": 1 1 2 1 1 1 2 1 1 1 ...
##  $ balance: num  730 1074 920 826 1161 ...
##  $ income : num  44362 31767 7492 24905 37469 ...
```

Let's check VIF

```
mod <-  lm(as.numeric(default) ~  balance + income, data = train_data)
summary(mod)
```

```
##
## Call:
## lm(formula = as.numeric(default) ~ balance + income, data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.23934 -0.06918 -0.02543  0.02124  0.95344
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 9.127e-01  7.220e-03 126.412   <2e-16 ***
## balance     1.296e-04  4.413e-06  29.357   <2e-16 ***
## income      3.353e-07  1.599e-07   2.097   0.0361 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1653 on 6044 degrees of freedom
## Multiple R-squared:  0.1253, Adjusted R-squared:  0.125
## F-statistic: 432.9 on 2 and 6044 DF,  p-value: < 2.2e-16
```

```
vif(mod)
```

```
##  balance   income
## 1.019817 1.019817
```

We see both variables are significant in linear regression and VIF is small ~ 1.019. This is good news.

Two-way contingency table of categorical outcome and predictors

Since we want to make sure there are not 0 cells

```
xtabs(~train_data$default + train_data$student, data = train_data)
```

```
##                  train_data$student
## train_data$default    0    1
##                  0 4122 1730
##                  1  118   77
```

```
xtabs(~test_data$default + test_data$student, data = test_data)
```

```
##                 test_data$student
## test_data$default    0    1
##                 0 2728 1087
##                 1   88   50
```

Great news. Unlike Assignment 5, we see here there is no issue of zero contingency and a normal logistic regression should be fine.

Fitting logistic model

```
set.seed(123)
mylogit <- glm(default ~ balance + income , data = train_data, family = binomial(link="logit"))
summary(mylogit)
```

```
##
## Call:
## glm(formula = default ~ balance + income, family = binomial(link = "logit"),
##     data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4579  -0.1367  -0.0517  -0.0183   3.3748
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.173e+01  5.768e-01 -20.340  < 2e-16 ***
## balance      5.787e-03  3.028e-04  19.115  < 2e-16 ***
## income       1.719e-05  6.466e-06   2.658  0.00785 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1723.03  on 6046  degrees of freedom
## Residual deviance:  901.62  on 6044  degrees of freedom
## AIC: 907.62
##
## Number of Fisher Scoring iterations: 8
```

We see that both balance and income are significant predictors of defaulting on the credit card. Our model has an AIC value of 907.62.

Let's predict the outcome for this model

```
predicted <- predict(mylogit, test_data, type="response")
```

Deciding on optimal cutoff

```
optCutOff <- optimalCutoff(test_data$default, predicted)[1]
optCutOff
```

```
## [1] 0.5180984
```

This tells us that we can use this prob cutoff to classify an observation as 0 or 1. If the prob-value is below this threshold we can classify it as non-defaulter else the customer is a defaulter.

Mis-classification error

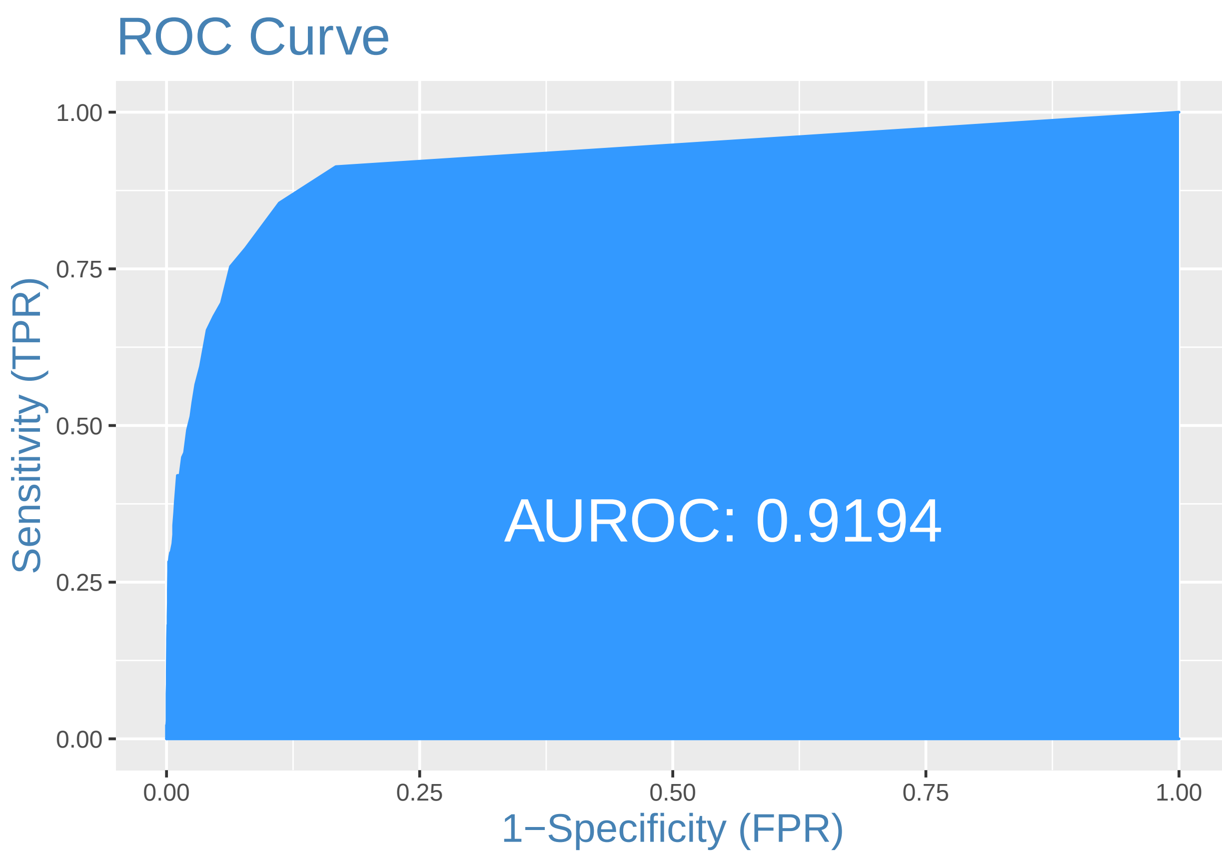
```
misClassError(test_data$default, predicted, threshold = optCutOff)
```

```
## [1] 0.0271
```

We note a misclassification error on test set of 2.7%

ROC curve

```
plotROC(test_data$default, predicted)
```



We see an AUC of 0.91 which is quite good.

Using optimal cutoff to determine accuracy measures with positive class as 1

```
threshold=optCutOff
predicted_values<-ifelse(predict(mylogit, test_data,
                                type="response")>threshold,1,0)
actual_values<-test_data$default
confusionMatrix(data = as.factor(predicted_values),
                reference = as.factor(actual_values),
                positive='1',mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3807   99
##          1    8   39
##
##                Accuracy : 0.9729
##                  95% CI : (0.9674, 0.9778)
##     No Information Rate : 0.9651
##     P-Value [Acc > NIR] : 0.003142
##
##                   Kappa : 0.4112
##
```

```
##  Mcnemar's Test P-Value : < 2.2e-16
##
##               Precision : 0.829787
##                  Recall : 0.282609
##                      F1 : 0.421622
##              Prevalence : 0.034910
##          Detection Rate : 0.009866
##    Detection Prevalence : 0.011890
##       Balanced Accuracy : 0.640256
##
##        'Positive' Class : 1
##
```

We see an overall accuracy of ~97.2% with precision of 0.79, recall of 0.28 and an F1-score of 0.41

Manipulating cutoff to increase recall

```
threshold=0.25
predicted_values<-ifelse(predict(mylogit, test_data,
                            type="response")>threshold,1,0)
actual_values<-test_data$default
confusionMatrix(data = as.factor(predicted_values),
                reference = as.factor(actual_values),
                positive='1',mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3741   73
##          1   74   65
##
##                Accuracy : 0.9628
##                  95% CI : (0.9564, 0.9685)
##     No Information Rate : 0.9651
##     P-Value [Acc > NIR] : 0.7961
##
##                   Kappa : 0.45
##
##  Mcnemar's Test P-Value : 1.0000
##
##               Precision : 0.46763
##                  Recall : 0.47101
##                      F1 : 0.46931
##              Prevalence : 0.03491
##          Detection Rate : 0.01644
##    Detection Prevalence : 0.03516
##       Balanced Accuracy : 0.72581
##
##        'Positive' Class : 1
##
```

We see an overall accuracy of ~96.2% with precision of 0.46, recall of 0.47 and an F1-score of 0.46

4b) Why is your model a good/ reasonable model ? Check AIC and pseudo R square values.

Our model is a decent model we can say as we recall the statistics.

AIC - 908.61 Misclassification rate - 2.7% AUC - 0.91 Accuracy - 97.2% Precision - 79% Recall - 28 % F1-Score - 0.41

Overall the model statistics are fine but we see a very low recall. Hence, we changed the probability cutoff to 0.25 instead of 0.51 to increase recall of the model. In this problem, we cannot afford such a low recall since we want to be able to identify the defaulter with more certainty.

Pseudo- Rsquare for Logistic regression

```
nullmod <- glm(train_data$default~1, family="binomial")
r_sq <- 1-logLik(mylogit)/logLik(nullmod)
r_sq
```

```
## 'log Lik.' 0.4767242 (df=3)
```

```
summary_glm <- summary(mylogit)
list( summary_glm$coefficient,
      round( 1 - ( summary_glm$deviance / summary_glm$null.deviance ), 2 ) )
```

```
## [[1]]
##                  Estimate   Std. Error    z value     Pr(>|z|)
## (Intercept) -1.173178e+01 5.767777e-01 -20.340213 5.668368e-92
## balance      5.787483e-03 3.027670e-04  19.115303 1.883230e-81
## income       1.718741e-05 6.465624e-06   2.658275 7.854172e-03
##
## [[2]]
## [1] 0.48
```

We get a pseduo R-square of 0.48.

4c) Give interpretation of regression coefficients.

We can write our model form as -

$$log(\frac{p}{1-p}) = -1.173178e + 01 + 5.787483e - 03 * balance + 1.718741e - 05 * income$$

```
coeffs <- round(coefficients(mylogit), digits = 4)

exp(coeffs[[1]]); exp(coeffs[[2]]); exp(coeffs[[3]])
```

```
## [1] 8.034225e-06
```

```
## [1] 1.005817
```

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

We now interpret the coefficients as -

Keeping income constant, for a unit increase in balance, the odds of defaulting on a credit card increase by 1.0058 or by 0.58%.

Similarly, Keeping balance constant, for a unit increase in income, the odds of defaulting on a credit card increase by 1.

We can interpet the intercept term as the odds of defaulting on a credit when balance and income are both 0.

4d) Form confusion matrix over test data. What % of the time are your predictions correct ?

We did this in 4a) itself however we show here again.

```r
threshold=optCutOff
predicted_values<-ifelse(predict(mylogit, test_data,
                             type="response")>threshold,1,0)
actual_values<-test_data$default
confusionMatrix(data = as.factor(predicted_values),
                reference = as.factor(actual_values),
                positive='1',mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3807   99
##          1    8   39
##
##                Accuracy : 0.9729
##                  95% CI : (0.9674, 0.9778)
##     No Information Rate : 0.9651
##     P-Value [Acc > NIR] : 0.003142
##
##                   Kappa : 0.4112
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##               Precision : 0.829787
##                  Recall : 0.282609
##                      F1 : 0.421622
##              Prevalence : 0.034910
##          Detection Rate : 0.009866
##    Detection Prevalence : 0.011890
##       Balanced Accuracy : 0.640256
##
##        'Positive' Class : 1
##
```

```r
threshold=0.25
predicted_values<-ifelse(predict(mylogit, test_data,
                             type="response")>threshold,1,0)
actual_values<-test_data$default
confusionMatrix(data = as.factor(predicted_values),
                reference = as.factor(actual_values),
                positive='1',mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 3741   73
##          1   74   65
```

```
##
##                Accuracy : 0.9628
##                  95% CI : (0.9564, 0.9685)
##     No Information Rate : 0.9651
##     P-Value [Acc > NIR] : 0.7961
##
##                   Kappa : 0.45
##
##  Mcnemar's Test P-Value : 1.0000
##
##               Precision : 0.46763
##                  Recall : 0.47101
##                      F1 : 0.46931
##              Prevalence : 0.03491
##          Detection Rate : 0.01644
##    Detection Prevalence : 0.03516
##       Balanced Accuracy : 0.72581
##
##        'Positive' Class : 1
##
```

Our model accuracy is 97.29% so 97.29% times our predictions are correct however we increased our recall in the second iteration changing the prob cutoff from 0.51 to 0.25 in order to predict defaulters better. For that model, we are correct 96.2% times.

4e) In your model, what is the estimated probability of default for a student with a credit card balance of of $2000 and income of $40000. What is the prob. of default for a non-student with the same credit card balance and income ?

Since, we did not include student to our model, we don't expect the probabilities of default to be different for the same levels of income and balance. Neverthless, we compute

```r
x0 = data.frame(student=1,balance = 2000, income = 40000, stringsAsFactors = FALSE)
predict(mylogit, x0, type="response")
```

```
##         1
## 0.6296417
```

```r
x0 = data.frame(student=0,balance = 2000, income = 40000, stringsAsFactors = FALSE)
predict(mylogit, x0, type="response")
```

```
##         1
## 0.6296417
```

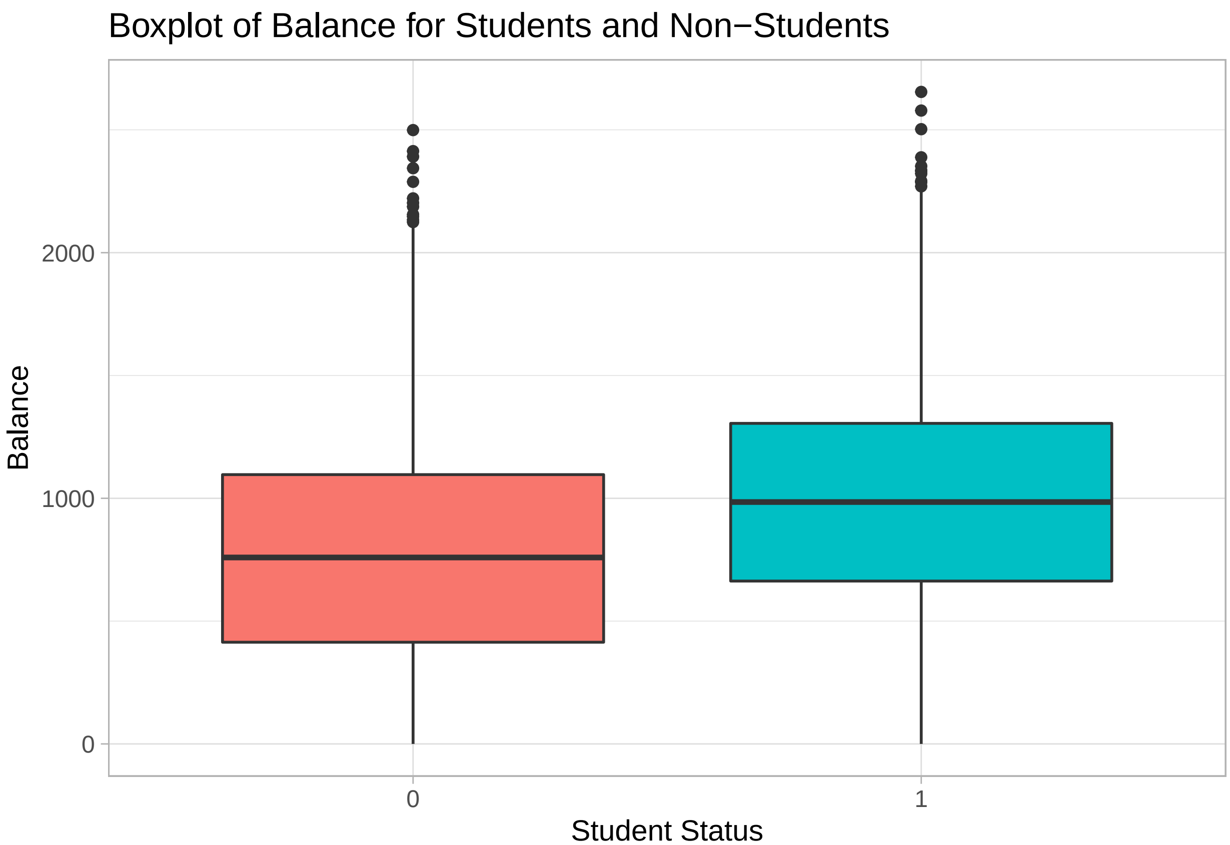
As, we can see the prob of default is 0.629 irrespective of whether the customer is a student or not.

4f) Are the variables balance and student correlated ? If yes, why ? If no, explain.

We first plot them together -

```r
ggplot(train_data,
       aes(x = factor(student),
           y = balance,
           group = student,
           fill = factor(student))) +
```

```r
  geom_boxplot() +
  theme_light() +
  theme(legend.position = "none") +
  labs(title = "Boxplot of Balance for Students and Non-Students",
       x = "Student Status",
       y = "Balance")
```



It seems the means of balance for student vs non-student are not that different.

And now we check the correlation between the variables -

```r
M<-cor(train_data$balance,as.numeric(train_data$student))
head(round(M,2))
```

```
## [1] 0.2
```

Recall earlier we computed this correlation in 4) as well. The variables are very weakly +vely correlated (0.2). We do not see why being a student should indicate a higher balance on the credit card though.

4g) Now let's add the binary variable student to the model.

```r
set.seed(123)
mylogit <- glm(default ~ balance + income + student + 0 ,
              data = train_data, family = binomial(link="logit"))
summary(mylogit)
```

```
##
## Call:
```

```
## glm(formula = default ~ balance + income + student + 0, family = binomial(link = "logit"),
##     data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4556  -0.1344  -0.0499  -0.0174   3.4155
##
## Coefficients:
##            Estimate Std. Error z value Pr(>|z|)
## balance   5.907e-03  3.102e-04  19.040   <2e-16 ***
## income   -5.013e-06  1.079e-05  -0.465    0.642
## student0 -1.091e+01  6.481e-01 -16.830   <2e-16 ***
## student1 -1.172e+01  5.825e-01 -20.114   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8382.92  on 6047  degrees of freedom
## Residual deviance:  895.02  on 6043  degrees of freedom
## AIC: 903.02
##
## Number of Fisher Scoring iterations: 8
```

We see that the variable income is no longer significant in this model.

4h) Does the data say that it is more likely for a student to default compared to non-student for different values of income level. Please comment.

We compute -

```
coeffs <- round(x = coefficients(mylogit),
                digits = 4)
coeffs
```

```
##  balance   income student0 student1
##   0.0059   0.0000 -10.9070 -11.7165
```

The coefficient of student1 is -11.7165 while the coefficient of student0 is -10.9070 hence the data suggests that the odds of defaulting for a student are lower compared to non-student. Hence, for different levels of income, the result would remain the same.

5. These days there is a lot of discussion on how heathcare system should look like in the US. For a scientific discussion, one needs to have a model of demand in the healthcare system. In this question, we work on dvisit which is about modeling the demand for doctor visits in terms of explanatory variables such as age, income, existence of health insurance, others.

Let us load the data and summarise the information

```r
# reading data
data(dvisits)
str(dvisits)
```

```
## 'data.frame':    5190 obs. of  19 variables:
##  $ sex     : int  1 1 0 0 0 1 1 1 1 0 ...
##  $ age     : num  0.19 0.19 0.19 0.19 0.19 0.19 0.19 0.19 0.19 0.19 ...
##  $ agesq   : num  0.0361 0.0361 0.0361 0.0361 0.0361 0.0361 0.0361 0.0361 0.0361 0.0361 ...
##  $ income  : num  0.55 0.45 0.9 0.15 0.45 0.35 0.55 0.15 0.65 0.15 ...
##  $ levyplus: int  1 1 0 0 0 0 0 0 1 1 ...
##  $ freepoor: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ freerepa: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ illness : int  1 1 3 1 2 5 4 3 2 1 ...
##  $ actdays : int  4 2 0 0 5 1 0 0 0 0 ...
##  $ hscore  : int  1 1 0 0 1 9 2 6 5 0 ...
##  $ chcond1 : int  0 0 0 0 1 1 0 0 0 0 ...
##  $ chcond2 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ doctorco: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ nondocco: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hospadmi: int  0 0 1 0 0 0 0 0 0 0 ...
##  $ hospdays: int  0 0 4 0 0 0 0 0 0 0 ...
##  $ medicine: int  1 2 2 0 3 1 0 1 1 1 ...
##  $ prescrib: int  1 1 1 0 1 1 0 1 0 1 ...
##  $ nonpresc: int  0 1 1 0 2 0 0 0 1 0 ...
```

This data is from the Australian health survey of 1977-78 where 5190 single adults where yound and old have been oversampled. The definitions are as follows -

1. sex : 1 if female, 0 if male

2. age : Age in years divided by 100 measured as mid point of 10 age groups from 15-19 to 65-69 with 70+ coded as 72

3. agesq : age squared

4. income : Annual income in australian dollars divided by 1000. 14000+ income is coded as 15000

5. levyplus : 1 if covered by private health insurance fund for private patient in public hospital with doctor of choice, 0 otherwise

6. freepoor : 1 if covered by govt. because low income, recent, immigrant, unemployed, 0 otherwise

7. freerepa : 1 if covered free by govt. because of old age or disability pension, or because invalid veteran or family of deceased veteran, 0 otherwise

8. illness : Number of illnesses in past two weeks capped at 5 for 5+

9. actdays : Number of days of reduced activity in past two weeks due to injury or illness
10. hscore : General health questionnaire score using Goldberg's method. High score indicates bad health
11. chcond1 : 1 if chronic condition but not limited in activity, 0 otherwise
12. chcond2 : 1 if chronic condition and limited in activity, 0 otherwise
13. doctorco : Number of consulations with a doc in last two weeks
14. nondocco : Number of consulations with non-doc in last two weeks
15. hospadmi : Number of admissions to a hospital in past 12 months
16. hospdays : Number of nights in a hospital during most recent admission. No admissions in past 12 months is coded as 0 and 80+ is coded as 80
17. medicine : Total number of prescribed and nonprescribed medications used in past 2 days
18. prescrib : Total number of prescribed medications in past 2 days
19. nonpresc : Total number of non-prescribed medications in past 2 days

We now analyse the data before developing the model.

```
#summary
summary(dvisits)
```

```
##       sex              age             agesq            income
##  Min.   :0.0000   Min.   :0.1900   Min.   :0.0361   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.2200   1st Qu.:0.0484   1st Qu.:0.2500
##  Median :1.0000   Median :0.3200   Median :0.1024   Median :0.5500
##  Mean   :0.5206   Mean   :0.4064   Mean   :0.2071   Mean   :0.5832
##  3rd Qu.:1.0000   3rd Qu.:0.6200   3rd Qu.:0.3844   3rd Qu.:0.9000
##  Max.   :1.0000   Max.   :0.7200   Max.   :0.5184   Max.   :1.5000
##     levyplus         freepoor          freerepa         illness
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.0000   Min.   :0.000
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.000
##  Median :0.0000   Median :0.00000   Median :0.0000   Median :1.000
##  Mean   :0.4428   Mean   :0.04277   Mean   :0.2102   Mean   :1.432
##  3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:2.000
##  Max.   :1.0000   Max.   :1.00000   Max.   :1.0000   Max.   :5.000
##     actdays           hscore          chcond1          chcond2
##  Min.   : 0.0000   Min.   : 0.000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median : 0.0000   Median : 0.000   Median :0.0000   Median :0.0000
##  Mean   : 0.8619   Mean   : 1.218   Mean   :0.4031   Mean   :0.1166
##  3rd Qu.: 0.0000   3rd Qu.: 2.000   3rd Qu.:1.0000   3rd Qu.:0.0000
##  Max.   :14.0000   Max.   :12.000   Max.   :1.0000   Max.   :1.0000
##     doctorco         nondocco          hospadmi          hospdays
##  Min.   :0.0000   Min.   : 0.0000   Min.   :0.0000   Min.   : 0.000
##  1st Qu.:0.0000   1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.: 0.000
##  Median :0.0000   Median : 0.0000   Median :0.0000   Median : 0.000
##  Mean   :0.3017   Mean   : 0.2146   Mean   :0.1736   Mean   : 1.334
```

```
##  3rd Qu.:0.0000   3rd Qu.: 0.0000   3rd Qu.:0.0000   3rd Qu.: 0.000
##  Max.   :9.0000   Max.   :11.0000   Max.   :5.0000   Max.   :80.000
##     medicine         prescrib         nonpresc
##  Min.   :0.000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :1.000   Median :0.0000   Median :0.0000
##  Mean   :1.218   Mean   :0.8626   Mean   :0.3557
##  3rd Qu.:2.000   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :8.000   Max.   :8.0000   Max.   :8.0000
```

Key observations -

1. We see age is +vely skewed but understandable due to oversampled dataset.

2. Given most of the capping logic already present, we may not see any outliers.

3. Our dependent variable seems to have a lot of 0 days given the 3rd quartile is also 0. On average however 1.3 nights are spent in hospital.

4. Medicine seems to be the sum of prescrib and nonpresc.

Any missing values ?

```
data <-na.omit(dvisits)
str(data)
```

```
## 'data.frame':    5190 obs. of  19 variables:
##  $ sex     : int  1 1 0 0 0 1 1 1 1 0 ...
##  $ age     : num  0.19 0.19 0.19 0.19 0.19 0.19 0.19 0.19 0.19 0.19 ...
##  $ agesq   : num  0.0361 0.0361 0.0361 0.0361 0.0361 0.0361 0.0361 0.0361 0.0361 0.0361 ...
##  $ income  : num  0.55 0.45 0.9 0.15 0.45 0.35 0.55 0.15 0.65 0.15 ...
##  $ levyplus: int  1 1 0 0 0 0 0 0 1 1 ...
##  $ freepoor: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ freerepa: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ illness : int  1 1 3 1 2 5 4 3 2 1 ...
##  $ actdays : int  4 2 0 0 5 1 0 0 0 0 ...
##  $ hscore  : int  1 1 0 0 1 9 2 6 5 0 ...
##  $ chcond1 : int  0 0 0 0 1 1 0 0 0 0 ...
##  $ chcond2 : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ doctorco: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ nondocco: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ hospadmi: int  0 0 1 0 0 0 0 0 0 0 ...
##  $ hospdays: int  0 0 4 0 0 0 0 0 0 0 ...
##  $ medicine: int  1 2 2 0 3 1 0 1 1 1 ...
##  $ prescrib: int  1 1 1 0 1 1 0 1 0 1 ...
##  $ nonpresc: int  0 1 1 0 2 0 0 0 1 0 ...
```

We did not find any missing values in the data.

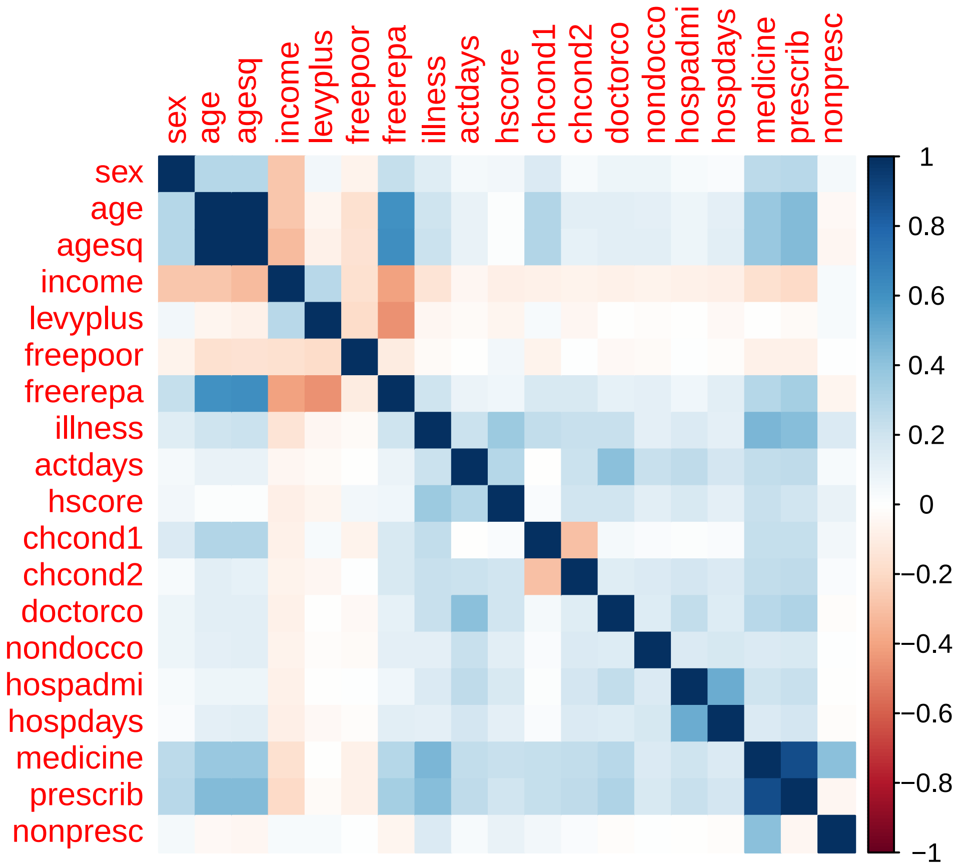
Correlation plot -

We check correlation before moving towards modeling exercise.

```
M<-cor(dvisits)
head(round(M,2))
```

```
##             sex   age agesq income levyplus freepoor freerepa illness
## sex        1.00  0.28  0.29  -0.28     0.06    -0.06     0.24    0.13
## age        0.28  1.00  0.99  -0.27    -0.06    -0.16     0.60    0.20
## agesq      0.29  0.99  1.00  -0.32    -0.08    -0.15     0.62    0.21
## income    -0.28 -0.27 -0.32   1.00     0.28    -0.16    -0.40   -0.15
## levyplus   0.06 -0.06 -0.08   0.28     1.00    -0.19    -0.46   -0.05
## freepoor  -0.06 -0.16 -0.15  -0.16    -0.19     1.00    -0.11   -0.03
##           actdays hscore chcond1 chcond2 doctorco nondocco hospadmi
## sex          0.04   0.06    0.15    0.03     0.08     0.08     0.04
## age          0.09   0.02    0.30    0.12     0.12     0.12     0.07
## agesq        0.09   0.02    0.30    0.11     0.12     0.12     0.08
## income      -0.05  -0.09   -0.08   -0.07    -0.08    -0.07    -0.07
## levyplus    -0.03  -0.05    0.03   -0.04    -0.01    -0.01     0.00
## freepoor    -0.01   0.06   -0.07    0.00    -0.04    -0.02     0.00
##           hospdays medicine prescrib nonpresc
## sex           0.03     0.27     0.27     0.05
## age           0.12     0.38     0.43    -0.04
## agesq         0.12     0.38     0.44    -0.05
## income       -0.09    -0.16    -0.20     0.04
## levyplus     -0.04     0.00    -0.02     0.03
## freepoor     -0.01    -0.07    -0.08     0.00
```

```
corrplot(M, method="color")
```



Key observations -

1. We see hospdays is correlated with hospadmi (+0.5) which seems to indicate that number of admissions in past 12 months is correlated with number of nights in hospital in the last admission. This maybe case of causation but we will explore more in detail later.

2. We also see a negative correlation between hospdays and each of levyplus, income and freepoor.

3. Age seems to be weakly positively correlated with medicine especially prescribed ones but negatievly with non prescribed ones.

4. Income is mostly negatively correlated with most other explanatory variables as well as hospdays.

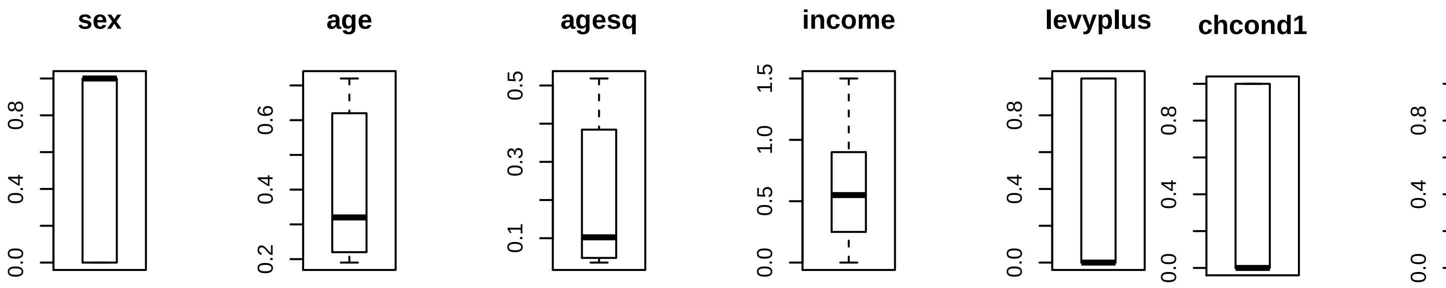
5. Levyplus and freerepa are negatively correlated (-0.46). This makes sense as the variables signify private care vs public care.
6. Doctorco and illness are correlated +vely as well.
7. prescrib and medicine are most +vely strongly correlated which is expected.

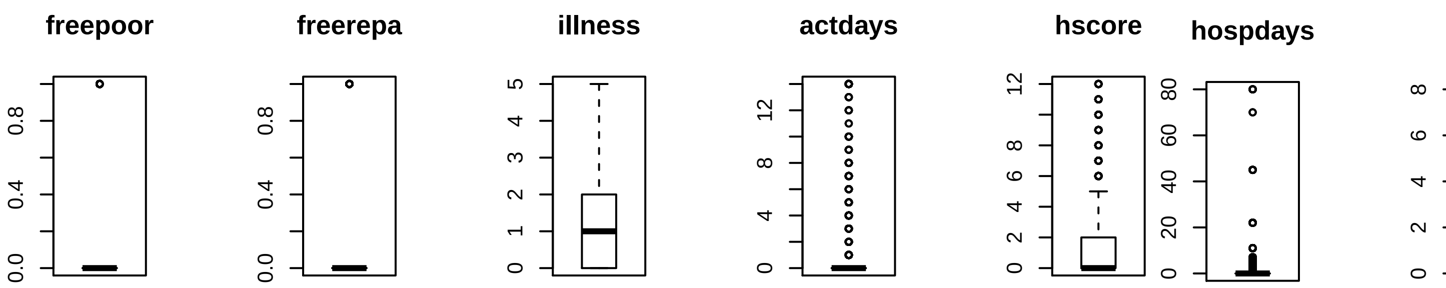
Outlier/ Univariate checks -

We confirm our earlier hypothesis of no outliers by looking at some univariate and outlier checks.

```
par(mfrow=c(2,5))
for (i in 1:length(dvisits)) {
```

```r
        boxplot(dvisits[,i], main=names(dvisits[i]), type="l")
}
```





We see actdays, hscore, doctorco, nondocco, hospadmi, hospadys and medicine (both prescrib and nonpresc) having some outliers.

Let's check how many zeros are in dataset

```r
colSums(dvisits==0)
```

```
##      sex      age    agesq   income levyplus freepoor freerepa  illness
##     2488        0        0       79     2892     4968     4099     1554
##  actdays   hscore  chcond1  chcond2 doctorco nondocco hospadmi hospdays
##     4454     3026     3098     4585     4141     4716     4491     4491
## medicine prescrib nonpresc
##     2229     3085     3814
```

```r
# Let's check their proportion to dataset as well
round(colSums(dvisits==0)/nrow(dvisits)*100,2)
```

```
##      sex      age    agesq   income levyplus freepoor freerepa  illness
##    47.94     0.00     0.00     1.52    55.72    95.72    78.98    29.94
##  actdays   hscore  chcond1  chcond2 doctorco nondocco hospadmi hospdays
##    85.82    58.30    59.69    88.34    79.79    90.87    86.53    86.53
## medicine prescrib nonpresc
##    42.95    59.44    73.49
```

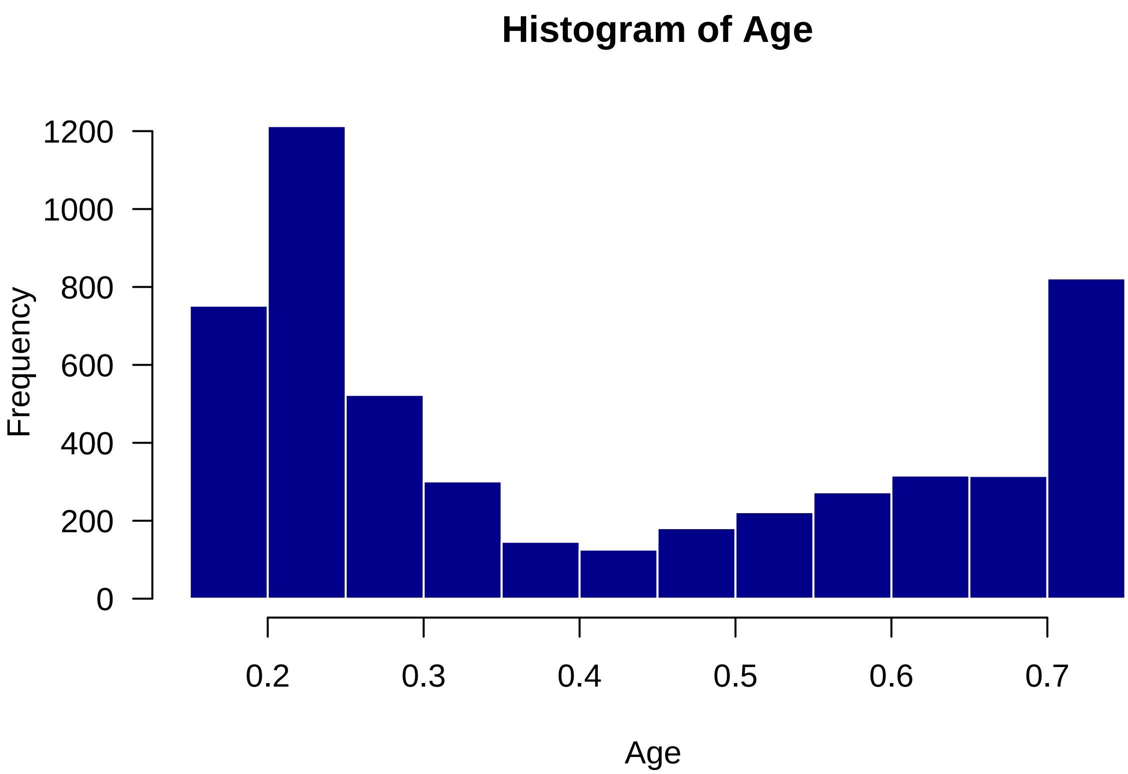
We see some variables like freepoor, actdays, chcond2, nondocco, hospadmi and hospdays with > 85% of values as 0.

Plotting the univariate plots -

We now plot the histogram/ density functions of some numerical variables.

Age

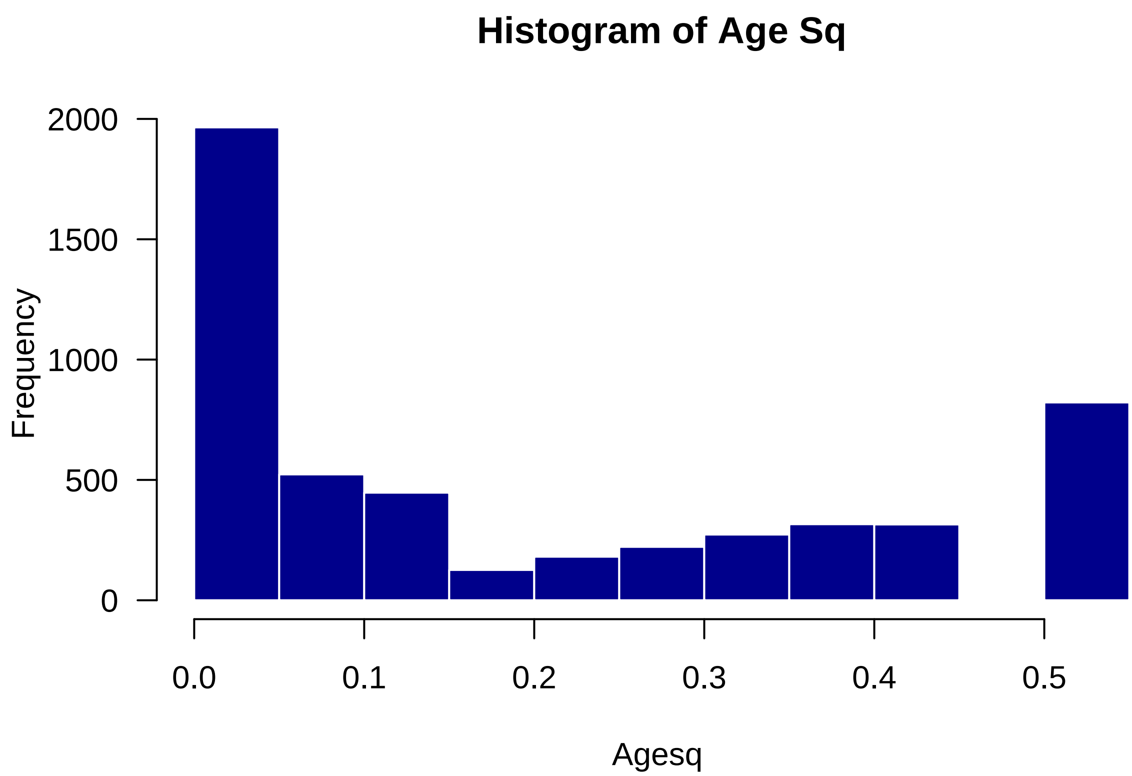
```
hist(dvisits$age, main="Histogram of Age",
     xlab="Age ",
     border="white",
     col="dark blue",
     las=1)
```



We confirm visually that age for young and old seem oversampled as mentioned.

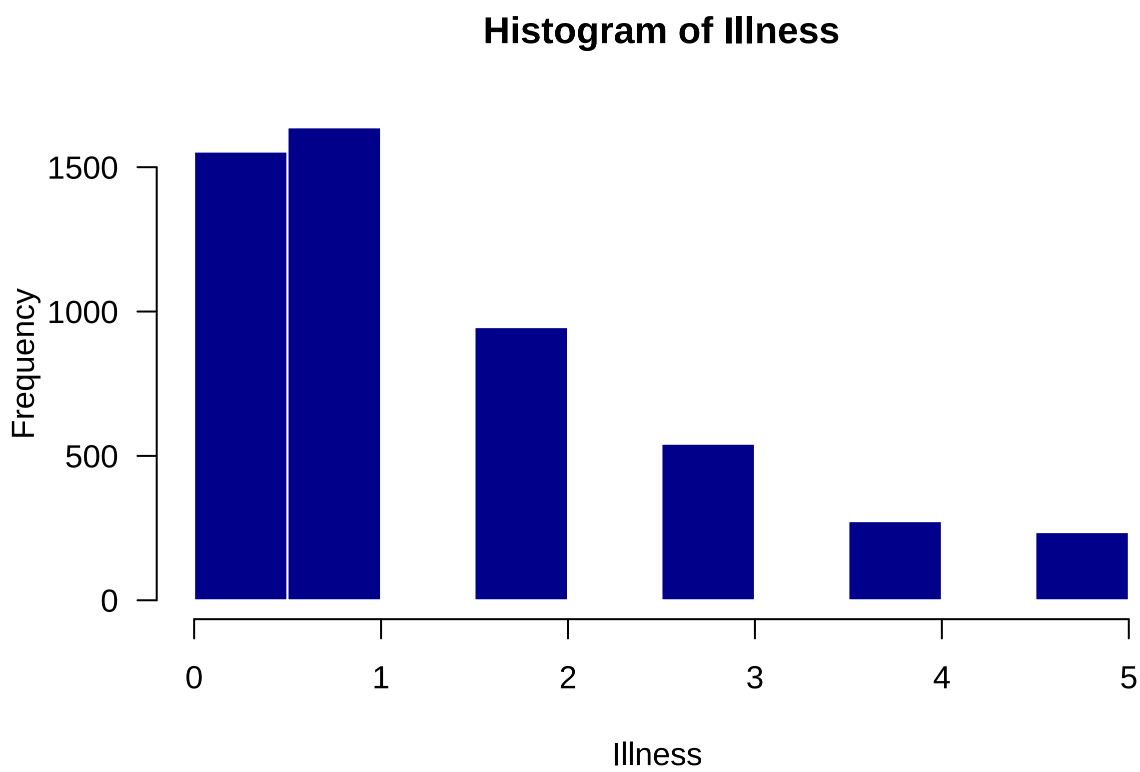
Age sq

```
hist(dvisits$agesq, main="Histogram of Age Sq",
     xlab="Agesq ",
     border="white",
     col="dark blue",
     las=1)
```



Illness

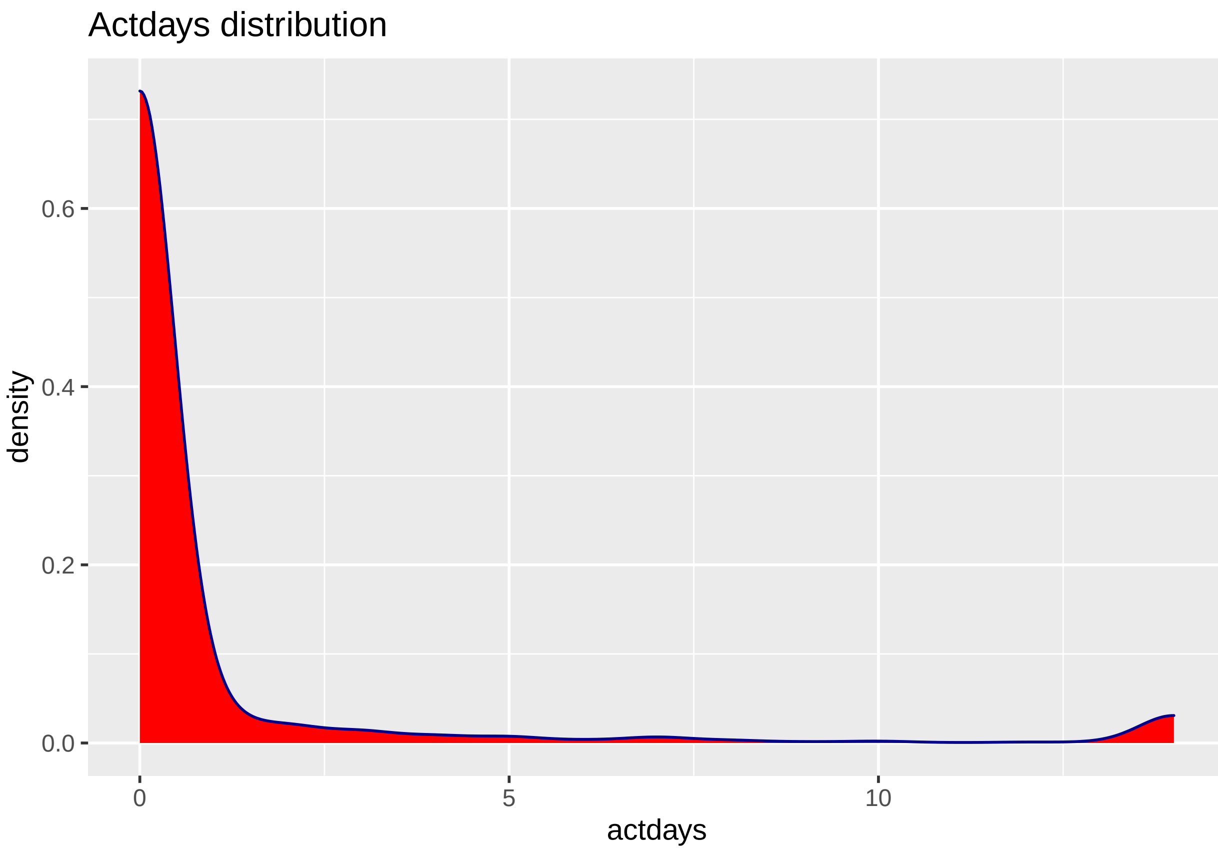
```r
hist(dvisits$illness, main="Histogram of Illness",
     xlab="Illness ",
     border="white",
     col="dark blue",
     las=1)
```



Majority of the illnesses are either 0 or 1 in the past two weeks.

Actdays

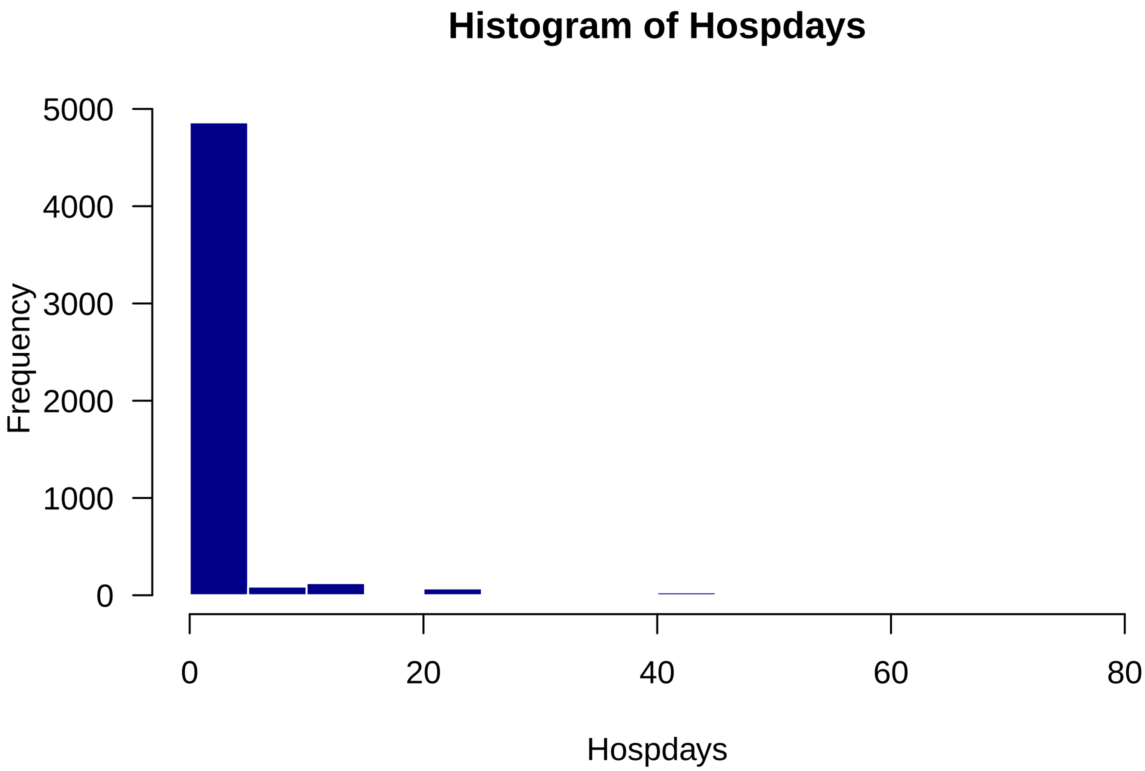
```r
ggplot(dvisits, aes(x=actdays)) +
  geom_density(color="darkblue", fill="red") +
  ggtitle("Actdays distribution")
```



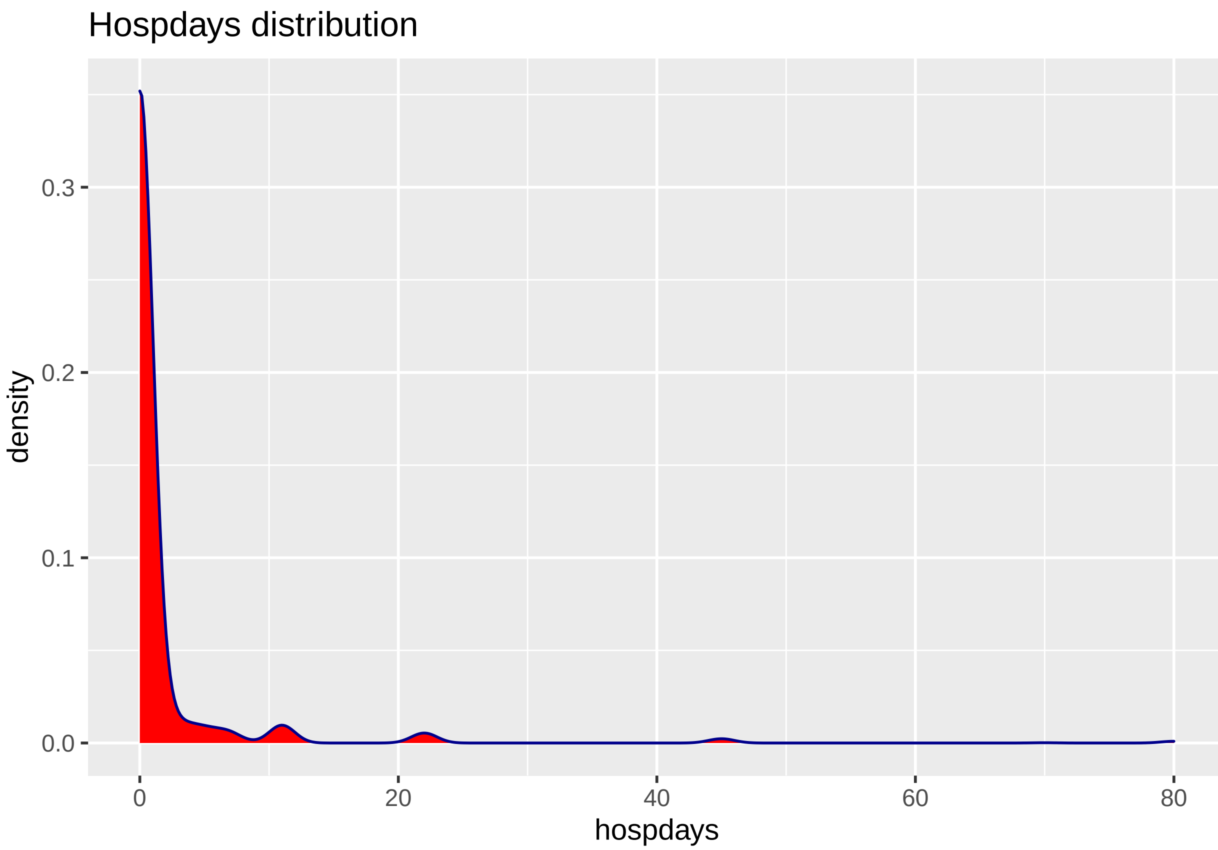
Majority of the days of reduced acitvity are concentrated around 0 with small spike at 15.

Hospdays

```
hist(dvisits$hospdays, main="Histogram of Hospdays",
     xlab="Hospdays",
     border="white",
     col="dark blue",
     las=1)
```



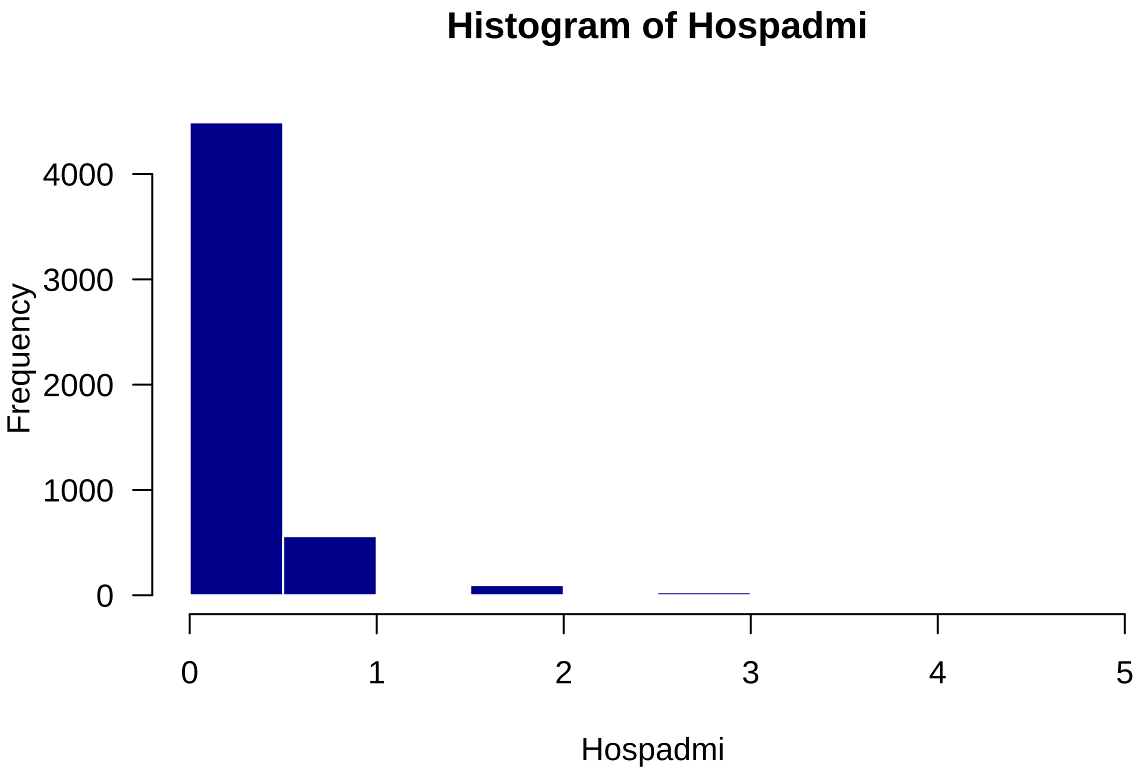

```
ggplot(dvisits, aes(x=hospdays)) +
  geom_density(color="darkblue", fill="red") +
  ggtitle("Hospdays distribution")
```



Most of the number of nights in hospital during recent admission are 0 however we see some outliers above 40 nights. We also note the incredibly skewed distribution and infer that a log transformation may be apt to model this response variable.

Hospadmi

```
hist(dvisits$hospadmi, main="Histogram of Hospadmi",
     xlab="Hospadmi ",
     border="white",
     col="dark blue",
     las=1)
```



Majority of the admissions to hospital were 0, and next 1 admission in the past 12 months.

5a) Using dvisits dataset fit a model with hospdays as response and other variables as potential predictors. Perform regression diagnostics on the model. Display any relevant plots.

We now perform linear regression using all variables in first iteration.

First iteration

We create the train and test split first 85% - 15% and fit the first model.

```
set.seed(123)
dvisits_train <- dvisits %>% sample_frac(.85)
dvisits_test  <- anti_join(dvisits, dvisits_train)
```

```
## Joining, by = c("sex", "age", "agesq", "income", "levyplus", "freepoor", "freerepa", "illness", "act
```

```
fit <- lm(hospdays~age+agesq+income+levyplus+
            freepoor+freerepa+illness+actdays+
            hscore+chcond1+chcond2+doctorco+
            nondocco+hospadmi+medicine+
            prescrib+nonpresc, data=dvisits_train)
summary(fit)
```

```
##
## Call:
## lm(formula = hospdays ~ age + agesq + income + levyplus + freepoor +
##     freerepa + illness + actdays + hscore + chcond1 + chcond2 +
##     doctorco + nondocco + hospadmi + medicine + prescrib + nonpresc,
##     data = dvisits_train)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -24.714  -0.907  -0.031   0.258  74.200
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.36745    0.58664   2.331 0.019799 *
## age         -9.15338    3.36192  -2.723 0.006501 **
## agesq       11.02237    3.76828   2.925 0.003462 **
## income       0.01168    0.26358   0.044 0.964650
## levyplus    -0.06473    0.20049  -0.323 0.746833
## freepoor    -0.23895    0.42161  -0.567 0.570910
## freerepa     0.70152    0.31075   2.258 0.024024 *
## illness     -0.05541    0.07152  -0.775 0.438543
## actdays      0.07613    0.03371   2.258 0.023980 *
## hscore       0.04316    0.04269   1.011 0.312021
## chcond1      0.19275    0.19553   0.986 0.324283
## chcond2      1.04588    0.29998   3.486 0.000494 ***
## doctorco    -0.15755    0.11733  -1.343 0.179417
## nondocco     0.48424    0.08784   5.513 3.73e-08 ***
## hospadmi     5.63065    0.17008  33.105  < 2e-16 ***
## medicine    -0.05606    0.11815  -0.474 0.635176
## prescrib     0.12939    0.13297   0.973 0.330564
## nonpresc          NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.373 on 4395 degrees of freedom
## Multiple R-squared:  0.2661, Adjusted R-squared:  0.2634
## F-statistic:  99.6 on 16 and 4395 DF,  p-value: < 2.2e-16
```

```r
vif(fit)
```

```
## Warning in v1 * v2: longer object length is not a multiple of shorter
## object length
```

```
##        age      agesq     income   levyplus   freepoor   freerepa
##  71.823686  73.975361   1.454113   1.517189   1.153218   2.439540
##    illness    actdays     hscore    chcond1    chcond2   doctorco
##   1.497668   1.378762   1.243777   1.403692   1.422775   1.333345
##   nondocco   hospadmi   medicine   prescrib   nonpresc
##   1.093128   1.147282   5.077233   5.294943 856.338107
```

We see an R-square of 27.5% however we now remove the non-significant variables and the highly correlated ones.

Second iteration

```r
fit <- lm(hospdays~agesq+freerepa+actdays+
            chcond2+nondocco+hospadmi, data=dvisits_train)
summary(fit)
```

```
##
## Call:
```

```
## lm(formula = hospdays ~ agesq + freerepa + actdays + chcond2 +
##     nondocco + hospadmi, data = dvisits_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -24.511  -0.973   0.050   0.243  74.593
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.29750    0.12584  -2.364 0.018118 *
## agesq        1.12011    0.55890   2.004 0.045119 *
## freerepa     0.84043    0.25508   3.295 0.000993 ***
## actdays      0.06760    0.03073   2.200 0.027850 *
## chcond2      0.87063    0.26480   3.288 0.001017 **
## nondocco     0.49973    0.08758   5.706 1.23e-08 ***
## hospadmi     5.65068    0.16650  33.939  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.375 on 4405 degrees of freedom
## Multiple R-squared:  0.2639, Adjusted R-squared:  0.2629
## F-statistic: 263.3 on 6 and 4405 DF,  p-value: < 2.2e-16
```

```r
vif(fit)
```

```
##    agesq freerepa  actdays  chcond2 nondocco hospadmi
## 1.626199 1.642721 1.144529 1.107857 1.085983 1.098645
```

```r
AIC(fit)
```

```
## [1] 27369.21
```

We now see an R-square of 26.3% and our VIF is good (maximum VIF of 1.6 for freerepa) and we have all our variables as significant. Our AIC however is quite large at 27,369.21

We can see that all of our variables have a positive sign indicating that as age, free coverage by govt., days of reduced activity, chronic condition and limited in activity, consultations with non doctor staff and number of admissions (our significant explanatory variables) go up/ increase, consequently the number of nights in most recent admission (our dependent variables) go up/ increase as well.

Third iteration

We now transform our response variables using four methods -

1. log transformation
2. GLM with log link
3. the boxcox transformation by first estimating lambda
4. transforming predictors the brute way

1. Log transform of the dependent variable

```r
fit <- lm(log(hospdays+1)~agesq+freerepa+actdays+
            chcond2+nondocco+hospadmi, data=dvisits_train)
summary(fit)
```

```
##
## Call:
## lm(formula = log(hospdays + 1) ~ agesq + freerepa + actdays +
##     chcond2 + nondocco + hospadmi, data = dvisits_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.8483 -0.0914 -0.0319 -0.0141  3.2612
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.007392   0.010387   0.712 0.476745
## agesq       0.139059   0.046133   3.014 0.002590 **
## freerepa    0.049053   0.021055   2.330 0.019864 *
## actdays     0.007591   0.002536   2.993 0.002777 **
## chcond2     0.043990   0.021857   2.013 0.044219 *
## nondocco    0.024927   0.007229   3.448 0.000569 ***
## hospadmi    1.119091   0.013743  81.431  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4436 on 4405 degrees of freedom
## Multiple R-squared:  0.6379, Adjusted R-squared:  0.6374
## F-statistic:  1293 on 6 and 4405 DF,  p-value: < 2.2e-16
```

```r
vif(fit)
```

```
##    agesq freerepa  actdays  chcond2 nondocco hospadmi
## 1.626199 1.642721 1.144529 1.107857 1.085983 1.098645
```

```r
AIC(fit)
```

```
## [1] 5358.145
```

We see a significant jump in R-square in this iteration to 63.7% which is great as our variables are still all significant. Our AIC comes down to 5,358 in this iteration.

2. GLM with log link

```r
# add a small value to hospdays
glm.mod <- glm(hospdays+0.0001~agesq+freerepa+actdays+
            chcond2+nondocco+hospadmi,
            data=dvisits_train, family=gaussian(link="log"))
summary(glm.mod)
```

```
##
## Call:
## glm(formula = hospdays + 1e-04 ~ agesq + freerepa + actdays +
##     chcond2 + nondocco + hospadmi, family = gaussian(link = "log"),
##     data = dvisits_train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -32.97   -1.80   -0.97   -0.88   78.61
##
```

```
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.213648   0.098809  -2.162 0.030654 *
## agesq        1.785174   0.242820   7.352 2.32e-13 ***
## freerepa     0.328940   0.080702   4.076 4.66e-05 ***
## actdays      0.001327   0.006857   0.193 0.846586
## chcond2      0.332885   0.087205   3.817 0.000137 ***
## nondocco     0.105572   0.010621   9.940  < 2e-16 ***
## hospadmi     0.454820   0.023479  19.371  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 34.24101)
##
##     Null deviance: 172884  on 4411  degrees of freedom
## Residual deviance: 150834  on 4405  degrees of freedom
## AIC: 28119
##
## Number of Fisher Scoring iterations: 14
```

```r
vif(glm.mod)
```

```
##      agesq   freerepa    actdays    chcond2   nondocco   hospadmi
## 0.25896799 0.13872244 0.04808063 0.10136797 0.01347595 0.01843293
```

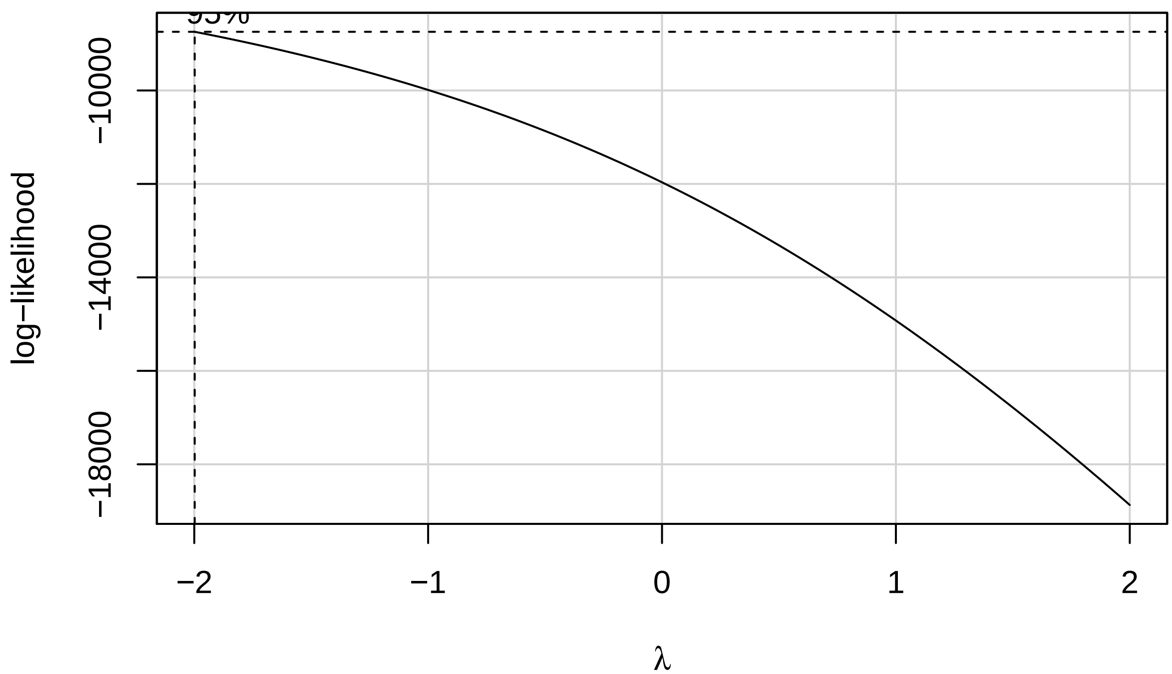
```r
AIC(glm.mod)
```

```
## [1] 28119.23
```

This isn't a great model as actdays isnt signifcant now and AIC has gone back up to 28,119.

3. the boxcox transformation by first estimating lambda

```r
boxCox(fit, family="yjPower", plotit = TRUE)
```



We see -2 as lambda where log likelihood is maximized.

```r
dvisits_train$depvar.transformed <- yjPower(dvisits_train$hospdays, -2)
```

```r
fit <- lm(depvar.transformed~agesq+freerepa+actdays+
            chcond2+nondocco+hospadmi, data=dvisits_train)
summary(fit)
```

```
##
## Call:
## lm(formula = depvar.transformed ~ agesq + freerepa + actdays +
##     chcond2 + nondocco + hospadmi, data = dvisits_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.89325 -0.01540 -0.01525 -0.01393  0.22071
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0155565  0.0018648   8.342   <2e-16 ***
## agesq       -0.0042253  0.0082819  -0.510    0.610
## freerepa     0.0057251  0.0037799   1.515    0.130
## actdays     -0.0002077  0.0004553  -0.456    0.648
## chcond2     -0.0021882  0.0039239  -0.558    0.577
## nondocco    -0.0002189  0.0012978  -0.169    0.866
## hospadmi     0.2733516  0.0024672 110.796   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07964 on 4405 degrees of freedom
## Multiple R-squared:  0.7533, Adjusted R-squared:  0.7529
## F-statistic:  2242 on 6 and 4405 DF,  p-value: < 2.2e-16
```

```r
vif(fit)
```

```
##    agesq freerepa  actdays  chcond2 nondocco hospadmi
## 1.626199 1.642721 1.144529 1.107857 1.085983 1.098645
```

```r
AIC(fit)
```

```
## [1] -9796.627
```

This model doesn't work as most of our predictors are now not signifcant.

4. Transforming the predictors in a brute force way

This method is used in applications in the real world to understand if a transformation of a predictor variable works better on the response. Let's see how this works.

We use multiple transformations of square, cube, log, inverse, inverse squared. We do not use sqrt here but for exposition, this method will work well.

```r
# we first add an insignificant value to variables which have a 0 in them
dvisits$actdays <- dvisits$actdays+0.00001
dvisits$nondocco <- dvisits$nondocco+0.00001
dvisits$hospadmi <- dvisits$hospadmi+0.00001

fit <- lm(log(hospdays+1) ~ agesq+I(agesq^2) +
            I(agesq^3) + I(agesq^-1) + I(agesq ^-2)
          + I(log(agesq+1))+
            freerepa+
actdays+I(actdays^2) + I(actdays^3) + I(actdays^-1)
+ I(actdays ^-2) + I(log(actdays+1))+
nondocco+I(nondocco^2) + I(nondocco^3) +
  I(nondocco^-1) + I(nondocco ^-2) + I(log(nondocco+1))+
hospadmi+I(hospadmi^2) + I(hospadmi^3) +
  I(hospadmi^-1) + I(log(hospadmi+1))+
chcond2
  , data=dvisits)
summary(fit)
```

```
##
## Call:
## lm(formula = log(hospdays + 1) ~ agesq + I(agesq^2) + I(agesq^3) +
##     I(agesq^-1) + I(agesq^-2) + I(log(agesq + 1)) + freerepa +
##     actdays + I(actdays^2) + I(actdays^3) + I(actdays^-1) + I(actdays^-2) +
##     I(log(actdays + 1)) + nondocco + I(nondocco^2) + I(nondocco^3) +
##     I(nondocco^-1) + I(nondocco^-2) + I(log(nondocco + 1)) +
##     hospadmi + I(hospadmi^2) + I(hospadmi^3) + I(hospadmi^-1) +
##     I(log(hospadmi + 1)) + chcond2, data = dvisits)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.49840 -0.04243  0.02547  0.03663  2.61416
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           1.439e+01  8.088e+00   1.779 0.075361 .
## agesq                -9.073e+01  5.941e+02  -0.153 0.878639
```

```
## I(agesq^2)            4.272e+01  2.552e+02   0.167 0.867069    
## I(agesq^3)           -1.653e+01  8.481e+01  -0.195 0.845454    
## I(agesq^-1)           4.583e-03  4.324e-02   0.106 0.915601    
## I(agesq^-2)          -1.709e-04  6.961e-04  -0.245 0.806094    
## I(log(agesq + 1))     9.085e+01  6.035e+02   0.151 0.880342    
## freerepa              2.556e-02  1.438e-02   1.777 0.075608 .  
## actdays              -7.749e-02  1.140e+00  -0.068 0.945816    
## I(actdays^2)          1.068e-02  7.329e-02   0.146 0.884195    
## I(actdays^3)         -3.857e-04  2.060e-03  -0.187 0.851502    
## I(actdays^-1)        -8.833e-02  1.050e+00  -0.084 0.932937    
## I(actdays^-2)         8.833e-07  1.050e-05   0.084 0.932936    
## I(log(actdays + 1))   5.361e-02  3.534e+00   0.015 0.987899    
## nondocco              6.676e+00  4.085e+00   1.634 0.102259    
## I(nondocco^2)        -5.735e-01  3.054e-01  -1.878 0.060453 .  
## I(nondocco^3)         2.255e-02  1.033e-02   2.184 0.029035 *  
## I(nondocco^-1)       -3.689e+00  2.761e+00  -1.336 0.181493    
## I(nondocco^-2)        3.689e-05  2.760e-05   1.336 0.181493    
## I(log(nondocco + 1)) -1.682e+01  1.135e+01  -1.483 0.138208    
## hospadmi              2.848e+01  1.500e+01   1.898 0.057709 .  
## I(hospadmi^2)        -5.054e+00  2.309e+00  -2.189 0.028632 *  
## I(hospadmi^3)         4.100e-01  1.652e-01   2.483 0.013075 *  
## I(hospadmi^-1)       -5.121e-05  2.949e-05  -1.737 0.082518 .  
## I(log(hospadmi + 1)) -3.916e+01  2.279e+01  -1.718 0.085870 .  
## chcond2               5.135e-02  1.495e-02   3.435 0.000598 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.3251 on 5164 degrees of freedom
## Multiple R-squared:  0.8043, Adjusted R-squared:  0.8034 
## F-statistic: 849.1 on 25 and 5164 DF,  p-value: < 2.2e-16
```

```r
vif(fit)
```

```
##                agesq           I(agesq^2)           I(agesq^3) 
##         5.972709e+08         3.260018e+07         9.427332e+05 
##          I(agesq^-1)          I(agesq^-2)    I(log(agesq + 1)) 
##         8.422300e+03         1.764345e+03         3.929351e+08 
##             freerepa              actdays         I(actdays^2) 
##         1.686289e+00         5.322423e+05         3.803704e+05 
##         I(actdays^3)        I(actdays^-1)        I(actdays^-2) 
##         5.700487e+04         6.584067e+13         6.583817e+13 
##  I(log(actdays + 1))             nondocco        I(nondocco^2) 
##         2.595313e+05         7.634037e+05         2.214618e+05 
##        I(nondocco^3)       I(nondocco^-1)       I(nondocco^-2) 
##         1.812712e+04         3.105623e+14         3.105514e+14 
## I(log(nondocco + 1))             hospadmi        I(hospadmi^2) 
##         7.310799e+05         2.846487e+06         5.280952e+05 
##        I(hospadmi^3)       I(hospadmi^-1) I(log(hospadmi + 1)) 
##         4.501359e+04         4.976046e+04         2.067991e+06 
##              chcond2 
##         1.130557e+00
```

Given we fitted all transformations its time to use only the significant ones but with good coefficient from the above result. We do this below.

```r
fit <- lm(log(hospdays+1) ~  I(log(agesq+1))+
freerepa+
I(actdays ^-2) + actdays +
nondocco +
I(hospadmi^3) + hospadmi +
chcond2
  , data=dvisits)
summary(fit)
```

```
##
## Call:
## lm(formula = log(hospdays + 1) ~ I(log(agesq + 1)) + freerepa +
##     I(actdays^-2) + actdays + nondocco + I(hospadmi^3) + hospadmi +
##     chcond2, data = dvisits)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.65713 -0.06173  0.00261  0.02054  2.92252
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -7.456e-02  2.148e-02  -3.472 0.000521 ***
## I(log(agesq + 1))  1.556e-01  4.418e-02   3.523 0.000430 ***
## freerepa           3.900e-02  1.610e-02   2.422 0.015461 *
## I(actdays^-2)      4.665e-12  2.182e-12   2.138 0.032533 *
## actdays            1.529e-02  2.723e-03   5.616 2.06e-08 ***
## nondocco           2.374e-02  5.532e-03   4.291 1.81e-05 ***
## I(hospadmi^3)     -5.711e-02  1.199e-03 -47.624  < 2e-16 ***
## hospadmi           1.550e+00  1.402e-02 110.541  < 2e-16 ***
## chcond2            5.487e-02  1.677e-02   3.272 0.001073 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3694 on 5181 degrees of freedom
## Multiple R-squared:  0.7465, Adjusted R-squared:  0.7461
## F-statistic:  1907 on 8 and 5181 DF,  p-value: < 2.2e-16
```

```r
vif(fit)
```

```
## I(log(agesq + 1))          freerepa     I(actdays^-2)           actdays
##          1.630753          1.636711          2.202593          2.351608
##          nondocco     I(hospadmi^3)          hospadmi           chcond2
##          1.084385          1.837577          1.924546          1.100910
```

```r
AIC(fit)
```

```
## [1] 4403.006
```

We are further able to improve our model R-square to 74.6% by using the above transformations and our AIC of the new model is 4403 (a significant improvement). This is great as our new model equation includes the log transformation of agesq, inverse squared of actdays, cube of hospadmi in addition to previous significant variables.

We will stop here for now and solve the questions asked before proceeding to more complex models which will be done at the end as continued third iteration onwards.

We note our two models as -

1) base model

```r
base_model <- lm(hospdays~agesq+freerepa+actdays+
            chcond2+nondocco+hospadmi, data=dvisits_train)
summary(base_model)
```

```
##
## Call:
## lm(formula = hospdays ~ agesq + freerepa + actdays + chcond2 +
##     nondocco + hospadmi, data = dvisits_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -24.511  -0.973   0.050   0.243  74.593
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.29750    0.12584  -2.364 0.018118 *
## agesq        1.12011    0.55890   2.004 0.045119 *
## freerepa     0.84043    0.25508   3.295 0.000993 ***
## actdays      0.06760    0.03073   2.200 0.027850 *
## chcond2      0.87063    0.26480   3.288 0.001017 **
## nondocco     0.49973    0.08758   5.706 1.23e-08 ***
## hospadmi     5.65068    0.16650  33.939  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.375 on 4405 degrees of freedom
## Multiple R-squared:  0.2639, Adjusted R-squared:  0.2629
## F-statistic: 263.3 on 6 and 4405 DF,  p-value: < 2.2e-16
```

```r
vif(base_model)
```

```
##    agesq freerepa  actdays  chcond2 nondocco hospadmi
## 1.626199 1.642721 1.144529 1.107857 1.085983 1.098645
```

```r
AIC(base_model)
```

```
## [1] 27369.21
```

2) transformed model

```r
transformed_model <- lm(log(hospdays+1) ~  I(log(agesq+1))+
freerepa+
I(actdays ^-2) + actdays +
nondocco +
I(hospadmi^3) + hospadmi +
chcond2
  , data=dvisits)
summary(transformed_model)
```

```
##
## Call:
## lm(formula = log(hospdays + 1) ~ I(log(agesq + 1)) + freerepa +
##     I(actdays^-2) + actdays + nondocco + I(hospadmi^3) + hospadmi +
##     chcond2, data = dvisits)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.65713 -0.06173  0.00261  0.02054  2.92252
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -7.456e-02  2.148e-02  -3.472 0.000521 ***
## I(log(agesq + 1))  1.556e-01  4.418e-02   3.523 0.000430 ***
## freerepa           3.900e-02  1.610e-02   2.422 0.015461 *
## I(actdays^-2)      4.665e-12  2.182e-12   2.138 0.032533 *
## actdays            1.529e-02  2.723e-03   5.616 2.06e-08 ***
## nondocco           2.374e-02  5.532e-03   4.291 1.81e-05 ***
## I(hospadmi^3)     -5.711e-02  1.199e-03 -47.624  < 2e-16 ***
## hospadmi           1.550e+00  1.402e-02 110.541  < 2e-16 ***
## chcond2            5.487e-02  1.677e-02   3.272 0.001073 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3694 on 5181 degrees of freedom
## Multiple R-squared:  0.7465, Adjusted R-squared:  0.7461
## F-statistic:  1907 on 8 and 5181 DF,  p-value: < 2.2e-16
```

```r
vif(transformed_model)
```

```
## I(log(agesq + 1))          freerepa     I(actdays^-2)           actdays
##          1.630753          1.636711          2.202593          2.351608
##          nondocco     I(hospadmi^3)          hospadmi           chcond2
##          1.084385          1.837577          1.924546          1.100910
```

```r
AIC(transformed_model)
```

```
## [1] 4403.006
```

Note : We answer the below questions now from the point of view of the base model

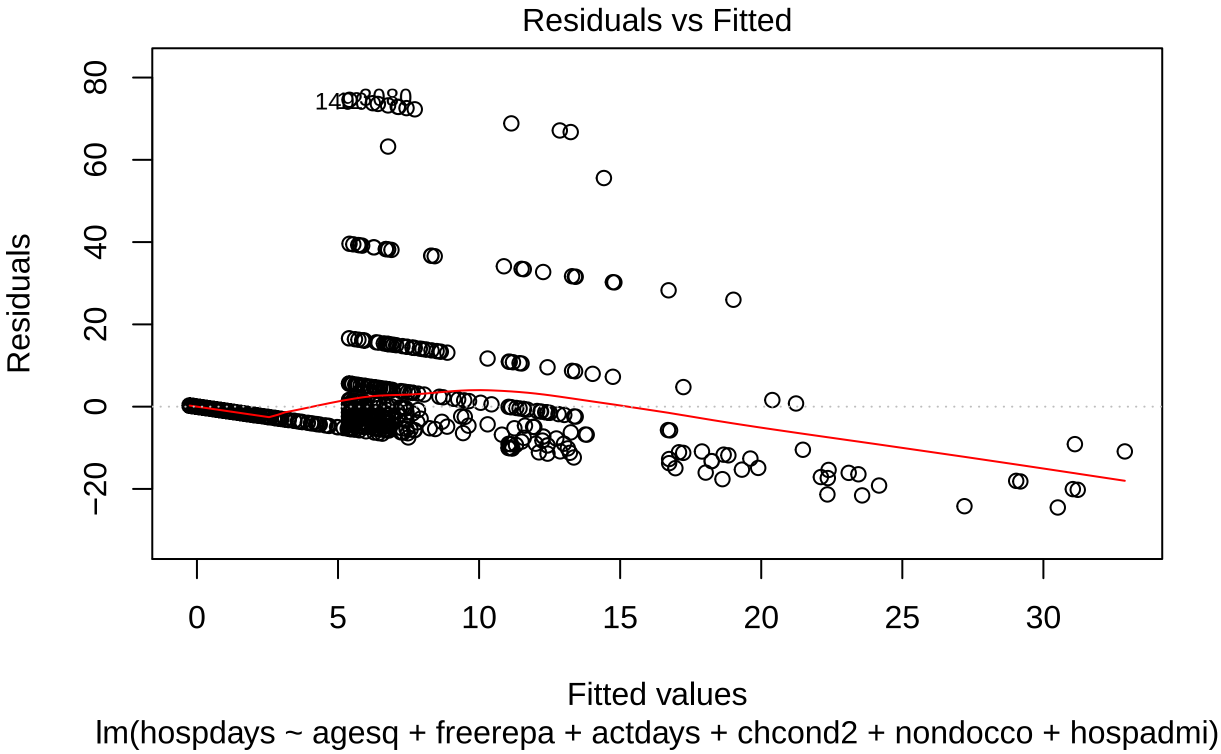
5b) Why is your model a good/ reasonable model ? Check the constant variance assumption for errors.

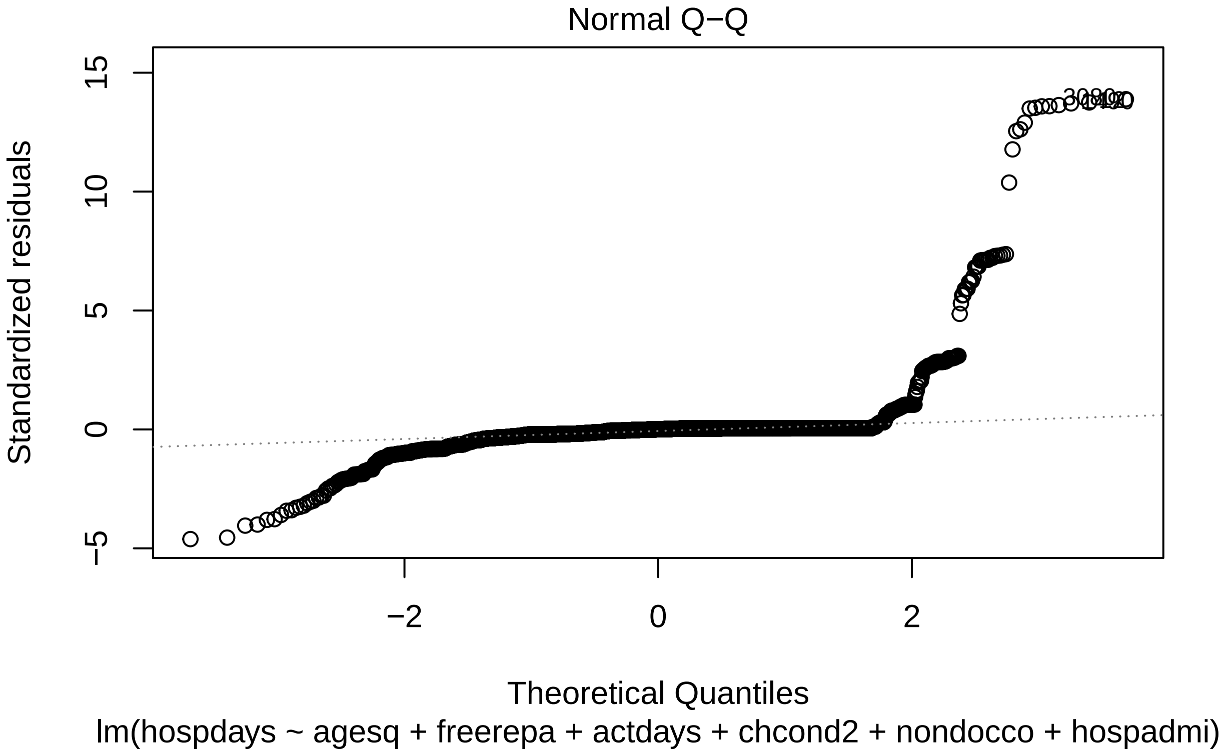
We explore this question first for our base model -

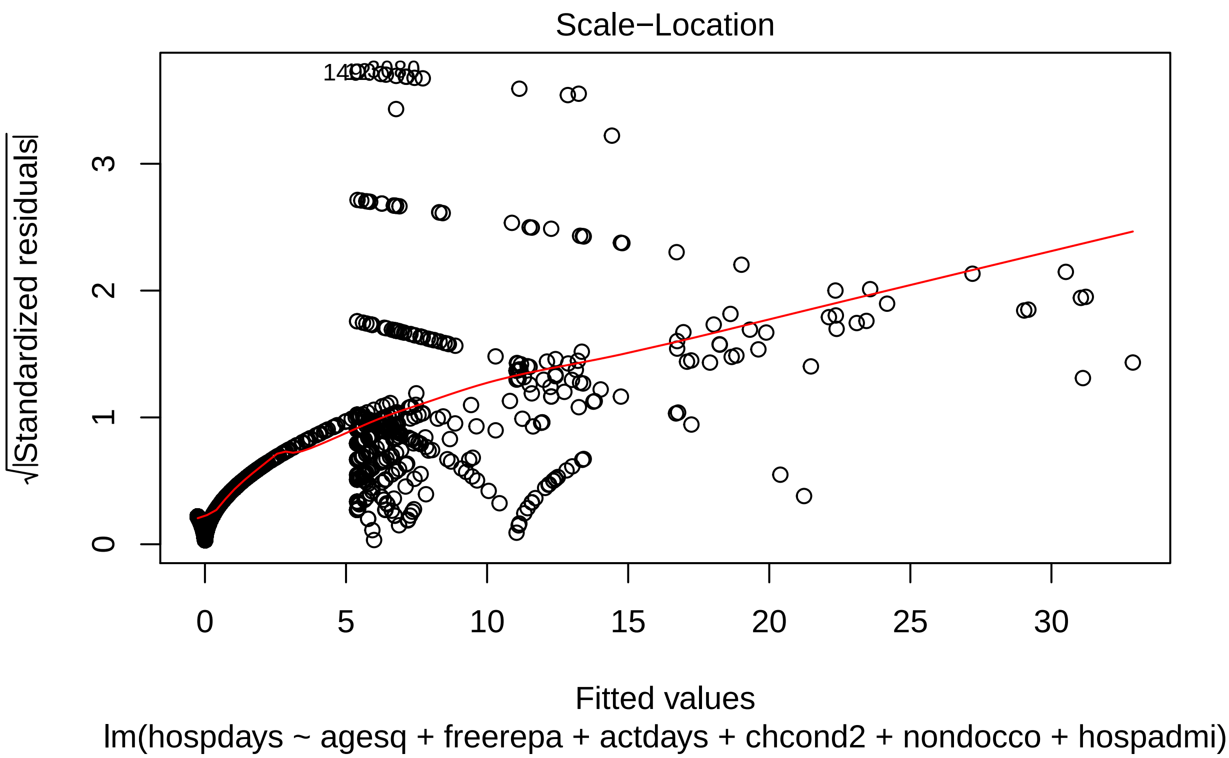
We see an R-square of 26.5% already and we saw our VIFs are good and indicate absence of multicollinearity between the variables. So we can now plot the model.

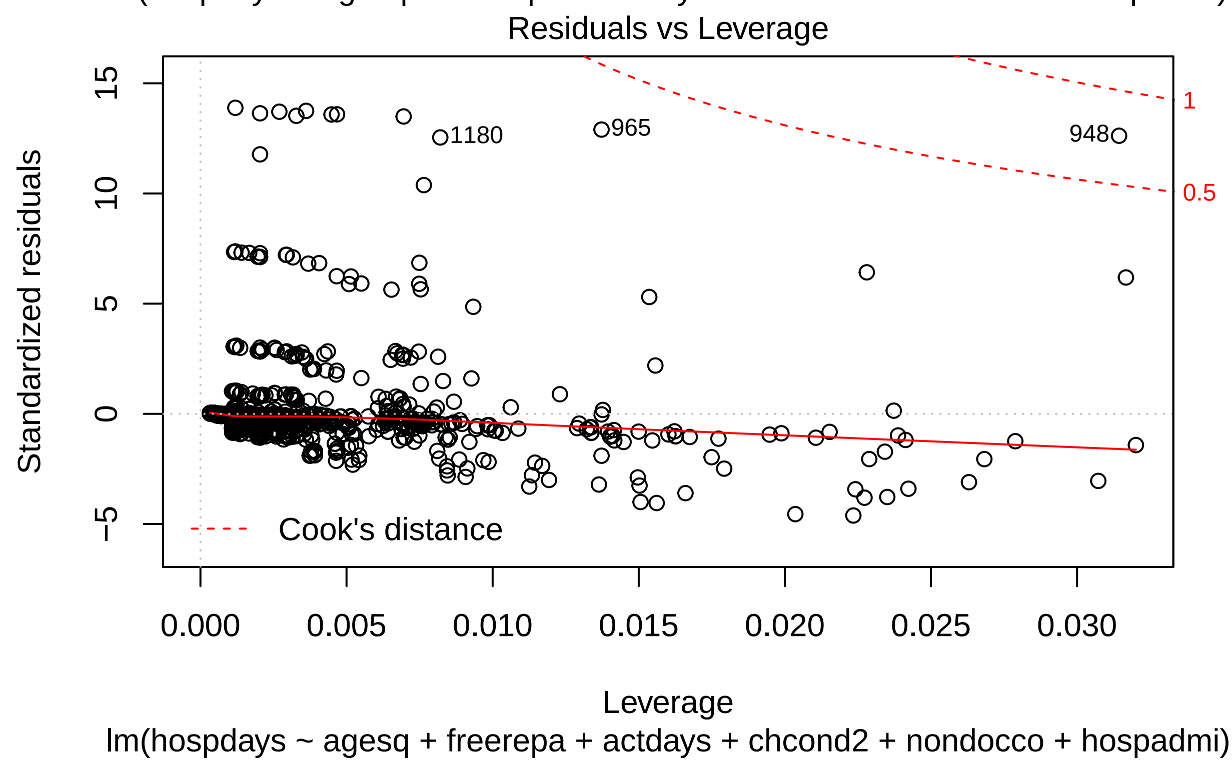
Let's plot the results

```r
plot(base_model)
```


Residuals vs Fitted
Residuals
Fitted values
lm(hospdays ~ agesq + freerepa + actdays + chcond2 + nondocco + hospadmi)


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(hospdays ~ agesq + freerepa + actdays + chcond2 + nondocco + hospadmi)





Checking constant variance assumption

We can see visually from the scale-location plot that the residuals increase with fitted values indicating heteroskedasticity.

ncvTest() For Homoscedasticity

```
ncvTest(base_model)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 13380.21, Df = 1, p = < 2.22e-16
```

We see a p-value $< .05$, indicating heteroscedasticity.

Breusch-Pagan Test For Homoscedasticity

```
bptest(base_model)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  base_model
## BP = 286.1, df = 6, p-value < 2.2e-16
```

We once again see a p-value $< .05$, indicating heteroscedasticity.

Let's predict the model and compute accuracies

```
#Create the evaluation metrics function

eval_metrics = function(model, df, predictions, target){
  resids = df[,target] - predictions
  resids2 = resids**2
  N = length(predictions)
  r2 = as.character(round(summary(model)$r.squared, 2))
  adj_r2 = as.character(round(summary(model)$adj.r.squared, 2))
  print(adj_r2) #Adjusted R-squared
  print(as.character(round(sqrt(sum(resids2)/N), 2))) #RMSE
}
```

```
#Predicting and evaluating the model on train data
predictions = predict(base_model, newdata = dvisits_train)
eval_metrics(base_model, dvisits_train, predictions, target = 'hospdays')
```

```
## [1] "0.26"
## [1] "5.37"
```

```
#Predicting and evaluating the model on test data
pred <- predict(base_model,dvisits_test)
rmse <- sqrt(sum((pred - dvisits_test$hospdays)^2)/length(dvisits_test$hospdays))
mape <- mean(abs(dvisits_test$hospdays - pred)/(dvisits_test$hospdays+0.01))
med_ape <- median(abs(dvisits_test$hospdays - pred)/(dvisits_test$hospdays+0.01))
c(R2=summary(base_model)$r.squared,RMSE = rmse, MAPE = mape, MED_APE=med_ape)
```

```
##        R2      RMSE       MAPE    MED_APE
##  0.2639369  4.9167414 52.8941722 24.3289435
```

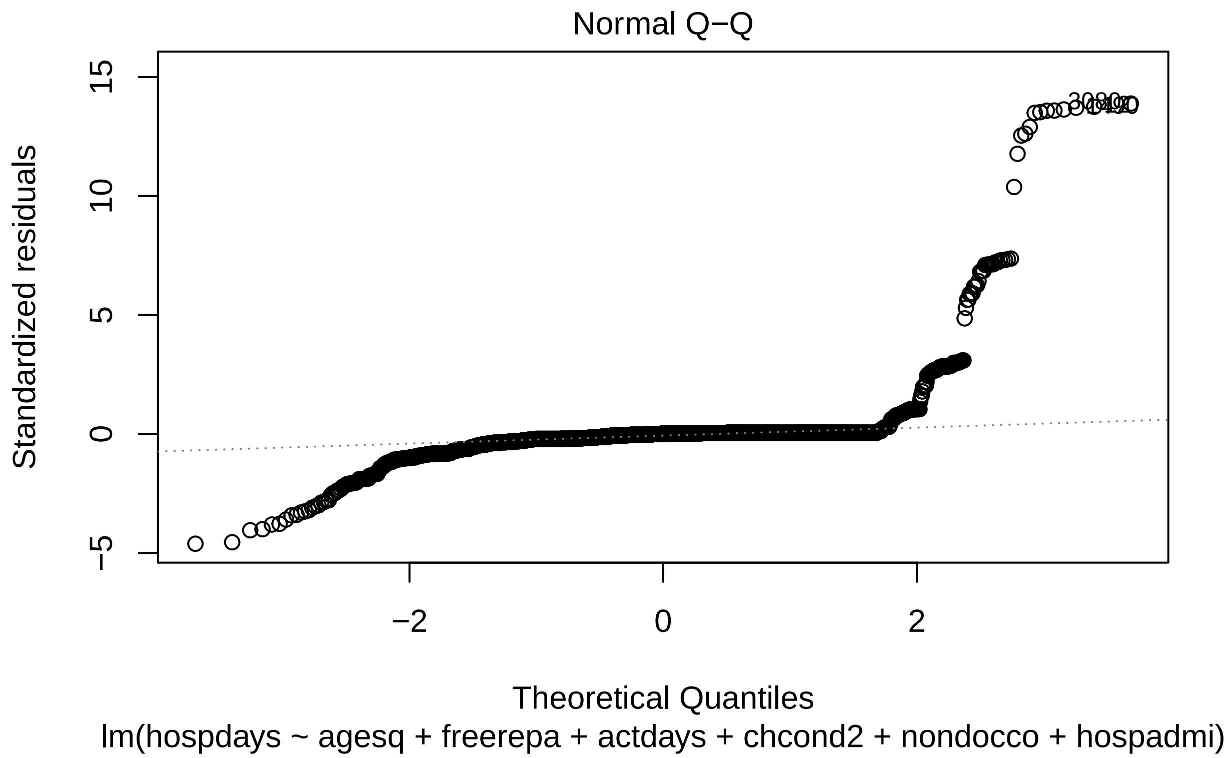
We see that our base model gives R-square of 26.3% and a MAPE of 5289% with median mape of 2432% and an RMSE of 4.9. Our AIC was very large at 27369.21 We violate the homoskedasticity assumption as well. Clearly, we do not feel this is not a good model.

5c) Check the normality assumption

We again do this in 2 ways - we look at QQ plot and perform the Shapiro Wilk normality test.

The normal probability plot of residuals should approximately follow a straight line.

```
plot(base_model,which=2)
```



We see points drastically falling away from the normal line.

Shapiro-Wilk Normality Test

```
resid <- studres(base_model)
shapiro.test(resid)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid
## W = 0.30906, p-value < 2.2e-16
```

From the p-value $< 2.2e-16 < 0.05$, we can see that the residuals are not normally distributed.

5d) Are the errors correlated ?

We can do this with the durbin watson statistic. The Durbin Watson examines whether the errors are autocorrelated with themselves. The null states that they are not autocorrelated.

```
durbinWatsonTest(base_model)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1    -0.003720016      2.007439   0.924
##  Alternative hypothesis: rho != 0
```

We see that p-value $= 0.924 > 0.05$ so the errors are not autocorrelated.

5e) Check for leverage points, outliers and influential points.

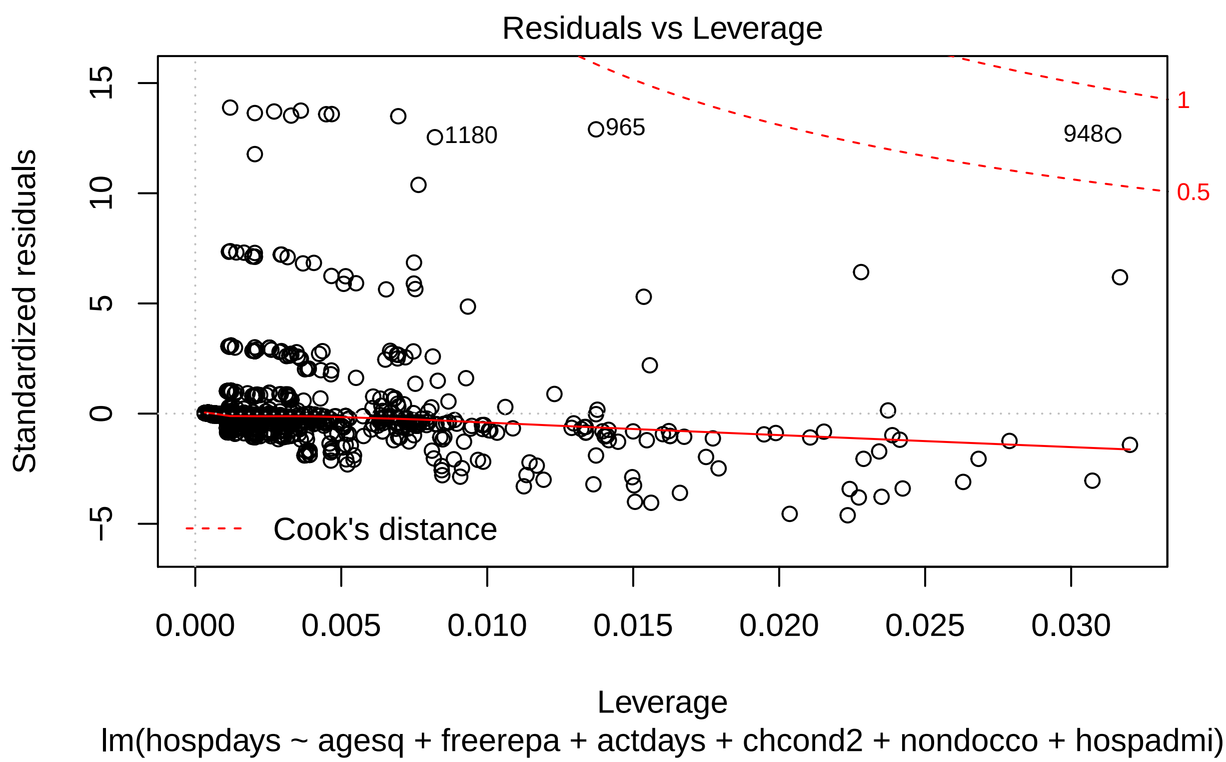
A standard way to check for outliers is to look at residuals above a certain threshold. An example would be as follows -

```
rstandard(base_model)[abs(rstandard(base_model)) > 13]
```

```
##       12      304      852     2050     2684      140     1490     3080
## 13.74931 13.58528 13.63784 13.52447 13.49482 13.59169 13.70961 13.88653
```

Here, we see points 12, 304, 852, 2050, 2684, 140, 1490 and 3080 with large residuals but note that not all of them or maybe none of them could be outliers. So we now look at the model plot of Residuals vs leverage.

```
plot(base_model, which=5)
```



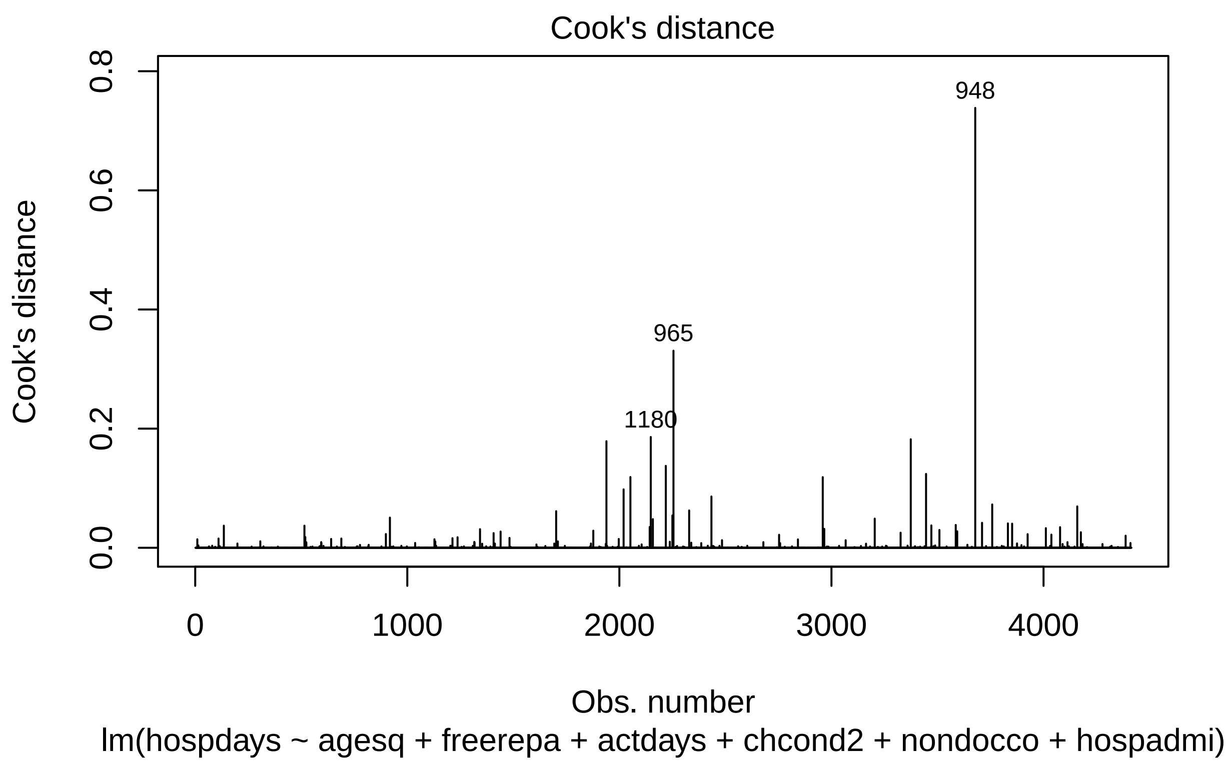
Leverage statistic is defined as -

$\hat{L} = \dfrac{2(p+1)}{n}$ where p is number of predictors and n is number of observations

In the above graph we see all points fall under the dashed lines of the cook's distance apart from point 948 indicating we clearly have one outlier for sure this time. We also see some points of high leverage and slightly higher residuals on the right corner as well. These however could be influential points.

We can plot the cook's distance with the below command -

```
#Cook's distance
plot(base_model, 4)
```



We see that apart from three points - 948, 965 & 1180, everyone else's cook's distance is below 0.2

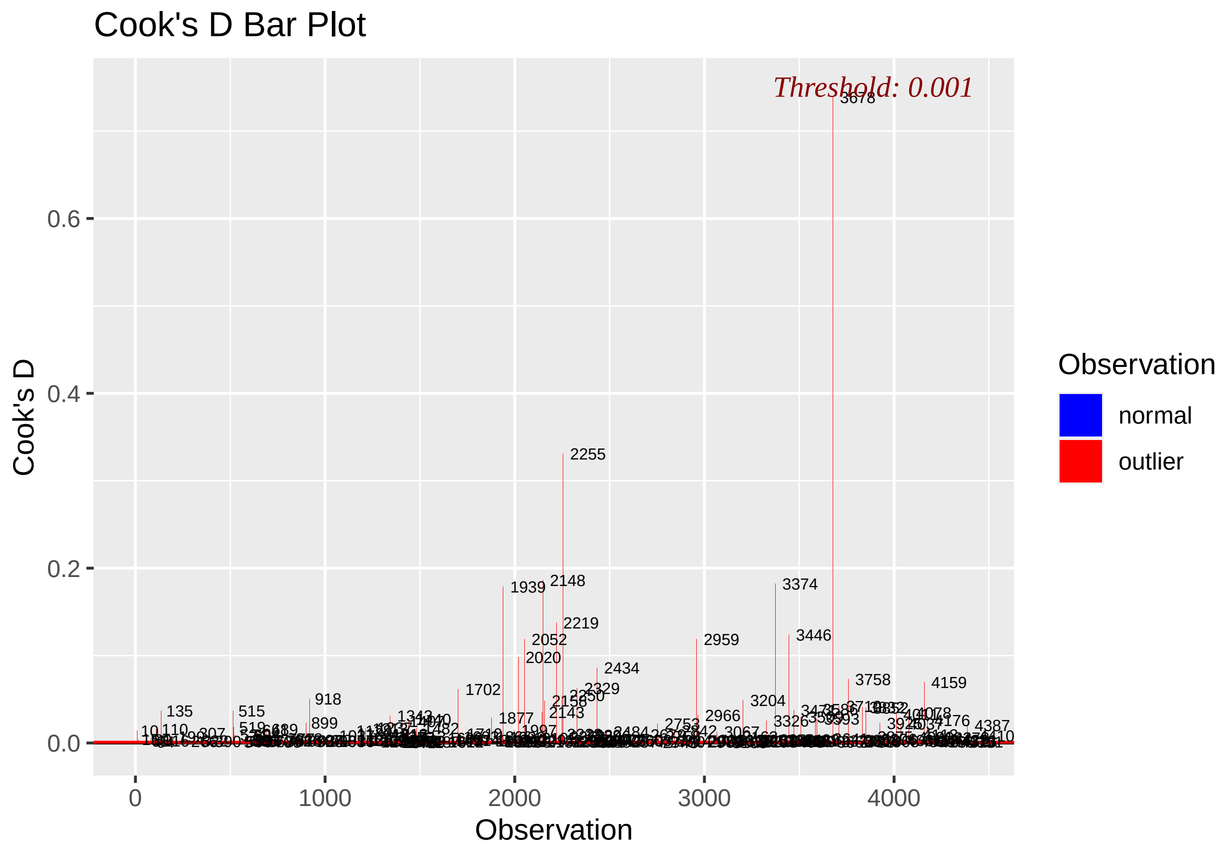
We claim the other two points as influential points.

A rule of thumb is that an observation has high influence if Cook's distance exceeds $\frac{4}{(n-p-1)}$

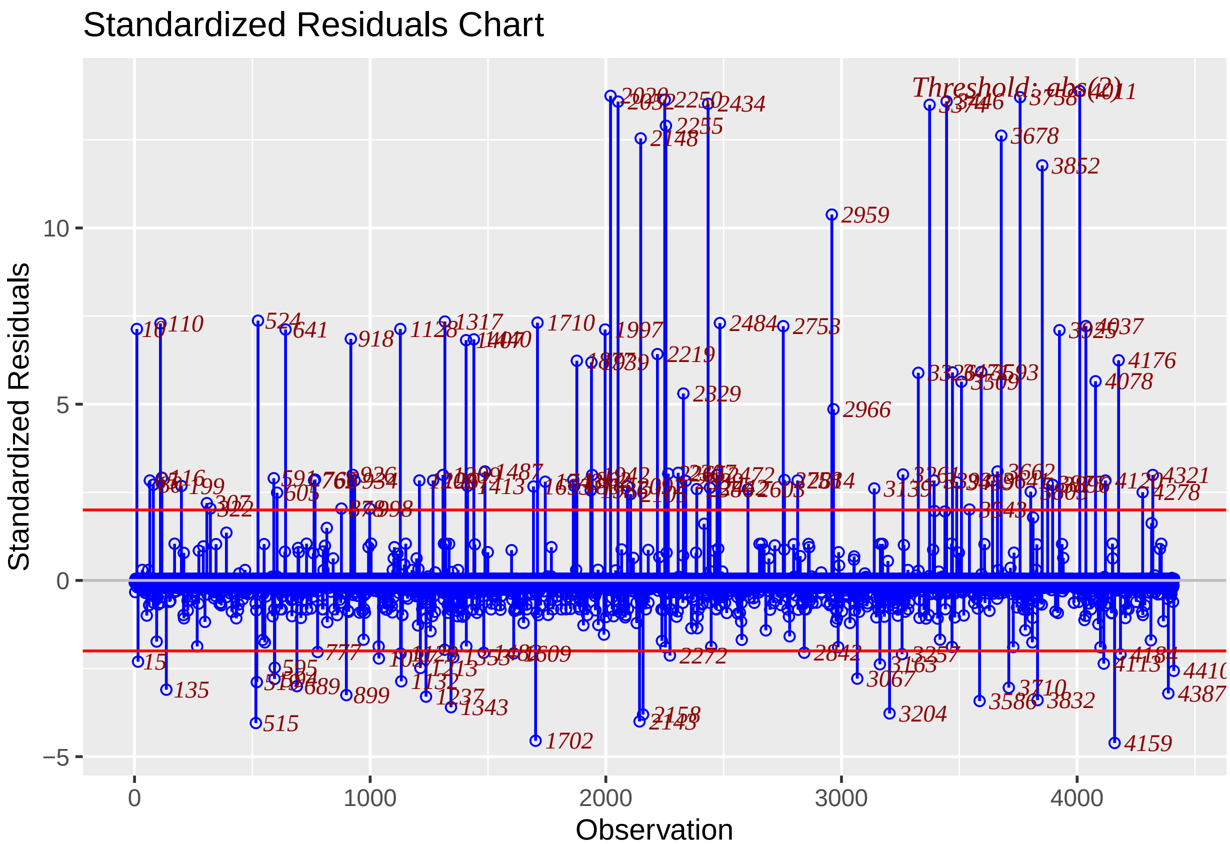
Note that given the size of the data, we do not compute cooks distance and hatvalues like we did earlier but of course this can be done as well.

We plot the results slightly better now -

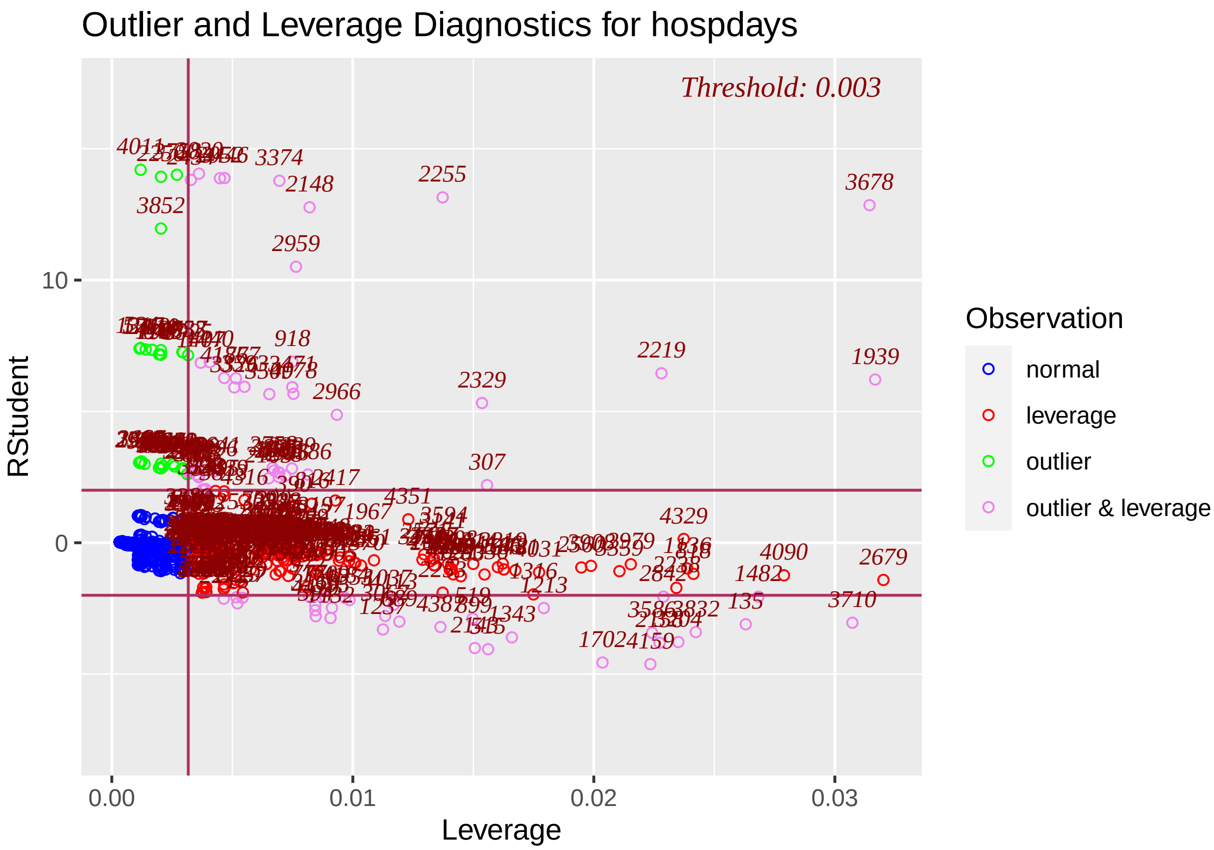
```
ols_plot_cooksd_bar(base_model)
```




```r
ols_plot_resid_stand(base_model)
```




```r
ols_plot_resid_lev(base_model)
```

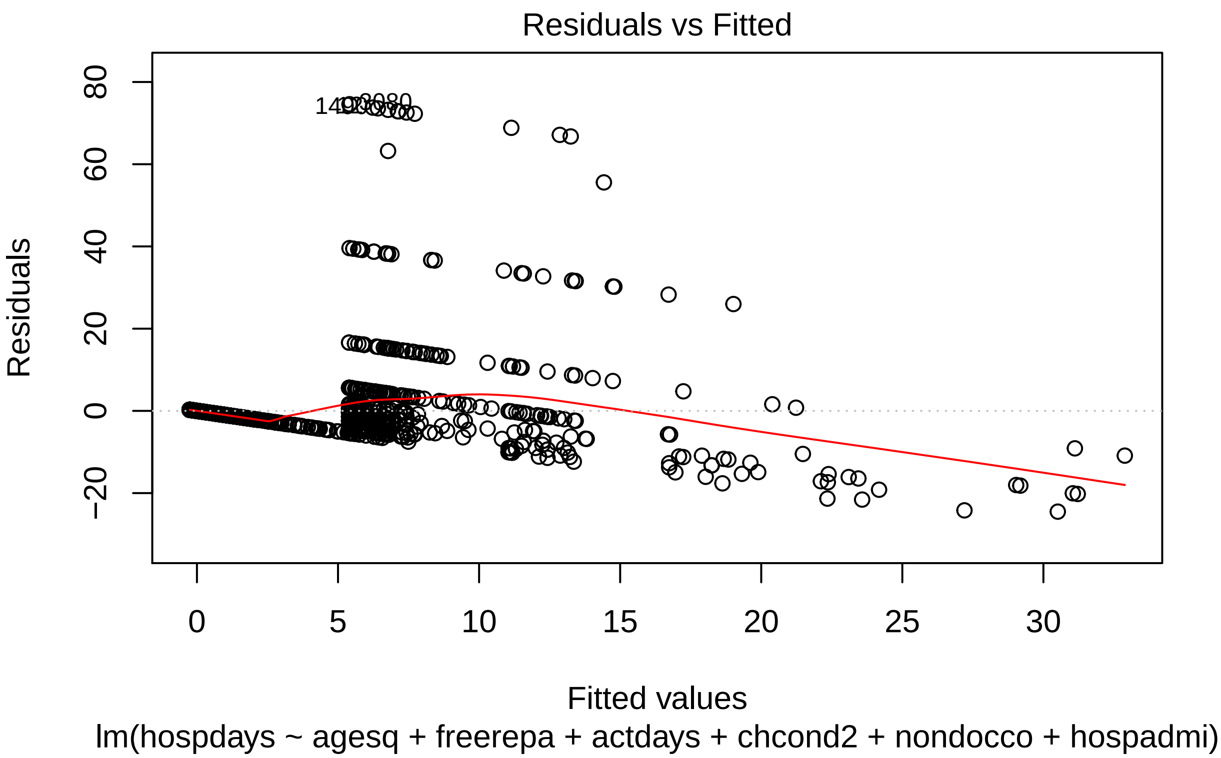


5f) Check the structure of the relationship between the predictors and the response.

This is interesting question, because while creating our transformed model we were able to partially answer this question. Refer to 5a) to see our transformed model.

Let us first check the linearity assumption in our base model.

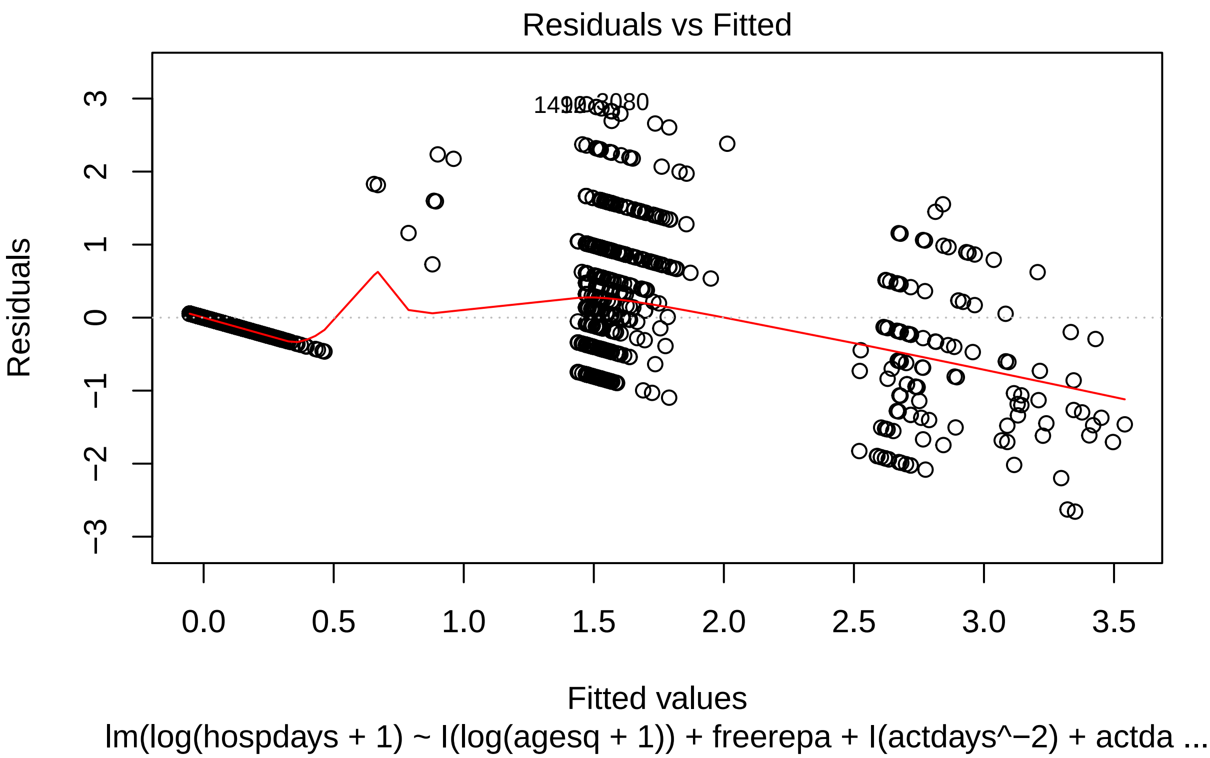
```
plot(base_model,which=1)
```



In this plot, we clearly see a pattern for residuals We see them decreasing from 0 to 3 and above 10 (fitted values) and increasing below 10 and above 3 (fitted values). This indicates we don't have linear relationship between our dependent and independent variables.

Let us also check the linearity assumption in our transformed model.

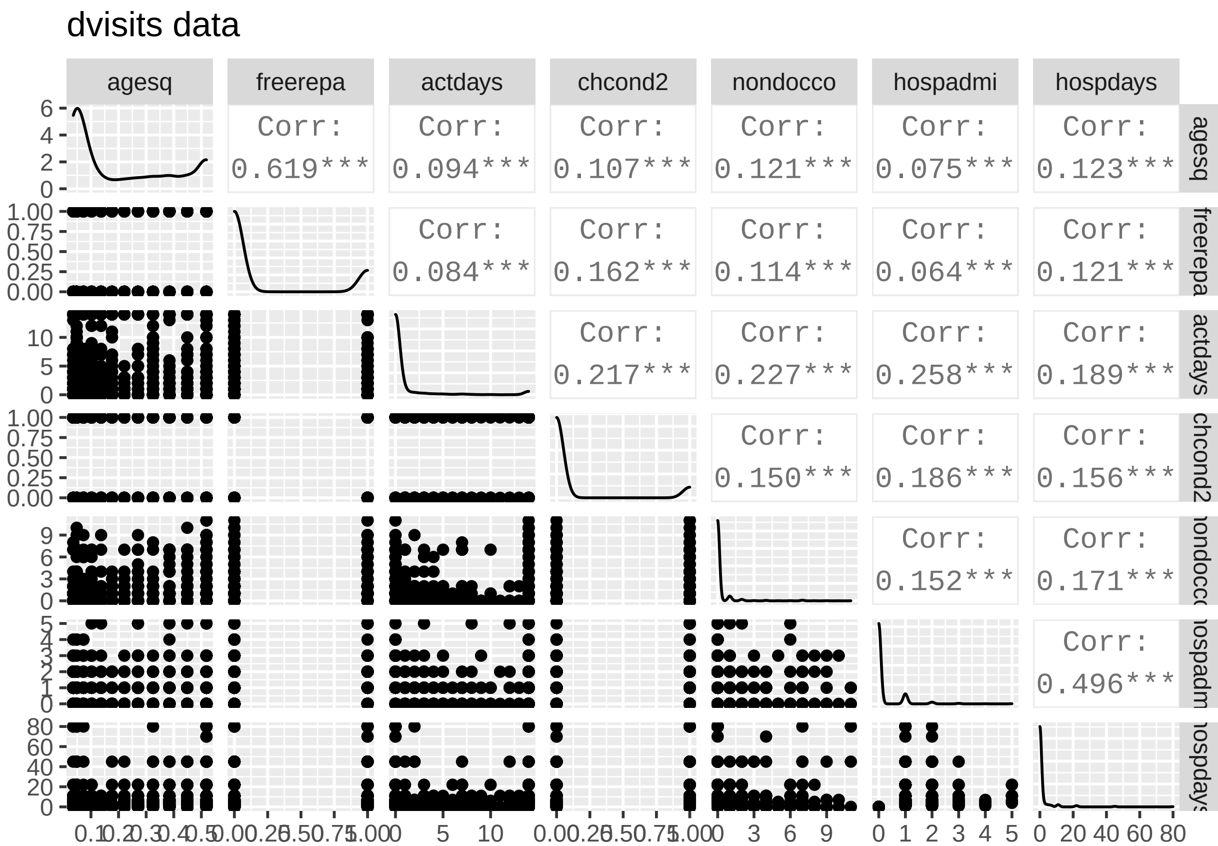
```
plot(transformed_model,which=1)
```



This plot also has some pattern and we see that even with transformation of the variable, we were not able to completely account for linearity.

Let's now check the structure of the data. Given ggpairs is intensive, we use only the significant predictors and response to plot the relationships.

```
ggpairs(data=dvisits, columns=c("agesq","freerepa","actdays",
            "chcond2","nondocco","hospadmi","hospdays"), title="dvisits data")
```



Given the discrete values in most of the variables, it's difficult to asses linear relationship above. We can however see the positive correlations mostly with the response variable.

6) The following data provides the Covid-19 cases per state since January. https://covidtracking.com/api/v1/states/daily.csv . The purpose is to predict the total number of cases in US per day with linear regression. Use data till Sept for training and rest for testing. Perform diagnostics and show why your model is good.

Let us load the file.

```
daily <-  read.csv('/Users/mac/Downloads/daily.csv')
str(daily)
```

```
## 'data.frame':    15969 obs. of  55 variables:
##  $ date                        : int  20201212 20201212 20201212 20201212 20201212 20201212 20201212 2
##  $ state                       : Factor w/ 56 levels "AK","AL","AR",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ positive                    : int  39101 292841 184252 0 402589 1521432 285634 146761 24643 44876
##  $ probableCases               : int  NA 52306 26377 NA 14891 NA 11767 8970 NA 1719 ...
##  $ negative                    : int  1094317 1458560 1685156 2140 2099165 25364698 1669992 3548330 74
##  $ pending                     : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ totalTestResultsSource      : Factor w/ 4 levels "posNeg","totalTestEncountersViral",..: 4 3 4 4 4
##  $ totalTestResults            : int  1133418 1696439 1843031 2140 4245154 26886130 3738103 3695091 77
##  $ hospitalizedCurrently       : int  146 2161 1071 NA 3534 13410 1607 1210 230 348 ...
##  $ hospitalizedCumulative      : int  883 28146 9911 NA 30687 NA 16092 12257 NA NA ...
##  $ inIcuCurrently              : int  NA NA 374 NA 799 2866 NA NA 59 52 ...
##  $ inIcuCumulative             : int  NA 2356 NA NA NA NA NA NA NA NA ...
##  $ onVentilatorCurrently       : int  18 NA 177 NA 515 NA NA NA 30 NA ...
##  $ onVentilatorCumulative      : int  NA 1350 1091 NA NA NA NA NA NA NA ...
##  $ recovered                   : int  7165 174805 159827 NA 60646 NA 15208 9800 17618 18851 ...
##  $ dataQualityGrade            : Factor w/ 8 levels "","#REF!","A",..: 3 3 4 7 4 5 3 6 4 4 ...
##  $ lastUpdateEt                : Factor w/ 7385 levels "","10/1/2020 00:00",..: 1714 1719 1709 1631 17
##  $ dateModified                : Factor w/ 7385 levels "","2020-02-26T00:00:00Z",..: 7373 7378 7368 70
##  $ checkTimeEt                 : Factor w/ 7385 levels "","02/25 19:00",..: 7373 7378 7368 7078 7368 7
##  $ death                       : int  176 4102 2911 0 7322 20847 3871 5363 713 815 ...
##  $ hospitalized                : int  883 28146 9911 NA 30687 NA 16092 12257 NA NA ...
##  $ dateChecked                 : Factor w/ 7385 levels "","2020-02-26T00:00:00Z",..: 7373 7378 7368 70
##  $ totalTestsViral             : int  1133418 NA 1843031 2140 4245154 26886130 NA 3686121 NA NA ...
##  $ positiveTestsViral          : int  47061 NA NA NA NA NA NA NA NA 44977 ...
##  $ negativeTestsViral          : int  1085086 NA 1685156 NA NA NA NA NA NA NA ...
##  $ positiveCasesViral          : int  NA 240535 157875 0 387698 1521432 273867 137791 NA 43157 ...
##  $ deathConfirmed              : int  176 3624 2619 NA 6749 NA 3305 4332 NA 718 ...
##  $ deathProbable               : int  NA 478 292 NA 573 NA 566 1031 NA 97 ...
##  $ totalTestEncountersViral    : int  NA NA NA NA NA NA 3738103 NA 770914 829032 ...
##  $ totalTestsPeopleViral       : int  NA 1696439 NA NA 2486863 NA 1943859 NA 327867 460700 ...
##  $ totalTestsAntibody          : int  NA NA NA NA 375676 NA 232118 NA NA NA ...
##  $ positiveTestsAntibody       : int  NA NA NA NA NA NA 22018 NA NA NA ...
##  $ negativeTestsAntibody       : int  NA NA NA NA NA NA 210100 NA NA NA ...
##  $ totalTestsPeopleAntibody    : int  NA 76958 NA NA NA NA NA NA NA NA ...
##  $ positiveTestsPeopleAntibody: int  NA NA NA NA NA NA NA NA NA NA ...
##  $ negativeTestsPeopleAntibody: int  NA NA NA NA NA NA NA NA NA NA ...
##  $ totalTestsPeopleAntigen     : int  NA NA 177317 NA NA NA NA NA NA NA ...
##  $ positiveTestsPeopleAntigen : int  NA NA 32114 NA NA NA NA NA NA NA ...
##  $ totalTestsAntigen           : int  NA NA 21856 NA NA NA NA 49816 NA NA ...
##  $ positiveTestsAntigen        : int  NA NA 3300 NA NA NA NA NA NA NA ...
##  $ fips                        : int  2 1 5 60 4 6 8 9 11 10 ...
##  $ positiveIncrease            : int  517 4066 2628 0 8077 35729 3961 0 286 1058 ...
##  $ negativeIncrease            : int  9356 0 12988 0 18295 278058 10966 0 7882 3263 ...
##  $ total                       : int  1133418 1751401 1869408 2140 2501754 26886130 1955626 3695091 77
```

```
## $ totalTestResultsIncrease   : int  9873 0 14852 0 57294 313787 46640 0 8168 9874 ...
## $ posNeg                     : int  1133418 1751401 1869408 2140 2501754 26886130 1955626 3695091 7
## $ deathIncrease              : int  18 16 36 0 77 225 25 0 4 8 ...
## $ hospitalizedIncrease       : int  14 0 63 0 385 0 143 0 0 0 ...
## $ hash                       : Factor w/ 15969 levels "0000831d10ed7915f0ccfcd211652c27ca1f259e",..
## $ commercialScore            : int  0 0 0 0 0 0 0 0 0 0 ...
## $ negativeRegularScore       : int  0 0 0 0 0 0 0 0 0 0 ...
## $ negativeScore              : int  0 0 0 0 0 0 0 0 0 0 ...
## $ positiveScore              : int  0 0 0 0 0 0 0 0 0 0 ...
## $ score                      : int  0 0 0 0 0 0 0 0 0 0 ...
## $ grade                      : logi  NA NA NA NA NA NA ...
```

Let's summarise this information -

```
summary(daily)
```

```
##       date              state         positive       probableCases
## Min.   :20200122   MA     :  326   Min.   :      0   Min.   :     0
## 1st Qu.:20200513   WA     :  326   1st Qu.:   2990   1st Qu.:   386
## Median :20200723   FL     :  319   Median :  24302   Median :  2033
## Mean   :20200737   NJ     :  307   Mean   :  86242   Mean   :  6061
## 3rd Qu.:20201002   NE     :  302   3rd Qu.: 101782   3rd Qu.:  6404
## Max.   :20201212   IN     :  290   Max.   :1521432   Max.   :142784
##                    (Other):14099   NA's   :152       NA's   :10014
##    negative           pending                     totalTestResultsSource
## Min.   :       0   Min.   :    0.0   posNeg                  :6481
## 1st Qu.:   52950   1st Qu.:   30.0   totalTestEncountersViral:3772
## Median :  303805   Median :  196.5   totalTestsPeopleViral   : 843
## Mean   : 1025763   Mean   : 1509.6   totalTestsViral         :4873
## 3rd Qu.: 1029180   3rd Qu.: 1022.8
## Max.   :25364698   Max.   :64400.0
## NA's   :310        NA's   :14255
## totalTestResults   hospitalizedCurrently hospitalizedCumulative
## Min.   :       0   Min.   :    0         Min.   :    1.0
## 1st Qu.:   54949   1st Qu.:  140         1st Qu.:  699.2
## Median :  376851   Median :  474         Median : 3121.5
## Mean   : 1229190   Mean   :  969         Mean   : 8337.9
## 3rd Qu.: 1293240   3rd Qu.: 1043         3rd Qu.: 9649.2
## Max.   :26886130   Max.   :18825         Max.   :89995.0
## NA's   :60         NA's   :3135          NA's   :6307
## inIcuCurrently   inIcuCumulative onVentilatorCurrently
## Min.   :   0.0   Min.   :   6    Min.   :   0.0
## 1st Qu.:  60.0   1st Qu.: 374    1st Qu.:  27.0
## Median : 151.0   Median : 996    Median :  77.0
## Mean   : 311.4   Mean   :1404    Mean   : 133.2
## 3rd Qu.: 322.0   3rd Qu.:1891    3rd Qu.: 146.0
## Max.   :5225.0   Max.   :6847    Max.   :2425.0
## NA's   :8004     NA's   :13197   NA's   :9566
## onVentilatorCumulative   recovered       dataQualityGrade
## Min.   :  32.0         Min.   :      2   A+     :4775
## 1st Qu.: 170.0         1st Qu.:   2394   A      :4296
## Median : 333.0         Median :  10096   B      :3915
## Mean   : 428.9         Mean   :  43315          :1261
## 3rd Qu.: 573.0         3rd Qu.:  48724   C      :1083
## Max.   :1350.0         Max.   :1167975   D      : 601
```

```
## NA's   :15016          NA's   :4570      (Other):  38
##          lastUpdateEt                dateModified        checkTimeEt
##                  :  487                        :  487              :  487
## 9/1/2020 00:00  :   46   2020-09-01T00:00:00Z:   46   08/31 20:00:   46
## 10/1/2020 00:00 :   44   2020-10-01T00:00:00Z:   44   09/30 20:00:   44
## 6/1/2020 00:00  :   43   2020-06-01T00:00:00Z:   43   05/31 20:00:   43
## 8/15/2020 00:00 :   41   2020-08-15T00:00:00Z:   41   08/14 20:00:   41
## 11/17/2020 00:00:   31   2020-11-17T00:00:00Z:   31   11/16 19:00:   31
## (Other)         :15277   (Other)            :15277   (Other)    :15277
##     death        hospitalized                  dateChecked
## Min.   :    0   Min.   :    1.0                       :  487
## 1st Qu.:   99   1st Qu.:  699.2   2020-09-01T00:00:00Z:   46
## Median :  670   Median : 3121.5   2020-10-01T00:00:00Z:   44
## Mean   : 2554   Mean   : 8337.9   2020-06-01T00:00:00Z:   43
## 3rd Qu.: 2870   3rd Qu.: 9649.2   2020-08-15T00:00:00Z:   41
## Max.   :27675   Max.   :89995.0   2020-11-17T00:00:00Z:   31
## NA's   :826     NA's   :6307      (Other)             :15277
## totalTestsViral    positiveTestsViral negativeTestsViral
## Min.   :       0   Min.   :      0    Min.   :       1
## 1st Qu.:   98068   1st Qu.:   7002    1st Qu.:  167088
## Median :  494837   Median :  34760    Median :  521524
## Mean   : 1415790   Mean   :  98288    Mean   : 1081633
## 3rd Qu.: 1602848   3rd Qu.: 130273    3rd Qu.: 1272023
## Max.   :26886130   Max.   :1430692    Max.   :10741096
## NA's   :5359       NA's   :10577      NA's   :12467
## positiveCasesViral deathConfirmed  deathProbable
## Min.   :      0    Min.   :    0   Min.   :   0.0
## 1st Qu.:   6438    1st Qu.:  451   1st Qu.:  41.0
## Median :  38930    Median : 1778   Median : 148.0
## Mean   : 100093    Mean   : 2764   Mean   : 254.8
## 3rd Qu.: 122110    3rd Qu.: 3697   3rd Qu.: 266.0
## Max.   :1521432    Max.   :15864   Max.   :1868.0
## NA's   :3545       NA's   :9056    NA's   :10691
## totalTestEncountersViral totalTestsPeopleViral totalTestsAntibody
## Min.   :       0         Min.   :      0       Min.   :     5
## 1st Qu.:   92059         1st Qu.: 121395       1st Qu.: 14144
## Median :  476617         Median : 315042       Median : 57683
## Mean   : 1507956         Mean   : 680704       Mean   :124330
## 3rd Qu.: 1469002         3rd Qu.: 873220       3rd Qu.:193476
## Max.   :21757045         Max.   :7839137       Max.   :650372
## NA's   :12213            NA's   :9183          NA's   :12856
## positiveTestsAntibody negativeTestsAntibody totalTestsPeopleAntibody
## Min.   :    1         Min.   :   587        Min.   :     1
## 1st Qu.:  359         1st Qu.:  9482        1st Qu.: 33759
## Median : 4403         Median : 57298        Median : 68919
## Mean   :10973         Mean   :109328        Mean   :120511
## 3rd Qu.:13489         3rd Qu.:136814        3rd Qu.:125459
## Max.   :71851         Max.   :578113        Max.   :630838
## NA's   :13679         NA's   :14938         NA's   :14669
## positiveTestsPeopleAntibody negativeTestsPeopleAntibody
## Min.   :    0              Min.   :     1
## 1st Qu.: 2693              1st Qu.: 46014
## Median : 3807              Median : 71709
## Mean   :12342              Mean   :143855
```

```
##  3rd Qu.:12358               3rd Qu.:143409
##  Max.   :69600               Max.   :560932
##  NA's   :15140               NA's   :15177
##  totalTestsPeopleAntigen positiveTestsPeopleAntigen totalTestsAntigen
##  Min.   :     3          Min.   :    3             Min.   :     2
##  1st Qu.: 13446          1st Qu.: 1628             1st Qu.: 16184
##  Median : 50660          Median : 4591             Median : 36880
##  Mean   : 71976          Mean   :10996             Mean   : 84574
##  3rd Qu.:113699          3rd Qu.:15947             3rd Qu.:106780
##  Max.   :296649          Max.   :66306             Max.   :864141
##  NA's   :15395           NA's   :15528             NA's   :14691
##  positiveTestsAntigen      fips       positiveIncrease  negativeIncrease
##  Min.   :    0        Min.   : 1.00   Min.   :-7757.0   Min.   :-340903
##  1st Qu.: 1004        1st Qu.:17.00   1st Qu.:   45.0   1st Qu.:    497
##  Median : 3438        Median :31.00   Median :  333.0   Median :   3353
##  Mean   :10333        Mean   :32.42   Mean   :  999.2   Mean   :  10640
##  3rd Qu.:15318        3rd Qu.:46.00   3rd Qu.: 1014.0   3rd Qu.:  10734
##  Max.   :71382        Max.   :78.00   Max.   :71734.0   Max.   : 956990
##  NA's   :15111
##      total          totalTestResultsIncrease     posNeg
##  Min.   :       0   Min.   :-336933          Min.   :       0
##  1st Qu.:   50197   1st Qu.:    825          1st Qu.:   49633
##  Median :  315794   Median :   4872          Median :  315661
##  Mean   : 1091433   Mean   :  13598          Mean   : 1091271
##  3rd Qu.: 1103038   3rd Qu.:  14921          3rd Qu.: 1102924
##  Max.   :26886130   Max.   : 313787          Max.   :26886130
##
##  deathIncrease     hospitalizedIncrease
##  Min.   :-201.00   Min.   :-4124.00
##  1st Qu.:   0.00   1st Qu.:    0.00
##  Median :   5.00   Median :    0.00
##  Mean   :  18.13   Mean   :   38.35
##  3rd Qu.:  18.00   3rd Qu.:   32.00
##  Max.   : 951.00   Max.   :16373.00
##
##                                       hash       commercialScore
##  0000831d10ed7915f0ccfcd211652c27ca1f259e:    1   Min.   :0
##  0001483bf70821fe534d3eabdba98552fdc84698:    1   1st Qu.:0
##  00025e0c779ad783aec2a7ea2fb204a5457bac98:    1   Median :0
##  0004a2a995e3e7e6f1747de720e302f65153f29a:    1   Mean   :0
##  000a11f702a1508be90c76de2668f513b274193b:    1   3rd Qu.:0
##  000b85b58233668c24bb4bd6be9d4a6ae7b1e72f:    1   Max.   :0
##  (Other)                                 :15963
##  negativeRegularScore negativeScore positiveScore     score
##  Min.   :0            Min.   :0     Min.   :0     Min.   :0
##  1st Qu.:0            1st Qu.:0     1st Qu.:0     1st Qu.:0
##  Median :0            Median :0     Median :0     Median :0
##  Mean   :0            Mean   :0     Mean   :0     Mean   :0
##  3rd Qu.:0            3rd Qu.:0     3rd Qu.:0     3rd Qu.:0
##  Max.   :0            Max.   :0     Max.   :0     Max.   :0
##
##   grade
##  Mode:logical
##  NA's:15969
```

```
##
##
##
##
##
```

Key observations -

1. The data is at state level and the question wants us to come up with a model for "total number of cases per day in the US" so we need to roll up the data at a day-level.

2. The date variable needs to be converted to date/ ts format.

3. We see quite a few NAs in the data.

4. The total column is the sum of positive, negative and pending cases which makes sense and will be our pendent variable.

5. Our variables of interest seem to be date, state (we can take a count of state per day), positive, negative, probableCases, pending, totalTestResults, death, total. There is a reason for not choosing variables like hospitalizedCurrently or inIcuCumulative since we do not expect total cases per day to be influenced by such factors, however we do expect more total cases per day if there are more tests conducted.

6. We have geographical identifier in 'FIPS' but we will later add latitude and longitude while making map plots.

However do note that in our test time frame, we will of course not have such breakdowns of total cases present, hence we won't be able to use these explanatory variables for forecasting. So our modeling form will most likely be -

$y_t = \alpha * y_{t-1} + \beta * y_{t-2} + ...\epsilon$

i.e our total number of cases per day will be regressed on previous day and significant lags in the past to predict for future. This is what Time Series Regression methods like Autoregression, ARIMA, SARIMA, and more complex models like LSTM (a type of neural network) can perform.

Data prep -

For now we limit our dataset to the following variables and construct a rolled up day level data -

```
# subset the data for above variables
state_daily <- daily[,c("date","state","positive",
     "negative", "probableCases", "pending",
    "totalTestResults", "death", "total","totalTestResultsIncrease")]
str(state_daily)
```

```
## 'data.frame':    15969 obs. of  10 variables:
##  $ date                    : int  20201212 20201212 20201212 20201212 20201212 20201212 20201212 2020
##  $ state                   : Factor w/ 56 levels "AK","AL","AR",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ positive                : int  39101 292841 184252 0 402589 1521432 285634 146761 24643 44876 ...
##  $ negative                : int  1094317 1458560 1685156 2140 2099165 25364698 1669992 3548330 7462
##  $ probableCases           : int  NA 52306 26377 NA 14891 NA 11767 8970 NA 1719 ...
```

```
##  $ pending                 : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ totalTestResults        : int  1133418 1696439 1843031 2140 4245154 26886130 3738103 3695091 7709
##  $ death                   : int  176 4102 2911 0 7322 20847 3871 5363 713 815 ...
##  $ total                   : int  1133418 1751401 1869408 2140 2501754 26886130 1955626 3695091 7709
##  $ totalTestResultsIncrease: int  9873 0 14852 0 57294 313787 46640 0 8168 9874 ...
```

We replace the NAs by 0 so we can sum them up

```
state_daily[is.na(state_daily)] <- 0
str(state_daily)
```

```
## 'data.frame':    15969 obs. of  10 variables:
##  $ date                    : int  20201212 20201212 20201212 20201212 20201212 20201212 20201212 2020
##  $ state                   : Factor w/ 56 levels "AK","AL","AR",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ positive                : num  39101 292841 184252 0 402589 ...
##  $ negative                : num  1094317 1458560 1685156 2140 2099165 ...
##  $ probableCases           : num  0 52306 26377 0 14891 ...
##  $ pending                 : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ totalTestResults        : num  1133418 1696439 1843031 2140 4245154 ...
##  $ death                   : num  176 4102 2911 0 7322 ...
##  $ total                   : int  1133418 1751401 1869408 2140 2501754 26886130 1955626 3695091 7709
##  $ totalTestResultsIncrease: int  9873 0 14852 0 57294 313787 46640 0 8168 9874 ...
```

Roll up the data to day level -

```
overall_daily <- state_daily %>%
  dplyr::group_by(date) %>%
  dplyr::summarise(pos=sum(positive),neg=sum(negative),
                   probable = sum(probableCases),
                   pend = sum(pending),
                   totalTest = sum(totalTestResults),
                   totalTestInc = sum(totalTestResultsIncrease),
                   deaths = sum(death),
                   totals = sum(total)
                   )
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

We look at the most recent data and correlate it with the official numbers. There were ~15 Mn cases reported as of Dec 9th, 2020 in US and our pos column shows 15.2 Mn positive COVID-19 cases. In order to predict the Total COVID-19 cases per day, we want to accurately predict this "pos" column.

Visualise the Data

We use maps for this now and aggregate the data at a state level.

```r
require(usmap)
```

```
## Loading required package: usmap
```

```r
require(openintro)
```

```
## Loading required package: openintro

## Loading required package: airports

## Loading required package: cherryblossom

## Loading required package: usdata

##
## Attaching package: 'openintro'

## The following objects are masked from 'package:fma':
##
##     books, housing

## The following objects are masked from 'package:fpp2':
##
##     goog, marathon, prison

## The following object is masked from 'package:faraway':
##
##     orings

## The following object is masked from 'package:reshape':
##
##     tips

## The following object is masked from 'package:caret':
##
##     dotPlot

## The following object is masked from 'package:car':
##
##     densityPlot

## The following object is masked from 'package:survival':
##
##     transplant

## The following objects are masked from 'package:lattice':
##
##     ethanol, lsegments

## The following objects are masked from 'package:MASS':
##
##     housing, mammals
```

```r
require(maps)
```

```
## Loading required package: maps

##
## Attaching package: 'maps'
```

```
## The following object is masked from 'package:fma':
##
##     ozone
## The following object is masked from 'package:faraway':
##
##     ozone
## The following object is masked from 'package:purrr':
##
##     map
## The following object is masked from 'package:plyr':
##
##     ozone
```

```
covidus <- daily
abb=covidus$state
# We convert the region names to full names with abbr2state
region=abbr2state(abb)


covidus2=cbind(covidus,region)
covidus2['fips'] <- fips(covidus2$state)
states=map_data("state")

####Aggregating the data ####

states2=states %>%
  dplyr::group_by(region) %>%
  dplyr::summarise(lat = max(lat),long=max(long))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
covid_agg=covidus2 %>%
  dplyr::group_by(region,state) %>%
  dplyr::summarise(total_pos = sum(positive,na.rm = TRUE),
                   total_neg=sum(negative,na.rm = TRUE),
            total_nt=sum(pending,na.rm = TRUE),
            total_t=sum(total,na.rm = TRUE),
            total_deaths=sum(death,na.rm = TRUE)
            )
```

```
## `summarise()` regrouping output by 'region' (override with `.groups` argument)
```

```
covid_agg$region=tolower(covid_agg$region)

map.df <- merge(covid_agg,states2, by="region", all.x = T)
```

convert date to date format

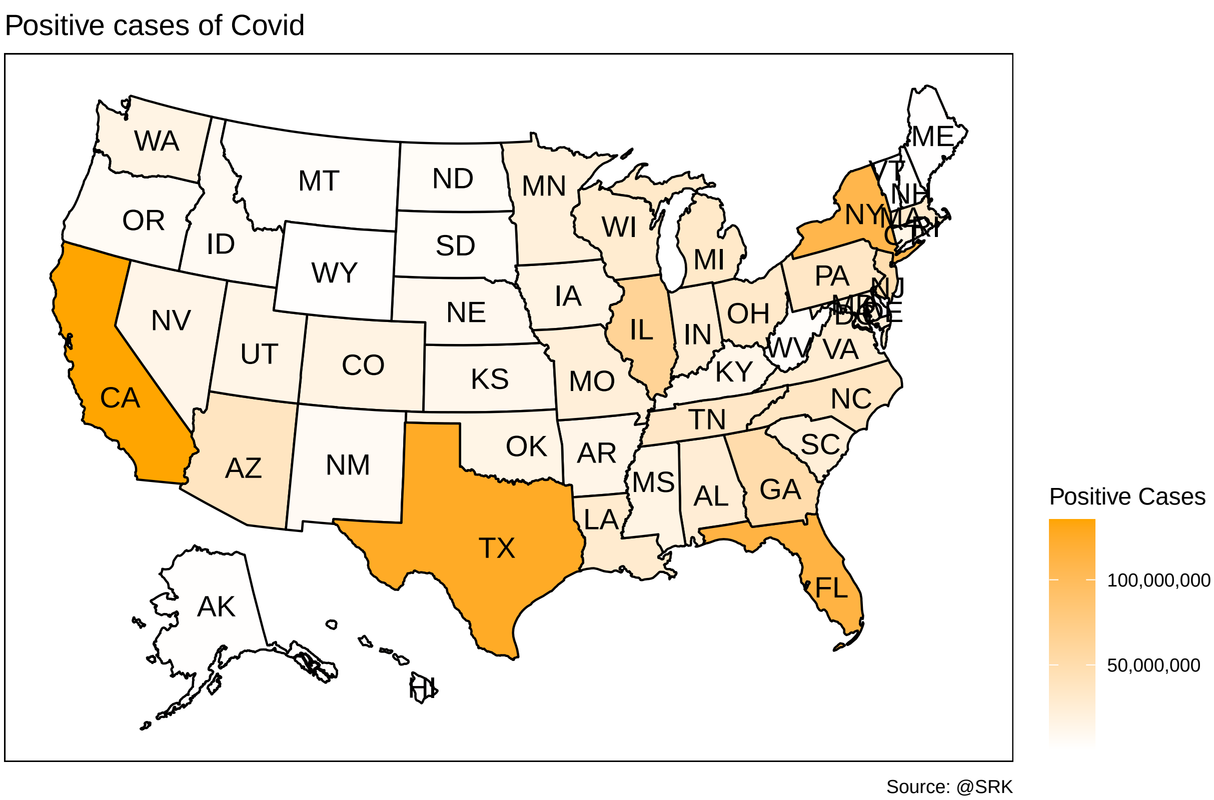
```
state_daily$date <- as.Date(as.character(state_daily$date),format="%Y%m%d")
#daily$date <- as.Date(as.character(daily$date),format="%Y%m%d")
covidus2$date <- as.Date(as.character(covidus2$date),format="%Y%m%d")
str(state_daily)
```

```
## 'data.frame':    15969 obs. of  10 variables:
##  $ date                   : Date, format: "2020-12-12" "2020-12-12" ...
##  $ state                  : Factor w/ 56 levels "AK","AL","AR",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ positive               : num  39101 292841 184252 0 402589 ...
##  $ negative               : num  1094317 1458560 1685156 2140 2099165 ...
##  $ probableCases          : num  0 52306 26377 0 14891 ...
##  $ pending                : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ totalTestResults       : num  1133418 1696439 1843031 2140 4245154 ...
##  $ death                  : num  176 4102 2911 0 7322 ...
##  $ total                  : int  1133418 1751401 1869408 2140 2501754 26886130 1955626 3695091 7709
##  $ totalTestResultsIncrease: int  9873 0 14852 0 57294 313787 46640 0 8168 9874 ...
```

We create a map.df and now we look at US maps plot.

Total positive cases by state in the US

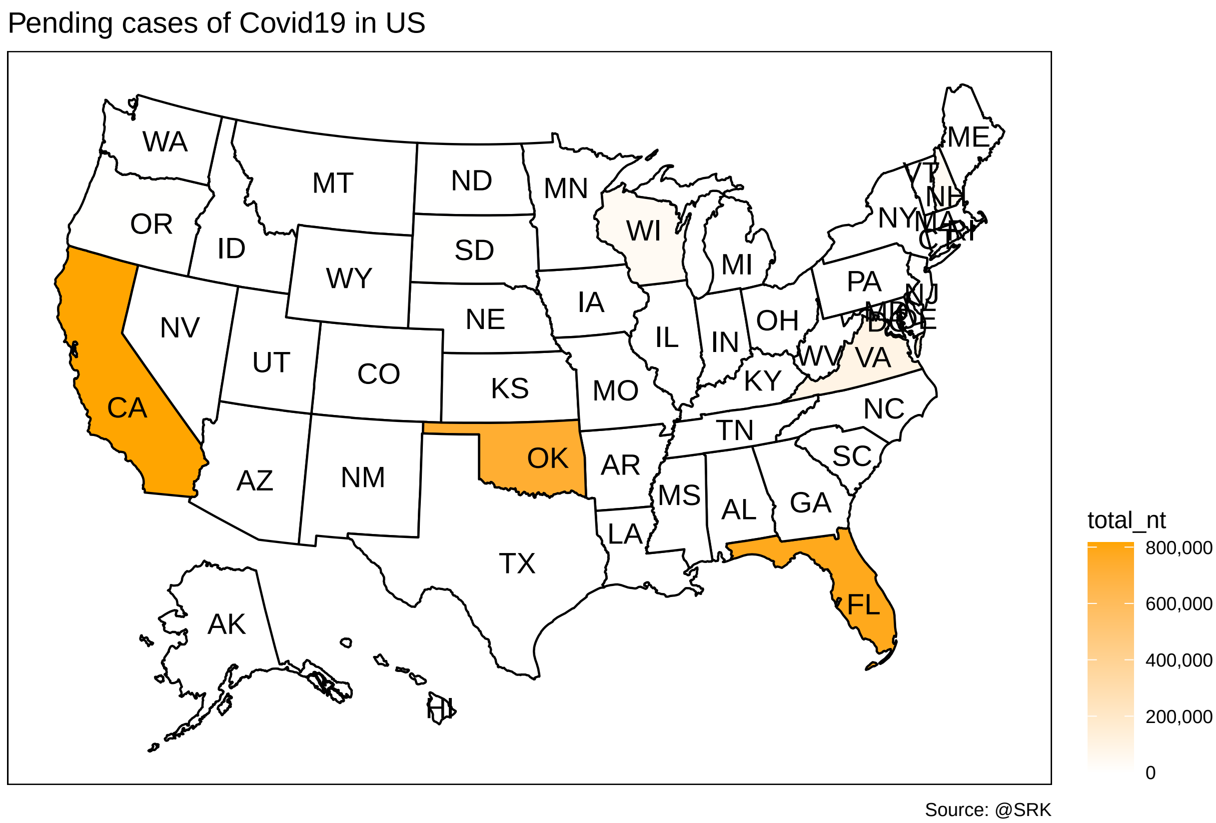
```r
plot_usmap(data = map.df, values = "total_pos",   labels=TRUE) +
  scale_fill_continuous( low = "white", high = "orange",
                         name = "Positive Cases", label = scales::comma
  ) +
  theme(legend.position = "right") +
  theme(panel.background = element_rect(colour = "black")) +
  labs(title = "Positive cases of Covid", caption = "Source: @SRK")
```



We see few states like California, Texas, New York, Florida have really high cases.

Total pending cases by state in the US

```r
####total pending cases in US####
plot_usmap(data = map.df, values = "total_nt",   labels=TRUE) +
  scale_fill_continuous( low = "white", high = "orange",
                         name = "total_nt", label = scales::comma
  ) +
  theme(legend.position = "right") +
  theme(panel.background = element_rect(colour = "black")) +
  labs(title = "Pending cases of Covid19 in US", caption = "Source: @SRK")
```

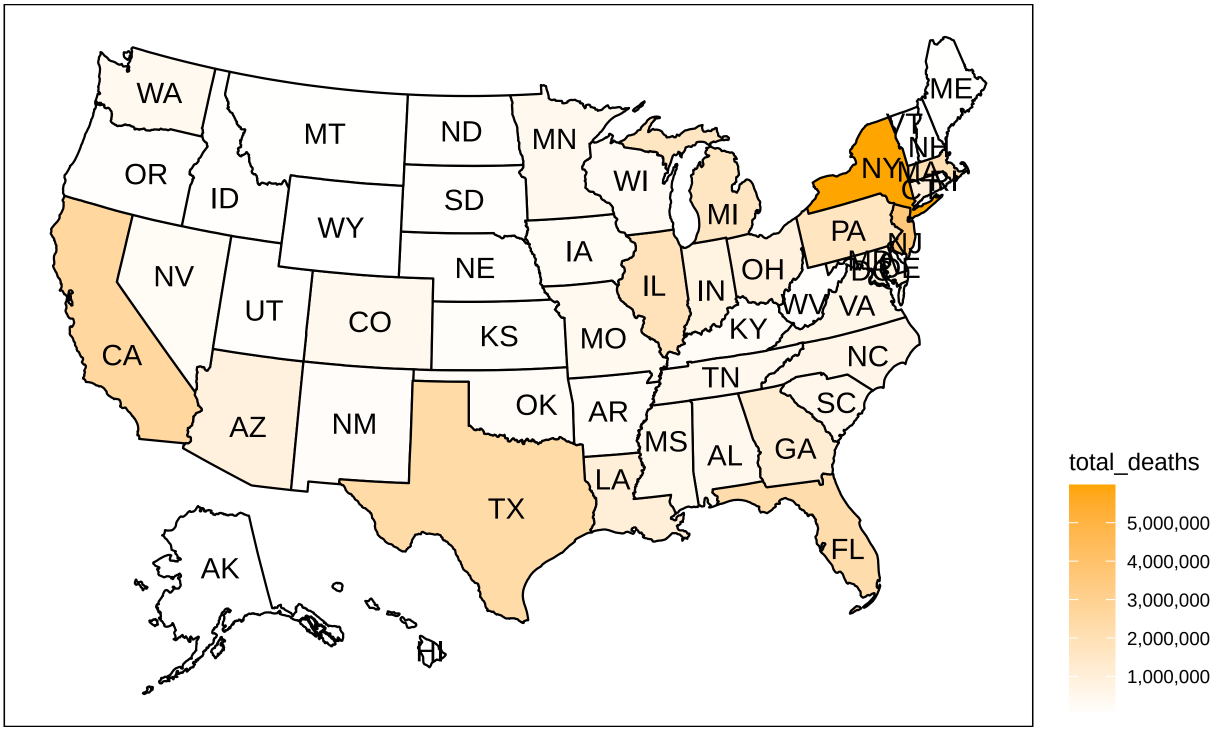


We ee very high pending cases in CA, OK and FL.

Total deaths by state in the US

```r
####total non treated cases in US####
plot_usmap(data = map.df, values = "total_deaths",   labels=TRUE) +
  scale_fill_continuous( low = "white", high = "orange",
                         name = "total_deaths", label = scales::comma
  ) +
  theme(legend.position = "right") +
  theme(panel.background = element_rect(colour = "black")) +
  labs(title = "Deaths in Covid19 in US", caption = "Source: @SRK")
```

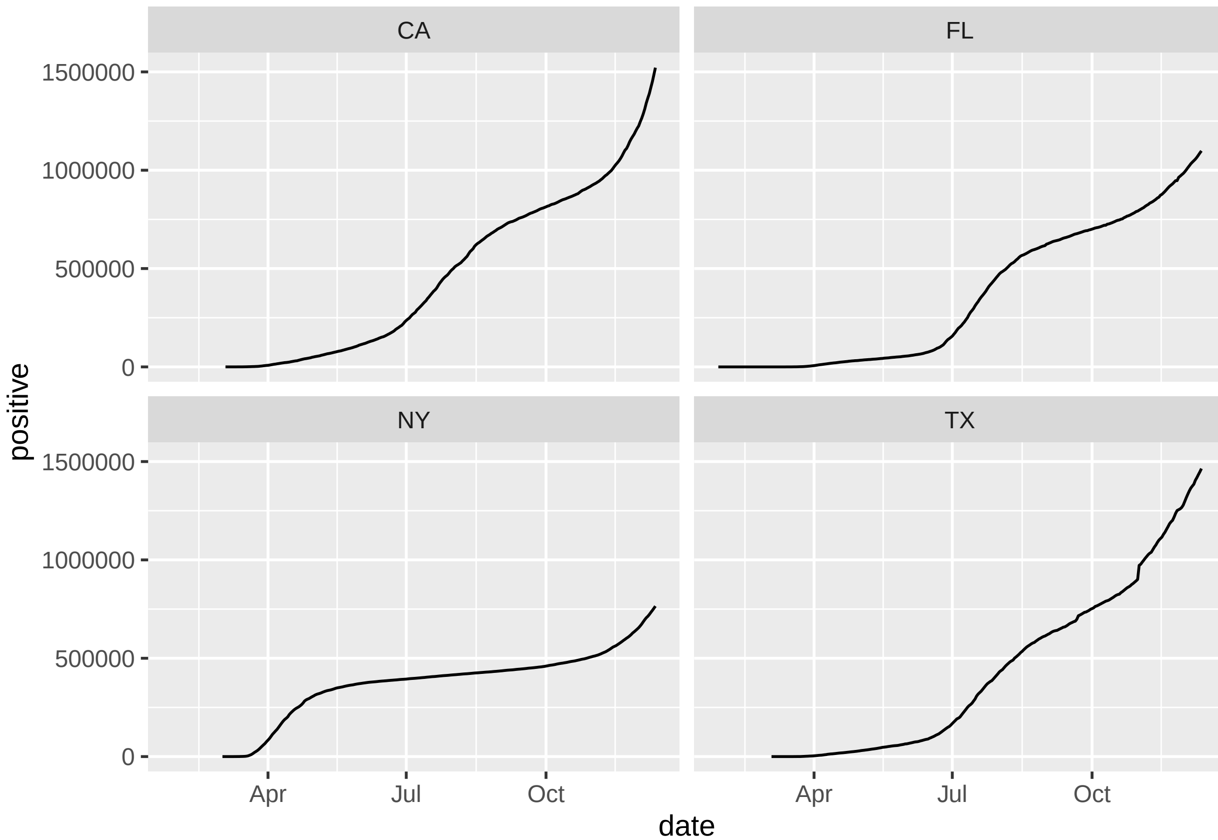
Deaths in Covid19 in US



Source: @SRK

Positive trend of cases in 4 states - TX, CA, FL, NY

```
### Subsetting the data for four states with high cases ####
sub_us=subset(covidus2 , subset =(covidus2$state %in% c('TX','CA','FL','NY')))

#### Infected patients growth in 4 states
p <- ggplot(data = sub_us, aes(x = date, y = positive),color= state)

p + geom_line(aes(group = state)) + facet_wrap(~ state)
```

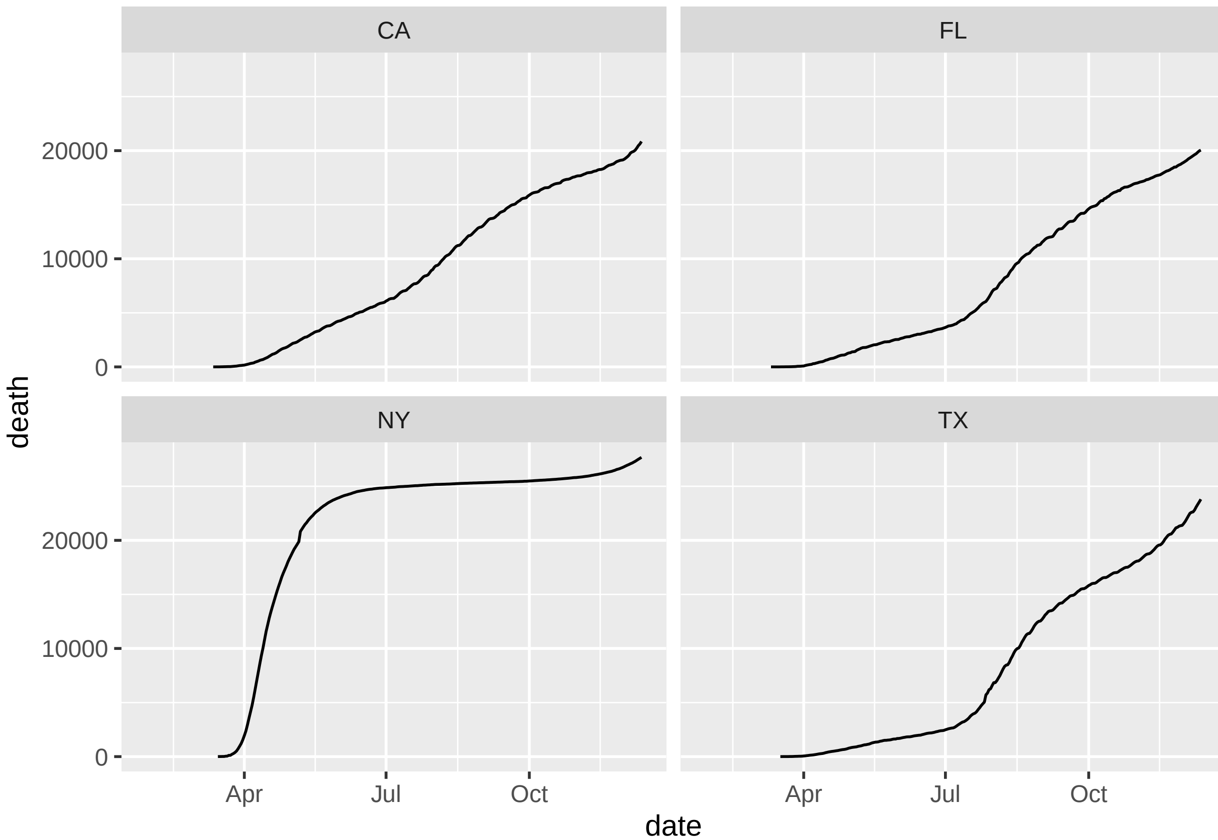


We notice a not so sharp increase in NY in Jul, while in Florida and Texas between Apr-Jul, we see cases spiking slowly. This maybe due to limited testing at the time, and later months look quite grim.

Death trend of cases in 4 states - TX, CA, FL, NY

```
p <- ggplot(data = sub_us, aes(x = date, y = death))
p + geom_line(aes(group = state)) + facet_wrap(~ state)
```

```
## Warning: Removed 76 row(s) containing missing values (geom_path).
```

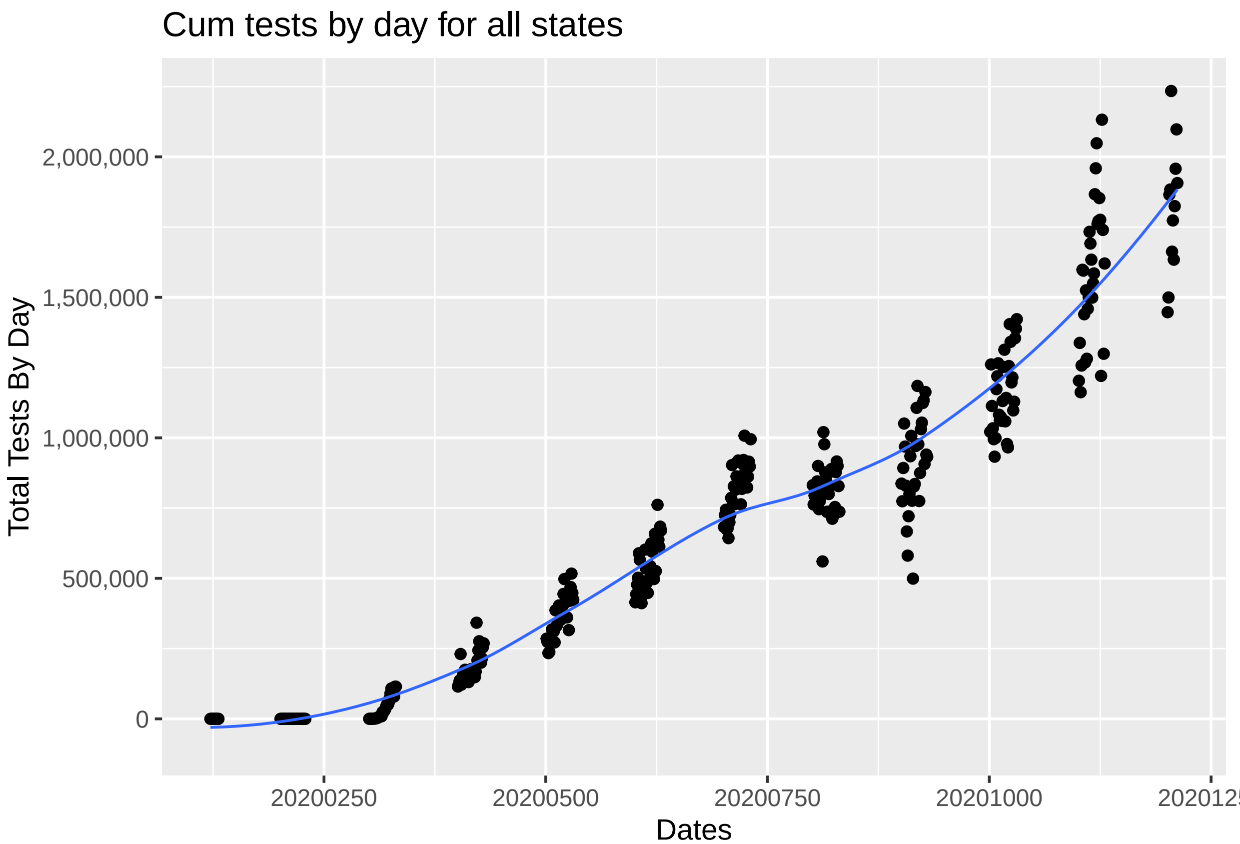


Unfortunately, the number of deaths spiked in NY in the months of Apr-June and then stabilising. the other states mostly have a linear increase in death rate.

Cumulative tests by day for all state

```
###Tests By Day###
totaltest <- (aggregate(daily["totalTestResultsIncrease"]
  , by=daily["date"], sum, na.rm=TRUE, na.action=NULL))

ggplot(data=totaltest, aes(x=date,
y=totalTestResultsIncrease)) +geom_point() +
  geom_smooth(fill=NA, size=0.5) + labs(x = "Dates") +
ggtitle("Cum tests by day for all states") +
  scale_y_continuous(name="Total Tests By Day", labels = scales::comma)
```

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



We see that overall cumulative trend has continuously gone up.

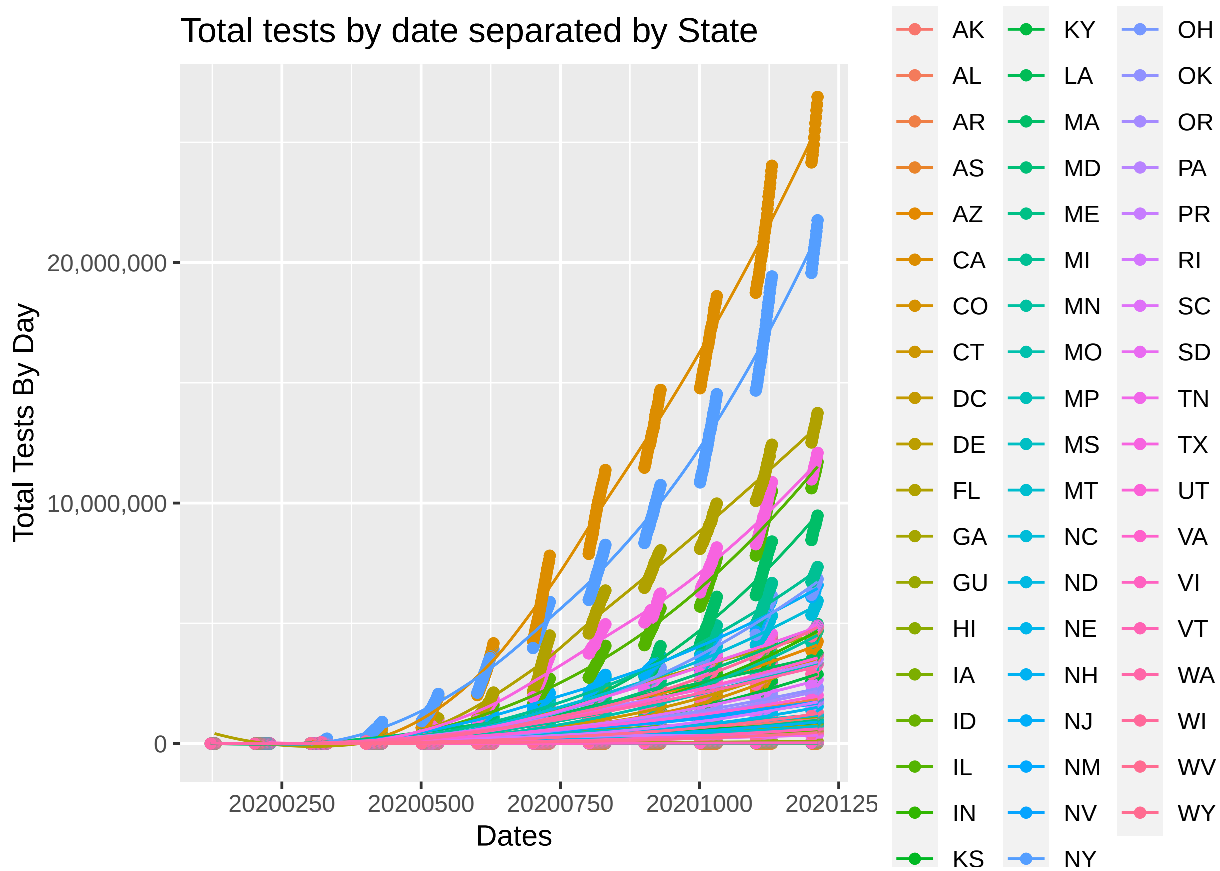
Total tests by date separated by State

```
ggplot(data=daily, aes(x=date,
  y=totalTestResults, colour = state)) +geom_point() +
  geom_smooth(fill=NA, size=0.5) + labs(x = "Dates") +
  ggtitle("Total tests by date separated by State") +
  scale_y_continuous(name="Total Tests By Day", labels = scales::comma)
```

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

```
## Warning: Removed 60 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 60 rows containing missing values (geom_point).
```



We can further look at more visuals however we feel we can now proceed to modeling the question.

Let's start by building a time series model with 'pos' - total number of positive cases daily in US in our overall_daily dataset.

For time series regression, we will first perform some checks and then build an ARIMA model folowed by ARIMAX. The major factors that may influence the covid cases, like income, gdp, education level, social distancing practice, are not in this particular dataset so we do not incorporate those.

Converting our rolled up data to date format

```
overall_daily$date <- as.Date(as.character(overall_daily$date),format="%Y%m%d")
str(overall_daily)
```

```
## tibble [326 x 9] (S3: tbl_df/tbl/data.frame)
##  $ date        : Date[1:326], format: "2020-01-22" "2020-01-23" ...
##  $ pos         : num [1:326] 0 0 0 0 0 0 0 0 0 0 ...
##  $ neg         : num [1:326] 0 0 0 0 0 0 0 0 0 0 ...
##  $ probable    : num [1:326] 0 0 0 0 0 0 0 0 0 0 ...
##  $ pend        : num [1:326] 0 0 0 0 0 0 0 0 0 0 ...
##  $ totalTest   : num [1:326] 1 2 2 2 2 3 3 5 5 8 ...
##  $ totalTestInc: int [1:326] 0 1 0 0 0 1 0 1 0 3 ...
##  $ deaths      : num [1:326] 0 0 0 0 0 0 0 0 0 0 ...
##  $ totals      : int [1:326] 0 0 0 0 0 0 0 0 0 0 ...
```

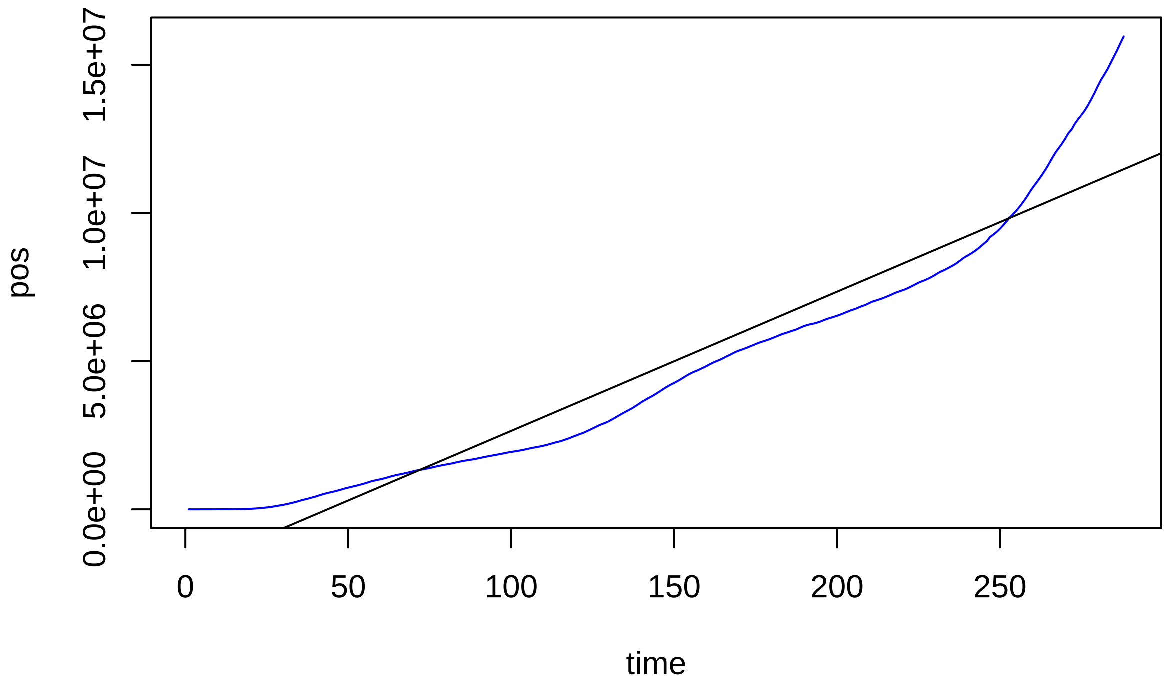
We prepare our data as below -

```
#creating additional variables
data <- overall_daily %>%
         dplyr::mutate(
         Date=as.Date(date, format="%Y%m%d"),
         day_of_week = wday(Date, label = TRUE),
         day_of_week_index = wday(Date),
         wk_of_month=ceiling(day(Date) / 7 ),
         month = format(Date, "%m"))
```

Removing the rows with 0 cases

```
data <- filter(data, pos != 0)
```

Visualizing the pos (covid positive cases) series



Seasonality (monthly) check for 1 year of daily data

We now check if monthly averages over 1 years of data show any seasonality or pattern by month i.e are there months wherein covid cases rose and fell ?

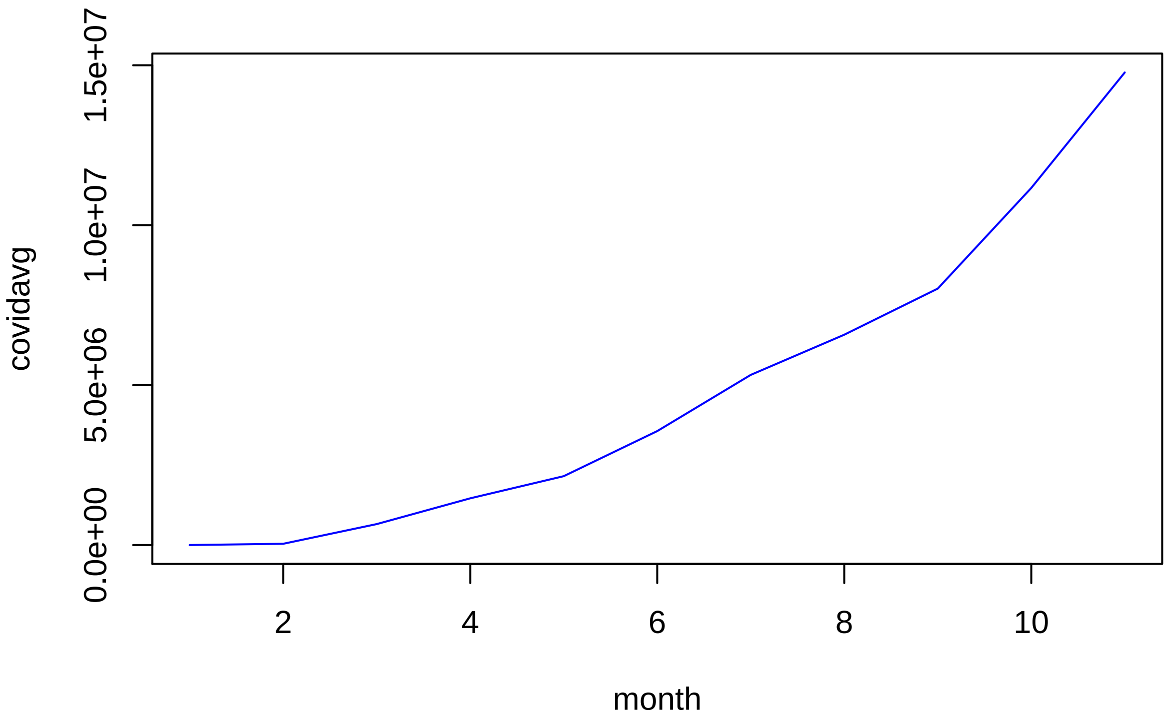
```
#do monthly averages show seasonality ?
seasonalchk <- data %>% dplyr::group_by(month) %>%
  dplyr::summarize(mean_pos=mean(pos)) %>%
  dplyr::group_by(month) %>%
  dplyr::summarize(avg_pos=mean(mean_pos))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
head(seasonalchk)
```

```
## # A tibble: 6 x 2
##   month  avg_pos
##   <chr>    <dbl>
## 1 02          18
## 2 03       39276
## 3 04      654813.
## 4 05     1460682.
## 5 06     2152834.
## 6 07     3562159.
```

```
#plot the series of mean prices by month
plot(seasonalchk$avg_pos,xlab="month",ylab="covidavg",type='l',col='blue')
```



We ee that cases didn't start increasing until month 3 - March but then took a huge turn (almost exponential increase).

splitting into train and test

```
train <- filter(data,month %in% c("01","02","03","04","05",
                              "06","07","08","09"))
test <- filter(data,month %in% c("10","11","12"))

str(train)
```

```
## tibble [215 x 14] (S3: tbl_df/tbl/data.frame)
##  $ date             : Date[1:215], format: "2020-02-29" "2020-03-01" ...
##  $ pos              : num [1:215] 18 50 94 145 278 ...
##  $ neg              : num [1:215] 24 85 243 446 1397 ...
##  $ probable         : num [1:215] 0 0 0 0 0 0 0 0 0 0 ...
##  $ pend             : num [1:215] 0 0 0 0 124 206 497 533 347 314 ...
##  $ totalTest        : num [1:215] 6552 6661 6873 7165 8177 ...
##  $ totalTestInc     : int [1:215] 62 94 212 285 322 608 874 875 1165 2144 ...
##  $ deaths           : num [1:215] 5 8 11 14 16 20 26 27 31 35 ...
##  $ totals           : int [1:215] 42 135 337 591 1799 2515 3712 4593 5513 7570 ...
##  $ Date             : Date[1:215], format: "2020-02-29" "2020-03-01" ...
##  $ day_of_week      : Ord.factor w/ 7 levels "Sun"<"Mon"<"Tue"<..: 7 1 2 3 4 5 6 7 1 2 ...
##  $ day_of_week_index: num [1:215] 7 1 2 3 4 5 6 7 1 2 ...
##  $ wk_of_month      : num [1:215] 5 1 1 1 1 1 1 1 2 2 ...
##  $ month            : chr [1:215] "02" "03" "03" "03" ...
```

```
str(test)
```

```
## tibble [73 x 14] (S3: tbl_df/tbl/data.frame)
##  $ date             : Date[1:73], format: "2020-10-01" "2020-10-02" ...
```

```
##  $ pos              : num [1:73] 7211040 7260293 7311230 7349212 7386968 ...
##  $ neg              : num [1:73] 93723319 94698422 95543345 96366378 97210565 ...
##  $ probable         : num [1:73] 141996 144788 147962 149309 150240 ...
##  $ pend             : num [1:73] 13003 10813 11464 11471 11544 ...
##  $ totalTest        : num [1:73] 1.13e+08 1.14e+08 1.15e+08 1.16e+08 1.17e+08 ...
##  $ totalTestInc     : int [1:73] 1021405 1261155 1113384 1033879 995060 932914 999154 1173237 121910
##  $ deaths           : num [1:73] 199943 200787 201530 201903 202233 ...
##  $ totals           : int [1:73] 100947362 101969528 102866039 103727061 104609077 105361737 1061474
##  $ Date             : Date[1:73], format: "2020-10-01" "2020-10-02" ...
##  $ day_of_week      : Ord.factor w/ 7 levels "Sun"<"Mon"<"Tue"<..: 5 6 7 1 2 3 4 5 6 7 ...
##  $ day_of_week_index: num [1:73] 5 6 7 1 2 3 4 5 6 7 ...
##  $ wk_of_month      : num [1:73] 1 1 1 1 1 1 1 2 2 2 ...
##  $ month            : chr [1:73] "10" "10" "10" "10" ...
```

Converting series to a ts object

In order to analyze a time series in R, we need to convert the data frame as a ts (time series) object. The R language uses many functions to create, manipulate and plot the time series data. The data for the time series is stored in an R object called time-series object. It is also an R data object like a vector or data frame. The time series object is created by using the ts() function and is easier to manipulate post conversion.

```r
#is the positive cases field a ts object or not
print(is.ts(data$pos))
```

```
## [1] FALSE
```

```r
#converting to ts object
train.ts <- ts(train$pos,frequency = 30)
test.ts <- ts(test$pos,frequency = 30)
print(is.ts(train.ts))
```

```
## [1] TRUE
```

```r
frequency(train.ts)
```

```
## [1] 30
```

Our time series training set has 215 observations from Jan-Sep while our test set has 73 observations from Oct-Dec.

Series Decomposition

Time series decomposition is a mathematical procedure which transforms a time series into multiple different time series. The original time series is often split into 3 component series:

1. Seasonal: Patterns that repeat with a fixed period of time. For example, a website might receive more visits during weekends; this would produce data with a seasonality of 7 days.

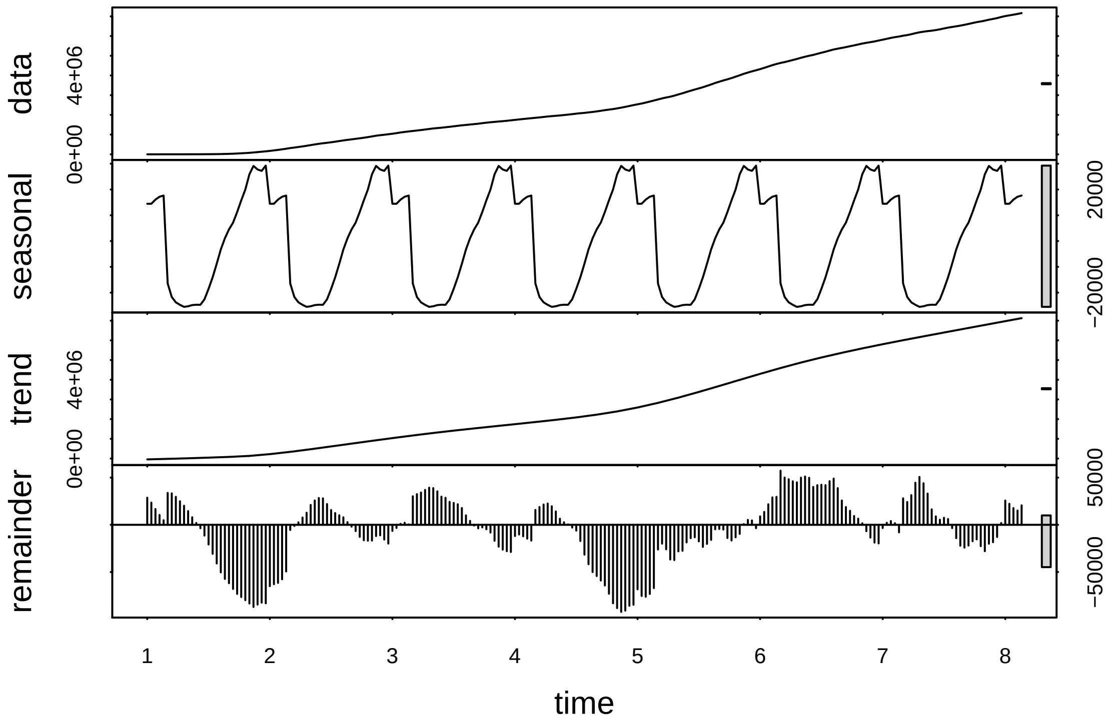
2. Trend: The underlying trend of the metrics. A website increasing in popularity should show a general trend that goes up.

3. Random: Also call "noise", "irregular" or "remainder," this is the residuals of the original time series after the seasonal and trend series are removed.

We will try and decompose the covid positive cases using the stl() function in R.

Series Decomposition with stl()

```
#decompose covid cases into trend, seasonal and remainder using stl
dcomposeclose=stl(train.ts, "periodic")
plot(dcomposeclose)
```



We see a clear trend and seasonal pattern in the data.

Testing the series for stationarity, autocorrelation and normality

We now want to analyze the covid series for properties that make forecasting simpler and give us more insight into its nature.

1. Stationarity - A stationary time series is one whose statistical properties such as mean, variance, autocorrelation, etc. are all constant over time. Most statistical forecasting methods are based on the assumption that the time series can be rendered approximately stationary (i.e., "stationarized") through the use of mathematical transformations. A stationarized series is relatively easy to predict. We use adf.test (Augmented Dickey Fuller Test) function in R to check whether a series is stationary or not. This function computes the augmented Dickey-Fuller statistic for testing the null hypothesis that there exists a unit root at the zero frequency.

2. Autocorrelation - The Box Ljung test will be used to determine if the residuals are independent or not. We will use the Box.test function in R for this.

3. Normality - can be tested by the Jarque-Bera Test. The test statistic is always nonnegative. If it is far from zero, it signals the data does not have a normal distribution. We will use jarque.bera.test function in R for this.

Stationarity

```r
#testing stationarity for covid
adf.test(train.ts)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
## [1,]   0 26.360   0.990
## [2,]   1  1.106   0.927
## [3,]   2  1.201   0.940
## [4,]   3  0.788   0.870
## [5,]   4  0.185   0.697
## Type 2: with drift no trend
##      lag    ADF p.value
## [1,]   0 12.150   0.990
## [2,]   1  1.070   0.990
## [3,]   2  1.197   0.990
## [4,]   3  0.813   0.990
## [5,]   4  0.225   0.973
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -5.78   0.010
## [2,]   1 -2.00   0.576
## [3,]   2 -1.97   0.586
## [4,]   3 -1.94   0.599
## [5,]   4 -2.01   0.571
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

We see a p-value of 0.01 which indicates that we can reject the null and conclude that the series is stationary.

Autocorrelation

```r
#box jung test
Box.test(train.ts,type='Ljung')
```

```
##
##  Box-Ljung test
##
## data:  train.ts
## X-squared = 212.74, df = 1, p-value < 2.2e-16
```

We have significant p-value indicating the presence of autocorrelation in residuals.

normality

```r
#jarque bera test
tseries::jarque.bera.test(train.ts)
```

```
##
```

```
##  Jarque Bera Test
##
## data:  train.ts
## X-squared = 19.56, df = 2, p-value = 5.659e-05
```

We again have significant p-values indicating the presence of non-normal covid cases.

First iteration - ARIMA model

ARIMA is an acronym that stands for AutoRegressive Integrated Moving Average. It is a class of model that captures a suite of different standard temporal structures in time series data. This acronym is descriptive, capturing the key aspects of the model itself. Briefly, they are:

1. AR: Autoregression. A model that uses the dependent relationship between an observation and some number of lagged observations.

2. I: Integrated. The use of differencing of raw observations (e.g. subtracting an observation from an observation at the previous time step) in order to make the time series stationary.

3. MA: Moving Average. A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.

Each of these components are explicitly specified in the model as a parameter. A standard notation is used of ARIMA(p,d,q) where the parameters are substituted with integer values to quickly indicate the specific ARIMA model being used.

The parameters of the ARIMA model are defined as follows:

1. p: The number of lag observations included in the model, also called the lag order.

2. d: The number of times that the raw observations are differenced, also called the degree of differencing.

3. q: The size of the moving average window, also called the order of moving average.

A key note here is that adopting an ARIMA model for a time series assumes that the underlying process that generated the observations is an ARIMA process. Whether this is true or not, can be found out once we look at the results of the modeling process.

ARIMA model form

We use auto.arima () for this

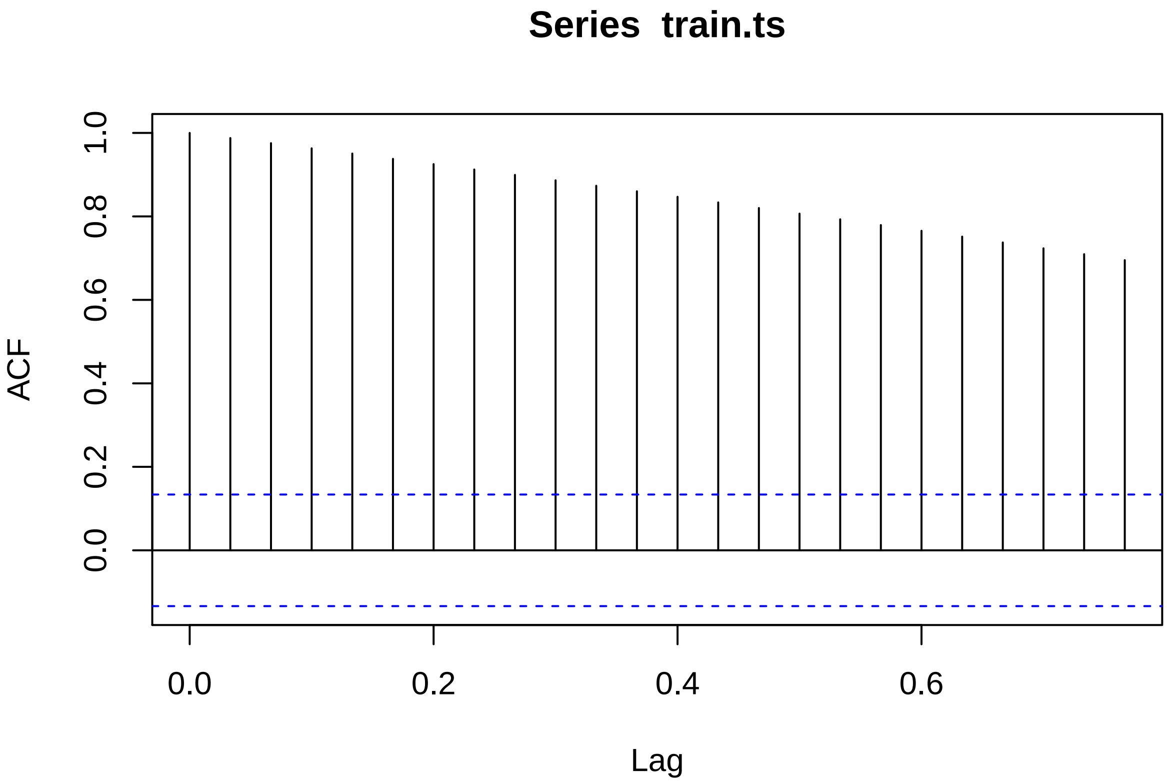
```r
#arima to check order for covid cases for last year
fit_diff_covid <-auto.arima(train.ts)
fit_diff_covid
```

```
## Series: train.ts
## ARIMA(2,2,2)
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.2367  -0.9553  -1.3477  0.8910
## s.e.  0.0313   0.0454   0.0531  0.0634
##
## sigma^2 estimated as 15359547:  log likelihood=-2063.76
## AIC=4137.52   AICc=4137.81   BIC=4154.33
```

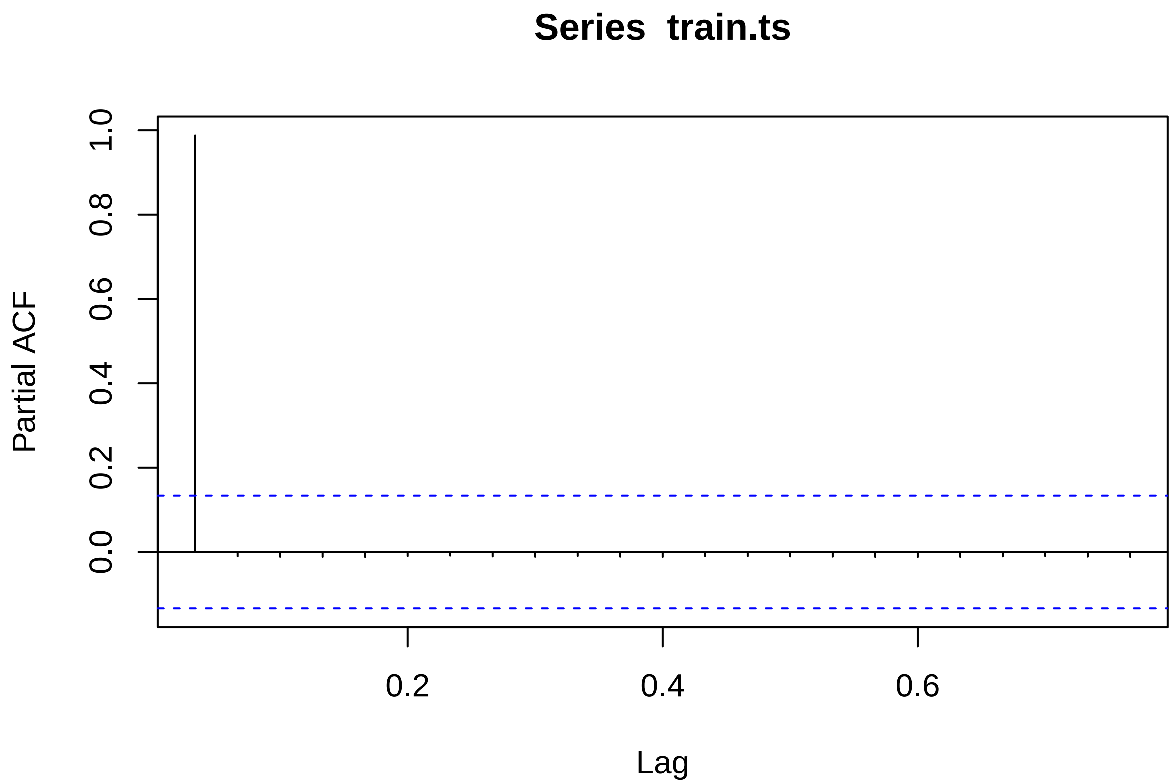
We get a arima model of the form ARIMA(2,2,2) indicating difference of 2 AR of 2 and MA of 2. Our AIC is 4137.

Visually understanding the AR and MA terms through ACF and PACF plots

```
#acf and pacf plots for covid series
acf(train.ts)
```




```
pacf(train.ts)
```



We don't see any meaningful insight by analyzing covid series. However, the significant lag at 1 in PACF plot and all lags significant in ACF indicate that autocorrelation from lag 2 to n is propogated by the autocorrelation at lag 1.

Estimating Final ARIMA model selected on AIC

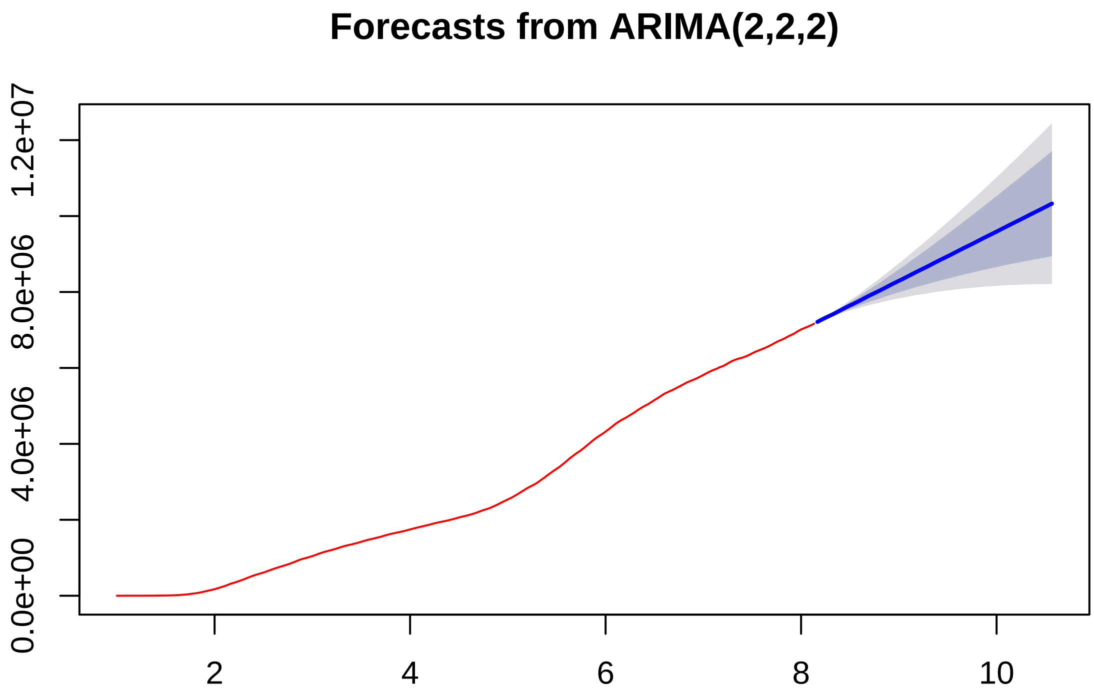
```
predicted <- forecast::forecast(fit_diff_covid,h=73)
predicted
```

```
##          Point Forecast   Lo 80   Hi 80   Lo 95   Hi 95
##  8.166667        7211850 7206827 7216873 7204169 7219531
##  8.200000        7257982 7247247 7268717 7241565 7274400
##  8.233333        7301799 7284986 7318612 7276086 7327513
##  8.266667        7343019 7320034 7366004 7307866 7378171
##  8.300000        7383237 7353742 7412731 7338129 7428345
##  8.333333        7424698 7387857 7461540 7368354 7481043
##  8.366667        7468654 7423234 7514075 7399190 7538119
##  8.400000        7514507 7459284 7569731 7430050 7598964
##  8.433333        7560323 7494486 7626160 7459634 7661012
##  8.466667        7604281 7527540 7681021 7486916 7721645
##  8.500000        7645977 7558325 7733628 7511925 7780028
##  8.533333        7686649 7587996 7785303 7535772 7837527
##  8.566667        7728218 7618145 7838292 7559875 7896561
##  8.600000        7771873 7649639 7894106 7584933 7958812
##  8.633333        7817250 7682023 7952476 7610439 8024061
##  8.666667        7862765 7713906 8011625 7635105 8090426
##  8.700000        7906807 7744006 8069607 7657825 8155789
##  8.733333        7948892 7772072 8125713 7678469 8219316
```

```
## 8.766667      7989967 7799039  8180896 7697967  8281968
## 8.800000      8031662 7826328  8236995 7717631  8345692
## 8.833333      8075086 7854810  8295362 7738203  8411969
## 8.866667      8120060 7884205  8355915 7759352  8480768
## 8.900000      8165296 7913333  8417259 7779951  8550641
## 8.933333      8209377 7941003  8477751 7798935  8619819
## 8.966667      8251778 7966875  8536682 7816057  8687500
## 9.000000      8293207 7991673  8594740 7832051  8754362
## 9.033333      8335036 8016632  8653439 7848079  8821992
## 9.066667      8378290 8042593  8713986 7864886  8891693
## 9.100000      8422924 8069420  8776427 7882286  8963561
## 9.133333      8467902 8096137  8839668 7899337  9036468
## 9.166667      8511990 8121674  8902306 7915053  9108927
## 9.200000      8554645 8145635  8963656 7929117  9180173
## 9.233333      8596381 8168563  9024199 7942089  9250673
## 9.266667      8638348 8191514  9085182 7954974  9321722
## 9.300000      8681479 8215280  9147678 7968489  9394468
## 9.333333      8725829 8239835  9211822 7982565  9469092
## 9.366667      8770574 8264388  9276760 7996429  9544719
## 9.400000      8814643 8287988  9341299 8009194  9620093
## 9.433333      8857500 8310217  9404783 8020503  9694497
## 9.466667      8899502 8331468  9467536 8030769  9768235
## 9.500000      8941605 8352632  9530579 8040848  9842363
## 9.533333      8984651 8374440  9594862 8051414  9917888
## 9.566667      9028765 8396947  9660582 8062483  9995046
## 9.600000      9073300 8419523  9727077 8073434 10073165
## 9.633333      9117335 8441335  9793335 8083482 10151188
## 9.666667      9160350 8461959  9858740 8092253 10228446
## 9.700000      9202580 8481667  9923493 8100039 10305121
## 9.733333      9244815 8501204  9988426 8107560 10382070
## 9.766667      9287805 8521235 10054375 8115438 10460173
## 9.800000      9331725 8541873 10121576 8123751 10539698
## 9.833333      9376072 8562622 10189523 8132007 10620137
## 9.866667      9420061 8582761 10257361 8139521 10700601
## 9.900000      9463198 8601876 10324520 8145919 10780477
## 9.933333      9505623 8620141 10391106 8151395 10859852
## 9.966667      9547983 8638174 10457792 8156550 10939416
## 10.000000     9590941 8656571 10525310 8161946 11019936
## 10.033333     9634701 8675485 10593918 8167706 11101697
## 10.066667     9678883 8694532 10663235 8173447 11184319
## 10.100000     9722819 8713094 10732544 8178578 11267060
## 10.133333     9766049 8730775 10801322 8182735 11349362
## 10.166667     9808639 8747675 10869603 8186035 11431243
## 10.200000     9851115 8764300 10937929 8188974 11513255
## 10.233333     9894058 8781179 11006937 8192057 11596059
## 10.266667     9937690 8798490 11076890 8195434 11679946
## 10.300000     9981727 8815941 11147512 8198812 11764641
## 10.333333    10025606 8833008 11218203 8201685 11849526
## 10.366667    10068904 8849319 11288488 8203710 11934097
## 10.400000    10111633 8864916 11358350 8204944 12018322
## 10.433333    10154215 8880210 11428220 8205792 12102638
## 10.466667    10197157 8895667 11498647 8206700 12187614
## 10.500000    10240686 8911477 11569895 8207836 12273536
## 10.533333    10284596 8927424 11641768 8208981 12360212
```

```
## 10.566667       10328418 8943067 11713768 8209707 12447129
```

```r
plot(predicted,col = 'red')
```



Computing accuracy of the model

We compute MAPE to check accuracy of the arima model.

```r
set.seed(7)

prediction_test <- as.data.frame(predicted)

comparison <- as.data.frame(cbind(prediction_test$`Point Forecast`,test$pos))
comparison$mape <- (abs(comparison$V2-comparison$V1)/abs(comparison$V2))
summary(comparison$mape)
```

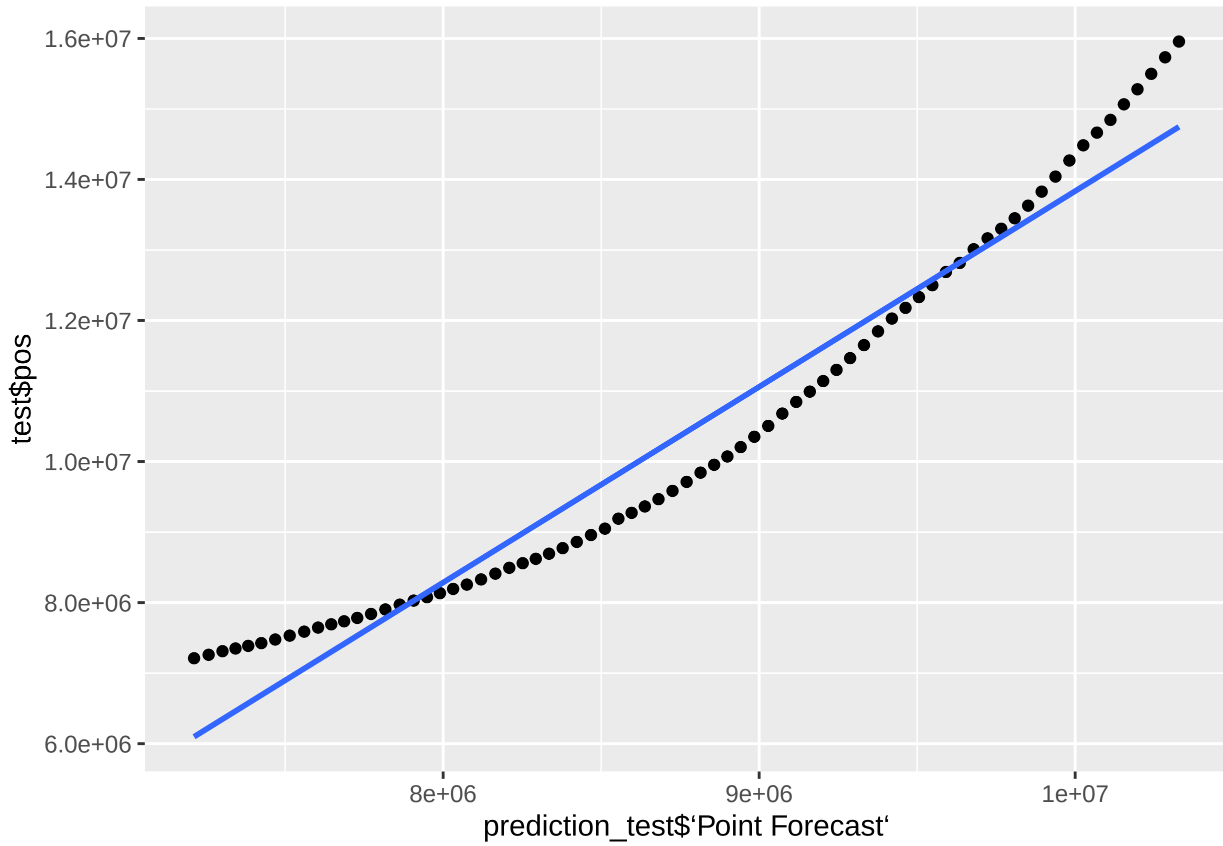
```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0001123 0.0176894 0.0969712 0.1295928 0.2362380 0.3527114
```

We get an accuracy median MAPE of 9.6% and average MAPE of 12.9% which is decent for a base ARIMA model.

We plot the predicted vs actual plot for the test set.

```r
ggplot(comparison,aes(x = prediction_test$`Point Forecast`, y = test$pos) ) +
     geom_point() +
     geom_smooth(method = "lm", se = FALSE)
```

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


1.6e+07
1.4e+07
1.2e+07
1.0e+07
8.0e+06
6.0e+06
test$pos
8e+06
9e+06
1e+07
prediction_test$‘Point Forecast‘

We now explore ARIMAX

ARIMAX is basically ARIMA but with an x variable - our key hypothesis is that the total positive cases per day (detected) can be influenced by testing i.e more the tests conducted, more will be the detected positive cases per day. We use the totalTest and totalTestInc variables as our explantory variable in this iteration

We first put these variables into a matrix form -

```
totaltest_mat <- as.matrix(train$totalTest,train$totalTestInc)
```

and then pass it in the xreg parameter -

```
#arima to check order for covid cases for last year
fit_diff_covid <- auto.arima(train.ts,xreg = totaltest_mat)
fit_diff_covid
```

```
## Series: train.ts
## Regression with ARIMA(2,2,2) errors
##
## Coefficients:
##          ar1      ar2      ma1     ma2    xreg
##       1.2106  -0.9047  -1.3834  0.8557  0.0130
## s.e.  0.0397   0.0599   0.0549  0.0556  0.0041
##
## sigma^2 estimated as 14651615:  log likelihood=-2057.92
## AIC=4127.84   AICc=4128.25   BIC=4148.01
```

We see a model of form 2,2,2 with AIC of 4127.8

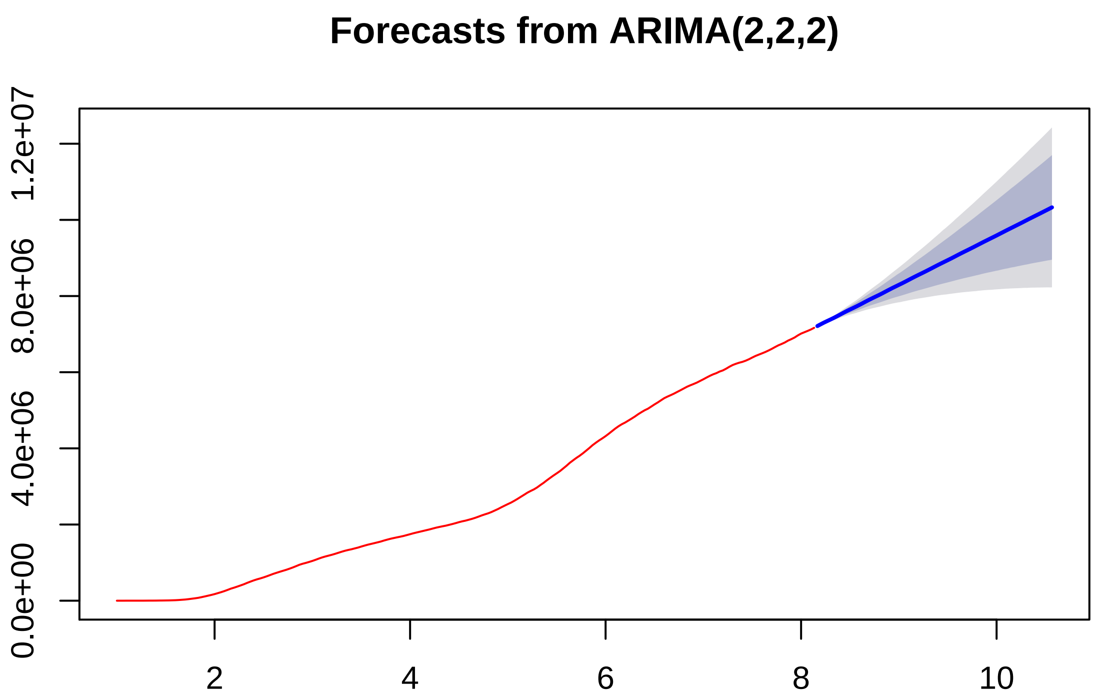
```
#estimation
arima.final <- arima(train.ts,c(2,2,2))
```

```
predicted <- forecast::forecast(arima.final,h=73)
predicted
```

```
##          Point Forecast   Lo 80   Hi 80   Lo 95   Hi 95
## 8.166667        7211850 7206875 7216825 7204241 7219459
## 8.200000        7257982 7247349 7268616 7241720 7274245
## 8.233333        7301799 7285145 7318454 7276329 7327270
## 8.266667        7343019 7320251 7365787 7308198 7377839
## 8.300000        7383237 7354020 7412453 7338554 7427919
## 8.333333        7424698 7388204 7461192 7368885 7480511
## 8.366667        7468654 7423663 7513646 7399846 7537463
## 8.400000        7514507 7459805 7569210 7430847 7598167
## 8.433333        7560323 7495107 7625539 7460584 7660062
## 8.466667        7604281 7528264 7680297 7488024 7720538
## 8.500000        7645977 7559152 7732801 7513189 7778764
## 8.533333        7686649 7588926 7784372 7537195 7836104
## 8.566667        7728218 7619183 7837253 7561463 7894973
## 8.600000        7771873 7650792 7892953 7586697 7957049
## 8.633333        7817250 7683299 7951200 7612390 8022109
## 8.666667        7862765 7715311 8010220 7637253 8088278
## 8.700000        7906807 7745542 8068071 7660174 8153440
## 8.733333        7948892 7773740 8124045 7681020 8216765
## 8.766667        7989967 7800840 8179095 7700722 8279213
## 8.800000        8031662 7828265 8235058 7720593 8342730
## 8.833333        8075086 7856889 8293284 7741382 8408791
## 8.866667        8120060 7886431 8353689 7762755 8477365
```

```
## 8.900000         8165296 7915710  8414882 7783587  8547005
## 8.933333         8209377 7943535  8475219 7802807  8615947
## 8.966667         8251778 7969563  8533994 7820167  8683390
## 9.000000         8293207 7994518  8591895 7836402  8750012
## 9.033333         8335036 8019636  8650436 7852673  8817398
## 9.066667         8378290 8045760  8710819 7869730  8886850
## 9.100000         8422924 8072755  8773092 7887387  8958461
## 9.133333         8467902 8099645  8836160 7904701  9031104
## 9.166667         8511990 8125356  8898624 7920684  9103295
## 9.200000         8554645 8149493  8959797 7935019  9174271
## 9.233333         8596381 8172599  9020163 7948262  9244500
## 9.266667         8638348 8195729  9080967 7961421  9315275
## 9.300000         8681479 8219678  9143279 7975216  9387742
## 9.333333         8725829 8244420  9207237 7989577  9462080
## 9.366667         8770574 8269163  9271984 8003732  9537415
## 9.400000         8814643 8292957  9336330 8016793  9612494
## 9.433333         8857500 8315381  9399620 8028400  9686600
## 9.466667         8899502 8336827  9462177 8038965  9760039
## 9.500000         8941605 8358188  9525023 8049346  9833865
## 9.533333         8984651 8380197  9589105 8060218  9909084
## 9.566667         9028765 8402908  9654622 8071600  9985930
## 9.600000         9073300 8425691  9720909 8082867 10063732
## 9.633333         9117335 8447712  9786957 8093236 10141434
## 9.666667         9160350 8468548  9852151 8102330 10218369
## 9.700000         9202580 8488468  9916692 8110440 10294720
## 9.733333         9244815 8508219  9981411 8118289 10371341
## 9.766667         9287805 8528467 10047143 8126498 10449112
## 9.800000         9331725 8549325 10114125 8135147 10528302
## 9.833333         9376072 8570296 10181848 8143744 10608400
## 9.866667         9420061 8590660 10249462 8151602 10688520
## 9.900000         9463198 8610002 10316394 8158347 10768049
## 9.933333         9505623 8628495 10382752 8164171 10847076
## 9.966667         9547983 8646757 10449209 8169677 10926289
## 10.000000        9590941 8665386 10516495 8175427 11006454
## 10.033333        9634701 8684534 10584869 8181546 11087857
## 10.066667        9678883 8703818 10653948 8187650 11170117
## 10.100000        9722819 8722620 10723018 8193147 11252492
## 10.133333        9766049 8740542 10791555 8197672 11334425
## 10.166667        9808639 8757685 10859594 8201343 11415935
## 10.200000        9851115 8774553 10927676 8204655 11497574
## 10.233333        9894058 8791678 10996437 8208114 11580002
## 10.266667        9937690 8809237 11066143 8211870 11663509
## 10.300000        9981727 8826939 11136514 8215632 11747821
## 10.333333       10025606 8844259 11206952 8218892 11832319
## 10.366667       10068904 8860825 11276983 8221306 11916501
## 10.400000       10111633 8876678 11346588 8222932 12000334
## 10.433333       10154215 8892229 11416201 8224174 12084256
## 10.466667       10197157 8907946 11486369 8225478 12168836
## 10.500000       10240686 8924017 11557355 8227014 12254358
## 10.533333       10284596 8940228 11628965 8228563 12340630
## 10.566667       10328418 8956137 11700699 8229695 12427140
```

```r
plot(predicted,col = 'red')
```



Computing accuracy of the model

We compute MAPE to check accuracy of the arima model.

```
set.seed(7)

prediction_test <- as.data.frame(predicted)

comparison <- as.data.frame(cbind(prediction_test$`Point Forecast`,test$pos))
comparison$mape <- (abs(comparison$V2-comparison$V1)/abs(comparison$V2))
summary(comparison$mape)
```

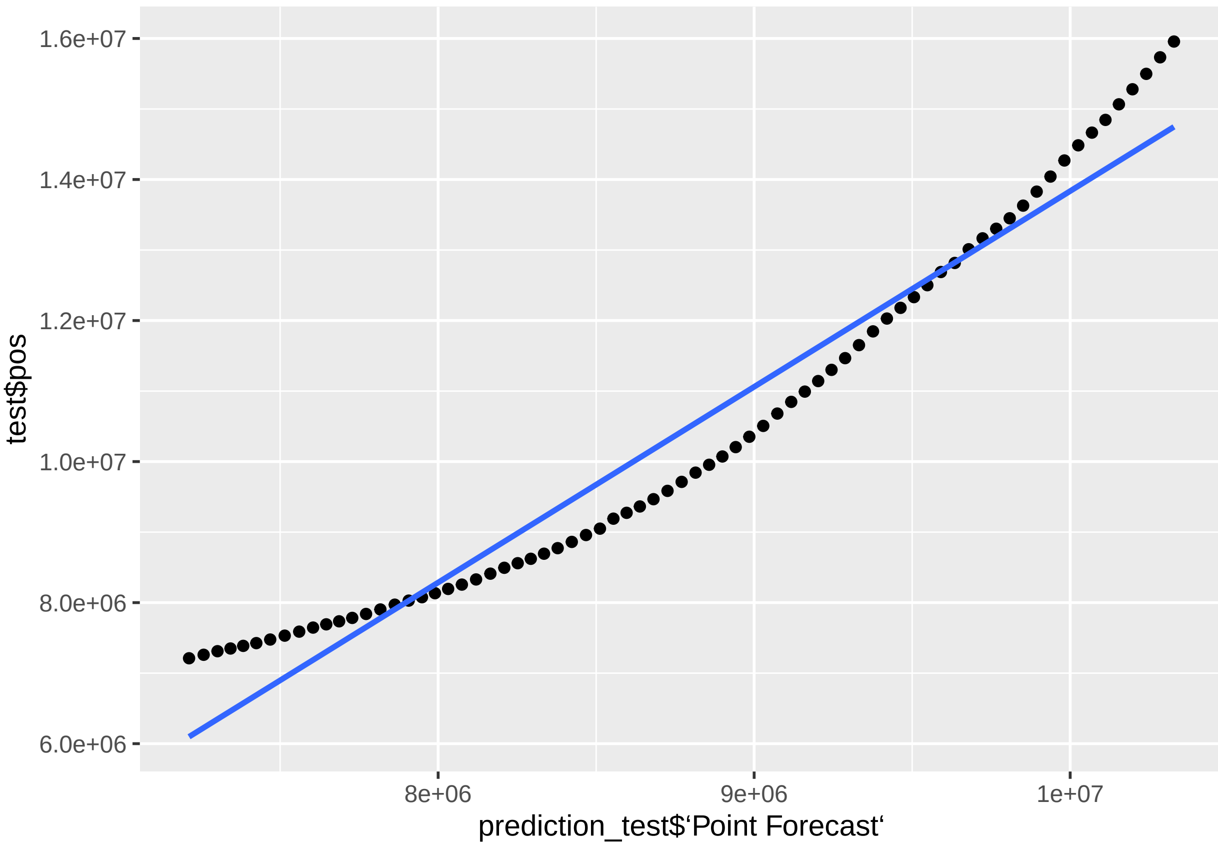
```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0001123 0.0176894 0.0969712 0.1295928 0.2362380 0.3527114
```

We get an accuracy median MAPE of 9.6% and average MAPE of 12.9% which is decent for a base ARIMAX model with two variables.

We plot the predicted vs actual once again for this model.

```
ggplot(comparison,aes(x = prediction_test$`Point Forecast`, y = test$pos) ) +
     geom_point() +
     geom_smooth(method = "lm", se = FALSE)
```

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



We don't see much improvement in adding the test variables to ARIMA.

We feel given the seasonality is unknown since we have less than a year of data, models like ARIMA and ARIMAX are not able to forecast with a much higher precision.

Future Scope - We also know about the COVID-19 Open Research Dataset Challenge (CORD-19)(https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge). This was created in response to the COVID-19 pandemic, the White House and a coalition of leading research groups have prepared the COVID-19 Open Research Dataset (CORD-19). CORD-19 is a resource of over 51,000 scholarly articles, including over 40,000 with full text, about COVID-19, SARS-CoV-2, and related coronaviruses. This freely available dataset is provided to the global research community to apply recent advances in natural language processing and other AI techniques to generate new insights in support of the ongoing fight against this infectious disease. This dataset is ideal for performing Text mining. Also besides this dataset, offered by US CDC, we can can explore if there are any significant factors such as income, gdp, and education level that may potentially affect COVID 19 outbreak. This can be extracted as well and we feel utilizing the above 3 datasets, we can have not only a good prediction of total cases per day, but also about what factors influence those cases. We can also try an LSTM model for this problem.

Appendix -

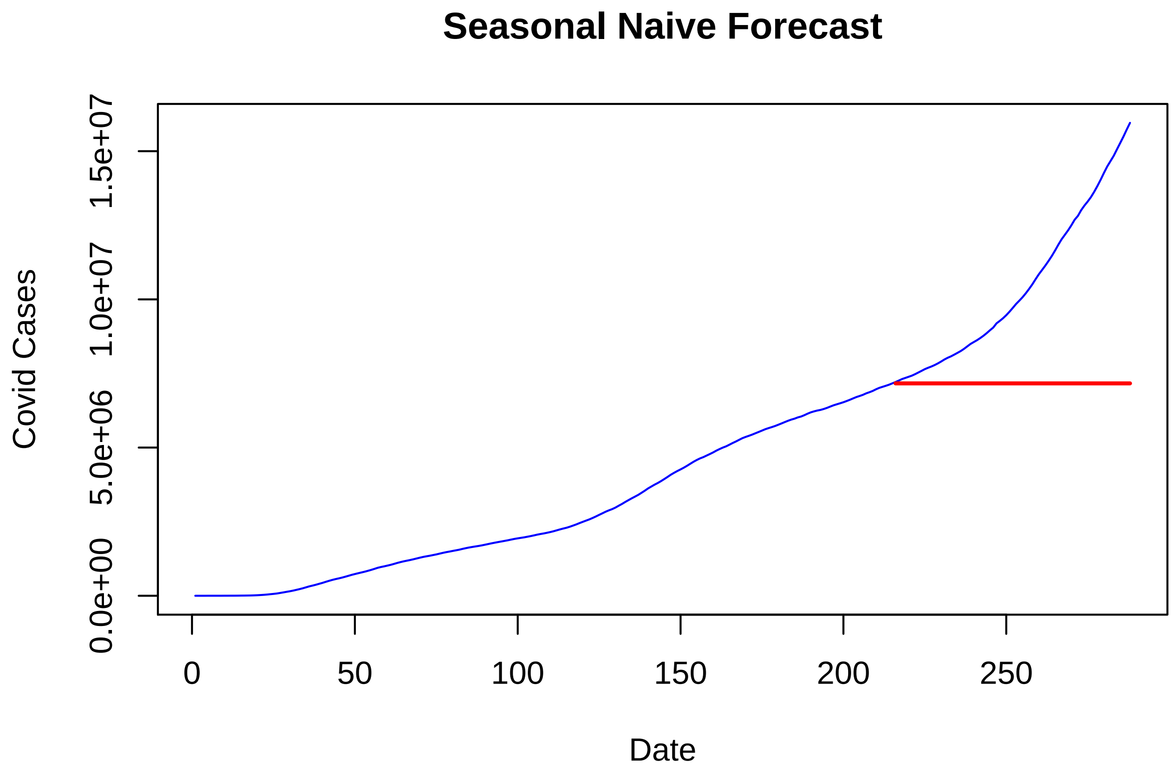
We tried some additional modeling methods and got the following results for COVID prediction however we did not include these methods as the MAPE wasn't as great as ARIMA models. We tried naive, ETS, TBATS models in R. Feel free to peruse.

Naive forecast

```
naive = snaive(train$pos, h=length(test$pos))
MAPE(naive$mean, test$pos) * 100
```

```
## [1] 27.1352
```

```
plot(data$pos, col="blue", xlab="Date", ylab="Covid Cases", main="Seasonal Naive Forecast", type='l')
lines(naive$mean, col="red", lwd=2)
```



ETS - Exponential models

```
ets_model = ets(train$pos, allow.multiplicative.trend = TRUE)
summary(ets_model)
```

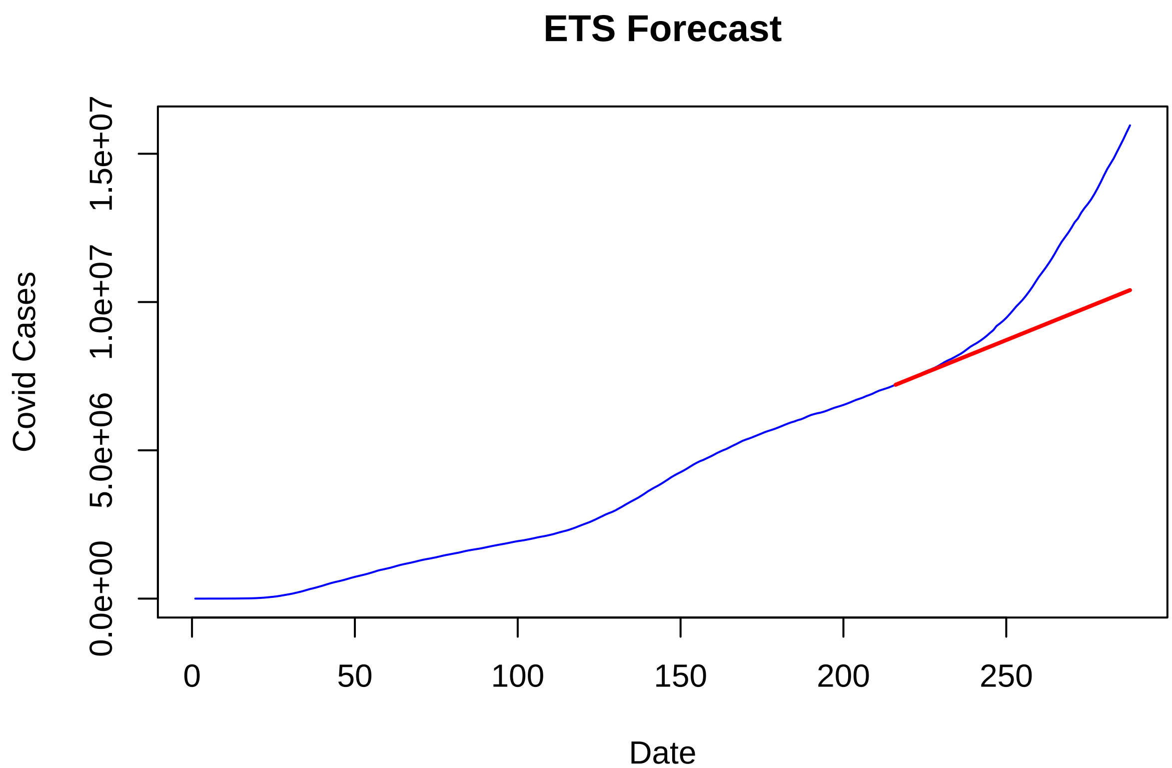
```
## ETS(A,A,N)
##
## Call:
##  ets(y = train$pos, allow.multiplicative.trend = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.9999
```

```
##     beta  = 0.9999
##
##   Initial states:
##     l = -235.5307
##     b = 140.0045
##
##   sigma:  4862.555
##
##      AIC     AICc      BIC
## 4811.057 4811.344 4827.910
##
## Training set error measures:
##                    ME    RMSE      MAE      MPE     MAPE      MASE
## Training set 205.6272 4817.11 3515.994 1.576347 5.958993 0.1050075
##                   ACF1
## Training set 0.0278641
```

```
ets_forecast = forecast::forecast(ets_model, h=length(test$pos))
MAPE(ets_forecast$mean, test$pos) *100
```

```
## [1] 12.64343
```

```
plot(data$pos, col="blue", xlab="Date", ylab="Covid Cases", main="ETS Forecast", type='l')
lines(ets_forecast$mean, col="red", lwd=2)
```



TBATS forecast

```
tbats_model = tbats(train$pos)
tbats_forecast = forecast::forecast(tbats_model, h=length(test$pos))
```

```r
MAPE(tbats_forecast$mean, test$pos) * 100
```

```
## [1] 12.45552
```

```r
plot(data$pos, col="blue", xlab="Date", ylab="Covid Cases", main="TBATS Forecast", type='l')
lines(tbats_forecast$mean, col="red", lwd=2)
```