Time Series Analysis

Assignment 4: Computer Based

Submit all Exercises

Submission date: 10pm 24th November 2019

You will need to load the `TSA` library before these tasks.

Upload a **single file** containing all typed code and your answers to the questions below. It may be Word, PDF, or Rmd in format. Failure to submit a single file with all solutions will result in a penalty of 10/50 marks, but you may re-submit if you need to and in that case **the most recent file only will be graded**. Please **do not email solutions** to the tutor or the lecturer. Issues relating to upload to Brightspace must be identified clearly and well in advance of the deadline.

Late submissions will be penalised by 5/50 up to 1 hour late, 10/50 up to 24 hours, 25/50 up to 48 hours and no grades will be awarded for work that is more than 48 hours late.

Exercise 1. Look at the `prescrip` time series dataset in the `TSA` package by running **data(prescrip)**. This is monthly U.S. average prescription costs for 68 months from August 1986.

(a) Use the Augmented Dickey-Fuller test to test for whether the time series has a unit root and report your conclusion, assuming a significance level of 0.05 and setting the lag order to be $k = 0$. **[5]**

(b) We'd now like to find the Φ_3 test statistic for the test whether $(\alpha, \beta, \phi) = (\alpha, 0, 1)$ in the model given by:

$$X_t = \alpha + \beta t + \phi X_{t-1} + \varepsilon_t.$$

where X_t is the average prescription cost in month t. This model can be re-written as

$$\Delta X_t = \alpha + \beta t + \omega X_{t-1} + \varepsilon_t.$$

(where $\omega = \phi - 1$). To do this you first fit a linear model to the time series to estimate the coefficients:

```
n=length(prescrip)
tt=2:n # convenience vector of time indices
y=diff(prescrip) # first difference of the series
fit=lm(y~tt+prescrip[-n]) # estimate alpha, omega x[t-1], beta
yhat=fitted(fit)
```

Once this is done, Φ_3 (equivalent to an F statistic) will be given by the ratio of explained to unexplained variance. This in turn is equal to the ratio of the sum of squares of the

model divided by the degrees of freedom of the model to the sum of squares of the errors divided by the degrees of freedom of the errors:

$$\Phi_3 = \frac{SSM/dof_m}{SSE/dof_e}$$

. where $SSM = \sum_{t=1}^{n}(\hat{y}_t - \bar{y}_t)^2$ and $SSE = \sum_{t=1}^{n}(y - \hat{y}_t)^2$. **[5]**

(c) Load the `urca` package and verify this test statistic. You can find it by using the `ur.df` function with arguments `type='trend'` and `lags=0` and then calling `summary()` on the result. You are looking for the `F-statistic:` What is the difference between this test and the test of part (a)? **[5]**

Exercise 2. Load the `beersales` dataset in the `TSA` package by running **data(beersales)**. You will model monthly beersales in millions of barrels and perform the following tasks.

(a) Create plots using **tsdisplay()** and comment on them. **[5]**

(b) Estimate a trend by smoothing the series and plot it. What order do you choose for the moving average smoother and why? **[5]**

(c) Hence decompose the time series as an **additive** model with trend, seasonal, and random components. Find the seasonal and random components and plot them against time. Centre the seasonal component on zero. **[5]**

Exercise 3. Load the **google** dataset in the TSA package by running **data(google)**. You should first de-mean the data by running **google=google-mean(google)**.

(a) Create plots of the time series and comment on the features you find. **[5]**

(b) Test the data for conditional heterskedasticity and report the result. **[5]**

(c) Determine the ARMA-GARCH model order for the data and include your reasoning. Note that you will have to fit several potential models to the data and examine statistical significance of estimated parameters. **[5]**

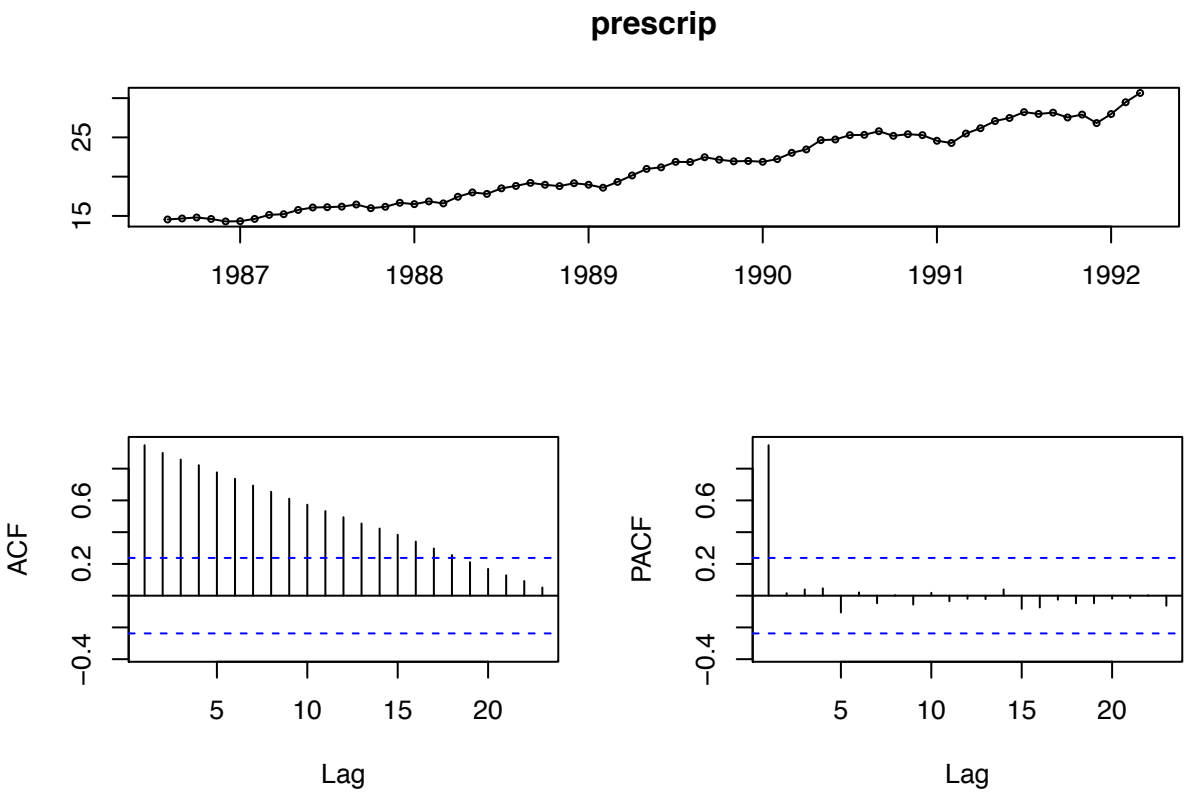
(d) Plot the conditional variances and the standardised residuals. **[5]**

16374403 - Assignment 4

Q1

Firstly, lets load in the data and visualise using the `tsdisplay()` function to get a general idea of what we are looking at.

```
library(TSA)
library(forecast)
data("prescrip")
tsdisplay(prescrip)
```



(a)

The code below will carry out an Augmented Dickey-Fuller (ADF) test with the lag order set to $k = 0$.

```
library(tseries)
adf.test(prescrip, k = 0)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  prescrip
## Dickey-Fuller = -2.7968, Lag order = 0, p-value = 0.2515
## alternative hypothesis: stationary
```

The null hypothesis for the test above is that the time series has a unit root and the alternate hypothesis is that the time series has no unit root.

The test yields a p-value of 0.2515 for $k = 0$. Assuming a significance level of 0.05 we therefore fail to reject the null hypothesis (since 0.2515>0.05). We conclude that there is *not* enough evidence at $\alpha = 0.05$ to conclude that the time series does *not* have a unit root.

This agrees with the general plot of the data which appears to have a non-constant mean over time and also a linear decay in ACF which would indicate a unit root.

(b)

Firstly, fit the linear model as instructed

```
n=length(prescrip)
tt=2:n # convenience vector of time indices
y=diff(prescrip) # first difference of the series
# Relating to the model given
  # tt is t
  #prescrip[-n] is X_{t-1}
fit=lm(y~tt+prescrip[-n]) # estimate alpha, omega x[t-1], beta
yhat=fitted(fit)
summary(fit)
```

```
##
## Call:
## lm(formula = y ~ tt + prescrip[-n])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.44289 -0.33147  0.01302  0.33358  1.04300
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.12039    1.08902   2.865  0.00563 **
## tt            0.05913    0.01987   2.975  0.00413 **
## prescrip[-n] -0.23663    0.08461  -2.797  0.00681 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4816 on 64 degrees of freedom
## Multiple R-squared:  0.1319, Adjusted R-squared:  0.1048
## F-statistic: 4.862 on 2 and 64 DF,  p-value: 0.01082
```

We get $\alpha = 3.12039, \beta = 0.05913$ and $\omega = -0.23662$. All of these estimates have p-values less than 0.05 and so we can conclude that they are significantly differenct from 0 at a 5% level of significance.

We can compute Φ_3 from the formula given in the question. The degrees of freedom (dof_m) of the model is equal to the number of explanatory variables in the model. In our model we have the trend (`tt` above) and X_{t-1} (`prescrip[-n]` in the code). The degrees of freedom of the error $dof_e = n - dof_m - 1 = 67 - 2 - 1 = 64$. Here n is the length of y and it is worth noting that this is not the same as the length of `prescrip` itself.

This is implemented using the following R code.

```
## SSM = SUM(\hat{y_i}-\bar{y})^2
SSM = sum((yhat-mean(y))^2)
# dof_m = 2 since 2 explanatory variables in the model
p=2
DOFm = p

## SSE = SUM(y_i-\hat{y_i})^2
SSE = sum((y - yhat)^2)
## DOFe = n - p - 1
DOFe = length(y) - p - 1

# From formula given
Phi_3 = (SSM/DOFm)/(SSE/DOFe)
Phi_3
```

```
## [1] 4.861987
```

We can check this by looking back at the 'F-statistic' output of `summary(fit)` above, both cases we get $\Phi_3 = 4.862$.

(c)

To verify the result in part (b) we can use the `urca` package and the `ur.df()` function. This is implemented below:

```
library(urca)
summary(ur.df(prescrip, type='trend', lags=0))
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.44289 -0.33147  0.01302  0.33358  1.04300
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.17952    1.10825   2.869  0.00557 **
## z.lag.1     -0.23663    0.08461  -2.797  0.00681 **
## tt           0.05913    0.01987   2.975  0.00413 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4816 on 64 degrees of freedom
## Multiple R-squared:  0.1319, Adjusted R-squared:  0.1048
## F-statistic: 4.862 on 2 and 64 DF,  p-value: 0.01082
##
##
## Value of test-statistic is: -2.7968 8.8117 4.862
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -4.04 -3.45 -3.15
## phi2  6.50  4.88  4.16
## phi3  8.73  6.49  5.47
```

As we can see, we get a F-statistic here of 4.862 which is identical to the value of Φ_3 we calculated in part (b).

Difference between this and (a)

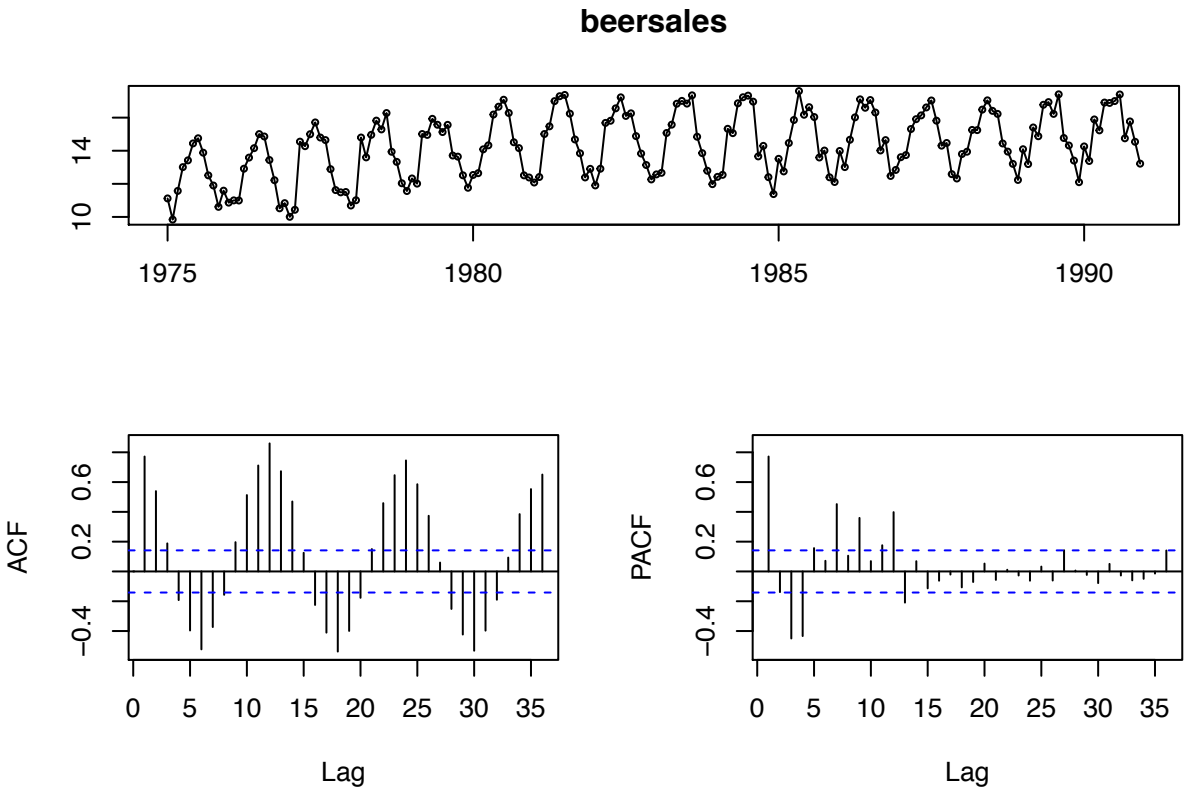
In part (a) we tested to see if the time series `prescrip` had a unit root. However, in part (a) we did not consider that there might be a trend component in the time series. The difference is that in this part of the question we accounted for the fact that the time series has a trend component.

Q2

(a)

Firstly, load the data and apply `tsdisplay()`

```
data('beersales')
tsdisplay(beersales)
```



The `beersales` data gives monthly beer sales in millions of barrels from 01/1975 to 12/1990.

From looking at the first plot above it is clear to see that there is a seasonal component to this time series (there is a repeating pattern of highs and lows at regular intervals). From examination of the ACF plot we can read of that the period is 12 months (the ACF has a clear cycle which repeats every 12 lags). This agrees with the intuition that people may be more likely to purchace beer at certain times of the year.

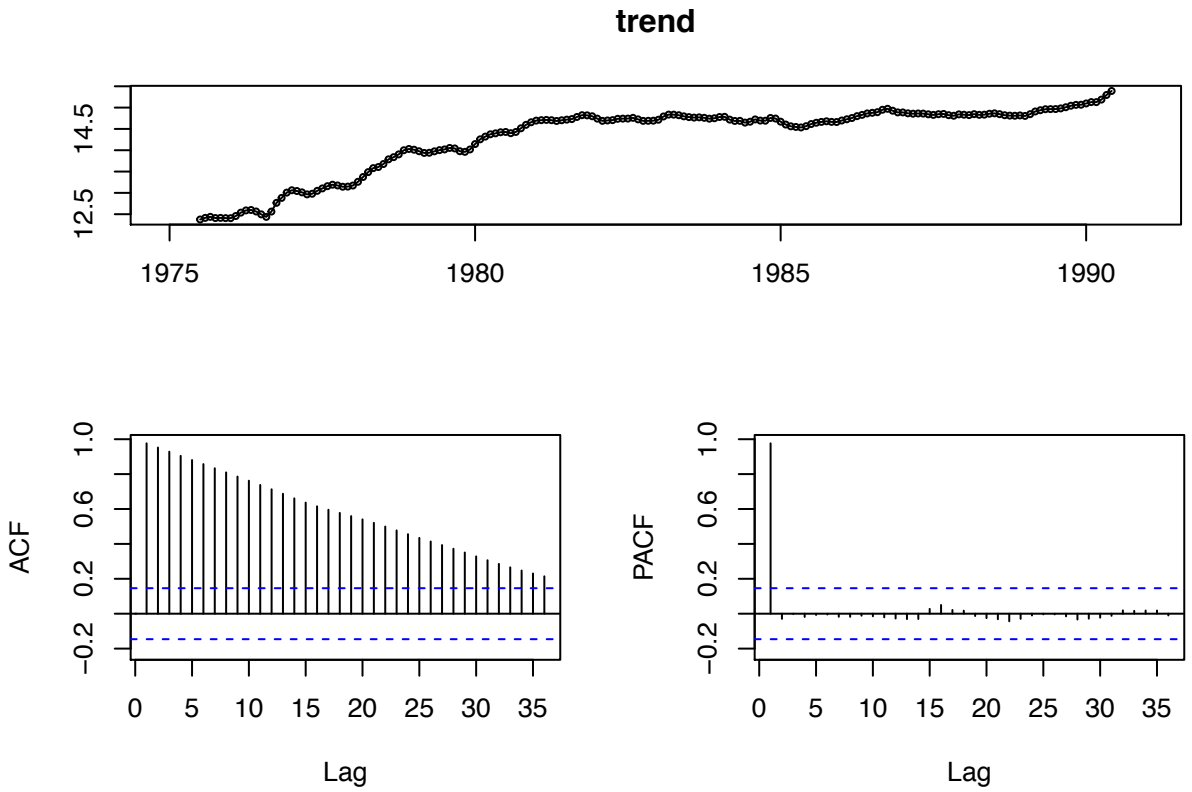
We can also see a possible trend component, particularly in the first 6 years of the data. Both the high points and low points are definitely increasing each year for the first 6 years which indicates a trend component to the time series.

(b)

We can estimate the trend component $\hat{TC}_t$ by smoothing out the seasonal component. We can do this using a moving average smoother where the order is equal to the period. We identified the period in part (a) as being 12 months.

The code below will plot the trend component of the time series.

```
trend = ma(beersales,12)
tsdisplay(trend)
```



We can see from the first plot that there is a clear trend component rising over the time period. This is further highlighted by the ACF plot which indicates that the trend is a non-stationary series itself.

When forecasting it can help to fit a linear model to the trend component. This can be done as follows:

```
lm(ma(beersales,12)~time(beersales))
```

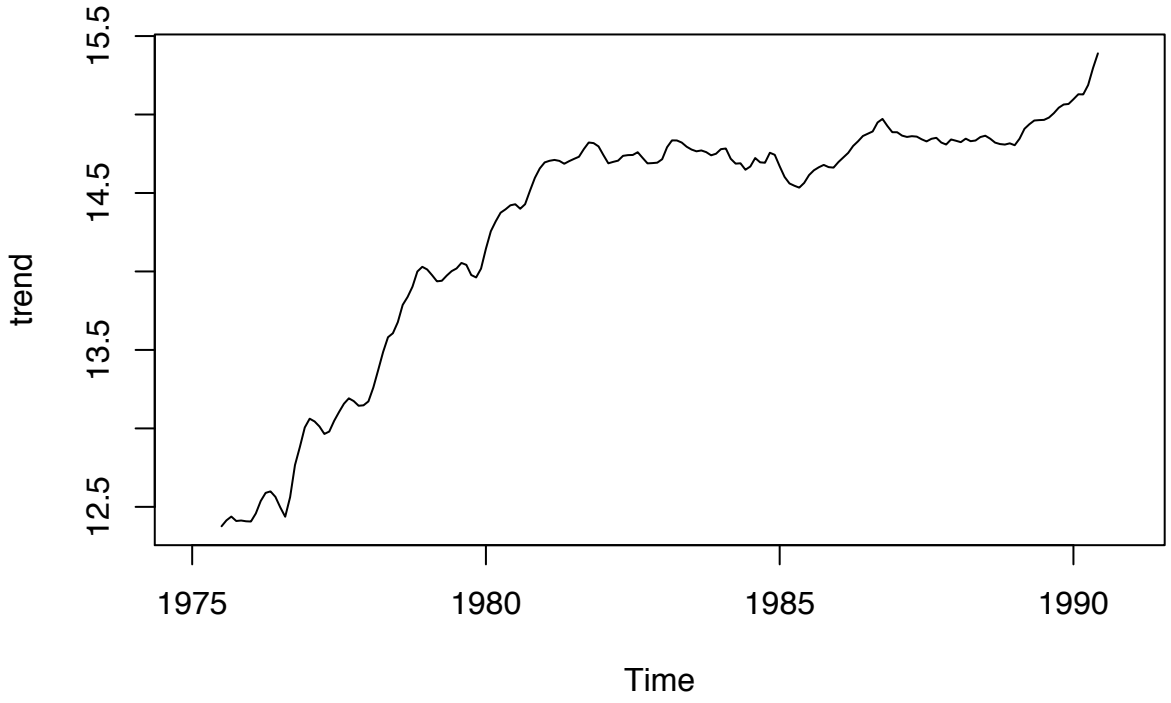
```
##
## Call:
## lm(formula = ma(beersales, 12) ~ time(beersales))
##
## Coefficients:
##     (Intercept)  time(beersales)
##       -297.2908           0.1571
```

(c)

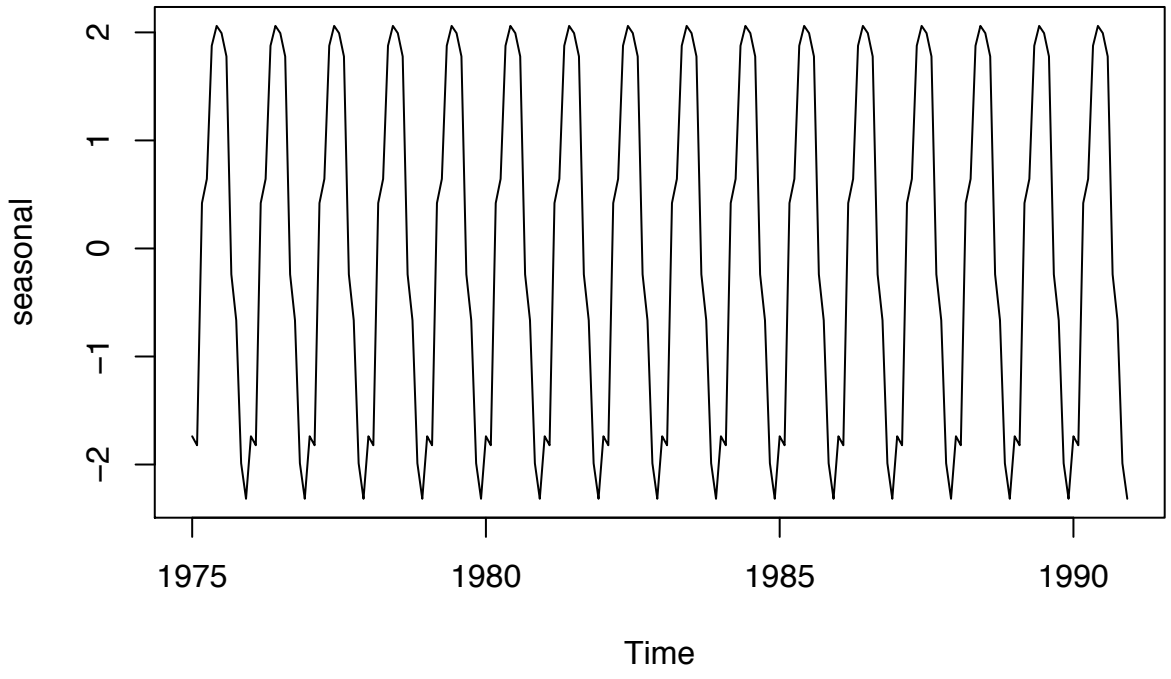
To decompose the time series as an additive model with trend, seasonal and random components we can use the `decompose()` function and specify `type='additive'`.

```
# Two methods to get the trend (will result in exact same)
trend = ma(beersales,12)
trend = decompose(beersales, type="additive")$trend
# Get seasonal component
```

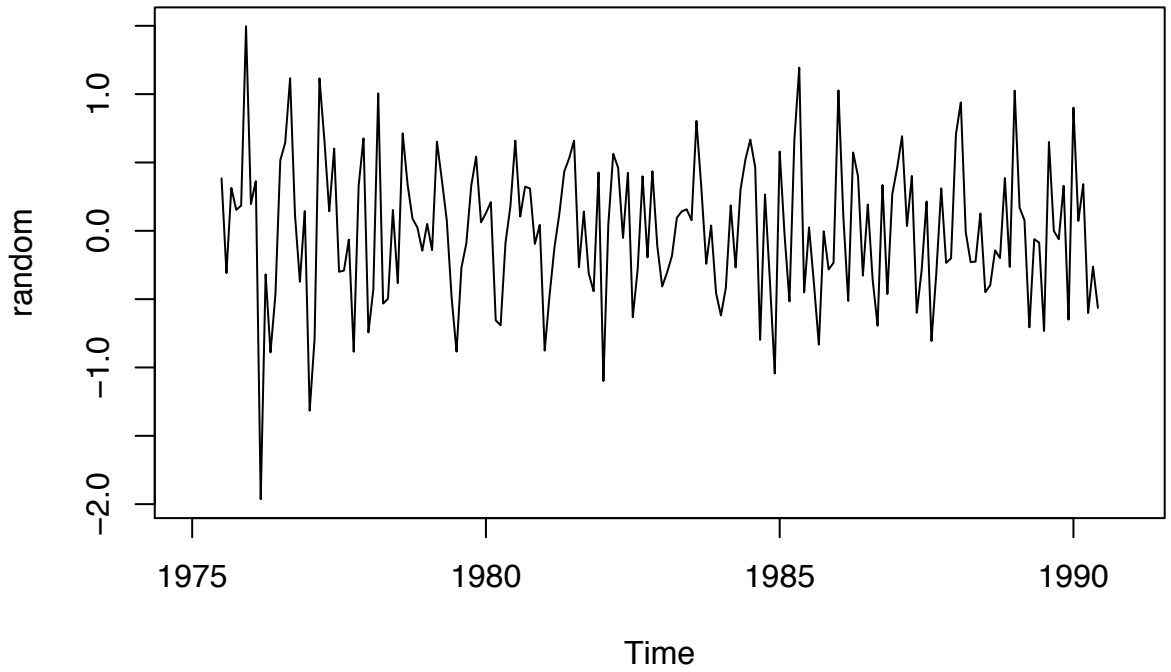
```r
seasonal = decompose(beersales, type="additive")$seasonal
# Get Random Component
random = decompose(beersales, type="additive")$random

# Plot the various components against time
plot(trend)
```




```r
plot(seasonal)
```

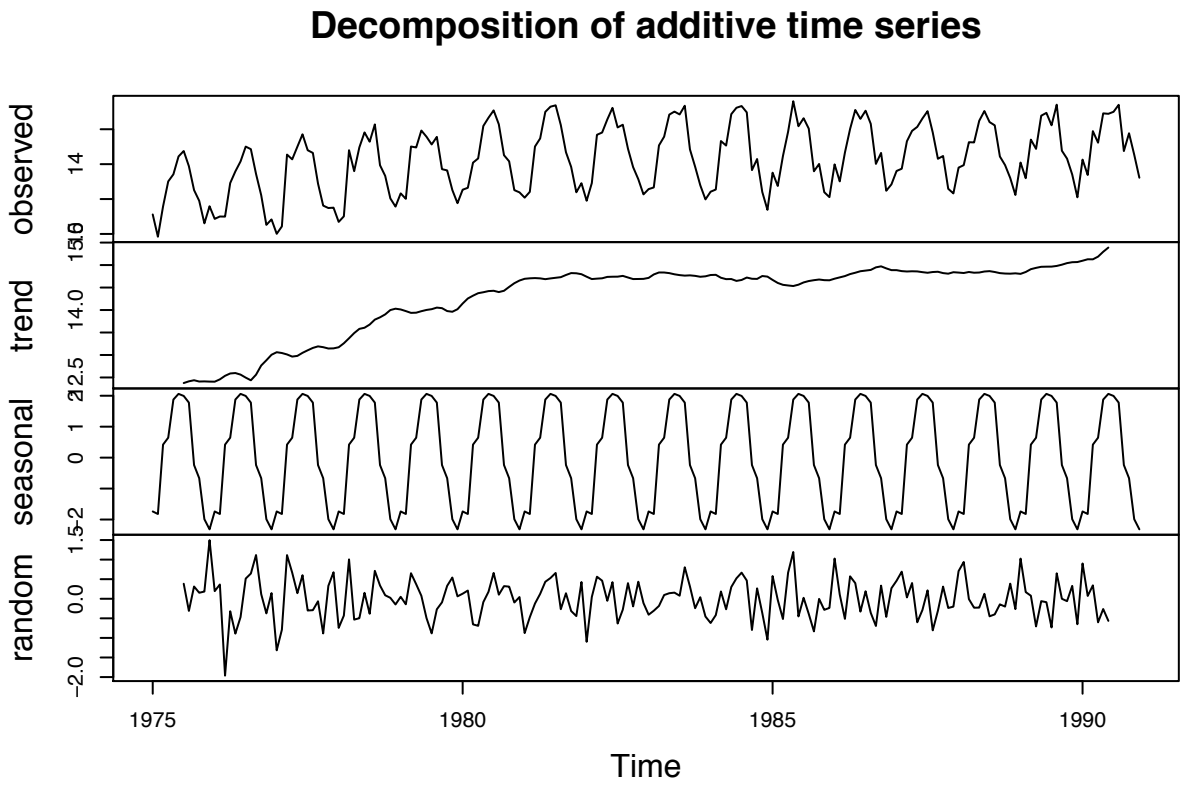


```
plot(random)
```



A more elegant way of obtaining these plots is simply:

```
plot(decompose(beersales, type="additive"))
```

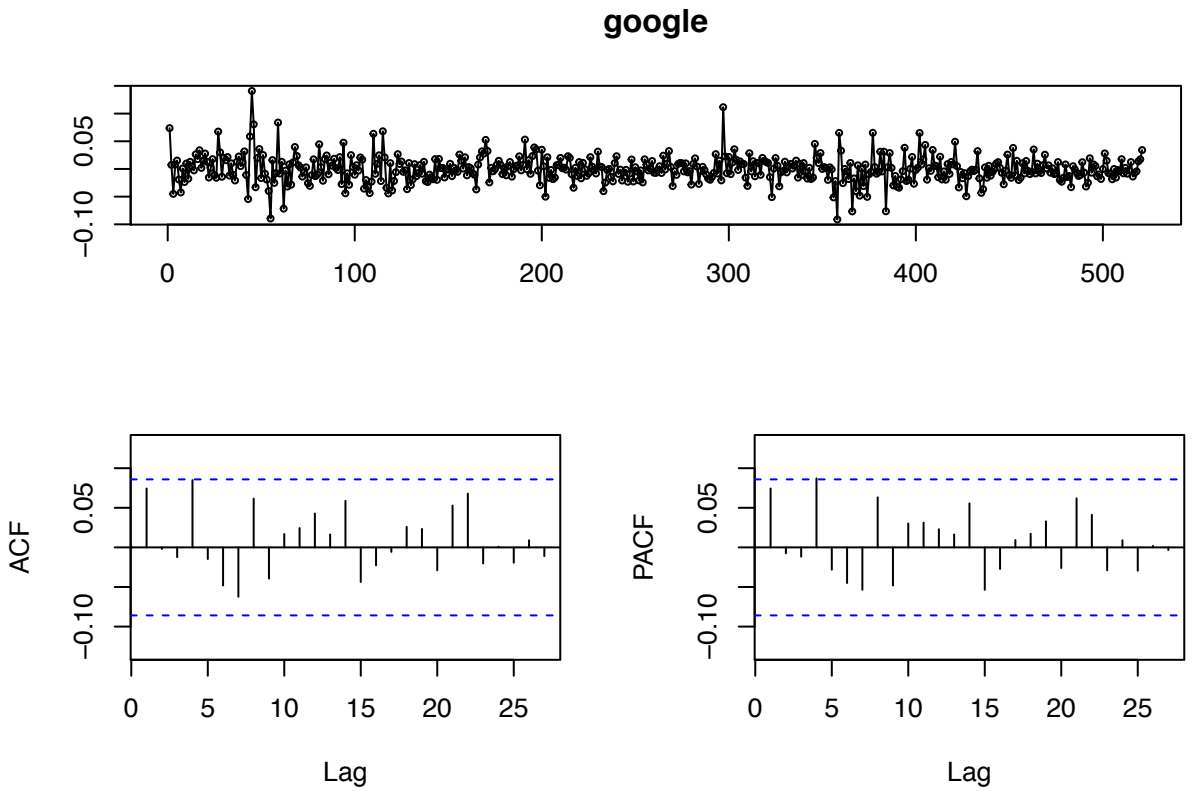


Q3

(a)

Plot the timeseries using `tsdisplay()`

```
data('google')
# De-mean the data
google = google-mean(google)
tsdisplay(google)
```



Comment

We can see from the plots that the ACF and PACF are null for all lags. From the first plot, the mean of the de-meaned series seems to be 0 and does not vary with time. This indicates that no trend component is present.

We can see that there seems to be periods of time with higher/lower variance than other periods. This is known as heteroskedasticity or in other words non-constant variance however we will need to do further tests to confirm this.

(b)

In order to test for conditional heteroscedascity in the data we can use a McLeod-Li test. The McLeod-Li test checks for the presence of conditional heteroscedascity by computing the Ljung-Box test with the squared residuals from an arima model fitted to the data.

What ARIMA model to fit?

We can check if we need to difference the data or not using an Augmented Dickey-Fuller test to test for a unit root.

```
adf.test(google)
```

```
## Warning in adf.test(google): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  google
## Dickey-Fuller = -7.982, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

This gives a small p-value and so reject H_0 and conclude that we don't need to difference the series and the ARIMA will have no integrated component.

Combining this with the ACF and PACF plots discussed in part (a) we fit an ARIMA(0,0,0) model to the `google` data.

Aside

We can do a quick test to make sure that an ARIMA(0,0,0) model is not too simple by fitting an ARIMA(1,0,0) and ARIMA(0,0,1) and testing the statistical significance of the AR and MA coefficients respectively.

```
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
# Test ARIMA(1,0,0)
model=arima(google,order=c(1,0,0))
coeftest(model)
```

```
##
## z test of coefficients:
##
##              Estimate Std. Error z value Pr(>|z|)
## ar1       7.5985e-02 4.4136e-02  1.7216  0.08514 .
## intercept 1.7151e-05 1.1278e-03  0.0152  0.98787
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

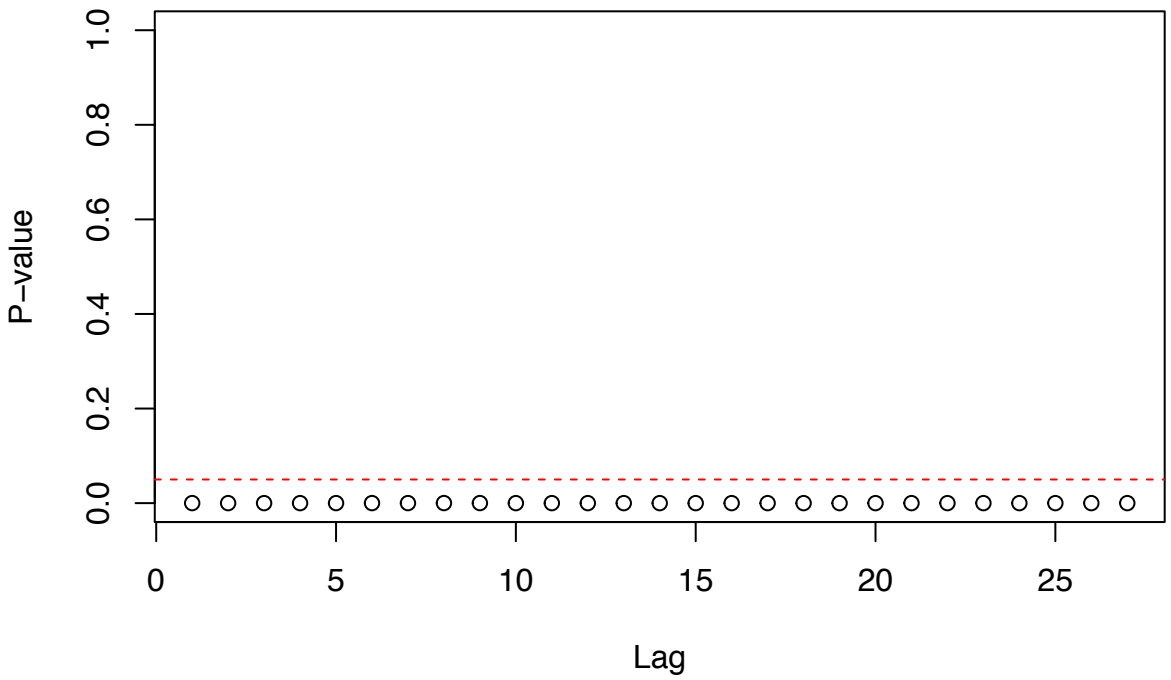
```r
# Test ARIMA(0,0,1)
model=arima(google,order=c(0,0,1))
coeftest(model)
```

```
##
## z test of coefficients:
##
##             Estimate Std. Error z value Pr(>|z|)
## ma1       7.6281e-02 4.3910e-02  1.7372  0.08235 .
## intercept 1.1185e-05 1.1216e-03  0.0100  0.99204
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From this output we can see that the p-value for each of the new terms is greater than 0.05 and so at a 5% level of significance we can fail to reject H_0 and conclude that the coefficients are not statistically different from 0 and therefore should not be included. Condlude ARIMA(0,0,0) is acceptable fit to data.

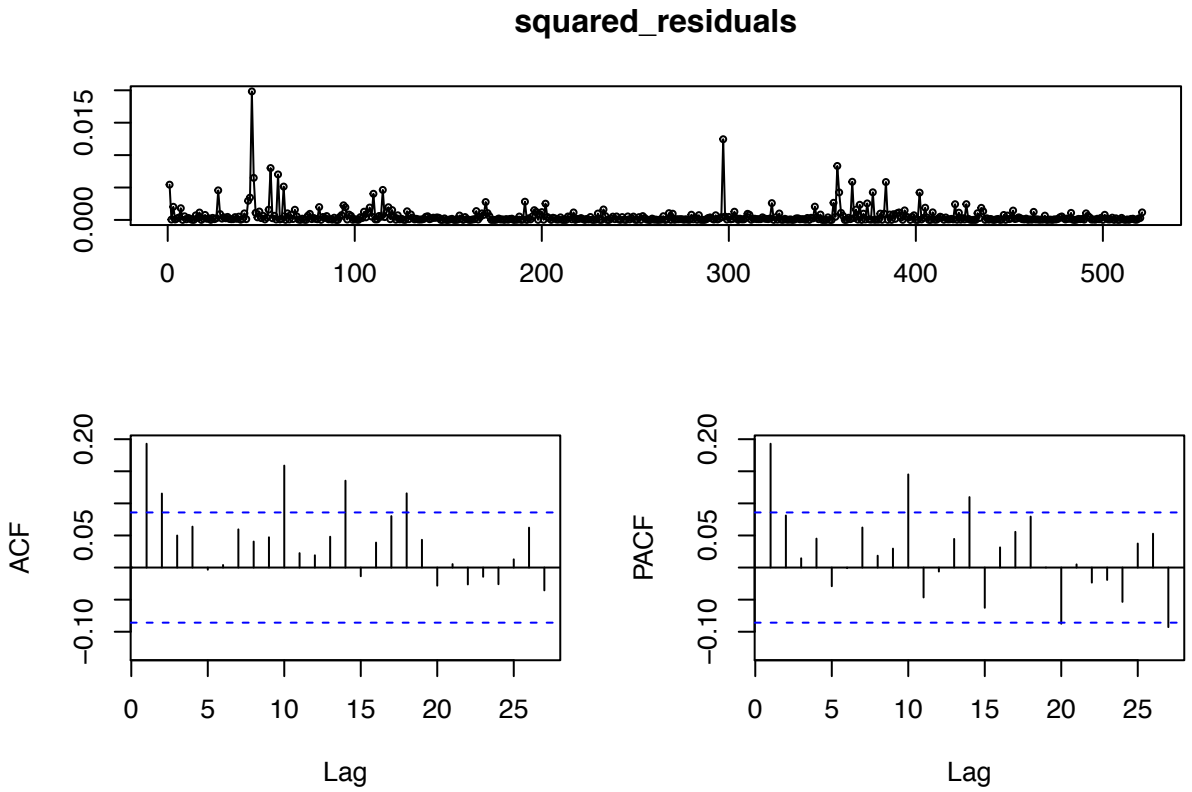
Test for conditional heteroscedascity

```r
# Fit ARIMA(0,0,0)
m1.google=arima(google,order=c(0,0,0))
# Get model residuals
res.google=residuals(m1.google)
McLeod.Li.test(y=res.google)
```



From the above output we can see that the p-value is less than 0.05 for all lags and so the residuals fail the McLeod-Li test. This indicates conditional heteroscedascity in the series. We can also see this by examining the squared residuals as follows:

```
squared_residuals = res.google^2
tsdisplay(squared_residuals)
```



We can see from the above output that significant ACF and PACF terms of the squared residuals are clearly visible and that there seems to be periods of spikes in the first plots. This further confirms the presence of conditional heteroscedascity (non-constant variance) in the data (ie. evidence of ARCH type behaviour in the fitted ARIMA model).

(c)

ARMA Part

From part (b) we saw the below code

```
library(lmtest)
# Test ARIMA(1,0,0)
model=arima(google,order=c(1,0,0))
coeftest(model)
```

```
##
## z test of coefficients:
##
##              Estimate Std. Error z value Pr(>|z|)
## ar1       7.5985e-02 4.4136e-02  1.7216  0.08514 .
## intercept 1.7151e-05 1.1278e-03  0.0152  0.98787
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Test ARIMA(0,0,1)
model=arima(google,order=c(0,0,1))
coeftest(model)
```

```
##
## z test of coefficients:
##
##             Estimate Std. Error z value Pr(>|z|)
## ma1       7.6281e-02 4.3910e-02  1.7372  0.08235 .
## intercept 1.1185e-05 1.1216e-03  0.0100  0.99204
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This told us that the ARIMA(1,0,0) and ARIMA(0,0,1) had statistically insignificant parameters at a 5% level of significance. Thus, we can conclude that the ARIMA(0,0,0) model was the best fit for the ARMA component of the ARMA-GARCH model.

GARCH Part

We recall the fact that if $\eta_t \sim GARCH(p,q)$ then $\eta_t^2 \sim ARMA(max(p,q),p)$. And so we try to fit an ARMA model to the squared residuals. Both the ACF and PACF of the squared residuals displayed in part (b) seem to tail off which suggests fitting an ARIMA$(u > 0, 0, v > 0)$ to the squared residuals.

We can start with the most simple ARIMA(1,0,1) model and work up to find the best model. Doing this tells us that the ARIMA(1,0,1) is the only model with all significant parameters and so we choose it.

```r
# ARIMA(1,0,1) model
sq_resid_model=arima(squared_residuals,order=c(1,0,1))
coeftest(sq_resid_model)
```

```
##
## z test of coefficients:
##
##              Estimate  Std. Error z value Pr(>|z|)
## ar1         5.8020e-01  1.6780e-01  3.4576 0.000545 ***
## ma1        -4.0173e-01  1.8941e-01 -2.1209 0.033930 *
## intercept   5.7207e-04  9.7355e-05  5.8761  4.2e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

And so, we get that $max(p,q) = 1$ and $p = 1$. From this we can tell that either $q = 0$ or $q = 1$.

Using the `garchFit()` function we can fit our proposed models and compare them. Note that the `garchFit()` function specifies GARCH(q,p) rather than GARCH(p,q) and so we will need to reverse the order for the code implementation.

```r
library(fGarch)
```

```
## Loading required package: timeDate
```

```
##
## Attaching package: 'timeDate'
```

```
## The following objects are masked from 'package:TSA':
##
##     kurtosis, skewness

## Loading required package: timeSeries

##
## Attaching package: 'timeSeries'

## The following object is masked from 'package:zoo':
##
##     time<-

## Loading required package: fBasics
```

```
# Fit proposed models
fit1=garchFit(~arma(0,0)+garch(1,0),google,include.mean=F)
```

```
##
## Series Initialization:
##  ARMA Model:                arma
##  Formula Mean:              ~ arma(0, 0)
##  GARCH Model:               garch
##  Formula Variance:          ~ garch(1, 0)
##  ARMA Order:                0 0
##  Max ARMA Order:            0
##  GARCH Order:               1 0
##  Max GARCH Order:           1
##  Maximum Order:             1
##  Conditional Dist:          norm
##  h.start:                   2
##  llh.start:                 1
##  Length of Series:          521
##  Recursion Init:            mci
##  Series Scale:              0.02386202
##
## Parameter Initialization:
##  Initial Parameters:          $params
##  Limits of Transformations:   $U, $V
##  Which Parameters are Fixed?  $includes
##  Parameter Matrix:
##                         U            V params includes
##     mu     -3.033331e-16 3.033331e-16    0.0    FALSE
##     omega   1.000000e-06 1.000000e+02    0.1     TRUE
##     alpha1  1.000000e-08 1.000000e+00    0.1     TRUE
##     gamma1 -1.000000e+00 1.000000e+00    0.1    FALSE
##     delta   0.000000e+00 2.000000e+00    2.0    FALSE
##     skew    1.000000e-01 1.000000e+01    1.0    FALSE
##     shape   1.000000e+00 1.000000e+01    4.0    FALSE
##  Index List of Parameters to be Optimized:
##  omega alpha1
##      2      3
```

```
##  Persistence:                  0.1
##
##
## --- START OF TRACE ---
## Selected Algorithm: nlminb
##
## R coded nlminb Solver:
##
##   0:     1622.4355: 0.100000 0.100000
##   1:     738.98122:  1.06103 0.376444
##   2:     736.11284:  1.09683 0.230916
##   3:     731.71571:  1.04469 0.0904103
##   4:     726.73629: 0.734354 0.197000
##   5:     726.15882: 0.835086 0.214292
##   6:     725.91453: 0.802000 0.210734
##   7:     725.90869: 0.795433 0.210447
##   8:     725.90863: 0.795990 0.210506
##   9:     725.90863: 0.795980 0.210512
##
## Final Estimate of the Negative LLH:
##  LLH:  -1220.27    norm LLH:  -2.342169
##        omega       alpha1
## 0.0004532274 0.2105121051
##
## R-optimhess Difference Approximated Hessian Matrix:
##               omega      alpha1
## omega  -978653944.6 -251590.776
## alpha1    -251590.8    -258.686
## attr(,"time")
## Time difference of 0.007367134 secs
##
## --- END OF TRACE ---
##
##
## Time to Estimate Parameters:
##  Time difference of 0.02115703 secs
```

```
summary(fit1)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~arma(0, 0) + garch(1, 0), data = google,
##     include.mean = F)
##
## Mean and Variance Equation:
##  data ~ arma(0, 0) + garch(1, 0)
## <environment: 0x7f8d151f1600>
##  [data = google]
##
## Conditional Distribution:
##  norm
```

```
##
## Coefficient(s):
##      omega      alpha1
## 0.00045323  0.21051211
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## omega  4.532e-04   3.691e-05   12.279  < 2e-16 ***
## alpha1 2.105e-01   7.179e-02    2.932  0.00337 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1220.27    normalized:  2.342169
##
## Description:
##  Wed Nov 20 14:27:58 2019 by user:
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  171.8793  0
##  Shapiro-Wilk Test  R    W      0.9649166 8.019453e-10
##  Ljung-Box Test     R    Q(10)  15.45629  0.1162905
##  Ljung-Box Test     R    Q(15)  19.95652  0.1736071
##  Ljung-Box Test     R    Q(20)  20.91325  0.4022544
##  Ljung-Box Test     R^2  Q(10)  21.4502   0.01816404
##  Ljung-Box Test     R^2  Q(15)  31.87076  0.006704001
##  Ljung-Box Test     R^2  Q(20)  42.47288  0.002398588
##  LM Arch Test       R    TR^2   18.62256  0.09805348
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -4.676660 -4.660323 -4.676689 -4.670261
```

```
fit2=garchFit(~arma(0,0)+garch(1,1),google,include.mean=F)
```

```
##
## Series Initialization:
##  ARMA Model:                 arma
##  Formula Mean:               ~ arma(0, 0)
##  GARCH Model:                garch
##  Formula Variance:           ~ garch(1, 1)
##  ARMA Order:                 0 0
##  Max ARMA Order:             0
##  GARCH Order:                1 1
##  Max GARCH Order:            1
##  Maximum Order:              1
##  Conditional Dist:           norm
##  h.start:                    2
##  llh.start:                  1
```

```
##  Length of Series:          521
##  Recursion Init:            mci
##  Series Scale:              0.02386202
##
## Parameter Initialization:
##  Initial Parameters:          $params
##  Limits of Transformations:   $U, $V
##  Which Parameters are Fixed?  $includes
##  Parameter Matrix:
##                         U            V params includes
##     mu     -3.033331e-16 3.033331e-16    0.0    FALSE
##     omega   1.000000e-06 1.000000e+02    0.1     TRUE
##     alpha1  1.000000e-08 1.000000e+00    0.1     TRUE
##     gamma1 -1.000000e+00 1.000000e+00    0.1    FALSE
##     beta1   1.000000e-08 1.000000e+00    0.8     TRUE
##     delta   0.000000e+00 2.000000e+00    2.0    FALSE
##     skew    1.000000e-01 1.000000e+01    1.0    FALSE
##     shape   1.000000e+00 1.000000e+01    4.0    FALSE
##  Index List of Parameters to be Optimized:
##  omega alpha1  beta1
##      2      3      5
##  Persistence:                  0.9
##
##
## --- START OF TRACE ---
## Selected Algorithm: nlminb
##
## R coded nlminb Solver:
##
##   0:     711.39951: 0.100000 0.100000 0.800000
##   1:     711.06655: 0.0947083 0.100134 0.796505
##   2:     710.84400: 0.0935107 0.106249 0.797692
##   3:     710.72411: 0.0885976 0.107939 0.794973
##   4:     710.58164: 0.0891935 0.113493 0.796758
##   5:     710.41492: 0.0825071 0.121648 0.791626
##   6:     710.33960: 0.0885426 0.129459 0.785291
##   7:     710.29083: 0.0921507 0.132939 0.774687
##   8:     710.27059: 0.0894848 0.137934 0.775382
##   9:     710.26751: 0.0915809 0.139551 0.770329
##  10:     710.26620: 0.0933173 0.140791 0.769022
##  11:     710.26507: 0.0940046 0.141664 0.766780
##  12:     710.26416: 0.0925616 0.141326 0.768796
##  13:     710.26415: 0.0925620 0.141402 0.768802
##  14:     710.26414: 0.0925577 0.141431 0.768732
##  15:     710.26414: 0.0925630 0.141503 0.768707
##  16:     710.26412: 0.0926984 0.141654 0.768424
##  17:     710.26412: 0.0926815 0.141718 0.768376
##  18:     710.26412: 0.0926973 0.141707 0.768371
##  19:     710.26412: 0.0926965 0.141707 0.768373
##
## Final Estimate of the Negative LLH:
##  LLH:  -1235.914    norm LLH:  -2.372197
##        omega       alpha1        beta1
## 5.278102e-05 1.417067e-01 7.683727e-01
```

```
##
## R-optimhess Difference Approximated Hessian Matrix:
##               omega       alpha1         beta1
## omega  -25787474857 -7496722.651 -11412315.423
## alpha1     -7496723    -3657.488     -4145.602
## beta1     -11412315    -4145.602     -5756.696
## attr(,"time")
## Time difference of 0.005463123 secs
##
## --- END OF TRACE ---
##
##
## Time to Estimate Parameters:
##  Time difference of 0.02724504 secs
```

```
summary(fit2)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~arma(0, 0) + garch(1, 1), data = google,
##     include.mean = F)
##
## Mean and Variance Equation:
##  data ~ arma(0, 0) + garch(1, 1)
## <environment: 0x7f8d1332eb48>
##  [data = google]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##      omega      alpha1       beta1
## 5.2781e-05  1.4171e-01  7.6837e-01
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## omega  5.278e-05   2.046e-05    2.579  0.00990 **
## alpha1 1.417e-01   4.439e-02    3.192  0.00141 **
## beta1  7.684e-01   6.423e-02   11.964  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1235.914    normalized:  2.372197
##
## Description:
##  Wed Nov 20 14:27:58 2019 by user:
##
```

```
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  197.9118  0
##  Shapiro-Wilk Test  R    W      0.9683952 3.688883e-09
##  Ljung-Box Test     R    Q(10)  11.78695  0.2995681
##  Ljung-Box Test     R    Q(15)  16.42109  0.3546311
##  Ljung-Box Test     R    Q(20)  17.22032  0.6386243
##  Ljung-Box Test     R^2  Q(10)  5.091377  0.8849909
##  Ljung-Box Test     R^2  Q(15)  8.717807  0.8918041
##  Ljung-Box Test     R^2  Q(20)  13.24386  0.8666774
##  LM Arch Test       R    TR^2   5.023359  0.9571942
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -4.732877 -4.708372 -4.732943 -4.723278
```

From fitting the proposed models we can see that the GARCH(1,1) model has a lower AIC (-4.732877). Based on this I concluded it was a better fit to the data.

Conclusion

The best ARMA-GARCH model order is ARMA(0,0) and GARCH(1,1).

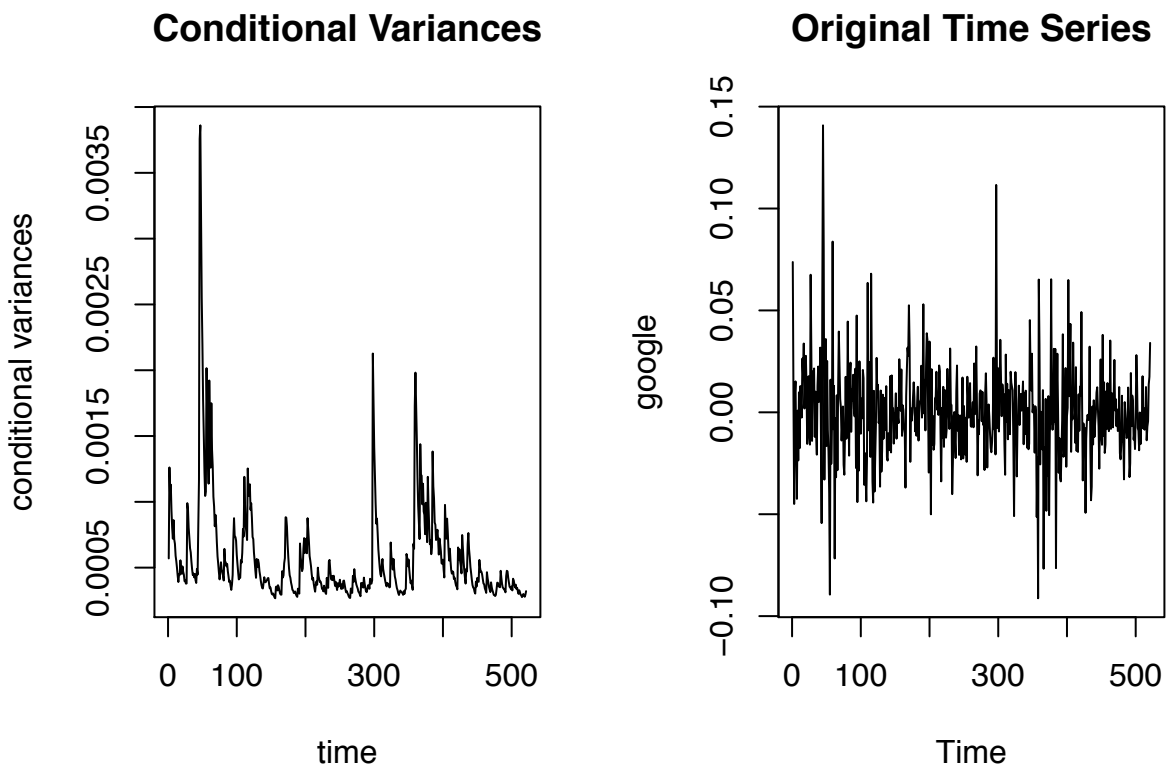
The McLeod-Li p-value of 0.9571942 (from the output of `summary(fit2)`) tells us that we can say with a high level of confidence that we have removed the ARCH behaviour from the time series. In addition to this, all of the co-efficients are statistically significant which is another good indicator of an acceptable model.

(d)

Conditional Variances

We can plot the conditional variances as follows (left plot).

```
par(mfrow=c(1,2))
plot(fit2@sigma.t^2,type= 'l' ,ylab= 'conditional variances', xlab='time',  main='Conditional Variances
plot(google, main='Original Time Series')
```

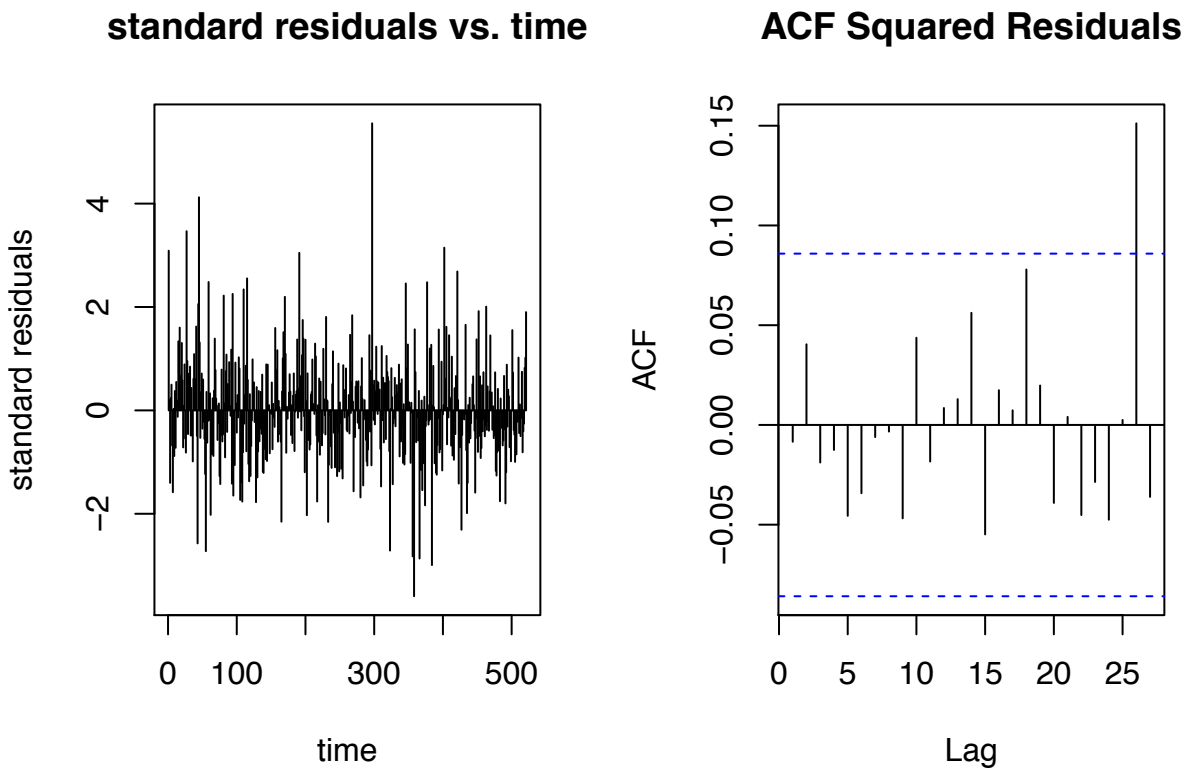


The conditional variances from the ARMA-GARCH model appear larger on the left plot at the same time as the areas of high variance in the original time series as we would hope.

Standardised residuals

Plot standard residuals is seen below.

```
par(mfrow=c(1,2))
plot(fit2@residuals/fit2@sigma.t,type= 'h' ,ylab= 'standard residuals', xlab='time', main= 'standard res
acf((fit2@residuals/fit2@sigma.t)^2, main='ACF Squared Residuals')
```



The plot of standard residuals appears to be a random scatter about zero as we would hope. The ACF of squared residuals plot convey that there is no significant autocorrelation between lags (we can disregard the one spike above the significance line as an error since at 5% level of significance we expect 1 in every 20 tests and we have more than 20 tests displayed here).

Conclusion

In summary, the above plots indicate that the chosen model is an acceptable fit to the data and has sucessfully modelled out the ARCH behaviour in the time series.