STAT30010 - Assignment 4

Jamie Kennedy - 17372983

I have followed the rules of the school plagiarism guideline
Signed: Jamie Kennedy

Exercise 1
(a) The augmented Dickey-Fuller test can be used to test whether the time series has a unit root. The null hypothesis of the test is $\omega = 0$ which means $\phi = 1$ because we define $\omega = \phi - 1$. We can perform the test using the following code

```
# load data
library(TSA)
data("prescrip")

# do augmented D-F test
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
```

Performing this test on our series returns a p value of 0.2515. This is greater than the significance level of 0.05 so we fail to reject the null hypothesis at this significance level. Therefore, we conclude that a unit root may be present.

(b) We can fit a linear model using the code provided in the assignment

```
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)
```

Then we can calculate the test statistic

```
SSM=sum((yhat-mean(yhat))^2)
SSE=sum((y-yhat)^2)
phi3 = (SSM/2)/(SSE/64)
phi3
```

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

(c) To verify the test statistic use the following code

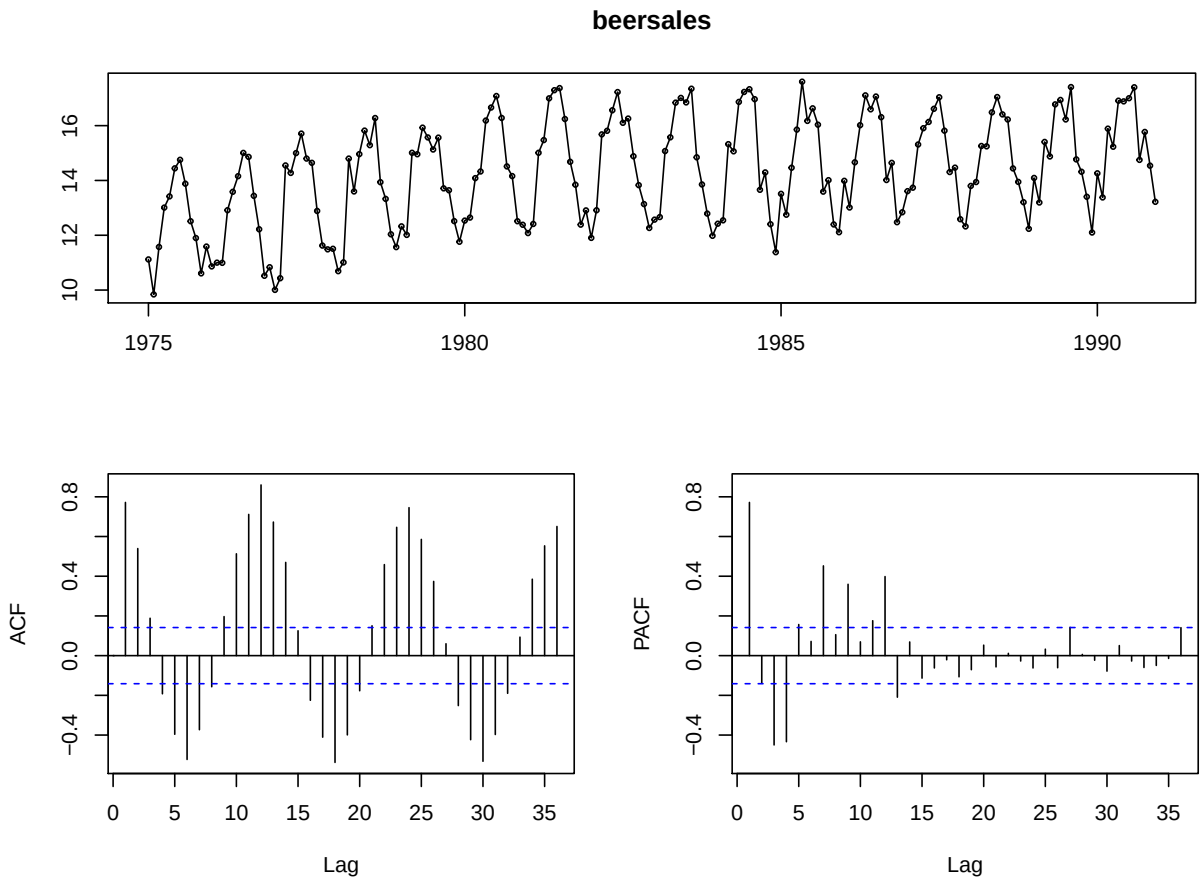
```
library(urca)
a = ur.df(prescrip,type = "trend",lags=0)
summary(a)
```

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

This returns an F-statistic of 4.862 which is the same as what we calculated in part(b).
The difference in the two tests is that the test in part(a) is an augmented Dickey-Fuller test which tests if there are any unit roots or not. Whereas, the test in part(b) tests the null hypothesis that $(\alpha, \beta, \phi) = (\alpha, 0, 1)$ i.e. it only tests for one unit root not for any number of unit roots. Basically, The augmented Dickey-Fuller test accomodates general ARMA(p,q) models with unkown orders.

Exercise 2
(a) First thing to do is load the data and create the plots

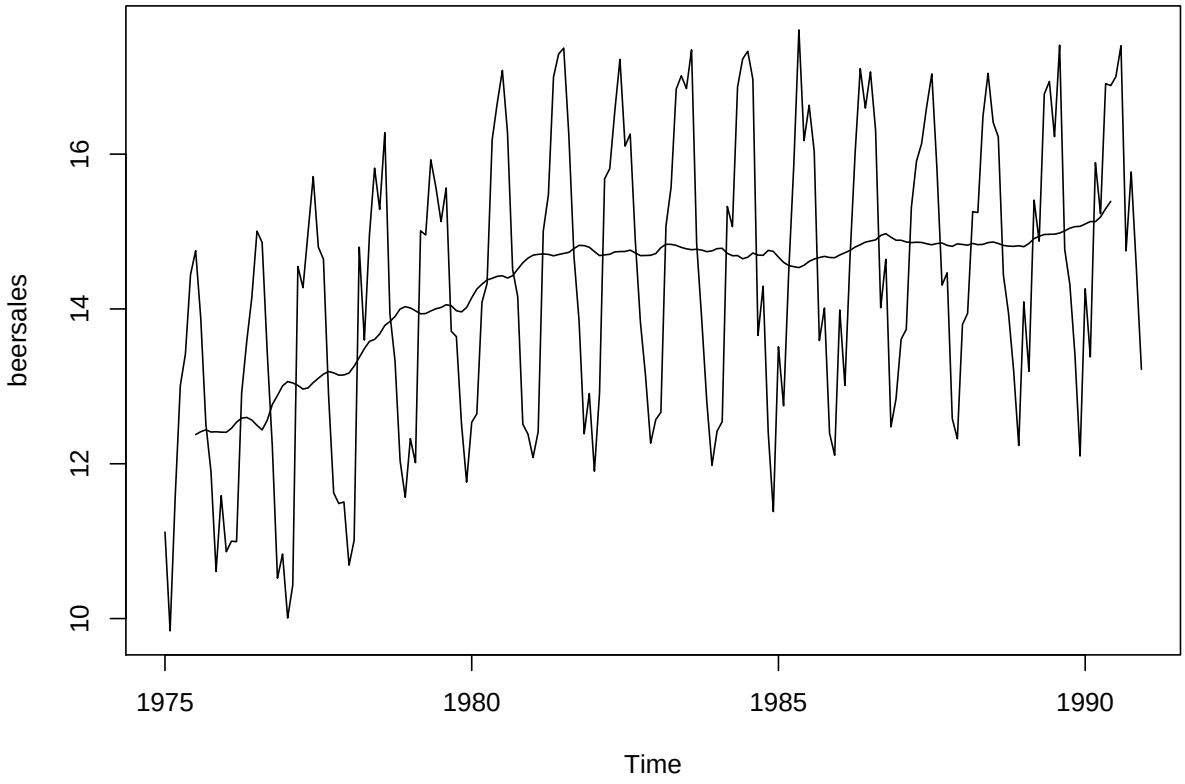
```
data("beersales")
library(forecast)
tsdisplay(beersales)
```



From the plot of the series we can that there appears to be a clear seasonal component in the time series due to the repeated pattern of up and down movements. This is confirmed when we look at the ACF plot. The ACF plot shows us that the period looks to be 12 months. When looking at the plot of the time series there also seems to be a slight upward trend.

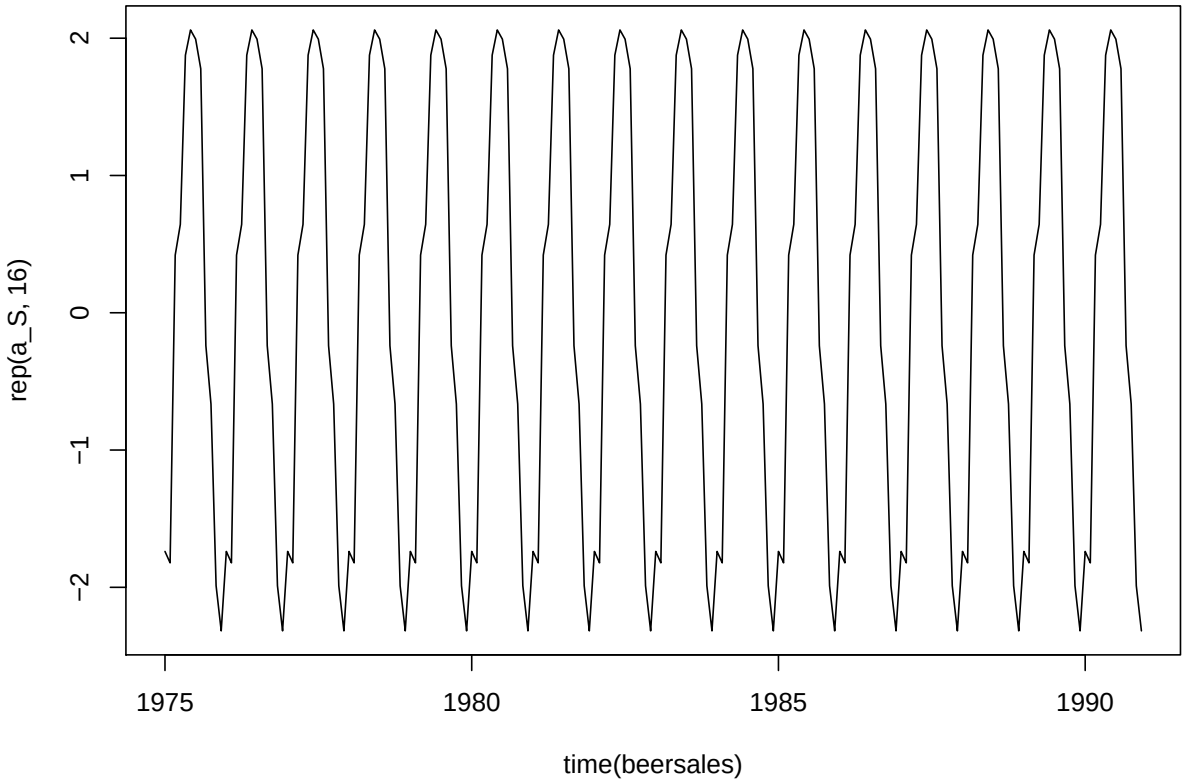
(*b*) To estimate the trend we can smooth the series as follows. We choose an order of 12 for the moving average smoother as this is the period of the time series. By choosing the order of the moving average smoother to be the period of the seasonal component we will effectively smooth out the seasonal component which then leaves the trend.

```
# estimate trend
TC=ma(beersales,12)
plot(beersales)
lines(TC)
```

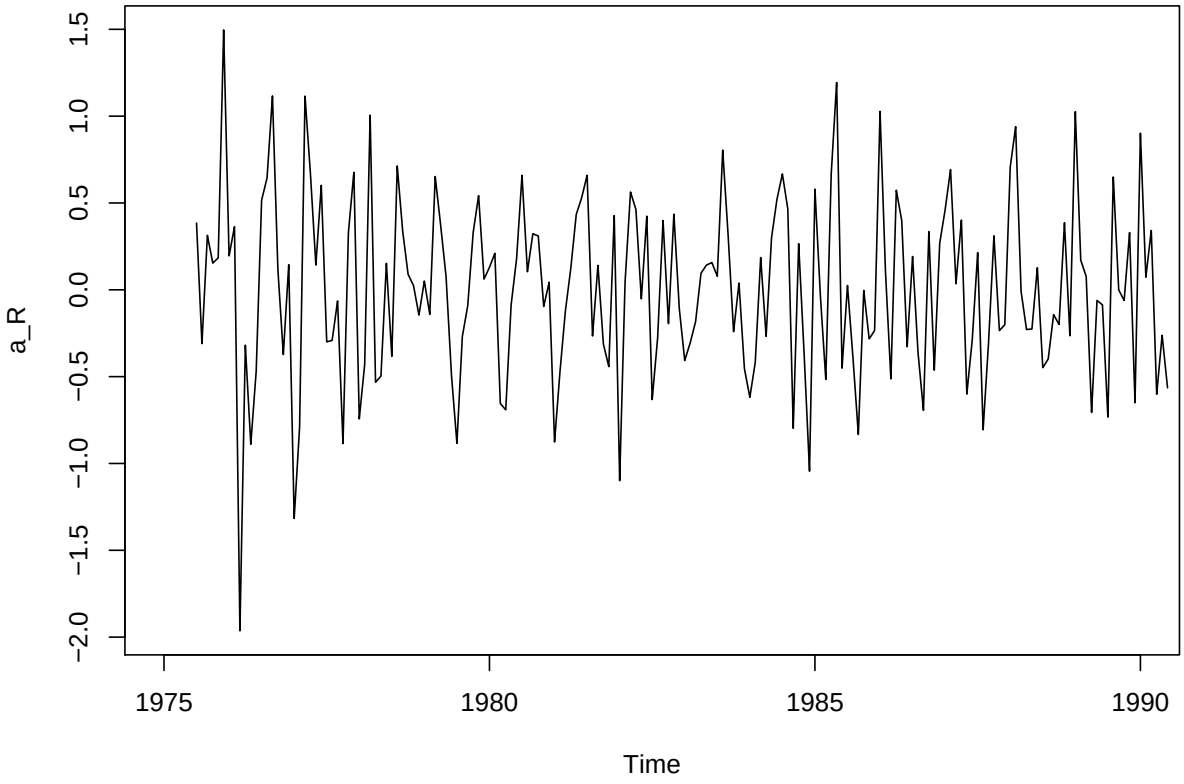


(c) To answer this question we use the following code

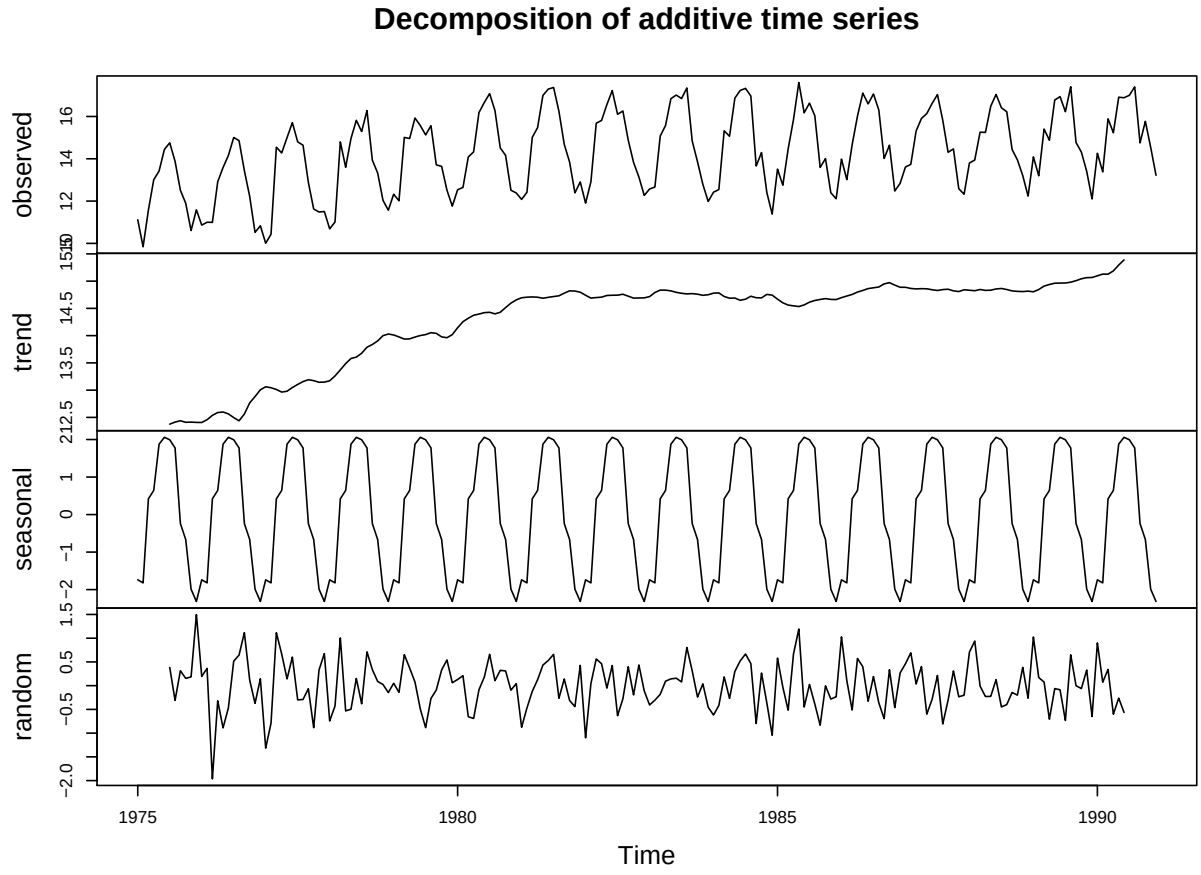
```
# we have trend component from part(b)

# seasonal component
a_pseudo_s = beersales-TC
a_matrix_s = matrix(a_pseudo_s,nrow=12)
a_S = rowMeans(a_matrix_s,na.rm=TRUE)
# centre seasonal component
a_S = a_S-mean(a_S)
# plot seasonal component
plot(time(beersales),rep(a_S,16),type="l")
```




```
# random component
a_R=beersales-(TC+rep(a_S,16))
plot(a_R,xlim=c(1975,1990),type="l")
```



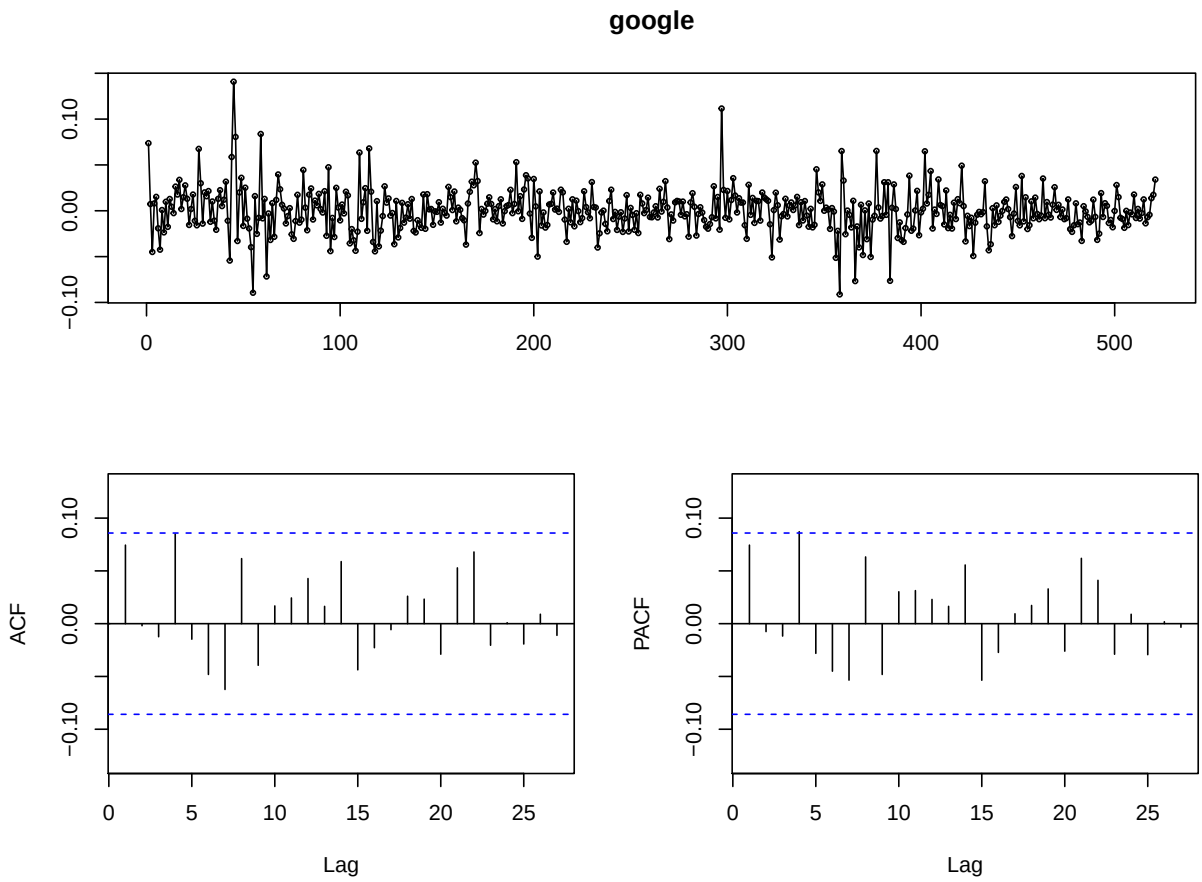

```
# check answers using decompose function
a_decomp=decompose(beersales,type="add")
plot(a_decomp)
```



Exercise 3

(a) First thing to do is load the data and plot it

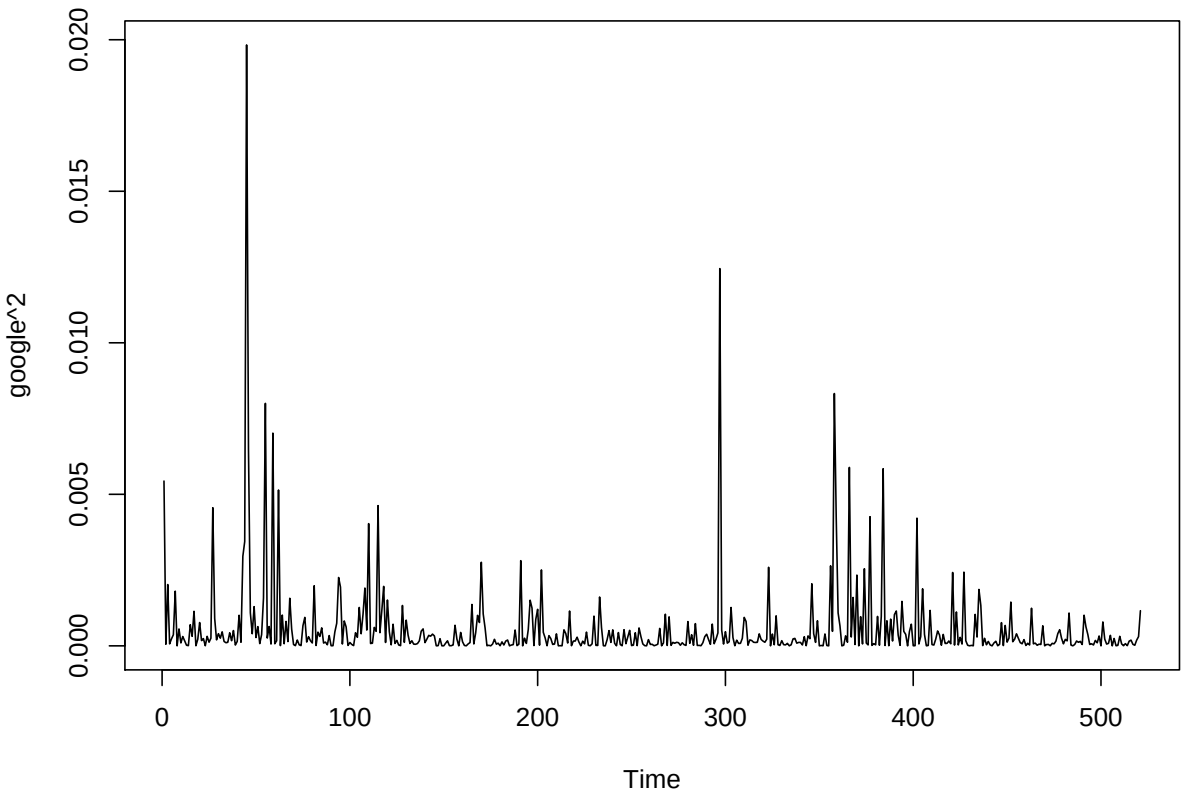
```
# load data
data("google")
# de-mean data
google = google-mean(google)
# create plots of series
tsdisplay(google)
```



We can see from these plots that the series looks to resemble white noise. The ACf and PACF plots return almost all null values so there are no obvious signs of the series being difference stationary. From the plot of the series there doesn't appear to be any trend so there are no obvious signs of the series being trend stationary. However, there does appear to be volatility clustering i.e. there are periods of relative calm but there are also periods of very high volatility. Therefore, it is possible the series is not stationary even though it doesn't appear to be trend stationary or difference stationary.

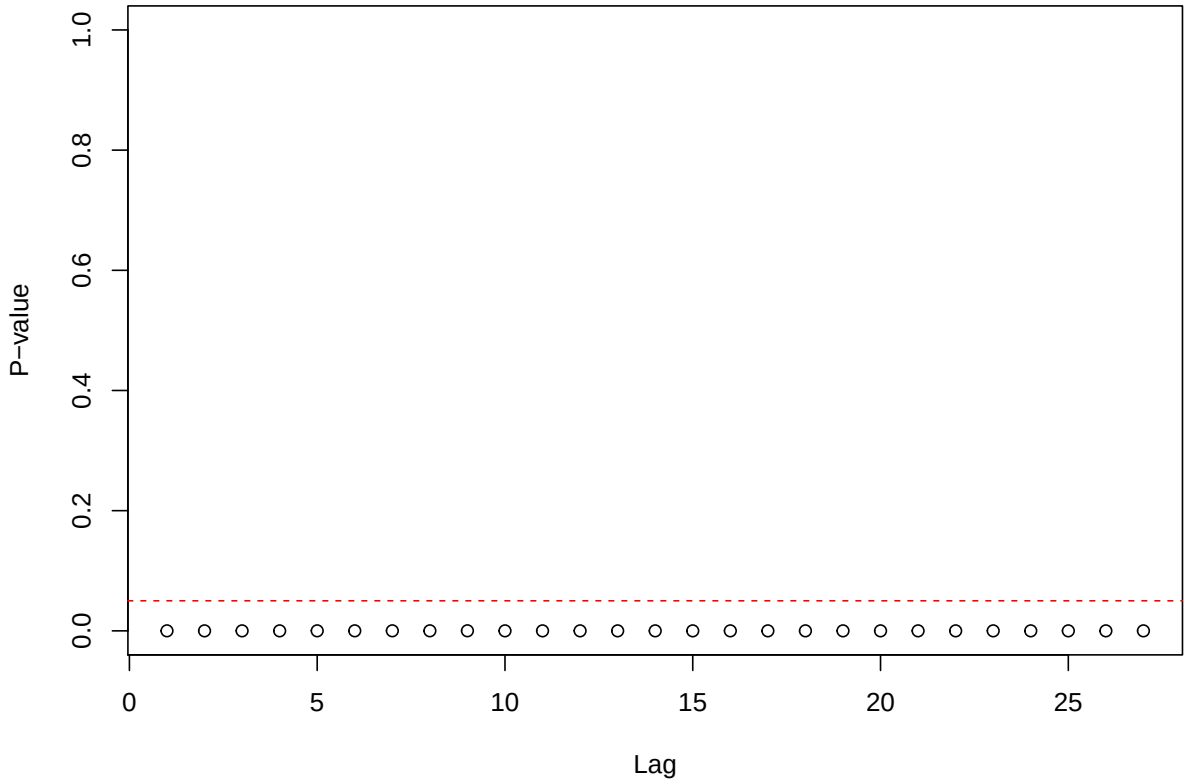
(*b*) Since volatility clustering is symmetric, looking at the squares of the series means we really get to see the volatility clustering.

```
plot(google^2)
```



We see high values followed by high values and low values followed by low values i.e. we have autocorrelation. It looks like there may be conditional heterskedasticity present so we perform a formal test to make sure. The test we use is the McLeod-Li test.

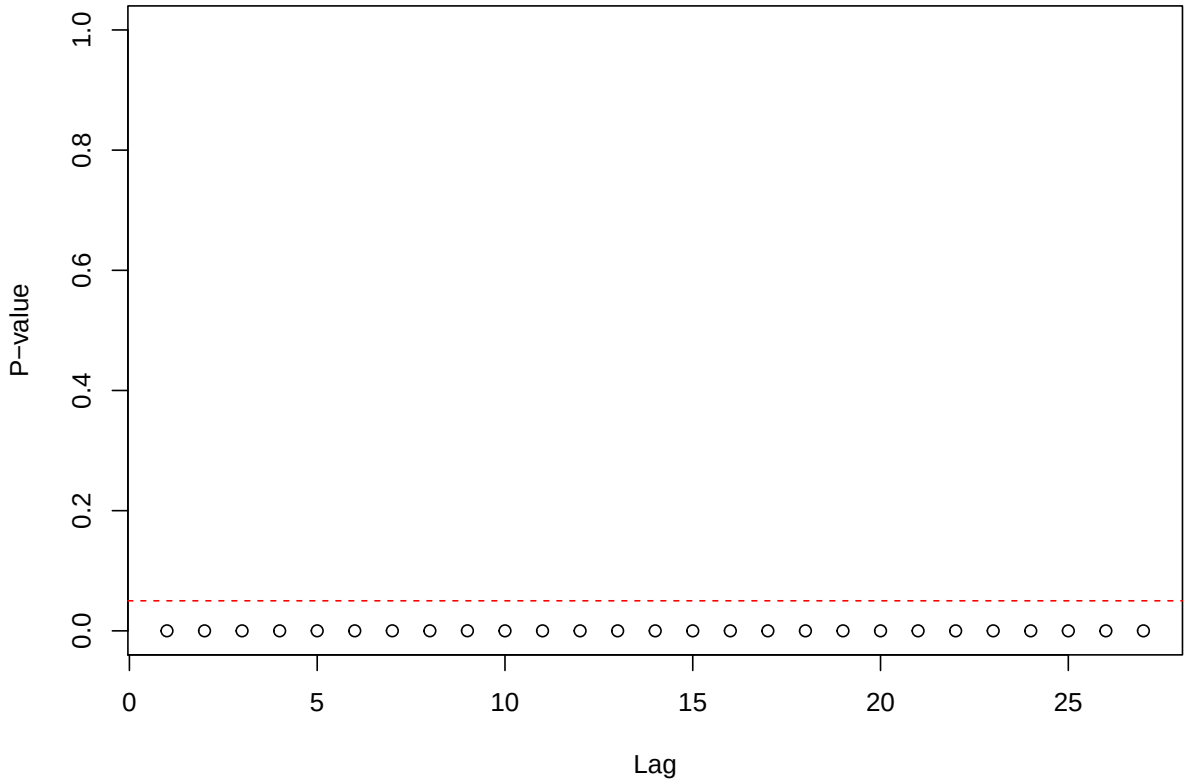
```
McLeod.Li.test(y=google)
```



The output of this test shows all p-values are below critical value so reject null hypothesis. The null hypothesis is that there is no autocorrelation so no ARCH behaviour. Thus, by rejecting the null hypothesis we are saying conditional heterskedasticity is present.

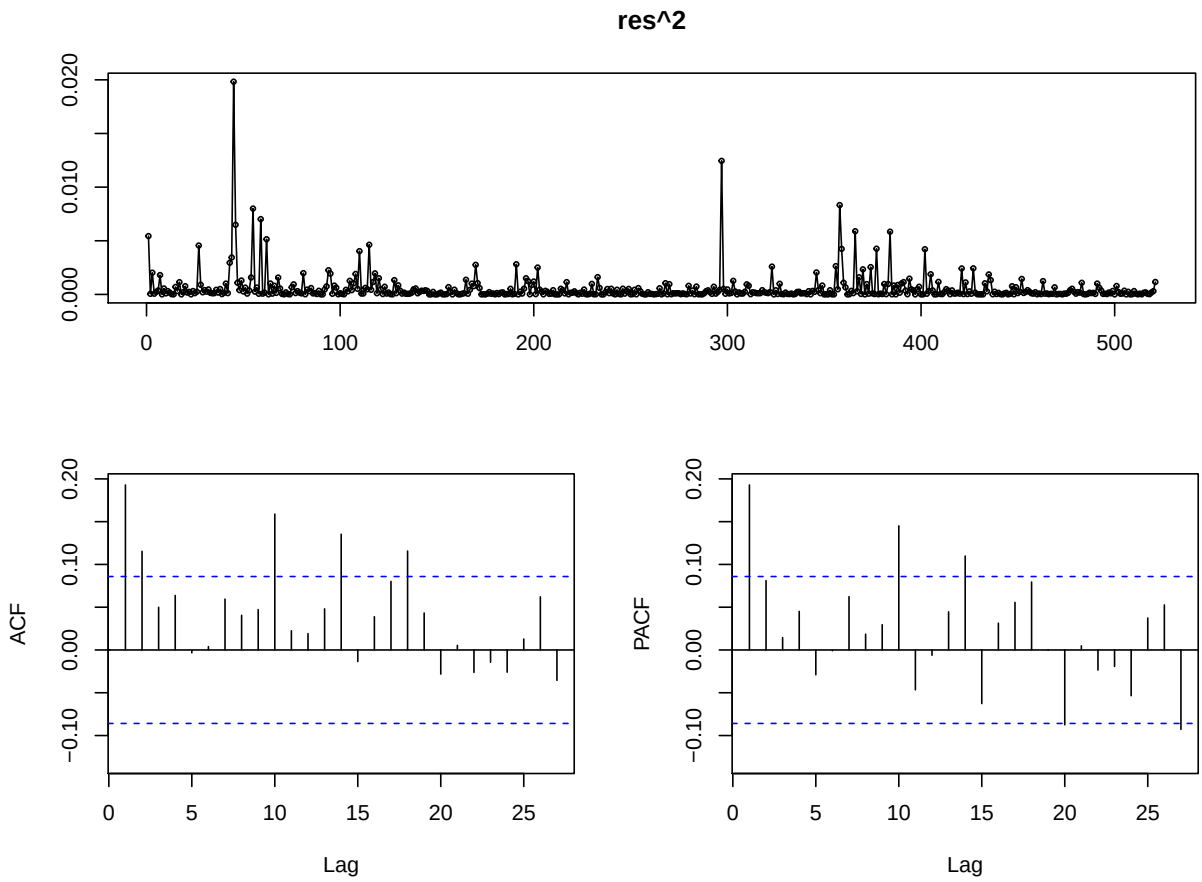
(c) From the ACF and PACF plots of the series we can see u=0 and v=0 so we can fit an ARMA(0,0) model to the series. Then we want to see if GARCH behaviour is present so perform McLeod-Li test on residuals of model.

```
m = arima((google),order=c(0,0,0))
res = residuals(m)
McLeod.Li.test(y=res)
```



This test shows that all p values are below critical level so there is volatility clustering. Therefore, we can proceed to determine the GARCH specification. We can do this by looking at the ACf and PACF of the squared residuals as these should follow an ARMA(max(p,q),p) model.

```
tsdisplay(res^2)
```



We can't determine p and q from these plots so we will have to fit several potential models to the data and examine statistical significance of estimated parameters.

```
library(fGarch)
summary(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 0.015589 secs
##
## --- END OF TRACE ---
##
##
## Time to Estimate Parameters:
##  Time difference of 0.04255199 secs
##
## 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: 0x000000001f2ad270>
##  [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:
##  Tue Dec 01 19:03:16 2020 by user: Kennedy
##
##
## 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
```

```
summary(garchFit(~arma(0,0)+garch(2,1),google,include.mean=F))
```

```
##
## Series Initialization:
##  ARMA Model:                arma
##  Formula Mean:              ~ arma(0, 0)
##  GARCH Model:               garch
##  Formula Variance:          ~ garch(2, 1)
##  ARMA Order:                0 0
##  Max ARMA Order:            0
##  GARCH Order:               2 1
##  Max GARCH Order:           2
##  Maximum Order:             2
##  Conditional Dist:          norm
##  h.start:                   3
##  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   0.00    FALSE
##     omega   1.000000e-06 1.000000e+02   0.10     TRUE
##     alpha1  1.000000e-08 1.000000e+00   0.05     TRUE
##     alpha2  1.000000e-08 1.000000e+00   0.05     TRUE
##     gamma1 -1.000000e+00 1.000000e+00   0.10    FALSE
##     gamma2 -1.000000e+00 1.000000e+00   0.10    FALSE
##     beta1   1.000000e-08 1.000000e+00   0.80     TRUE
##     delta   0.000000e+00 2.000000e+00   2.00    FALSE
##     skew    1.000000e-01 1.000000e+01   1.00    FALSE
##     shape   1.000000e+00 1.000000e+01   4.00    FALSE
##  Index List of Parameters to be Optimized:
##  omega alpha1 alpha2  beta1
##      2      3      4      7
##  Persistence:                  0.9
##
##
## --- START OF TRACE ---
## Selected Algorithm: nlminb
##
## R coded nlminb Solver:
```

```
##
##   0:     712.67842: 0.100000 0.0500000 0.0500000 0.800000
##   1:     712.14039: 0.0928745 0.0532717 0.0487738 0.794580
##   2:     711.83636: 0.0941552 0.0616684 0.0529096 0.796342
##   3:     711.07916: 0.0821285 0.0730120 0.0520005 0.786581
##   4:     710.66687: 0.0870600 0.0908023 0.0499684 0.781635
##   5:     710.32203: 0.0923348 0.0985931 0.0375749 0.770353
##   6:     710.15670: 0.0957391 0.115365 0.0313662 0.764191
##   7:     710.14739: 0.0922119 0.115498 0.0308554 0.762031
##   8:     710.10987: 0.0943451 0.117288 0.0331310 0.759921
##   9:     710.10205: 0.0965716 0.117822 0.0304801 0.757659
##  10:     710.08454: 0.0943601 0.125330 0.0305994 0.754784
##  11:     710.08018: 0.0951330 0.125556 0.0306330 0.755061
##  12:     710.07747: 0.0953168 0.125299 0.0299961 0.754591
##  13:     710.07417: 0.0964590 0.125544 0.0306324 0.753525
##  14:     710.07019: 0.0976623 0.125720 0.0315367 0.750472
##  15:     710.06678: 0.0968611 0.128793 0.0264285 0.753680
##  16:     710.06227: 0.0996719 0.128839 0.0293444 0.748196
##  17:     710.06212: 0.0996543 0.128788 0.0292333 0.748086
##  18:     710.06200: 0.0998086 0.128833 0.0291942 0.748079
##  19:     710.06177: 0.0999070 0.128940 0.0290943 0.747799
##  20:     710.06096: 0.100787 0.130087 0.0291903 0.745938
##  21:     710.06079: 0.102002 0.129836 0.0299085 0.744167
##  22:     710.06074: 0.101743 0.130099 0.0298089 0.744357
##  23:     710.06074: 0.101709 0.130095 0.0297340 0.744460
##  24:     710.06074: 0.101718 0.130090 0.0297557 0.744436
##
## Final Estimate of the Negative LLH:
##  LLH:  -1236.118    norm LLH:  -2.372587
##        omega       alpha1       alpha2        beta1
## 5.791765e-05 1.300897e-01 2.975569e-02 7.444359e-01
##
## R-optimhess Difference Approximated Hessian Matrix:
##               omega       alpha1       alpha2        beta1
## omega  -21261688197 -5949329.773 -6334564.829 -9358236.051
## alpha1      -5949330    -2878.124    -2831.629    -3261.302
## alpha2      -6334565    -2831.629    -3241.790    -3590.630
## beta1       -9358236    -3261.302    -3590.630    -4707.923
## attr(,"time")
## Time difference of 0.01695609 secs
##
## --- END OF TRACE ---
##
##
## Time to Estimate Parameters:
##  Time difference of 0.06582713 secs
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~arma(0, 0) + garch(2, 1), data = google,
##     include.mean = F)
##
```

```
## Mean and Variance Equation:
##  data ~ arma(0, 0) + garch(2, 1)
## <environment: 0x000000001dd7c758>
##  [data = google]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##      omega      alpha1      alpha2       beta1
## 5.7918e-05  1.3009e-01  2.9756e-02  7.4444e-01
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## omega  5.792e-05   2.693e-05    2.151   0.0315 *
## alpha1 1.301e-01   5.108e-02    2.547   0.0109 *
## alpha2 2.976e-02   7.154e-02    0.416   0.6775
## beta1  7.444e-01   9.412e-02    7.909 2.66e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1236.118    normalized:  2.372587
##
## Description:
##  Tue Dec 01 19:03:16 2020 by user: Kennedy
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  193.5159  0
##  Shapiro-Wilk Test  R    W      0.9686692 4.177974e-09
##  Ljung-Box Test     R    Q(10)  11.87735  0.2933488
##  Ljung-Box Test     R    Q(15)  16.35679  0.3587434
##  Ljung-Box Test     R    Q(20)  17.2408   0.6372869
##  Ljung-Box Test     R^2  Q(10)  5.22696   0.8755102
##  Ljung-Box Test     R^2  Q(15)  8.921716  0.881575
##  Ljung-Box Test     R^2  Q(20)  13.46368  0.8566157
##  LM Arch Test       R    TR^2   5.122092  0.9537761
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -4.729819 -4.697145 -4.729935 -4.717020
```

```
summary(garchFit(~arma(0,0)+garch(1,2),google,include.mean=F))
```

```
##
## Series Initialization:
##  ARMA Model:                 arma
##  Formula Mean:               ~ arma(0, 0)
##  GARCH Model:                garch
##  Formula Variance:           ~ garch(1, 2)
```

```
##  ARMA Order:                 0 0
##  Max ARMA Order:             0
##  GARCH Order:                1 2
##  Max GARCH Order:            2
##  Maximum Order:              2
##  Conditional Dist:           norm
##  h.start:                    3
##  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    0.4     TRUE
##     beta2   1.000000e-08 1.000000e+00    0.4     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  beta2
##      2      3      5      6
##  Persistence:                 0.9
##
##
## --- START OF TRACE ---
## Selected Algorithm: nlminb
##
## R coded nlminb Solver:
##
##   0:     712.79953: 0.100000 0.100000 0.400000 0.400000
##   1:     712.41717: 0.0960963 0.0999178 0.397109 0.396824
##   2:     712.18192: 0.0951223 0.105597 0.397813 0.396870
##   3:     712.10763: 0.0877545 0.111551 0.393916 0.391406
##   4:     711.56163: 0.0903552 0.122613 0.395854 0.390030
##   5:     711.36715: 0.0912852 0.129351 0.393006 0.381063
##   6:     711.15127: 0.100216 0.140121 0.397955 0.363207
##   7:     711.01497: 0.0921139 0.142280 0.414652 0.349421
##   8:     710.85474: 0.104486 0.141645 0.442634 0.308128
##   9:     710.83113: 0.104568 0.143675 0.443604 0.308573
##  10:     710.80932: 0.102992 0.144228 0.443544 0.307000
##  11:     710.78124: 0.102983 0.145627 0.447349 0.304847
##  12:     710.74243: 0.101219 0.145692 0.453483 0.298247
##  13:     710.46598: 0.0944919 0.139956 0.547358 0.219927
##  14:     710.43270: 0.115050 0.183353 0.594658 0.117547
##  15:     710.28505: 0.105983 0.168545 0.644798 0.0868752
```

```
##  16:     710.25736: 0.0832284 0.129872 0.722304 0.0681078
##  17:     710.16051: 0.0924951 0.145952 0.714219 0.0507226
##  18:     710.15759: 0.0919388 0.145138 0.719754 0.0474798
##  19:     710.15725: 0.0909871 0.143506 0.728234 0.0412231
##  20:     710.15720: 0.0912576 0.143974 0.726792 0.0420312
##  21:     710.15720: 0.0912572 0.143973 0.726849 0.0419769
##
## Final Estimate of the Negative LLH:
##  LLH:  -1236.021    norm LLH:  -2.372402
##        omega       alpha1        beta1        beta2
## 5.196143e-05 1.439731e-01 7.268490e-01 4.197687e-02
##
## R-optimhess Difference Approximated Hessian Matrix:
##               omega       alpha1         beta1         beta2
## omega  -25758829823 -7467964.629 -11388818.131 -11567905.313
## alpha1     -7467965    -3586.786     -4110.111     -4150.460
## beta1     -11388818    -4110.111     -5735.818     -5839.903
## beta2     -11567905    -4150.460     -5839.903     -5963.837
## attr(,"time")
## Time difference of 0.02112699 secs
##
## --- END OF TRACE ---
##
##
## Time to Estimate Parameters:
##  Time difference of 0.06071901 secs
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~arma(0, 0) + garch(1, 2), data = google,
##     include.mean = F)
##
## Mean and Variance Equation:
##  data ~ arma(0, 0) + garch(1, 2)
## <environment: 0x000000002323eeb0>
##  [data = google]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##      omega      alpha1       beta1       beta2
## 5.1961e-05  1.4397e-01  7.2685e-01  4.1977e-02
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## omega  5.196e-05   2.162e-05    2.403  0.01625 *
## alpha1 1.440e-01   5.017e-02    2.870  0.00411 **
## beta1  7.268e-01   3.049e-01    2.384  0.01713 *
```

```
## beta2  4.198e-02   2.636e-01    0.159  0.87346
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  1236.021    normalized:  2.372402
##
## Description:
##  Tue Dec 01 19:03:16 2020 by user: Kennedy
##
##
## Standardised Residuals Tests:
##                                 Statistic p-Value
##  Jarque-Bera Test   R    Chi^2  193.989   0
##  Shapiro-Wilk Test  R    W      0.9689277 4.701402e-09
##  Ljung-Box Test     R    Q(10)  11.95814  0.2878681
##  Ljung-Box Test     R    Q(15)  16.30685  0.3619563
##  Ljung-Box Test     R    Q(20)  17.13208  0.6443811
##  Ljung-Box Test     R^2  Q(10)  5.347539  0.8667866
##  Ljung-Box Test     R^2  Q(15)  9.029636  0.8759628
##  Ljung-Box Test     R^2  Q(20)  13.18301  0.8693944
##  LM Arch Test       R    TR^2   5.055834  0.956088
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -4.729448 -4.696775 -4.729565 -4.716650
```

```
# fit chosen model
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
```

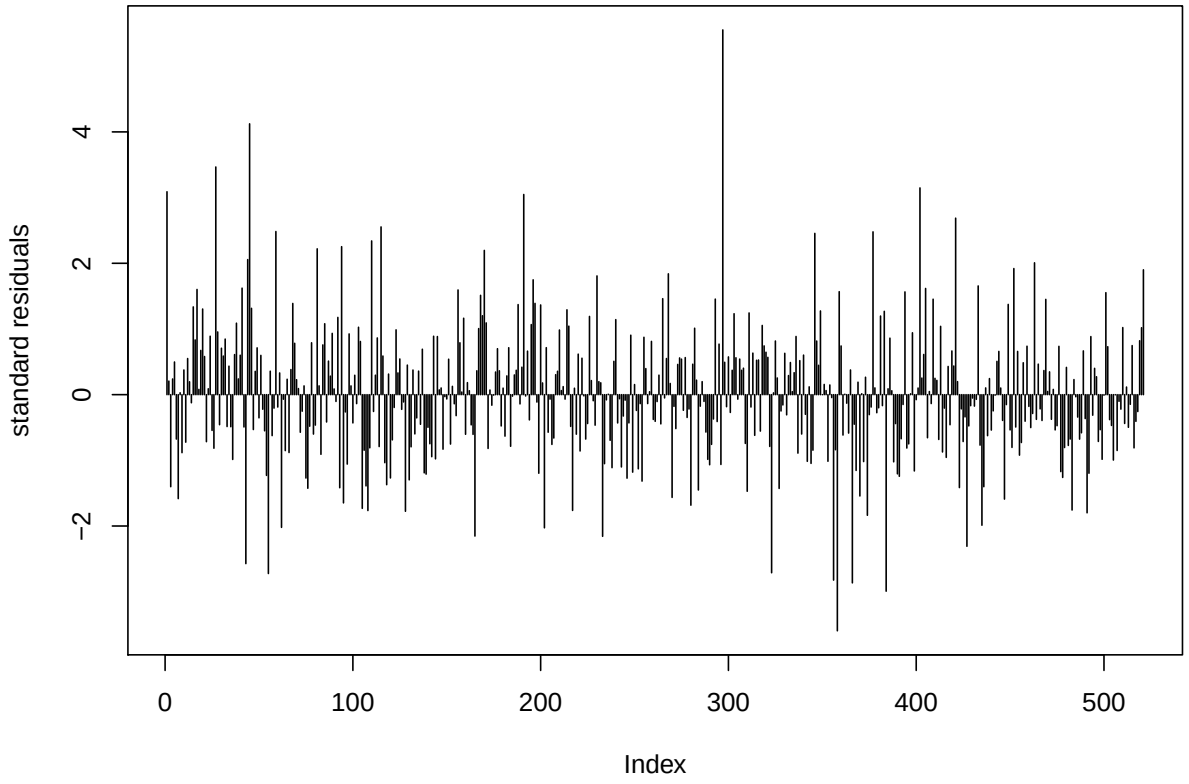
```
##     omega   1.000000e-06 1.000000e+02    0.1     TRUE
##     alpha1  1.000000e-08 1.000000e+00    0.1     TRUE
##     gamma1 -1.000000e+00 1.000000e+00    0.1    FALSE
##     beta1   1.000000e-08 1.000000e+00    0.8     TRUE
##     delta   0.000000e+00 2.000000e+00    2.0    FALSE
##     skew    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 0.009972095 secs
##
## --- END OF TRACE ---
##
```

```
##
## Time to Estimate Parameters:
##  Time difference of 0.05226111 secs
```

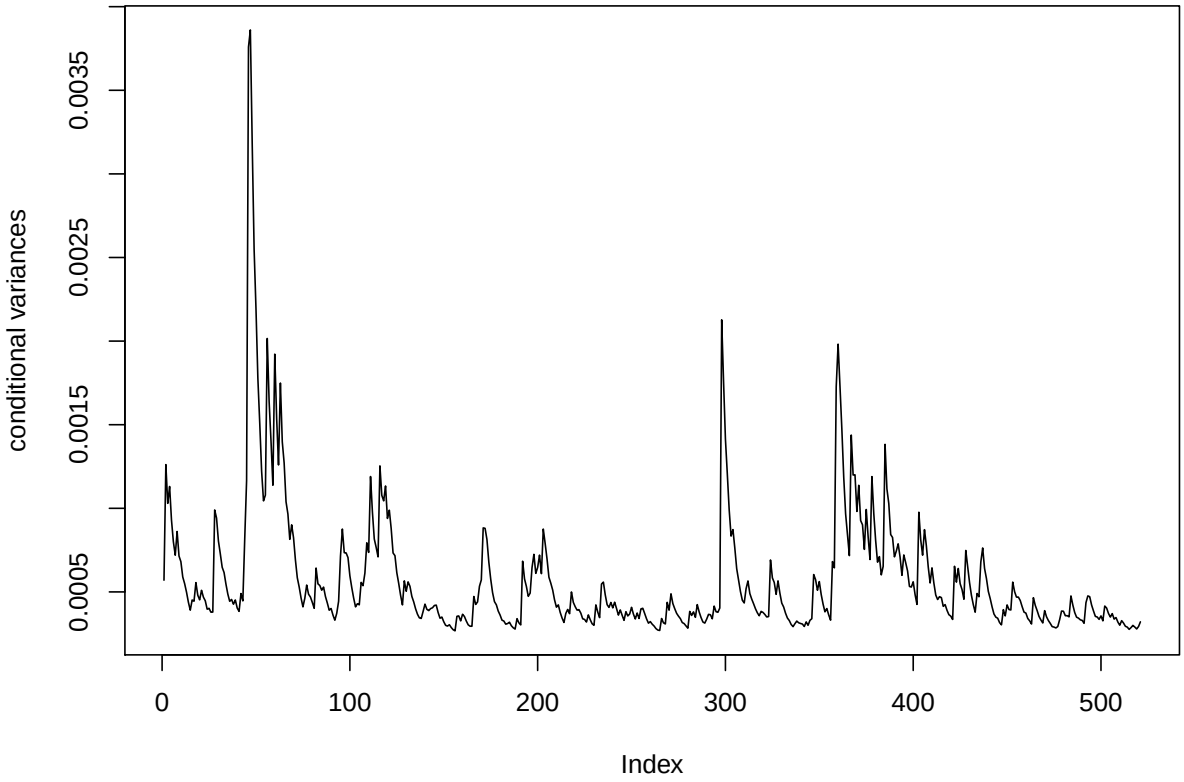
We find that all parameters are significant for GARCH(1,1) and if we increase p or q then we get un-needed parameters. Therfore, we choose an ARMA(0,0)+GARCH(1,1) model for the data.

(d) To plot the conditional variances and standardised residuals we use the following code

```
# plot standardised residuals
plot(fit2@residuals/fit2@sigma.t,type='h',ylab='standard residuals')
```




```
# plot conditional variances
plot(fit2@sigma.t^2,type='l',ylab='conditional variances')
```


conditional variances
0.0005
0.0015
0.0025
0.0035
0
100
200
300
400
500
Index