TIME SERIES PROJECT

Monthly Gasoline Prices in the U.S.A (1992 – 2017)

Team:

Apurva Kenekar
Heet Shah
Yu Ping Chuang (Amy)

MA611 Spring 2018
Prof. Pannapa Changpetch

INTRODUCTION: -

Source of data

Data: Monthly Gasoline Retail Prices in U.S.D. (Jan 1992 - Dec 2017)
Source: U.S. Energy Information Administration
https://www.eia.gov/petroleum/data.php - prices
Motor gasoline retail prices, U.S. city average
Release date: February 26, 2018 | data from: Monthly Energy Review
312 observations (Jan 1992 - Dec 2017)

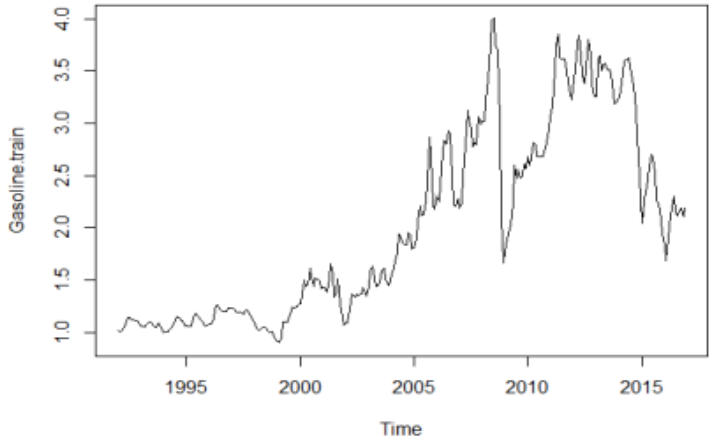
Our data is an univariate data which has only single variable gasoline retailer prices, period from Jan 1992-Dec 2017.

Interest and Objective

Gasoline price is relevant to almost everyone, from drivers who care about the price fluctuations, to the impact it has on the broadly industries which have chain effects on the macroeconomic, as well as the importance it is been treated a strategic raw material internationally. The significance of gasoline price driven us to study for the retail gasoline price, and motivated to create a most appropriate model to predict the future gasoline prices.

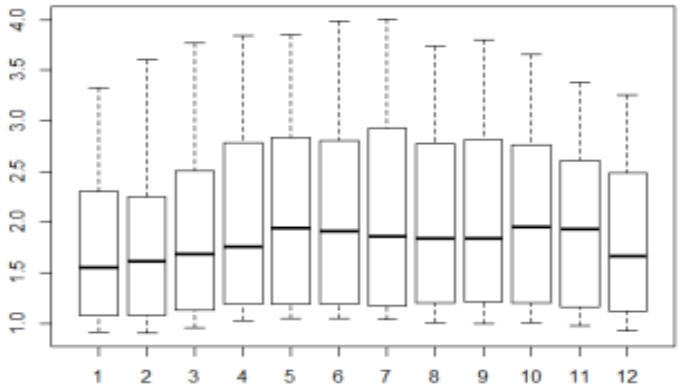
DATA VISUALIZATION: -

```
> Gasoline.train<-ts(Gasoline.Ratailer.prices, start=
c(1992,1),end=c(2016,12),freq=12)
> Gasoline.test<-ts(Gasoline.Ratailer.prices, start=
c(1992,1),end=c(2017,12),freq=12)
> Gasoline.test<-window(Gasoline.test,start = c(2017,1), end =
c(2017,12),frequency=12)
> plot(Gasoline.train)
```



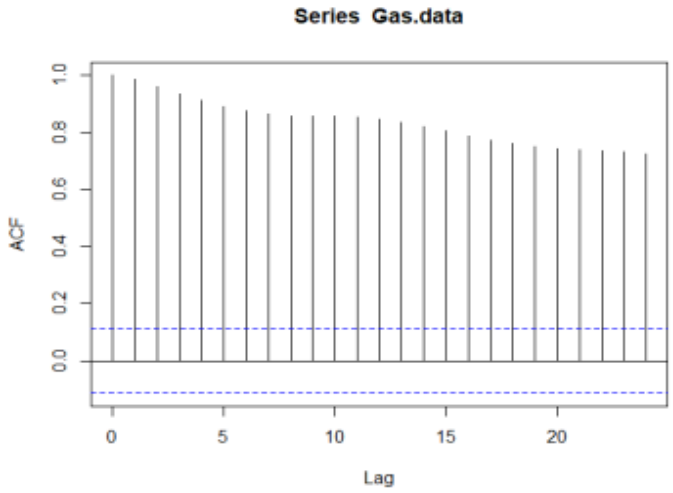
The original time series plot shows this is a non-stationary series, with an upward trend from year 1992-2007, then encountered the financial crisis in year 2008-2010, the gas price slumped. After that it gradually climb back between year 2010-2015, then it plummet again in year 2015-2016 caused by lowest crude oil price since year 2009.

```
> boxplot(Gasoline.train~cycle(Gasoline.train))
```



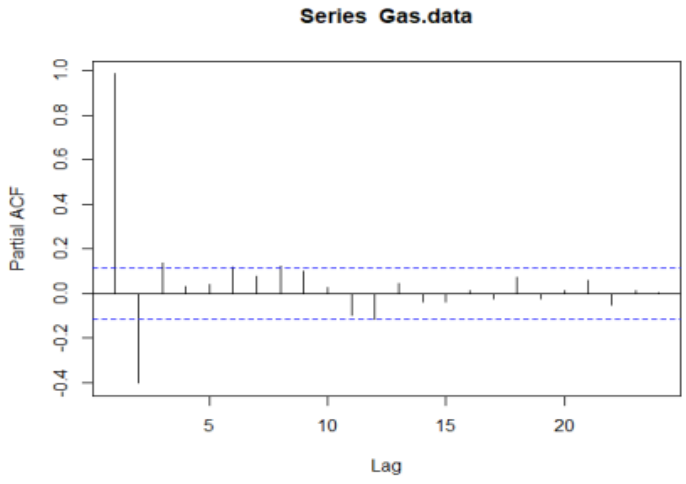
From the box plot we can see there is a seasonal effect of the gasoline price, though not very obvious. It shows higher prices in summer (May-Jun) and fall seasons (Oct-Nov), and lower prices in winter (Dec-Feb).

```
> Gas.data=coredata(Gasoline.train)
> sd(Gas.data)
[1] 0.9099515 ###this is the original data standard deviation
> acf(Gas.data)
```



The ACF plot shows slowly decay from value 1. We might need to difference our series when we perform the AIRMA models.

```
> pacf(Gas.data)
```



The PACF looks like cut off after lag 3, which is AR(3).

DECOMPOSITION MODELS: -

First, we read the data and created training & testing sets

```
> tsproject = read.csv(file = "611project.csv", header = FALSE, sep = ",")
> Gas.data = tsproject[,2]
> Gas.ts <- ts(tsproject[,2], start = c(1992,1), end = c(2016,12), freq = 12)
> test.gas <- ts(tsproject[,2], start = c(1992, 1), end = c(2017, 12),
frequency = 12)
> test.gas = window(test.gas, start = c(2017, 1), end = c(2017, 12),
frequency = 12)
> sd(Gas.ts[7:294])
[1] 0.9180352
```

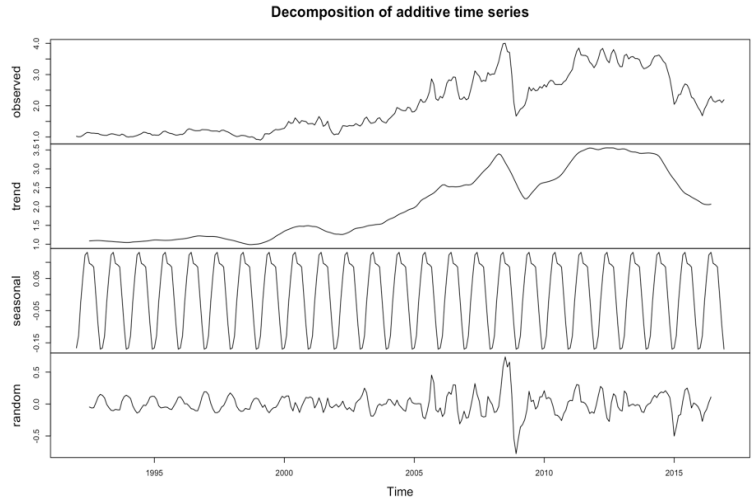
This is the standard deviation of the data after removing 12 observations from the data so we can accurately compare the models below with the standard deviation of the data.

Additive Model using Decompose function:

```
> #Additive model
> Gas.dadd <- decompose(Gas.ts, type = "additive")
> str(Gas.dadd)
List of 6
 $ x       : Time-Series [1:300] from 1992 to 2017: 1.02 1.01 1.01 1.05 1.11
...
 $ seasonal: Time-Series [1:300] from 1992 to 2017: -0.1658 -0.1276 -0.0226
0.0572 0.123 ...
 $ trend   : Time-Series [1:300] from 1992 to 2017: NA NA NA NA NA ...
 $ random  : Time-Series [1:300] from 1992 to 2017: NA NA NA NA NA ...
 $ figure  : num [1:12] -0.1658 -0.1276 -0.0226 0.0572 0.123 ...
 $ type    : chr "additive"
 - attr(*, "class")= chr "decomposed.ts"
```

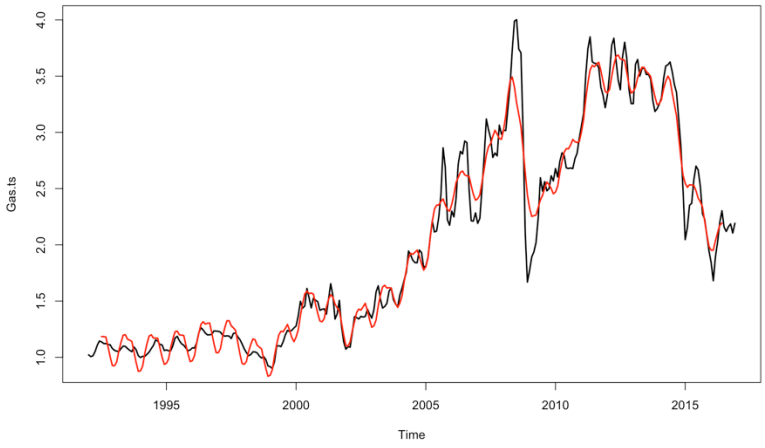
The structure of the additive model shows that since the decompose function uses moving average method to calculate the trend component, the initial values of the trend & random components are NAs.

```
> plot(Gas.dadd)
```



The plot of the additive decompose function shows the individual trend, seasonal & random components. We can see that both the trend & random components show some kind of pattern.

```
> predda = Gas.dadd$trend + Gas.dadd$seasonal
> layout(1:1)
> plot(Gas.ts, lwd=2)
> lines(predda, col = "red", lwd=2)
```



We can see that the additive decomposition model fits the data well.

```
> Gas.daddresid = window(Gas.dadd$random, start=c(1992, 7), end=c(2016, 6))
> rmse.dadd = sqrt(sum(Gas.daddresid^2)/length(Gas.daddresid))
> rmse.dadd
[1] 0.1646889
```

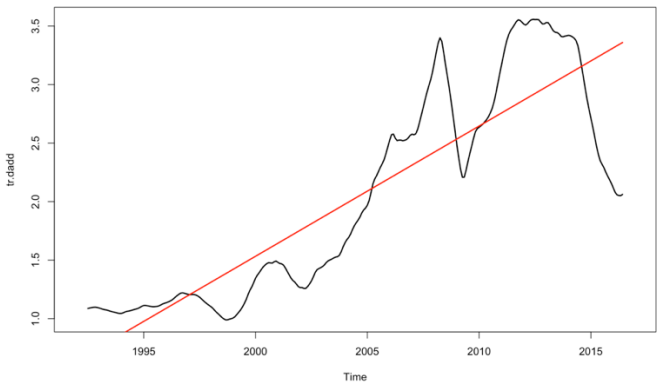
The RMSE of the decompose additive model shows that we have reduce the variability with a sd from 0.90995 to 0.1646889 on the training data set.

Now, we will try to fit the model on the testing data to find out the test RMSE.

```
> tr.dadd = Gas.dadd$trend
> plot(tr.dadd)
> tim.dadd = time(tr.dadd)
> ti.dadd = unclass(tim.dadd)
> trreg1 = lm(tr.dadd ~ ti.dadd)
> summary(trreg1)

Call:
lm(formula = tr.dadd ~ ti.dadd)

Residuals:
     Min       1Q   Median       3Q      Max
-1.29934 -0.29255  0.04244  0.28631  0.94825

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.210e+02  7.398e+00  -29.87   <2e-16 ***
ti.dadd      1.112e-01  3.691e-03   30.14   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.434 on 286 degrees of freedom
  (12 observations deleted due to missingness)
Multiple R-squared:  0.7606,  Adjusted R-squared:  0.7597
F-statistic: 908.5 on 1 and 286 DF,  p-value: < 2.2e-16
```

The model parameters are significant with 75.97% adjusted R^2 and significant global F-statistic.

```
> plot(tr.dadd, lwd = 2, type='l')
> lines(ti.dadd[7:294], fitted(trreg1), col='red', lwd=2)
```



Here, we show the model with fitted values over the actual trend component values. The model doesn't fit the data well as the actual trend values are not linear.

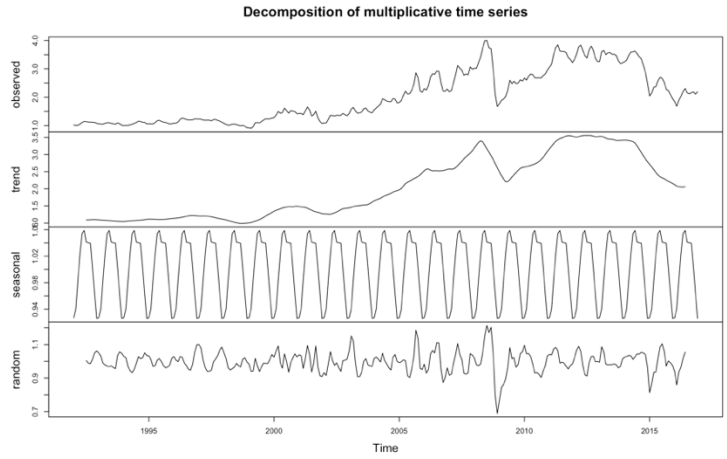
```
> predtime = seq(2017, 2017.917, by=1/12)
> predtime
 [1] 2017.000 2017.083 2017.167 2017.250 2017.333 2017.417 2017.500 2017.583
2017.667
[10] 2017.750 2017.833 2017.917
> preddata1 = predict(trreg1, data.frame(ti.dadd = predtime), se.fit = TRUE)
> preddata1
$fit
       1        2        3        4        5        6        7        8
9       10
3.424629 3.433900 3.443171 3.452441 3.461712 3.470983 3.480253 3.489524
3.498794 3.508065
      11       12
3.517336 3.526606

$se.fit
         1          2          3          4          5          6          7
8
0.05288433 0.05315377 0.05342362 0.05369388 0.05396453 0.05423559 0.05450702
0.05477885
         9         10         11         12
0.05505104 0.05532361 0.05559655 0.05586984

$df
[1] 286

$residual.scale
[1] 0.4339646

> season = c(-0.165838686, -0.127632089, -0.022583478,  0.057209925,
0.122956453,  0.131046730,  0.096753328,  0.092656105,  0.085411314, -
0.004895978, -0.095651186, -0.169432436)
> predvalues = preddata$fit
> spredvalues = predvalues + season
> resid.dadd = test.gas - spredvalues
> resid.dadd
            Jan        Feb        Mar        Apr        May        Jun
Jul
2017 -0.9737906 -1.0792679 -1.1775871 -1.1696512 -1.2816683 -1.3450292 -
1.3660065
```

```
            Aug        Sep        Oct        Nov        Dec
2017 -1.2851799 -1.0142058 -1.0731691 -0.9476845 -0.9691739
> rmse.dadd = sqrt(sum(resid.dadd^2)/ length(resid.dadd))
> rmse.dadd
[1] 1.149457
```

The RMSE of the decomposition additive model applied on the test data gives us a very high test RMSE of 1.149457 compared to the standard deviation of the data which is 0.9099515.

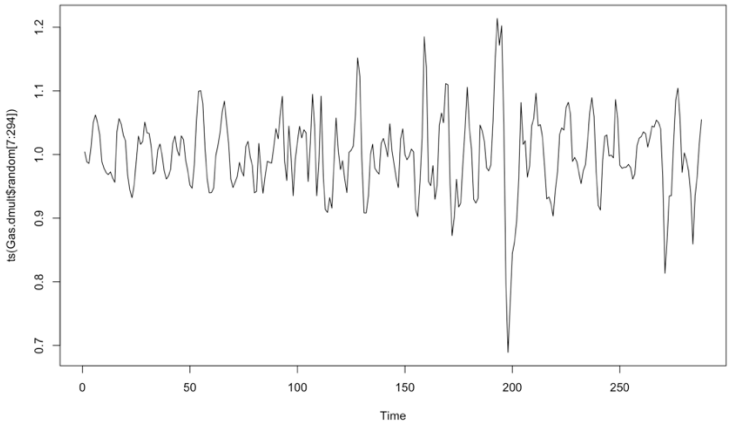
Multiplicative Model using decompose function:

```
> #Multiplicative model
> Gas.dmult <- decompose(Gas.ts, type = "mult")
> plot(Gas.dmult)
```



The plot of the multiplicative decomposition function shows that the trend component fits the data well.

```
> plot(ts(Gas.dmult$random[7:294]))
```

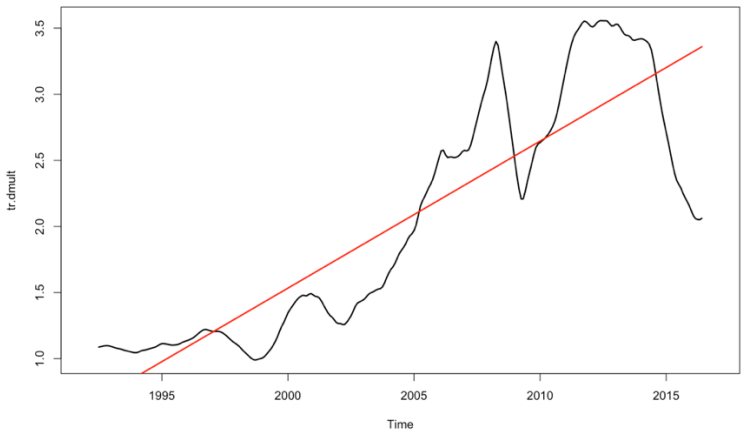


From the plot of the random component of the data, we can see that it is not exactly random but there is some pattern.

```
> sd(Gas.ts[7:294] - (Gas.dmult$trend[7:294]*Gas.dmult$seasonal[7:294]))
[1] 0.1544973
> preddm = Gas.dmult$trend*Gas.dmult$seasonal
> dmult.resid = Gas.ts - preddm
> preddmresid = window(dmult.resid, start = c(1992, 7), end = c(2016, 6))
> rmsedm = sqrt(sum(preddmresid^2)/ length(preddmresid))
> rmsedm
[1] 0.1542288
```

This is the training RMSE of the multiplicative decomposition model.

```
> tr.dmult = Gas.dmult$trend
> plot(tr.dmult)
> tim.dmult = time(tr.dmult)
> ti.dmult = unclass(tim.dmult)
> trreg2 = lm(tr.dmult ~ ti.dmult)
> summary(trreg2)

Call:
lm(formula = tr.dmult ~ ti.dmult)

Residuals:
     Min       1Q   Median       3Q      Max
-1.29934 -0.29255  0.04244  0.28631  0.94825

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.210e+02  7.398e+00  -29.87   <2e-16 ***
ti.dmult     1.112e-01  3.691e-03   30.14   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.434 on 286 degrees of freedom
  (12 observations deleted due to missingness)
Multiple R-squared:  0.7606,  Adjusted R-squared:  0.7597
F-statistic: 908.5 on 1 and 286 DF,  p-value: < 2.2e-16

> plot(tr.dmult, lwd = 2, type='l')
> lines(ti.dmult[7:294], fitted(trreg2), col='red', lwd=2)
```



The plot shows the fitted values of the regression model using the trend component of the multiplicative decomposition model on the trend component.

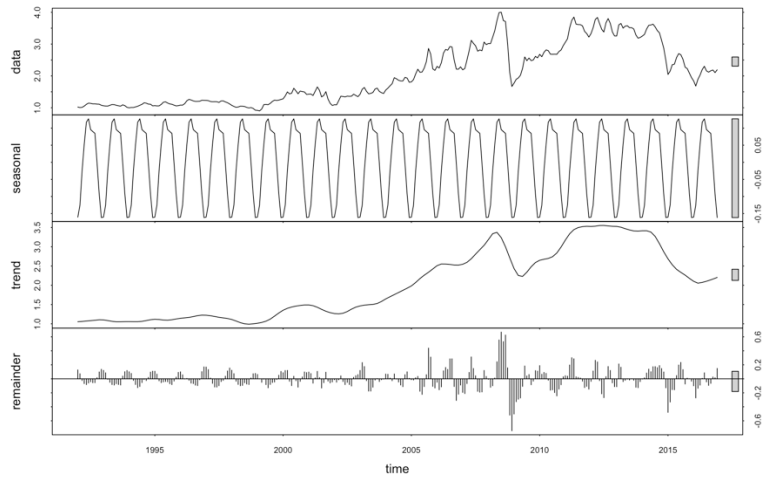
```
> predtime = seq(2017, 2017.917, by=1/12)
> predtime
 [1] 2017.000 2017.083 2017.167 2017.250 2017.333 2017.417 2017.500 2017.583
2017.667
[10] 2017.750 2017.833 2017.917
> preddata = predict(trreg2, data.frame(ti.dmult = predtime), se.fit = TRUE)
> preddata
$fit
       1        2        3        4        5        6        7        8
9       10
3.424629 3.433900 3.443171 3.452441 3.461712 3.470983 3.480253 3.489524
3.498794 3.508065
      11       12
3.517336 3.526606

$se.fit
         1          2          3          4          5          6          7
8
0.05288433 0.05315377 0.05342362 0.05369388 0.05396453 0.05423559 0.05450702
0.05477885
         9         10         11         12
0.05505104 0.05532361 0.05559655 0.05586984

$df
[1] 286

$residual.scale
[1] 0.4339646
```

```
> #Gas.dmult$seasonal
> season = c(0.9270145, 0.9403801, 0.9840679, 1.0231099, 1.0539975,
1.0587930, 1.0411766, 1.0403735, 1.0394977, 1.0026375, 0.9628965, 0.9260554)
> predvalues = preddata$fit
> spredvalues = predvalues * season
> resid.dmult = test.gas - spredvalues
> resid.dmult
            Jan        Feb        Mar        Apr        May        Jun
Jul
2017 -0.8896810 -1.0021712 -1.1453136 -1.1922268 -1.3456357 -1.4180520 -
1.4125582
            Aug        Sep        Oct        Nov        Dec
2017 -1.3334081 -1.0669888 -1.0873176 -0.9128303 -0.8778329
> rmse.dmult = sqrt(sum(resid.dmult^2)/ length(resid.dmult))
> rmse.dmult
[1] 1.15645
```

After applying the multiplicative decomposition model on the test set, we can see that the test RMSE is 1.15645 which is much higher than the additive model or the original data standard deviation so multiplicative decomposition model is not a good fit.

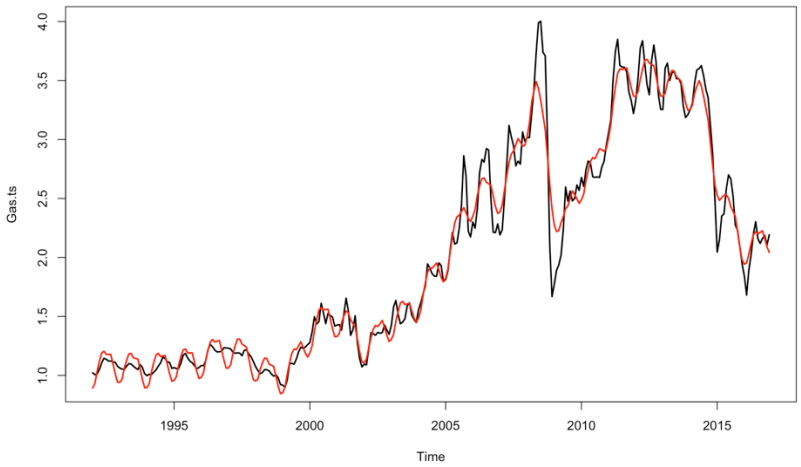
ADDITIVE MODEL USING STL FUNCTION:

```
> #STL Function Additive
> stl.add <- stl(Gas.ts, s.window = 'periodic')
> plot(stl.add)
```



The plot of the STL additive decomposition models shows the 3 different components applied on the training data.

```
> stladd.pred = stl.add$time.series[,'seasonal'] +
stl.add$time.series[,'trend']
> plot(Gas.ts, lwd=2)
> lines(stladd.pred, col='red', lwd=2)
```



The above plot shows the STL additive model seasonal & trend component on top of the time series plot.

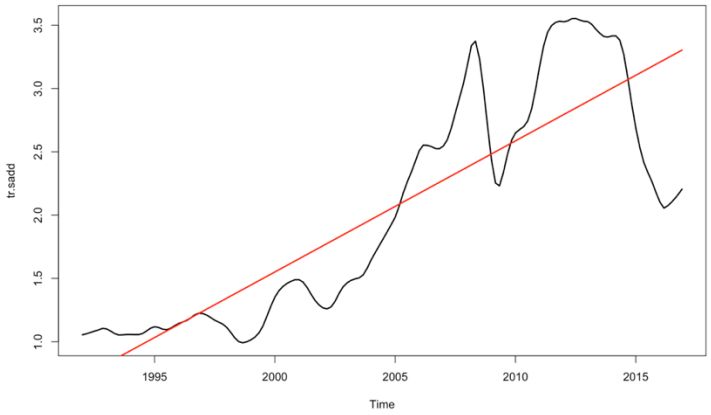
```
> stladd.rmse =
sqrt(sum(stl.add$time.series[,'remainder']^2/length(stl.add$time.series[,'rem
ainder'])))
> stladd.rmse
[1] 0.1527672
```

The STL additive model gives us a training RMSE of 0.1527672 which is good. Now, we apply it on the test data and try to find the test RMSE.

```
> tr.sadd = stl.add$time.series[,'trend']
> #plot(tr.sadd)
> tim.sadd = time(tr.sadd)
> ti.sadd = unclass(tim.sadd)
> trreg3 = lm(tr.sadd ~ ti.sadd)
> summary(trreg3)

Call:
lm(formula = tr.sadd ~ ti.sadd)

Residuals:
     Min       1Q   Median       3Q      Max
-1.17034 -0.29583  0.00706  0.30985  0.95944

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept) -2.056e+02  7.374e+00  -27.88   <2e-16 ***
ti.sadd      1.036e-01  3.679e-03   28.16   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4598 on 298 degrees of freedom
Multiple R-squared:  0.7268,  Adjusted R-squared:  0.7259
F-statistic: 792.8 on 1 and 298 DF,  p-value: < 2.2e-16
```

```
> plot(tr.sadd, lwd = 2, type='l')
> lines(ti.sadd, fitted(trreg3), col='red', lwd=2)
```



The above plot shows the fitted values obtained from the STL additive model on top of the trend component of that model.

```
> predtime = seq(2017, 2017.917, by=1/12)
> predtime
 [1] 2017.000 2017.083 2017.167 2017.250 2017.333 2017.417 2017.500 2017.583
2017.667
[10] 2017.750 2017.833 2017.917
> preddata = predict(trreg3, data.frame(ti.sadd = predtime), se.fit = TRUE)
> preddata
$fit
       1        2        3        4        5        6        7        8
9       10
3.311828 3.320460 3.329091 3.337723 3.346354 3.354985 3.363617 3.372248
3.380880 3.389511
      11       12
3.398143 3.406774

$se.fit
```

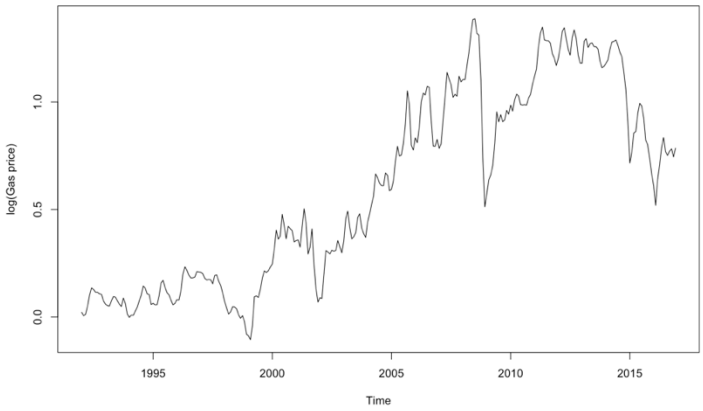
```
         1          2          3          4          5          6          7
8
0.05322974 0.05349566 0.05376202 0.05402880 0.05429601 0.05456362 0.05483165
0.05510007
         9         10         11         12
0.05536890 0.05563811 0.05590771 0.05617768

$df
[1] 298

$residual.scale
[1] 0.4598311

> #stl.add$time.series[,'seasonal']
> season = c(-0.16046903, -0.12521945, -0.02484988,  0.05270375,  0.11733735,
0.12723541,  0.09565351, 0.08998257,  0.08459164, -0.00213744, -0.09318650, -
0.16164193)
> predvalues = preddata$fit
> spredvalues = predvalues + season
> resid.sadd = test.gas - spredvalues
> resid.sadd
            Jan        Feb        Mar        Apr        May        Jun
Jul
2017 -0.8663593 -0.9682403 -1.0612413 -1.0504264 -1.1606914 -1.2252209 -
1.2482704
            Aug        Sep        Oct        Nov        Dec
2017 -1.1652309 -0.8954714 -0.9573738 -0.8309561 -0.8571321
> rmse.sadd = sqrt(sum(resid.sadd^2)/ length(resid.sadd))
> rmse.sadd
[1] 1.033828
```

The test RMSE is 1.033828 which is lower than the models using normal decompose() function but its still higher than standard deviation of the original data set so STL additive is not a good fit.

MULTIPLICATIVE MODEL USING STL FUNCTION:

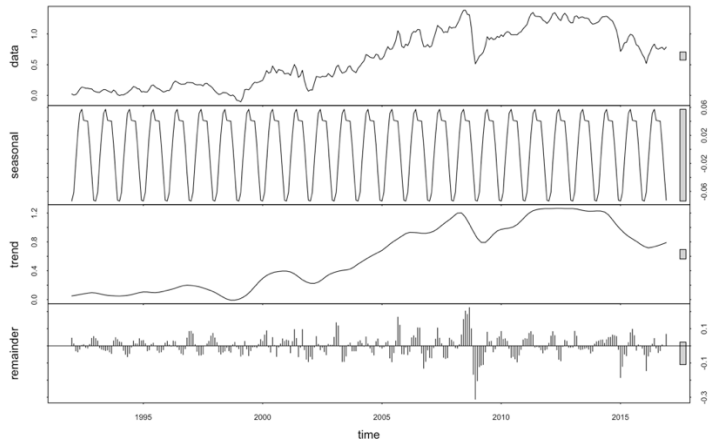
In order to fit multiplicative model using STL function, I have to log transform the model as STL only fits additive model.

```
> loggas = log(Gas.ts)
> plot(loggas, ylab='log(Gas price)')
```



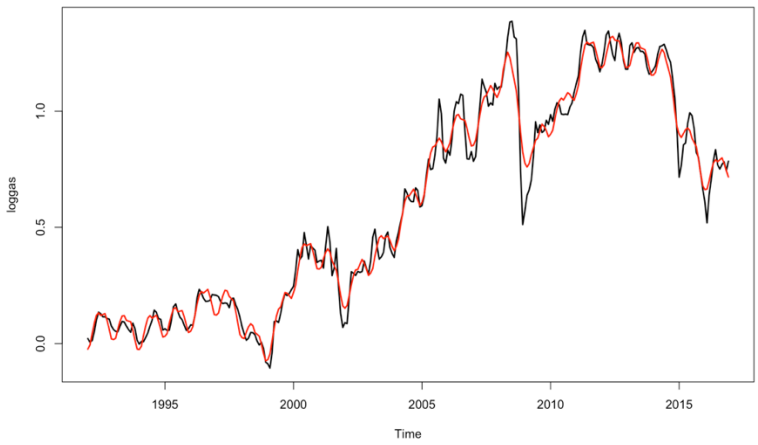
This is the plot of the time series data after log transforming so we can see that the pattern is exactly the same except for the y values.

```
> class(loggas)
[1] "ts"
> stl.mult = stl(loggas, s.window = 'periodic')
> plot(stl.mult)
```



The above plot shows us all the components of the STL multiplicative model over the time series data which is log transformed. We can see that the trend component is smoother than the models we have done in the past.

```
> stlmult.pred = stl.mult$time.series[,'seasonal']+stl.mult$time.series[,
'trend']
> plot(loggas, lwd=2)
> lines(stlmult.pred, col='red', lwd=2)
```



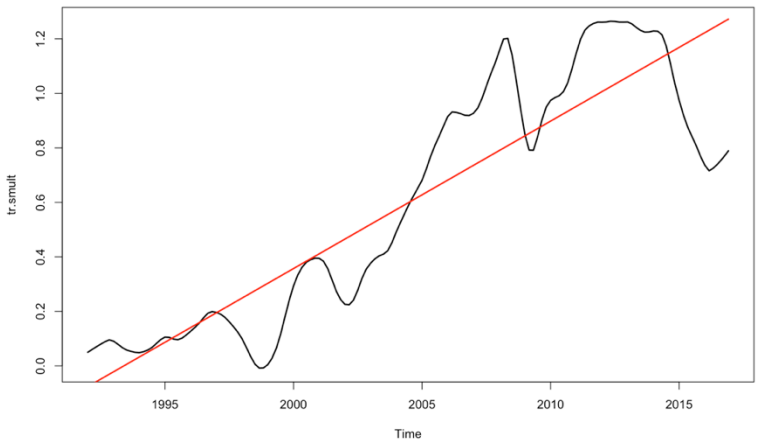
The above plot shows the STL multiplicative model values over the original time series data and it looks like a good fit.

```
> stlmult.resid = Gas.ts - exp(stlmult.pred)
> stlmult.rmse = sqrt(sum(stlmult.resid^2)/length(stlmult.resid))
> stlmult.rmse
[1] 0.1465828
```

The training RMSE of the multiplicative STL model is 0.1465828 which is good but now we will find the RMSE of the model on the test data set.

```
> tr.smult = stl.mult$time.series[,'trend']
> #plot(tr.smult)
> #tr.smult
> tim.smult = time(tr.smult)
> #tim.smult
> ti.smult = unclass(tim.smult)
> #ti.smult
> smultreg = lm(tr.smult ~ ti.smult)
> summary(smultreg)

Call:
lm(formula = tr.smult ~ ti.smult)

Residuals:
     Min       1Q   Median       3Q      Max
-0.51589 -0.12320  0.00894  0.13028  0.40019

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.079e+02  3.211e+00  -33.60   <2e-16 ***
```

```
ti.smult     5.412e-02  1.602e-03   33.78   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2002 on 298 degrees of freedom
Multiple R-squared:  0.793,   Adjusted R-squared:  0.7923
F-statistic:  1141 on 1 and 298 DF,  p-value: < 2.2e-16

> plot(tr.smult, lwd = 2)
> lines(ti.smult, fitted(smultreg), col = 'red', lwd=2)
```



The above plot shows the fitted values from the regression model using the trend component of the STL multiplicative model over the original trend values.

```
> predtime = seq(2017, 2017.917, by=1/12)
> predtime
 [1] 2017.000 2017.083 2017.167 2017.250 2017.333 2017.417 2017.500 2017.583
2017.667
[10] 2017.750 2017.833 2017.917
> pred.smult=predict(smultreg, data.frame(ti.smult=predtime), se.fit = TRUE)
> pred.smult
$fit
       1        2        3        4        5        6        7        8
9       10
1.277037 1.281546 1.286056 1.290566 1.295075 1.299585 1.304095 1.308604
1.313114 1.317623
      11       12
1.322133 1.326643

$se.fit
```

```
         1          2          3          4          5          6          7
8
0.02317857 0.02329436 0.02341035 0.02352652 0.02364287 0.02375940 0.02387611
0.02399299
         9         10         11         12
0.02411005 0.02422728 0.02434467 0.02446223

$df
[1] 298

$residual.scale
[1] 0.2002307

> #stl.mult$time.series[,'seasonal']
> season.smult = c(-0.073715095, -0.061369731,-0.017386538 ,0.021508037 ,
0.051374291 , 0.057317330, 0.041344497 , 0.040767272 , 0.040207932 ,
0.006917381 ,-0.034413096, -0.072552280)
> #season.smult
> predvalues = pred.smult$fit
> #predvalues
> spredvalues = predvalues + season.smult
> epredvalues=exp(spredvalues)
> epredvalues
       1        2        3        4        5        6        7        8
9       10
3.331163 3.387786 3.556117 3.713867 3.843754 3.884142 3.839872 3.855002
3.870260 3.760458
      11       12
3.624513 3.504649
> resid.smult = test.gas - epredvalues
> resid.smult
           Jan       Feb       Mar       Apr       May       Jun       Jul
Aug
2017 -1.046163 -1.160786 -1.313117 -1.373867 -1.540754 -1.627142 -1.628872 -
1.558002
           Sep       Oct       Nov       Dec
2017 -1.300260 -1.330458 -1.150513 -1.116649
> rmse.stl = sqrt(sum(resid.smult^2)/ length(resid.smult))
> rmse.stl
[1] 1.359754
```

The test RMSE of the model using STL multiplicative model is 1.359754 which is quite higher compared to other models or even the standard deviation of the original data so we can conclude that the model is not appropriate for our data set.

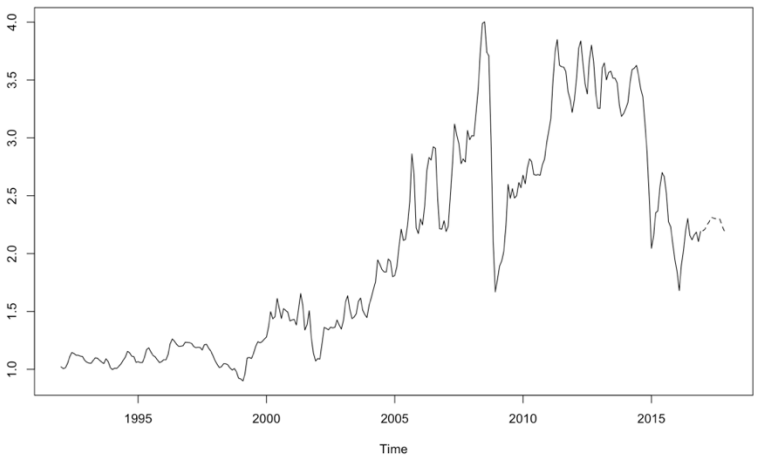
HOLT-WINTERS MODEL (MULTIPLICATIVE): -

```
> Gasw.hw = HoltWinters(Gas.ts, seasonal = "mult")
> Gasw.hw
Holt-Winters exponential smoothing with trend and multiplicative seasonal
component.

Call:
HoltWinters(x = Gas.ts, seasonal = "mult")

Smoothing parameters:
 alpha: 0.9713163
 beta : 0
 gamma: 1

Coefficients:
            [,1]
a    2.265866442
b   -0.002064685
s1   0.970706193
s2   0.976995251
s3   0.994849496
s4   1.007314598
s5   1.026116729
s6   1.023751516
s7   1.022093105
s8   1.025869329
s9   1.021981964
s10  0.998397261
s11  0.981060765
s12  0.967400355
> Gasw.hw$coef
           a            b           s1           s2           s3           s4
 2.265866442 -0.002064685  0.970706193  0.976995251  0.994849496  1.007314598
          s5           s6           s7           s8           s9          s10
 1.026116729  1.023751516  1.022093105  1.025869329  1.021981964  0.998397261
         s11          s12
 0.981060765  0.967400355
> Gasw.hw$SSE
[1] 5.428221
> sqrt(Gasw.hw$SSE/(length(Gas.data)-12))
[1] 0.1345142
> sd(Gas.data)
```

```
[1] 0.8946292
> regastw = Gas.ts - Gasw.hw$fitted[,'xhat']
> rmse = sqrt(sum(regastw^2)/length(regastw))
> rmse
[1] 0.137288
```

This is the training RMSE of the holt-winters multiplicative model which is quite good but now let's try the model on test data and get the testing RMSE.

```
> Gasw.predict = predict(Gasw.hw, n.ahead = 1*12)
> Gasw.predict
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
Sep
2017 2.197486 2.209706 2.248034 2.274121 2.314450 2.307002 2.301154 2.307538
2.296684
          Oct      Nov      Dec
2017 2.241621 2.200671 2.168031
> ts.plot(Gas.ts, Gasw.predict, lty = 1:2)
```



The above plot shows the original time series plot with the predicted values for the test data (2017) using the dotted line at the end.

```
> resid.hw = test.gas - Gasw.predict
> resid.hw
              Jan           Feb           Mar           Apr           May
Jun
2017  0.087513616  0.017293622 -0.005033933  0.065878807 -0.011450421 -
0.050001857
              Jul           Aug           Sep           Oct           Nov
Dec
2017 -0.090154363 -0.010538107  0.273316004  0.188378911  0.273328734
0.219968528
```

```
> rmse.hw = sqrt(sum(resid.hw^2)/ length(resid.hw))
> rmse.hw
[1] 0.1461951
```

The test RMSE of the model is 0.1461951 which is much better than any models we have done so far and also lower than the original standard deviation of 0.9099515 of the original time series so we can say that the holt-winters multiplicative model is a good fit.

REGRESSION ANALYSIS: -

To start with regression analysis, we firstly tried to fit the first order regression as shown in section 1. Next in section 2, assumption plots were examined to see the residual patterns, and diagnosis of the assumptions. According to the pattern of residuals, we tried to fit different models until we could find out the most appropriate one. In Section 3, we plot the residuals of each fitted model to observe the pattern of it. In section 4, we plotted the fitted values against actual data. Finally in section 5, we use test set to calculated the test RMSE for each fitted mode to see the prediction performance.

Section 1: the summary output of five different fitted models:

```
> Gas.time=time(Gasoline.train)
> Gas.seas=cycle(Gasoline.train)
```

First order regression

```
> Gas.reg=lm(Gasoline.train~

  0+Gas.time+factor(Gas.seas))
Call:
lm(formula = Gasoline.train ~ 0 + Gas.time + factor(Gas.seas))

Residuals:
     Min       1Q   Median       3Q      Max
-1.43291 -0.31697 -0.02126  0.32462  1.47767

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
Gas.time            0.10358    0.00414   25.02   <2e-16 ***
factor(Gas.seas)1  -205.74807    8.29789  -24.80   <2e-16 ***
factor(Gas.seas)2  -205.71730    8.29824  -24.79   <2e-16 ***
factor(Gas.seas)3  -205.62141    8.29858  -24.78   <2e-16 ***
factor(Gas.seas)4  -205.54836    8.29893  -24.77   <2e-16 ***
factor(Gas.seas)5  -205.48823    8.29927  -24.76   <2e-16 ***
factor(Gas.seas)6  -205.48407    8.29962  -24.76   <2e-16 ***
factor(Gas.seas)7  -205.52138    8.29996  -24.76   <2e-16 ***
factor(Gas.seas)8  -205.53297    8.30031  -24.76   <2e-16 ***
factor(Gas.seas)9  -205.54428    8.30065  -24.76   <2e-16 ***
factor(Gas.seas)10 -205.63551    8.30100  -24.77   <2e-16 ***
factor(Gas.seas)11 -205.73107    8.30134  -24.78   <2e-16 ***
factor(Gas.seas)12 -205.80330    8.30169  -24.79   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5171 on 287 degrees of freedom
Multiple R-squared:  0.9475,    Adjusted R-squared:  0.9451
F-statistic: 398.6 on 13 and 287 DF,  p-value: < 2.2e-16
```

Quadratic

```
> Gas.reg2=lm(Gasoline.train~0+Gas.time
  +I(Gas.time^2)+factor(Gas.seas))

Call:
lm(formula = Gasoline.train ~ 0 + Gas.time + I(Gas.time^2) +
    factor(Gas.seas))

Residuals:
     Min       1Q   Median       3Q      Max
-1.33946 -0.35538 -0.01303  0.36937  1.43751

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
Gas.time           4.617e+00  2.560e+00   1.804   0.0723 .
I(Gas.time^2)     -1.126e-03  6.386e-04  -1.763   0.0789 .
factor(Gas.seas)1 -4.730e+03  2.566e+03  -1.843   0.0663 .
factor(Gas.seas)2 -4.730e+03  2.566e+03  -1.843   0.0663 .
factor(Gas.seas)3 -4.729e+03  2.566e+03  -1.843   0.0663 .
factor(Gas.seas)4 -4.729e+03  2.566e+03  -1.843   0.0663 .
factor(Gas.seas)5 -4.729e+03  2.566e+03  -1.843   0.0663 .
factor(Gas.seas)6 -4.729e+03  2.566e+03  -1.843   0.0663 .
factor(Gas.seas)7 -4.729e+03  2.566e+03  -1.843   0.0663 .
factor(Gas.seas)8 -4.729e+03  2.566e+03  -1.843   0.0663 .
factor(Gas.seas)9 -4.729e+03  2.566e+03  -1.843   0.0663 .
factor(Gas.seas)10 -4.730e+03  2.566e+03  -1.843   0.0663 .
factor(Gas.seas)11 -4.730e+03  2.566e+03  -1.843   0.0663 .
factor(Gas.seas)12 -4.730e+03  2.566e+03  -1.843   0.0663 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5152 on 286 degrees of freedom
Multiple R-squared:  0.9481,    Adjusted R-squared:  0.9455
F-statistic: 373.1 on 14 and 286 DF,  p-value: < 2.2e-16
```

Cubic

```
Gas.reg3=lm(Gasoline.train~0+Gas.time
+I(Gas.time^2)+I(Gas.time^3)+factor(Gas.seas))
```

```
Call:
lm(formula = Gasoline.train ~ 0 + Gas.time + I(Gas.time^2) +
    I(Gas.time^3) + factor(Gas.seas))

Residuals:
     Min       1Q   Median       3Q      Max
-1.08108 -0.27960  0.00469  0.25518  1.06029

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
Gas.time           -1.443e+04  8.687e+02  -16.61   <2e-16 ***
I(Gas.time^2)       7.198e+00  4.334e-01   16.61   <2e-16 ***
I(Gas.time^3)      -1.197e-03  7.207e-05  -16.61   <2e-16 ***
factor(Gas.seas)1   9.637e+06  5.804e+05   16.60   <2e-16 ***
factor(Gas.seas)2   9.637e+06  5.804e+05   16.60   <2e-16 ***
factor(Gas.seas)3   9.637e+06  5.804e+05   16.60   <2e-16 ***
factor(Gas.seas)4   9.637e+06  5.804e+05   16.60   <2e-16 ***
factor(Gas.seas)5   9.637e+06  5.804e+05   16.60   <2e-16 ***
factor(Gas.seas)6   9.637e+06  5.804e+05   16.60   <2e-16 ***
factor(Gas.seas)7   9.637e+06  5.804e+05   16.60   <2e-16 ***
factor(Gas.seas)8   9.637e+06  5.804e+05   16.60   <2e-16 ***
factor(Gas.seas)9   9.637e+06  5.804e+05   16.60   <2e-16 ***
factor(Gas.seas)10  9.637e+06  5.804e+05   16.60   <2e-16 ***
factor(Gas.seas)11  9.637e+06  5.804e+05   16.60   <2e-16 ***
factor(Gas.seas)12  9.637e+06  5.804e+05   16.60   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3679 on 285 degrees of freedom
Multiple R-squared:  0.9736,    Adjusted R-squared:  0.9722
F-statistic: 701.4 on 15 and 285 DF,  p-value: < 2.2e-16
```

Log y + Cubic

```
> log.gas<-log(Gasoline.train)
> log.gas.reg <-lm(log.gas~0+Gas.time+
  I(Gas.time^2)+I(Gas.time^3)+factor(Gas.seas))
```

```
Call:
lm(formula = log.gas ~ 0 + Gas.time + I(Gas.time^2) + I(Gas.time^3) +
    factor(Gas.seas))

Residuals:
     Min       1Q   Median       3Q      Max
-0.46945 -0.10331  0.01429  0.10905  0.32297

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
Gas.time           -6.522e+03  3.375e+02  -19.32   <2e-16 ***
I(Gas.time^2)       3.255e+00  1.684e-01   19.33   <2e-16 ***
I(Gas.time^3)      -5.415e-04  2.800e-05  -19.34   <2e-16 ***
factor(Gas.seas)1   4.356e+06  2.255e+05   19.32   <2e-16 ***
factor(Gas.seas)2   4.356e+06  2.255e+05   19.32   <2e-16 ***
factor(Gas.seas)3   4.356e+06  2.255e+05   19.32   <2e-16 ***
factor(Gas.seas)4   4.356e+06  2.255e+05   19.32   <2e-16 ***
factor(Gas.seas)5   4.356e+06  2.255e+05   19.32   <2e-16 ***
factor(Gas.seas)6   4.356e+06  2.255e+05   19.32   <2e-16 ***
factor(Gas.seas)7   4.356e+06  2.255e+05   19.32   <2e-16 ***
factor(Gas.seas)8   4.356e+06  2.255e+05   19.32   <2e-16 ***
factor(Gas.seas)9   4.356e+06  2.255e+05   19.32   <2e-16 ***
factor(Gas.seas)10  4.356e+06  2.255e+05   19.32   <2e-16 ***
factor(Gas.seas)11  4.356e+06  2.255e+05   19.32   <2e-16 ***
factor(Gas.seas)12  4.356e+06  2.255e+05   19.32   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1429 on 285 degrees of freedom
Multiple R-squared:  0.9653,    Adjusted R-squared:  0.9635
F-statistic: 528.7 on 15 and 285 DF,  p-value: < 2.2e-16
```

Addd inflation

```
> inf.reg=lm(Gasoline.train~0+Gas.time+factor(Gas.seas)+inf.train)
```

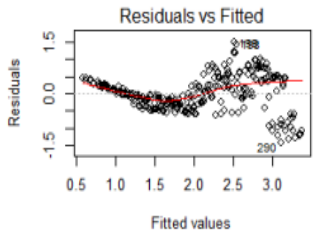
```
Call:
lm(formula = Gasoline.train ~ 0 + Gas.time + factor(Gas.seas) +
    inf.train)

Residuals:
     Min       1Q   Median       3Q      Max
-1.34750 -0.32333 -0.00171  0.32449  1.43894

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
Gas.time           1.053e-01  4.076e-03   25.84  < 2e-16 ***
factor(Gas.seas)1 -2.094e+02  8.173e+00  -25.62  < 2e-16 ***
factor(Gas.seas)2 -2.094e+02  8.174e+00  -25.61  < 2e-16 ***
factor(Gas.seas)3 -2.093e+02  8.176e+00  -25.60  < 2e-16 ***
factor(Gas.seas)4 -2.092e+02  8.174e+00  -25.59  < 2e-16 ***
factor(Gas.seas)5 -2.091e+02  8.173e+00  -25.58  < 2e-16 ***
factor(Gas.seas)6 -2.091e+02  8.173e+00  -25.58  < 2e-16 ***
factor(Gas.seas)7 -2.091e+02  8.172e+00  -25.58  < 2e-16 ***
factor(Gas.seas)8 -2.091e+02  8.173e+00  -25.58  < 2e-16 ***
factor(Gas.seas)9 -2.091e+02  8.174e+00  -25.59  < 2e-16 ***
factor(Gas.seas)10 -2.092e+02  8.172e+00  -25.59  < 2e-16 ***
factor(Gas.seas)11 -2.092e+02  8.170e+00  -25.60  < 2e-16 ***
factor(Gas.seas)12 -2.092e+02  8.170e+00  -25.61  < 2e-16 ***
inf.train          4.073e+01  1.086e+01    3.75 0.000214 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5057 on 286 degrees of freedom
Multiple R-squared:  0.95,    Adjusted R-squared:  0.9475
F-statistic:   388 on 14 and 286 DF,  p-value: < 2.2e-16
```

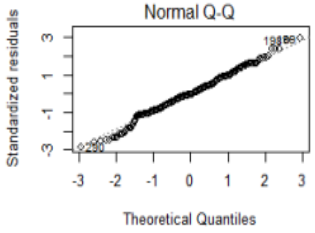
Because in regression model, errors are positively auto-correlated, the residual mean square may seriously underestimate the error variance, as a result the p-value are underestimated, and the t-value and f-value are overestimated. Therefore the summary output is no longer reliable in the regression analysis. However for comparison, we can see the residual standard error dropped significantly from the first order regression to the log y + cubic model, which has the lowest value. Whereas added another predictor-inflation, didn't helped to improve the model but worsen. This proved inflation would not a good predictor for the gas price.

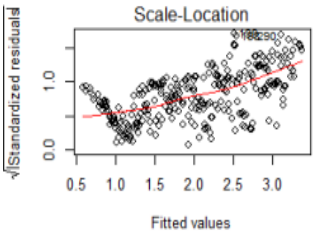
Section 2: examine the assumption plots of the fitted model.

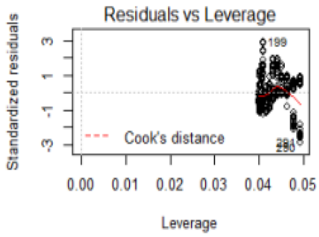
First order regression

```
> plot(Gas.reg)
```



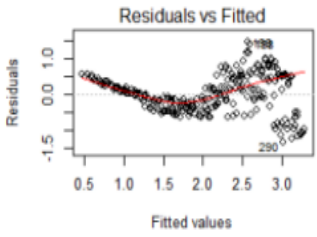


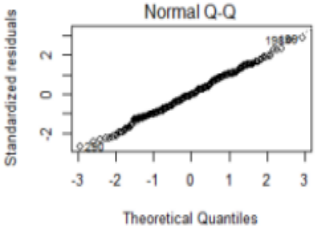


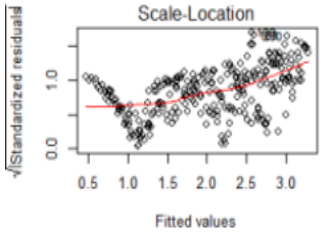


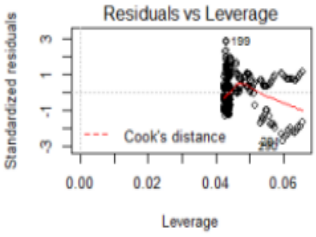
Quadratic

```
> plot(Gas.reg2)
```



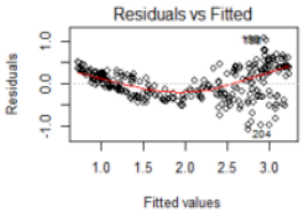


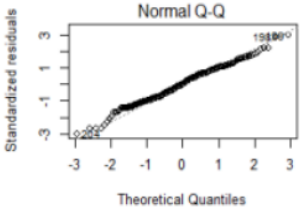


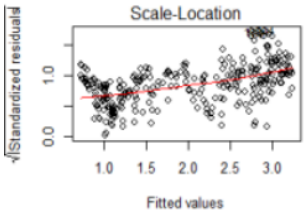


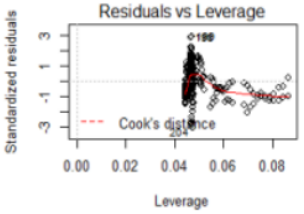
Cubic

```
> plot(Gas.reg3)
```



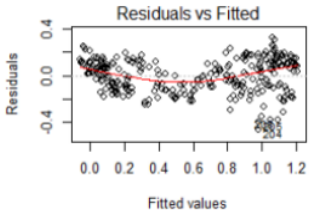


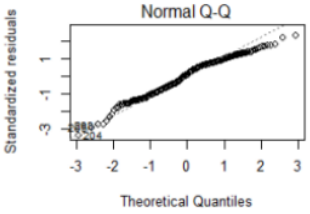


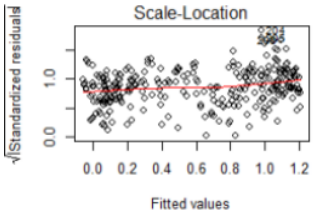


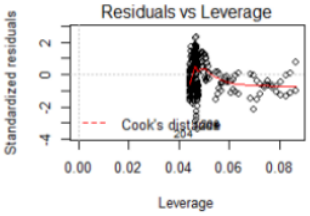
Log y+Cubic

```
> plot(log.gas.reg)
```

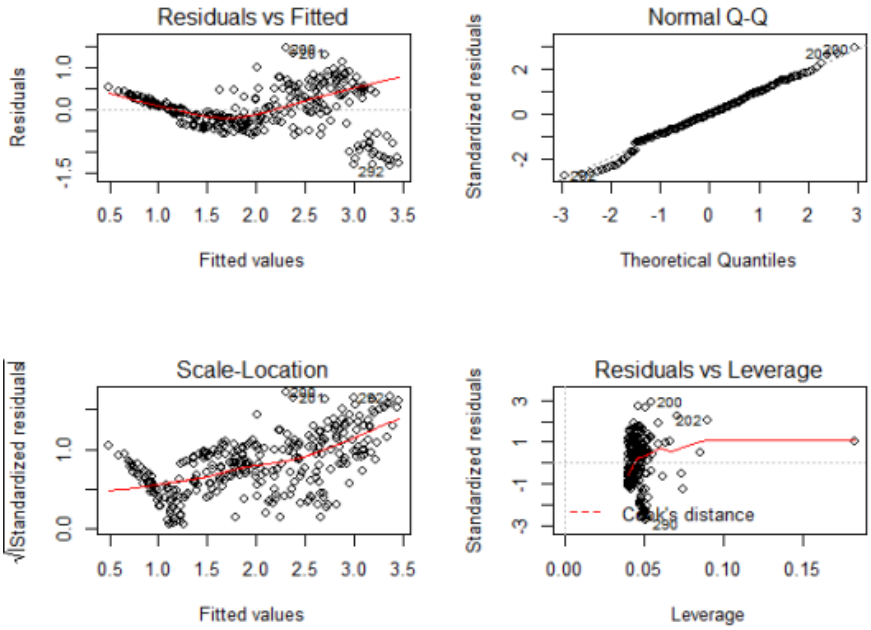








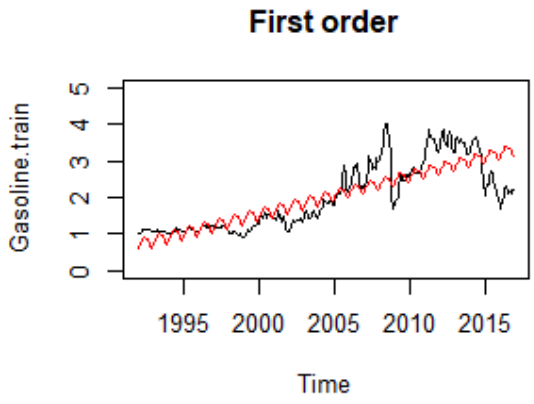
Add inflation

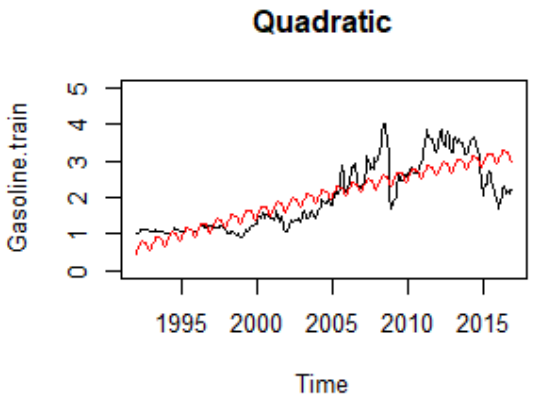


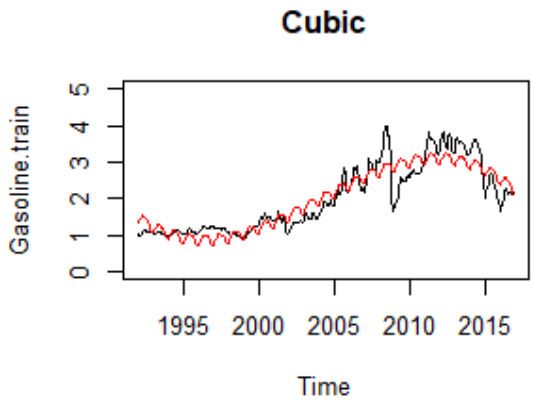
From section 2, we can compare the performance of the assumption plots among 5 models. The Residuals vs. Fitted of the first order regression shows there was a curvilinear, so a quadratic or a cubic term were considered to add into the model to see if they would improve the model. We saw the model indeed shows an improvement by adding the cubic term, but not by adding the quadratic term. However we still went on to try log y with cubic term, because our series has non constant variances. The result of log y shows even better than without the log y; the Residuals vs Fitted is less curvilinear and residuals seem scattered around the 0 (constant mean and variance assumptions are valid), the Normal QQ shows close to the line (normality assumption is valid), the Scale-Location almost flat (constant variance assumption is valid), the residual vs leverage doesn't seems very good but is acceptable. Whereas the inflation model shows it is not a good fit.

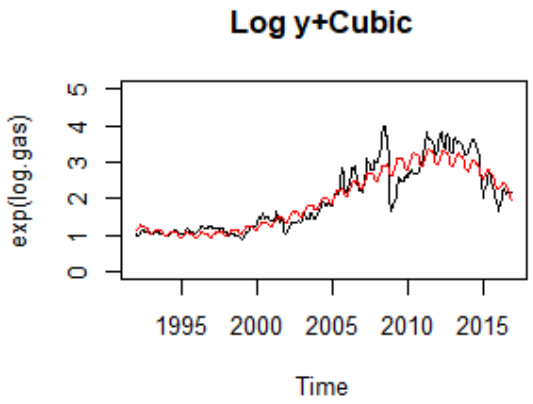
Section 3: plot the fitted values against the actual values

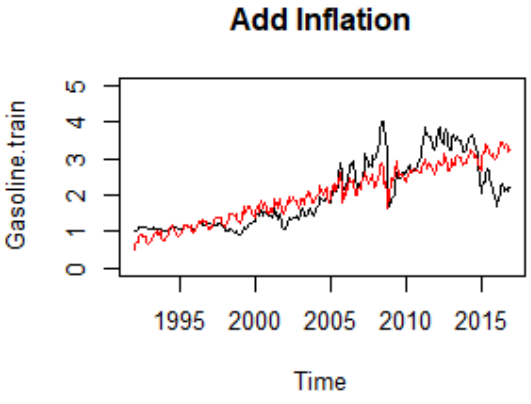
```
> yhat=fitted(Gas.reg)
> plot(Gasoline.train,ylim=c(0,5),main='First order')
> lines.default(Gas.time,yhat,col='red')
```







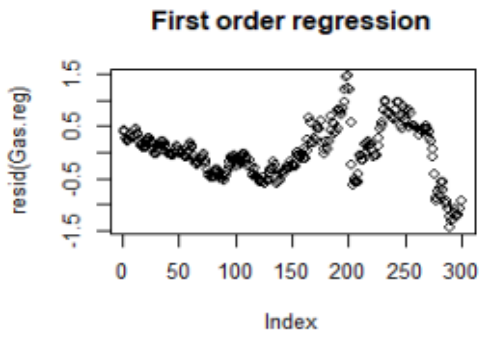


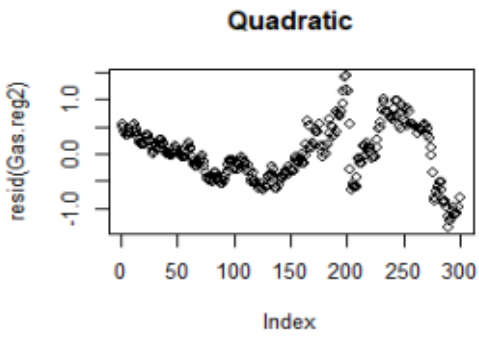


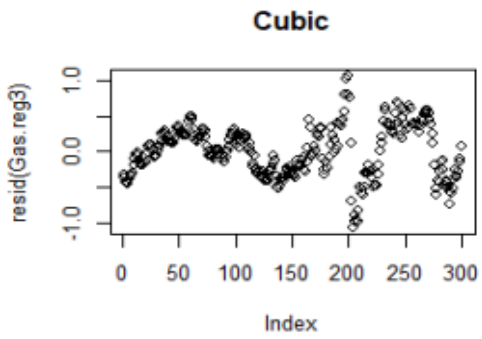
From the plots we can clearly see the Log Y+ cubic model fits best, as the fitted values red line is the closest to the actual data. However, we don't think the model fits our data good enough.

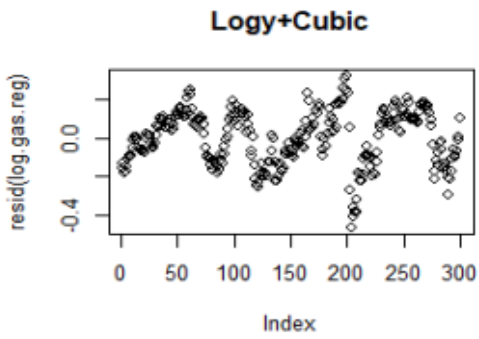
Section 4: residual plots of each fitted model

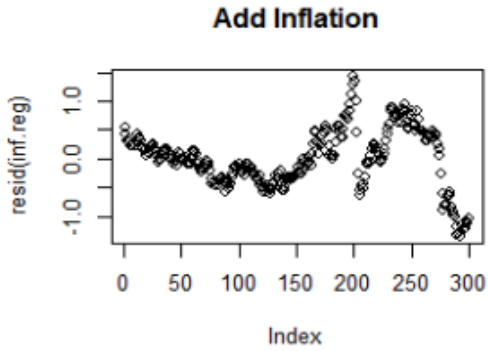
```
> resid.Gas.reg=resid(Gas.reg)
> plot(resid.Gas.reg)
```











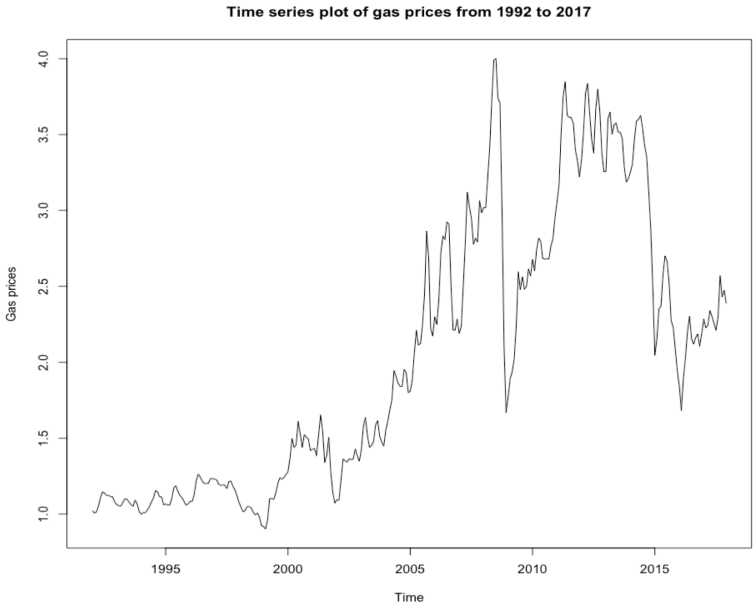
From the residual plots we can see there was gradual improvement from the first order regression to the Log y + Cubic model. The Log y + Cubic model displays residuals scattered around 0, and fluctuated within a constant band. In addition, added the inflation predictor did not show the residuals scattered around 0.

Section 5: Test RMSE of fitted models

```
> Gas.time=time(Gasoline.train)
> Gas.seas=cycle(Gasoline.train)
> pred.t=seq(2017,2017.917,by=1/12)
> pred.sea=Gas.seas[1:12]
> preddata1=predict(Gas.reg, data.frame(Gas.time=pred.t,Gas.seas=pred.sea),s
e.fit = TRUE)
> predvalues1=preddata1$fit
> resid.Gas.reg=Gasoline.test-predvalues1
> rmse.Gas.reg=sqrt(sum(resid.Gas.reg^2)/ length(resid.Gas.reg)
> rmse.Gas.reg
[1] 1.034282

> rmse.quadra
[1] 0.9046279

> rmse.cubic
[1] 0.366155

> rmse.inf
[1] 0.493808

> rmse.log.cubic
[1] 0.3247484
```

We can see the test RMSE has dropped significantly from the first order regression 1.034 to the lowest Log y + cubic model 0.325. This indicates that the Log y + Cubic model is the most appropriate model among all the regression models we have tried. However, regression model is not the best model for our data, as the lowest RMSE is higher than the Holt Winters Model.

ARIMA MODELLING: -



The series of gas prices, as seen from the time series plot is non stationary because it does not have a constant mean, or variance and the covariance(x_t , x_{t+k}) depends on both time and the lag. Hence, it can be classified as a non-stationary model. This can be further confirmed by using the Augmented Dickey Fuller test.

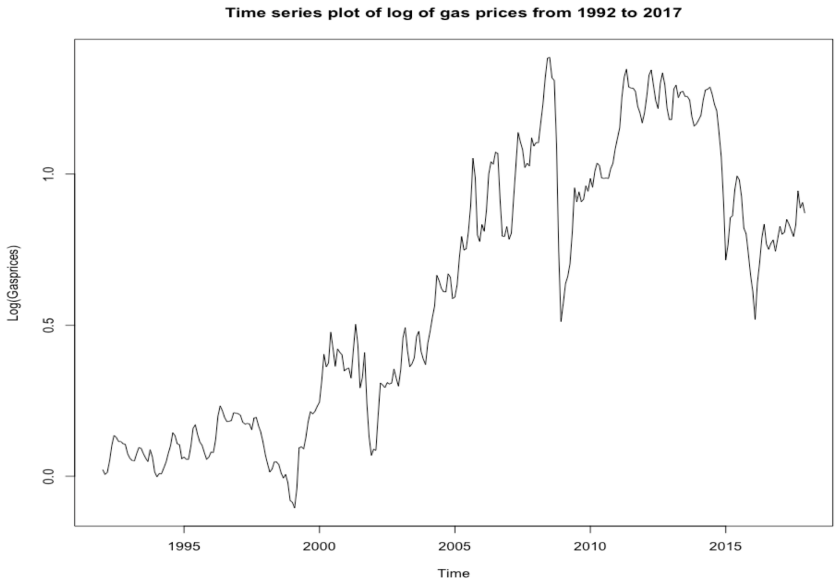
Augmented Dickey Fuller test for stationarity:

Null Hypothesis : The series is not stationary
Alternate Hypothesis: The series is stationary

```
> adfts=adf.test(gasprices.ts)
> adfts

        Augmented Dickey-Fuller Test

data:  gasprices.ts
Dickey-Fuller = -1.9706, Lag order = 6, p-value = 0.5888
alternative hypothesis: stationary
```

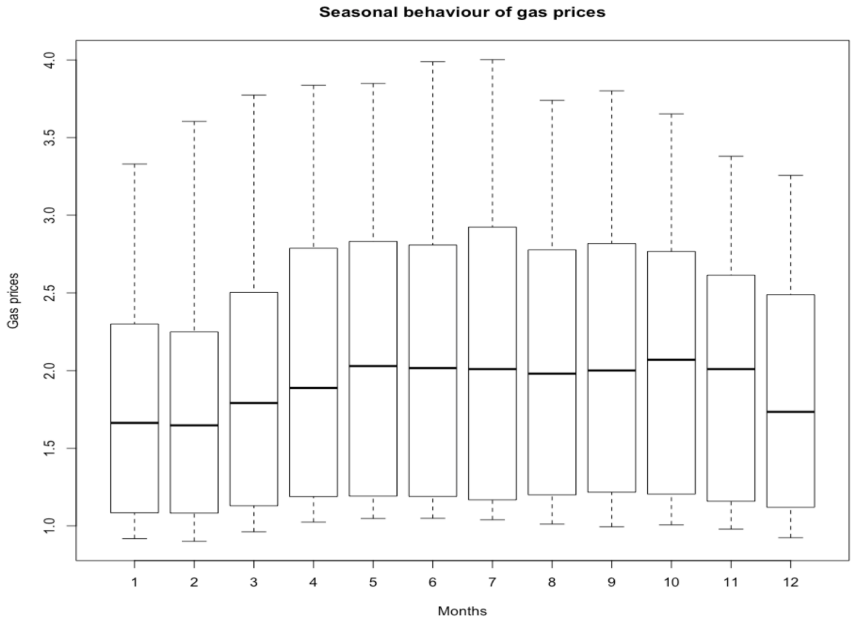
On performing the ADF test on the entire time series of gas prices, it can be seen that the p-value>0.05. Hence , we fail to reject the hypothesis. This strengthens our observation that the series is non stationary.

To conduct ARIMA modelling, it would be necessary to have a stationary model. This can be done by differencing the series. One adjustment to be made before differencing the series is that since increase in variance can be observed in the series, a log transformation becomes necessary.

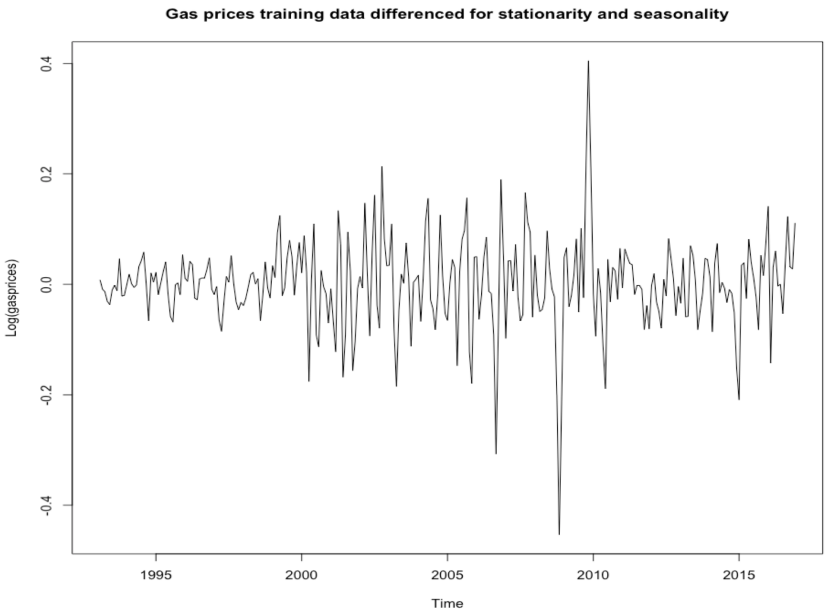


From the log of the series, it can be seen that there is some level of stabilization of the variance in the series.

Another aspect to be kept in mind before ARIMA modelling, is the seasonality of the data. The seasonality can be observed with peaks in the month of June and July , signaling the increase in demand of gas in the summer season.



A seasonal pattern emerges every 12 months , hence the ARIMA modelling will have to take this seasonality into consideration.
With these aspects in mind, the ARIMA modelling can be conducted.

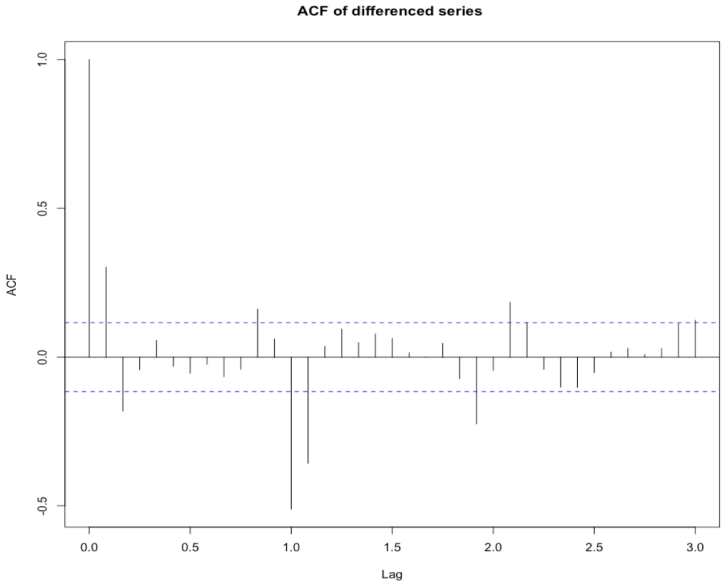


After differencing the log of the series to make it stationary and to account for seasonality, the above plot can be observed , which appears to be stationary.

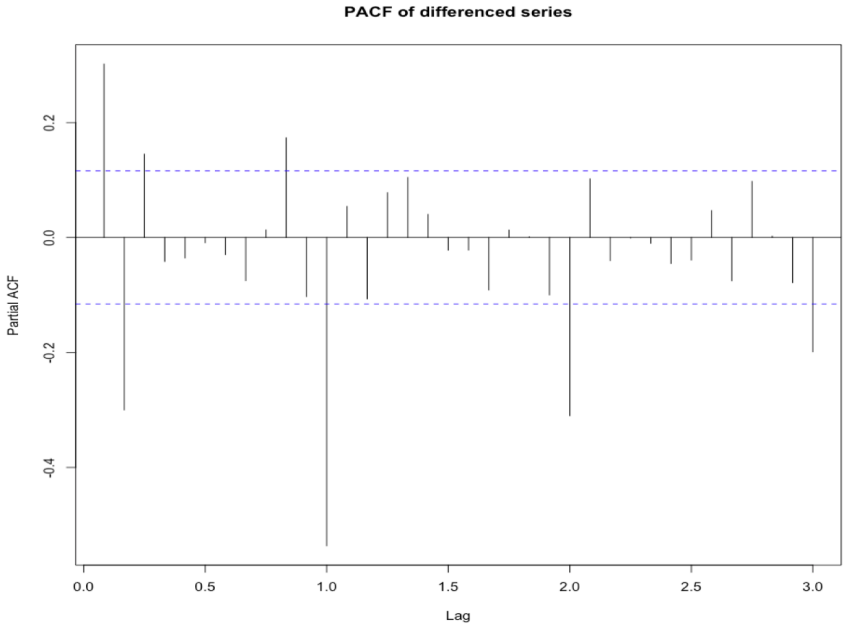
The ADF test can again be used to confirm the stationarity :

```
        Augmented Dickey-Fuller Test

data:  gpdiff
Dickey-Fuller = -6.52, Lag order = 6, p-value = 0.01
alternative hypothesis: stationary
```

Thus the differencing of the series was effective.



From the ACF of the differenced series , the following observations can be made:
Non-seasonal components : Cuts off after lag 2
Seasonal components : Cuts off after lag 12



From the PACF of the differenced series , the following observations can be made:
Non-seasonal components : Mixture of exponential decay and damped sinusoid function
Seasonal components : Lag 12, 24,36 decay exponentially

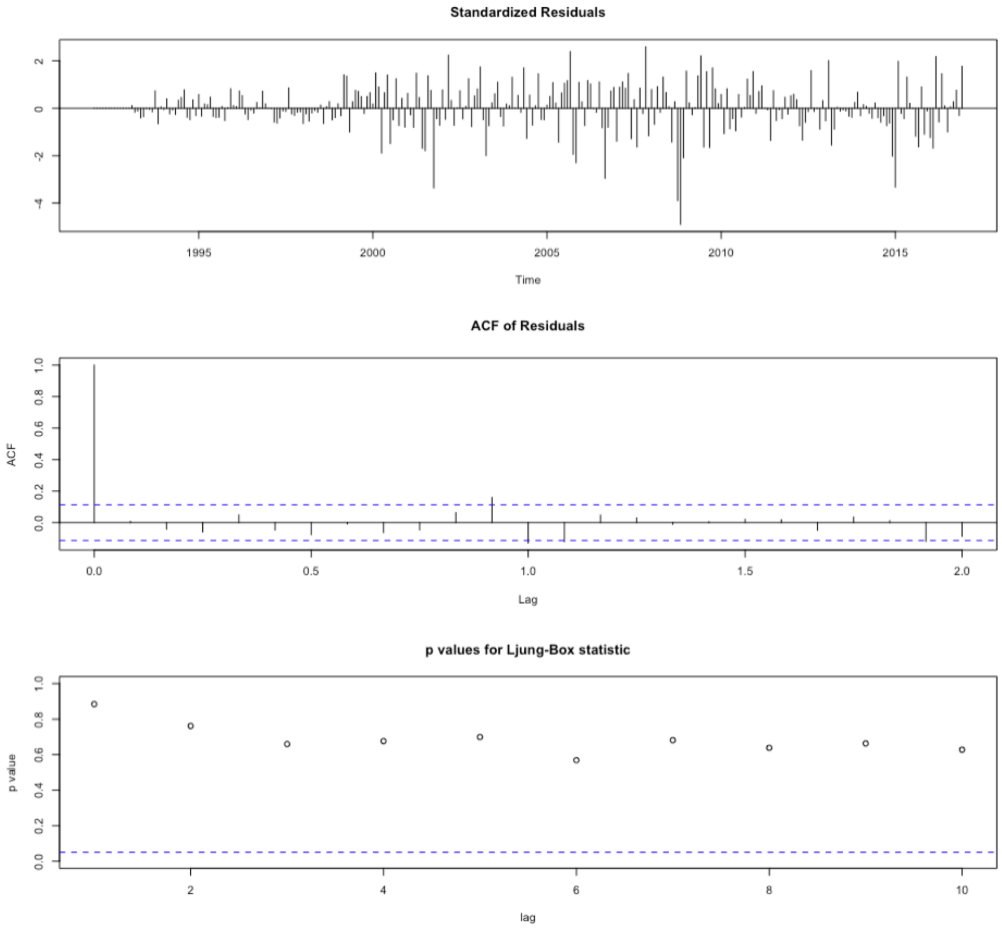
From these observations, it would be ideal to use the ARIMA (0,1,2)x(0,1,1)s=12

```
> fit <- arima(logGPtrain,order = c(0,1,2),
+              seas = list(order=c(0,1,1),
+                          frequency=12))
> fit

Call:
arima(x = logGPtrain, order = c(0, 1, 2), seasonal = list(order = c(0, 1, 1),
    frequency = 12))

Coefficients:
         ma1      ma2     sma1
      0.4502  -0.0794  -1.0000
s.e.  0.0621   0.0631   0.0968

sigma^2 estimated as 0.002422:  log likelihood = 437.67,  aic = -867.33
>
> exp(predict(fit,12)$pred)
         Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct      Nov      Dec
2017 2.290819 2.309222 2.419527 2.521850 2.604876 2.625368 2.588662 2.592112 2.595614 2.517337 2.421870 2.338237
```

```
> sarimapred=exp(predict(fit,12)$pred)
> residsarima=gaspricestest.ts-sarimapred
> rmsesarima=sqrt(sum(residsarima^2)/length(residsarima))
> rmsesarima
[1] 0.2123375
```

For a series with a standard deviation of 0.8946, a RMSE of 0.212 on the test data is a substantial improvement.

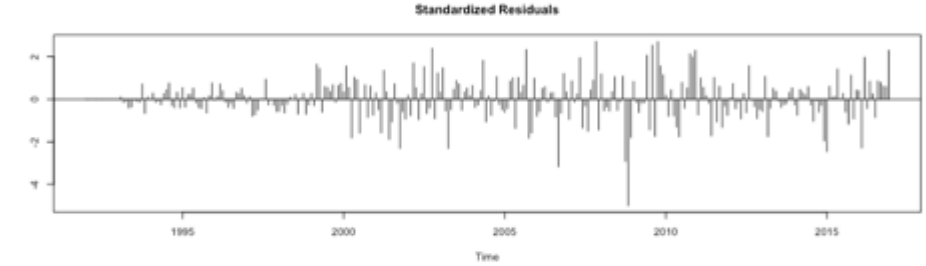
A diagnostic test of the residual would provide further insight into the performance of the model.

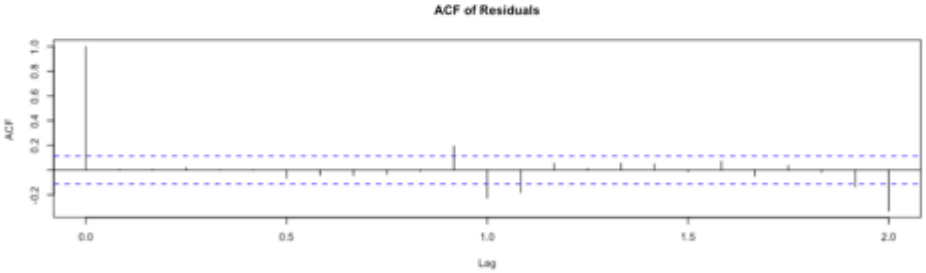


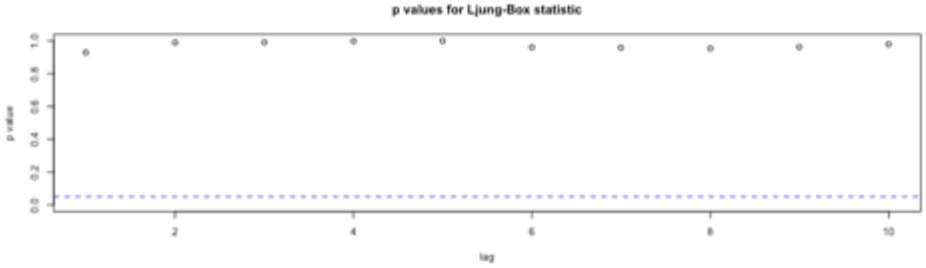
No clear pattern emerges in the plot of the standardized residuals. The ACF of the residuals resembles white noise and the p-values are well above the dotted line, hence the model can be considered to be a good fit for the data.

Further, the auto.arima() function can be used to check if there is scope for further improvement.

```
> autofitD1=auto.arima(logGPtrain,max.p=2, max.q=2, max.P=2, max.Q=2,D=1,d=1)
> autofitD1
Series: logGPtrain
ARIMA(2,1,2)(1,1,0)[12]

Coefficients:
          ar1      ar2     ma1     ma2     sar1
      -0.1795  -0.3095  0.6288  0.2312  -0.5213
s.e.   0.2580   0.1161  0.2614  0.1770   0.0503

sigma^2 estimated as 0.003878:  log likelihood=389.98
AIC=-767.97   AICc=-767.67   BIC=-746.01
```







The diagnostics are similar to that of the model obtained by using the ARIMA function.

```
> sarimaAApred=exp(predict(autofitD1,12)$pred)
> residAAsarima=gaspricestest.ts-sarimaAApred
> rmsesAAarima=sqrt(sum(residAAsarima^2)/length(residAAsarima))
> rmsesAAarima
[1] 0.2292546
```

In terms of the test RMSE, it can be seen that the model suggest by auto.arima() performs comparitively worse than the previous model.

Hence, the model ARIMA (0,1,2)x(0,1,1)s=12 would be the most appropriate model.

CHANGEPOINT ANALYSIS: -

Changepoint analysis for time series is an increasingly important aspect of statistics. Simply put, a changepoint is an instance in time where the statistical properties before and after this time point differ. With potential changes naturally occurring in data and many statistical methods assuming a "no change" setup, changepoint analysis is important in both applied and theoretical statistics.

Change-point analysis is a powerful new tool for determining whether a change has taken place. It is capable of detecting subtle changes missed by control charts. Further, it better characterizes the changes detected by providing confidence levels and confidence intervals. When collecting online data, a change-point analysis is not a replacement for control charting. But, because a change-point analysis can provide further information, the two methods can be used in a complementary fashion. When analyzing historical data, especially when dealing with large data sets, change-point analysis is preferable to control charting. A change-point analysis is more powerful, better characterizes the changes, controls the overall error rate, is robust to outliers, is more flexible and is simpler to use. This article describes how to perform a change-point analysis and demonstrates its capabilities through a number of examples.

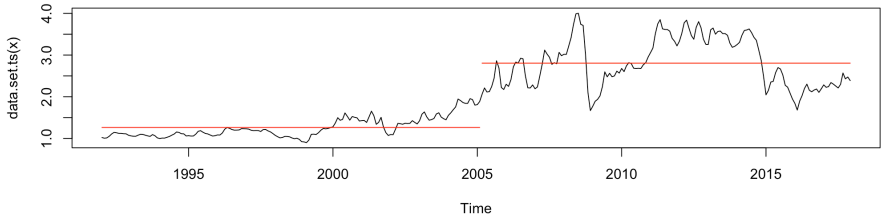
Traditionally, control charts are used to detect changes. The major difference between change-point analysis and control charting is that control charts can be updated following the collection of each data point while a change-point analysis can only be performed once all the data is collected. Control charts are generally better at detecting isolated abnormal points and at detecting a major change quickly while a change-point analysis can detect subtle changes frequently missed by control charts. The two methods can be used in a complementary fashion.

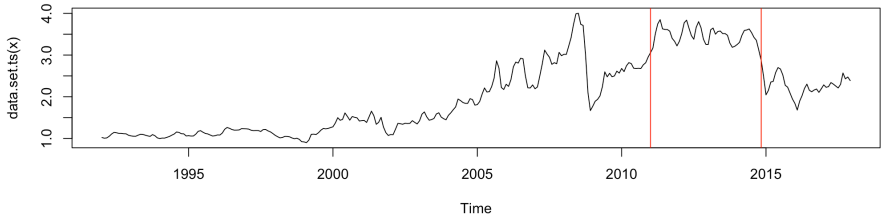
When analyzing historical data, especially when such data sets are large, a change-point analysis is preferable to control charting the data. One benefit of a change-point analysis is that it controls the change-wise error rate. As a result, each change detected is likely to be real. Control charts control the point-wise error rate. When there are thousands of data points, numerous points can exceed the control limits even when no change has occurred. Change-point analysis offers many other benefits as well. This article also compares and contrasts these two methods of detecting change.

The PELT algorithm is exact and under mild conditions has a computational cost that is linear in the number of data points. PELT is more accurate than binary segmentation and faster as than other exact search methods. However, there is scanty literature on the sensitivity/power of PELT algorithm as the changepoints approach the extremes and as the size of change increases.

```
> mvalue = cpt.mean(Gas.tse, method="PELT") #mean changepoints using PELT
> cpts(mvalue)
[1] 158
> tsproject[158,]
                V1    V2
158 2005 February 1.886
> plot(mvalue)
```

```
> vvalue = cpt.var(Gas.tse, method="PELT")
> cpts(vvalue)
[1] 229 275
> tsproject[c(229,275), ]
               V1    V2
229  2011 January 3.058
275 2014 November 2.875
> plot(vvalue)
```

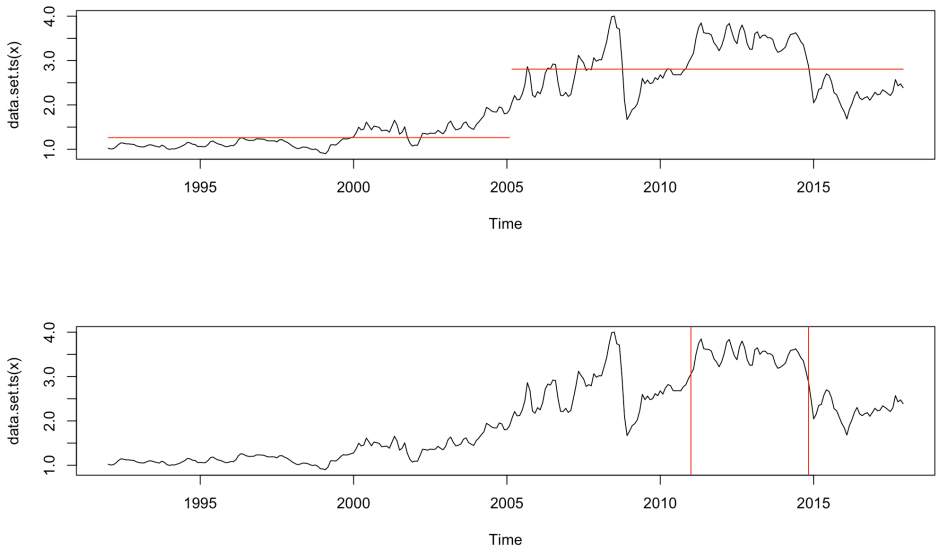




The above plot shows that after February 2005, there is shift in the mean of the plot. Although the biggest drop and change in the plot occurs around Nov. 2008 during recession, we can see that the overall shift in the mean of the plot started in Feb 2005. There was also the effect of hurricane Karina in late 2005 which might have contributed to the shift in the mean.

Binary Segmentation is another method with which we can use Changepoint analysis. It is not an exact procedure like PELT is but is approximate and is fast computationally.

```
> mvalue = cpt.mean(Gas.tse, method="BinSeg")
> cpts(mvalue)
[1] 158
> plot(mvalue)
> vvalue = cpt.var(Gas.tse, method="BinSeg")
> cpts(vvalue)
[1] 229 275
> plot(vvalue)
```



Binary segmentation procedure gives us the exact same results as PELT in the changes in mean & variance. So, we can say that the changes are definitely happening at those points as we can see that both the procedures give us the same results.

There are 2 changes in variance around January 2011 and November 2014, we can attribute the first change to the after effects of recession in the oil prices and next change can be said to be the cause of fluctuation in dollar prices in the US.

FINAL MODEL: -

Since Holt-winters gives us the lowest RMSE of all the models applied so far, we applied Holt-winters to the entire data set and predicted the values for 1 year ahead (12 months).

```
> #Entire data set
> Gasw.hw = HoltWinters(Gas.tse, seasonal = "mult")
> Gasw.hw
Holt-Winters exponential smoothing with trend and multiplicative seasonal
component.

Call:
HoltWinters(x = Gas.tse, seasonal = "mult")

Smoothing parameters:
 alpha: 0.9704061
 beta : 0
 gamma: 1

Coefficients:
            [,1]
a    2.471019422
b   -0.002064685
s1   0.971741417
s2   0.976477170
s3   0.995343084
s4   1.008718755
s5   1.025470281
s6   1.023403169
s7   1.021200412
s8   1.026834867
s9   1.025424362
s10  0.997369364
s11  0.981444938
s12  0.966402764
> Gasw.hw$coef
          a           b          s1          s2          s3          s4
 2.471019422 -0.002064685  0.971741417  0.976477170  0.995343084  1.008718755
         s5          s6          s7          s8          s9         s10
 1.025470281  1.023403169  1.021200412  1.026834867  1.025424362  0.997369364
        s11         s12
 0.981444938  0.966402764
> Gasw.hw$SSE
```

```
[1] 5.557898
> sqrt(Gasw.hw$SSE/(length(Gas.data)))
[1] 0.1334683
> sd(Gas.data)
[1] 0.8946292
> regastw = Gas.tse - Gasw.hw$fitted[,'xhat']
> rmse = sqrt(sum(regastw^2)/length(regastw))
> rmse
[1] 0.1361115
> Gasw.predict = predict(Gasw.hw, n.ahead = 1*12)
> Gasw.predict
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
Sep
2018 2.399186 2.408862 2.453347 2.484233 2.523371 2.516171 2.508647 2.520368
2.514789
          Oct      Nov      Dec
2018 2.443927 2.402879 2.364056
```