Volatility Modeling - ARCH/GARCH

Jai Vrat Singh

11/09/2018

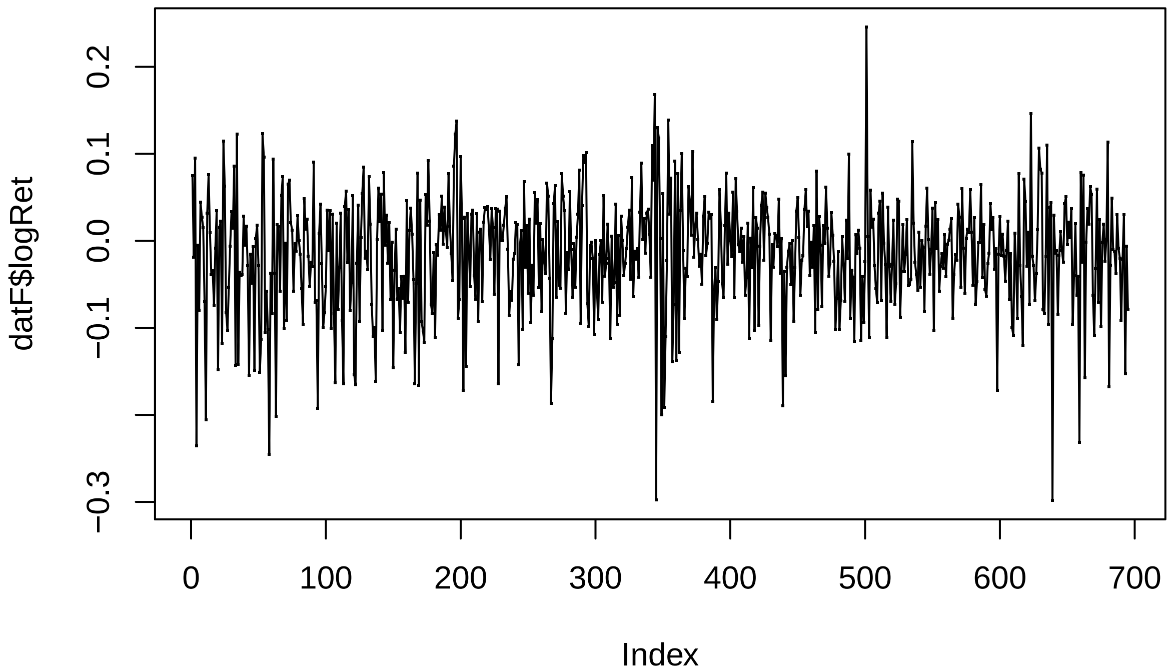
R Markdown

```
ret.what  <- list(date=numeric(), returns=numeric())
ret.widths <- c(12, 8)
strip.white <- c(TRUE, TRUE)

datF = scan(file = "http://faculty.chicagobooth.edu/ruey.tsay/teaching/fts2/m-3m4603.txt",
             what = ret.what, strip.white = strip.white)
datF = as.data.frame(datF)


#Above are simple returns, Convert returns to logReturns
datF[["logRet"]] = log(1- datF$returns)

head(datF)
```

```
##       date  returns       logRet
## 1 19460228 -0.07792  0.075033258
## 2 19460330  0.01859 -0.018764966
## 3 19460430 -0.10000  0.095310180
## 4 19460531  0.20988 -0.235570446
## 5 19460628  0.00513 -0.005143204
## 6 19460731  0.07653 -0.079616965
```

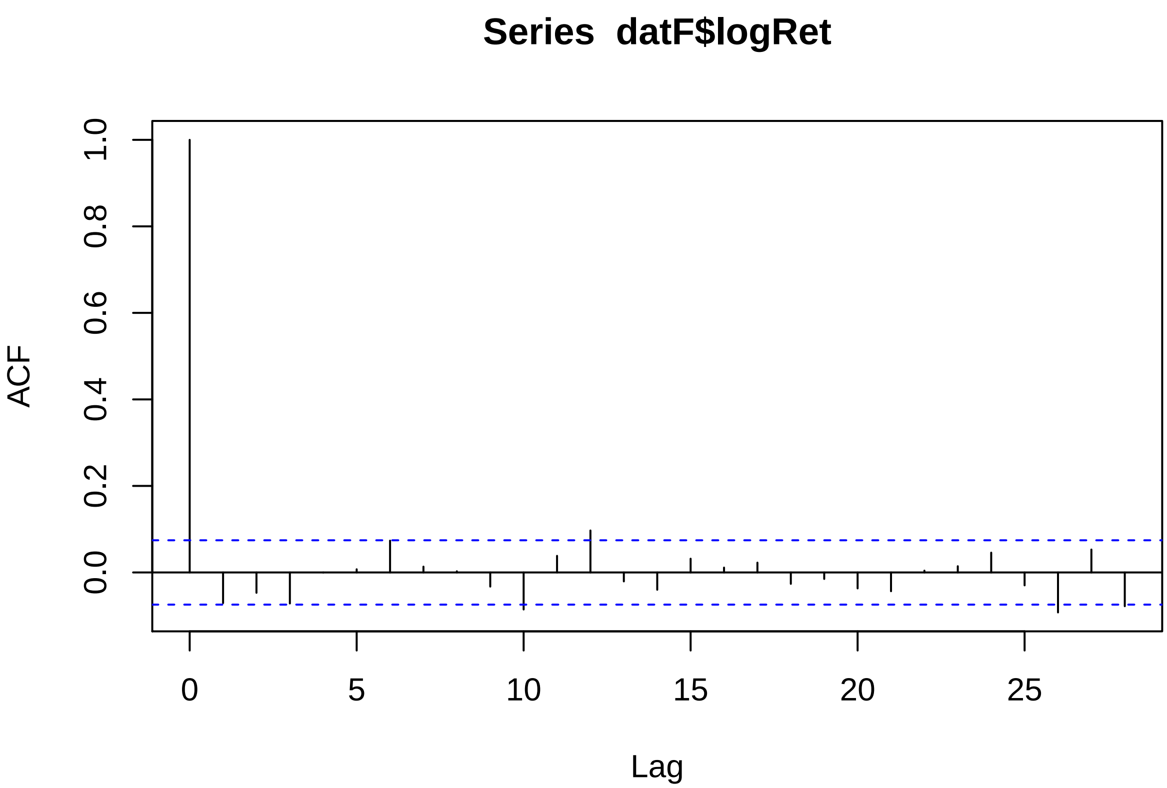
Evidence of ARCH effect

```
plot(datF$logRet, type = "o", pch = ".")
```

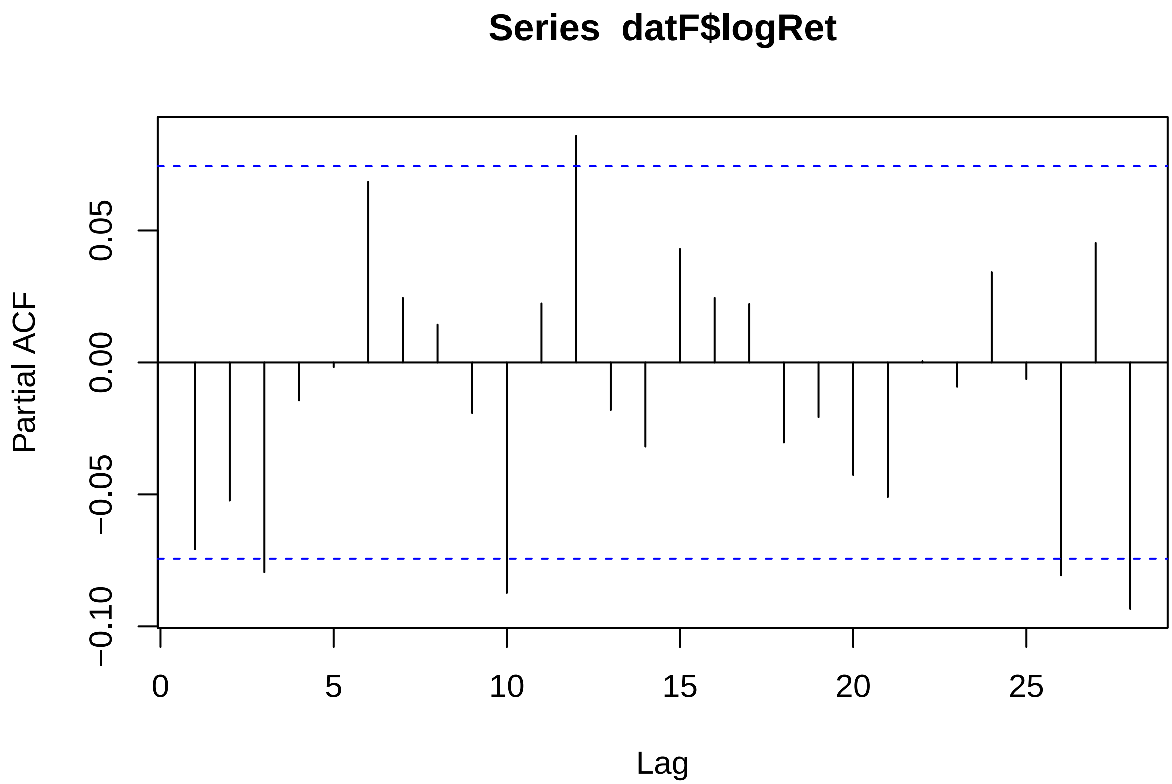


This plot reflects a pattern around the line logRet = 0, We need to look into ACF and PACF to get lagged correlations and its exact nature before the ARCH effect can be detected.

```
#par(mfrow=c(2,1))
acf(datF$logRet,)
```



```r
acf(datF$logRet, type = "partial")
```



ACF till lags 12 => MA(12) PACF till 3, 12 => AR(3), AR(12)

Try these

```r
#AR(12)
logret.fit = arima(x = datF$logRet, order = c(12,0,0))
logret.fit
```

```
##
## Call:
## arima(x = datF$logRet, order = c(12, 0, 0))
##
## Coefficients:
##           ar1      ar2      ar3      ar4     ar5     ar6     ar7     ar8
##       -0.0838  -0.0500  -0.0708  -0.0027  0.0041  0.0649  0.0181  0.0088
## s.e.   0.0378   0.0379   0.0380   0.0383  0.0383  0.0383  0.0383  0.0383
##           ar9     ar10    ar11    ar12  intercept
##       -0.0197  -0.0816  0.0307  0.0882    -0.0161
## s.e.   0.0383   0.0382  0.0384  0.0383     0.0023
##
## sigma^2 estimated as 0.004346:  log likelihood = 903.57,  aic = -1779.14
```

```r
#aic = -1779.14

#AR(3)
logret.fit = arima(x = datF$logRet, order = c(3,0,0))
logret.fit
```

```
##
```

```
## Call:
## arima(x = datF$logRet, order = c(3, 0, 0))
##
## Coefficients:
##           ar1      ar2      ar3  intercept
##       -0.0794  -0.0584  -0.0808    -0.0161
## s.e.   0.0379   0.0379   0.0380     0.0021
##
## sigma^2 estimated as 0.004445:  log likelihood = 895.85,  aic = -1781.7
```

```
#aic = -1781.7

#MA(12)
logret.fit = arima(x = datF$logRet, order = c(0,0,12))
logret.fit
```

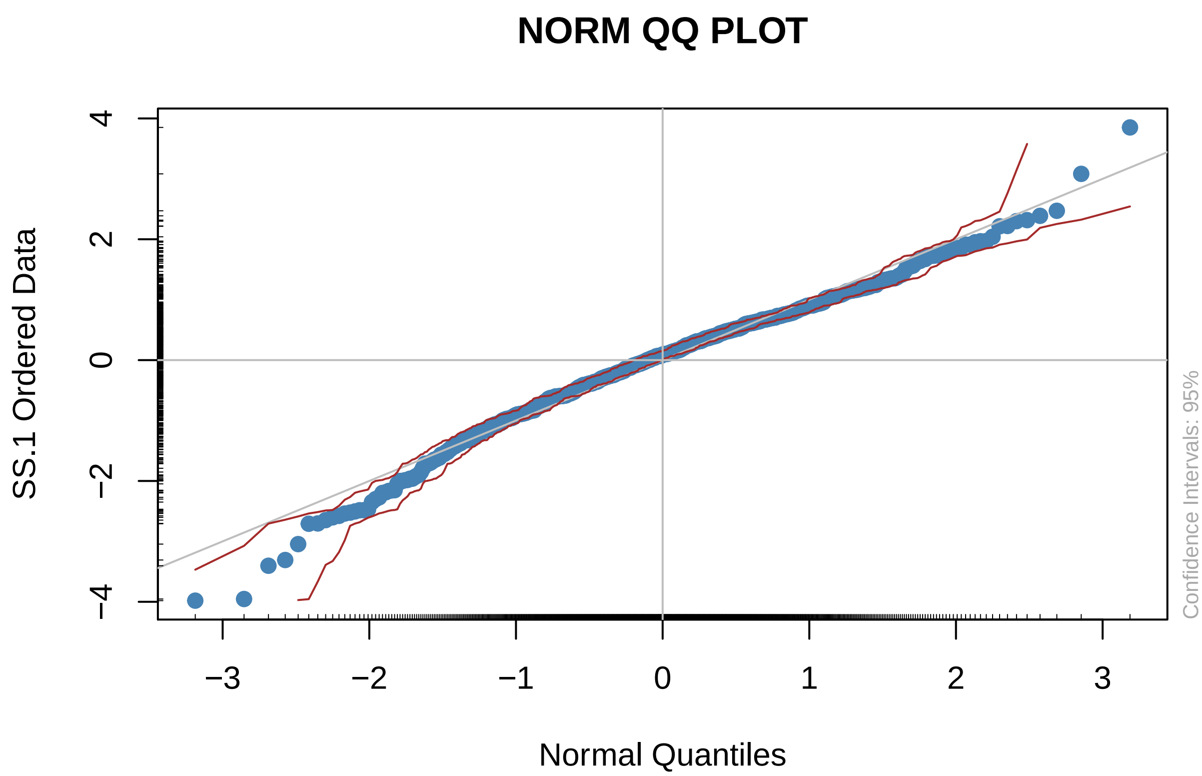
```
##
## Call:
## arima(x = datF$logRet, order = c(0, 0, 12))
##
## Coefficients:
##           ma1      ma2      ma3     ma4     ma5     ma6     ma7      ma8
##       -0.0821  -0.0402  -0.0691  0.0066  0.0121  0.0800  0.0165  -0.0001
## s.e.   0.0378   0.0378   0.0377  0.0382  0.0383  0.0386  0.0380   0.0390
##           ma9     ma10    ma11    ma12  intercept
##       -0.0316  -0.0909  0.0377  0.1035    -0.0161
## s.e.   0.0405   0.0396  0.0382  0.0376     0.0024
##
## sigma^2 estimated as 0.004338:  log likelihood = 904.24,  aic = -1780.48
```

```
#aic = -1780.48
```

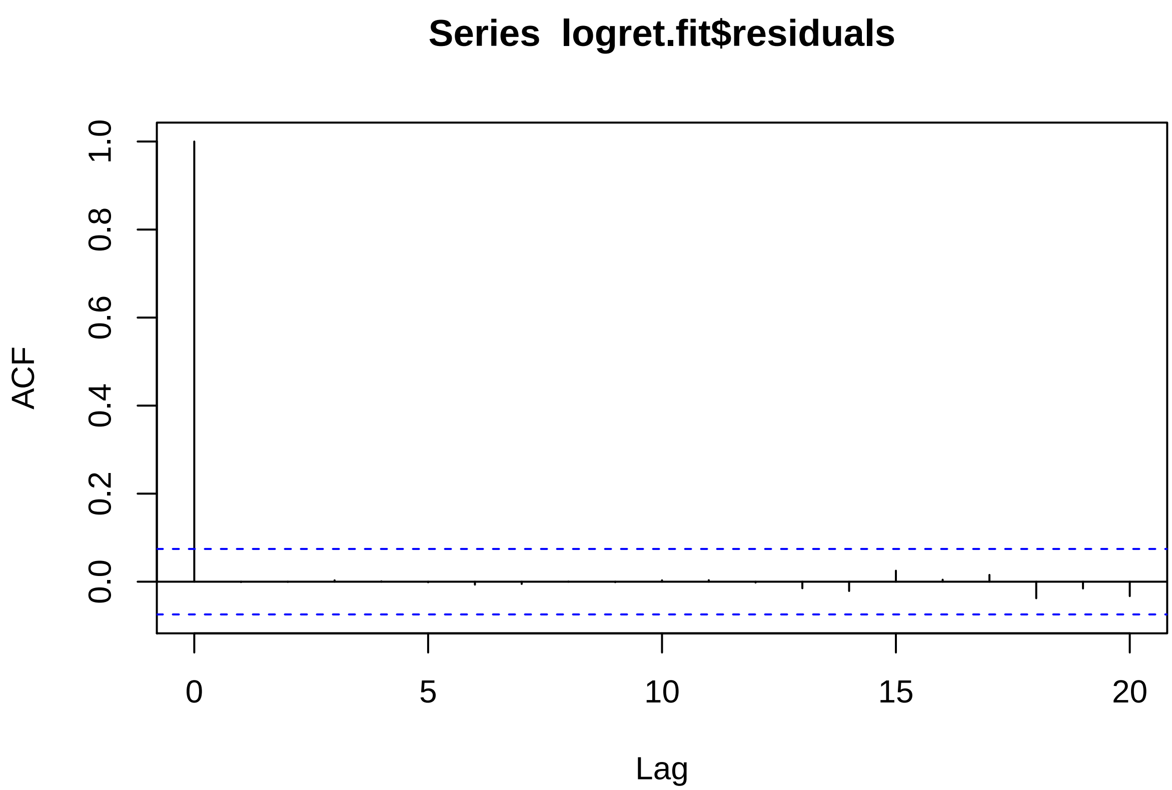
Lowest AIC is for AR(3) => seems it is best fit

Residue analysis of above fit

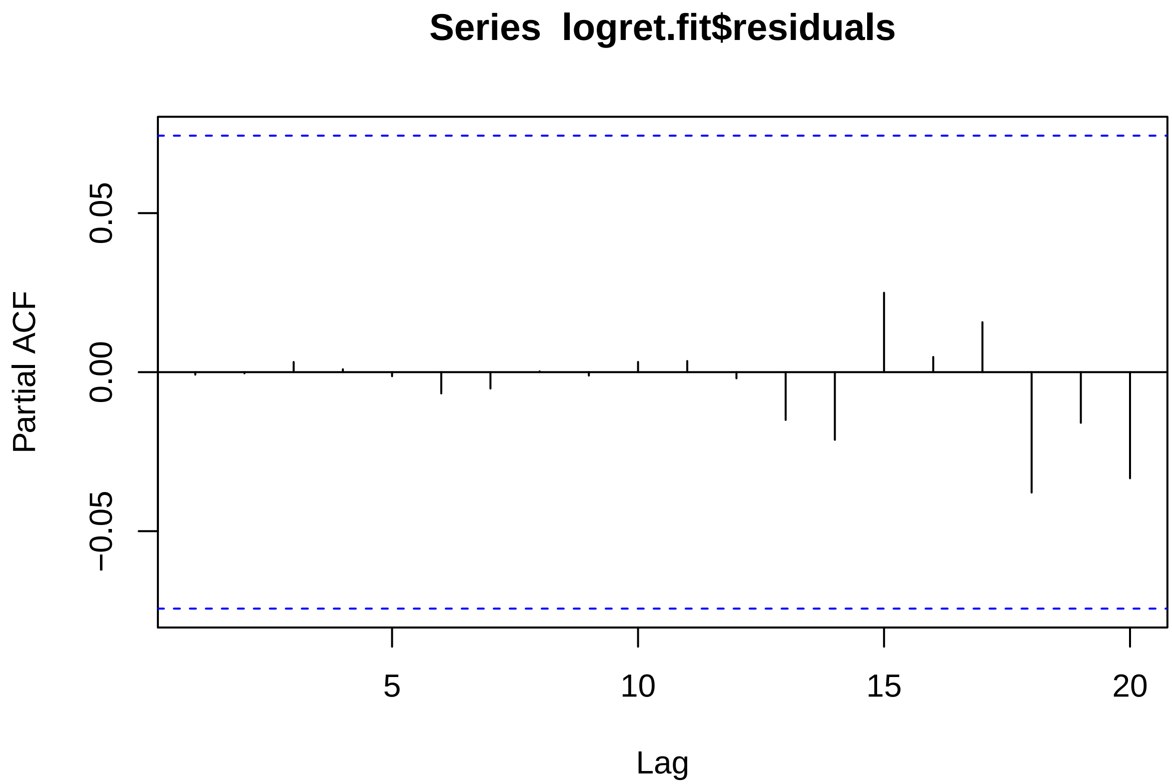
```
#logret.fit$residuals
qqnormPlot(logret.fit$residuals)
```




```
acf(logret.fit$residuals, lag.max = 20)
```




```
acf(logret.fit$residuals, type = "partial", lag.max = 20)
```



Trying various models and Residual analysis, seems that MA(12) is the best fit model.

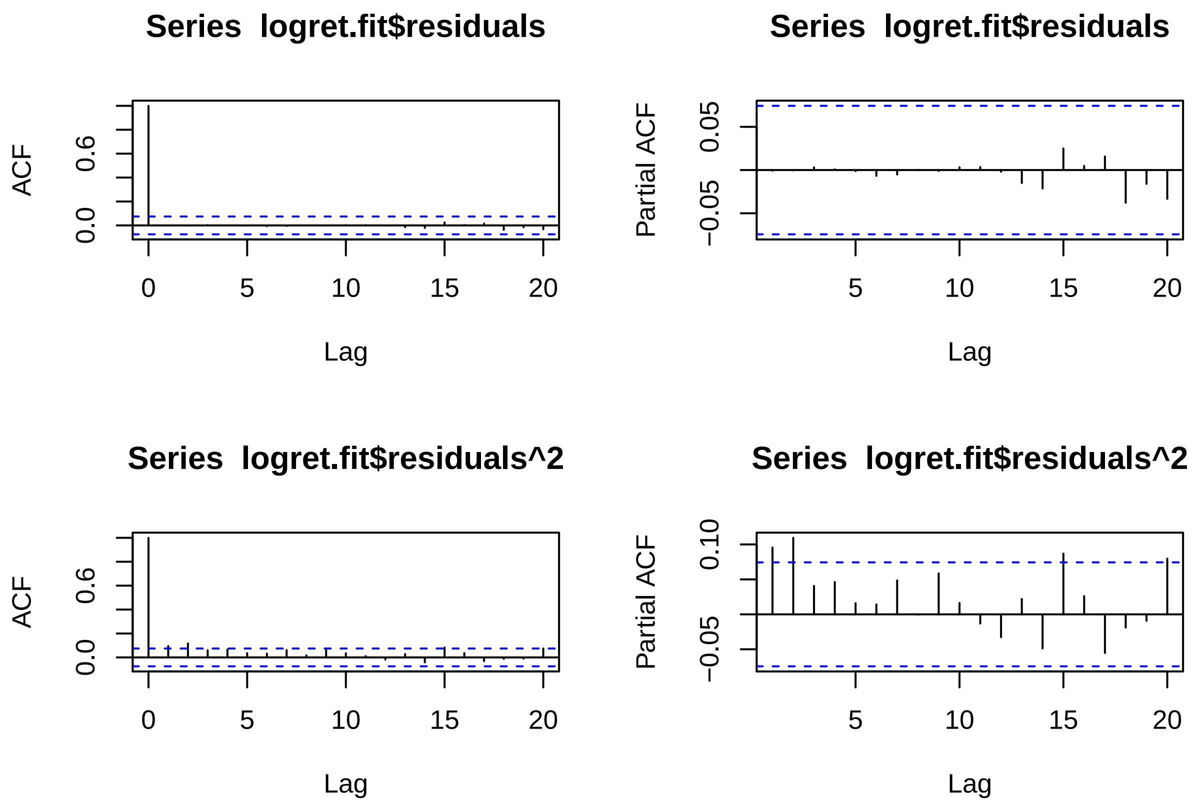
```
lags <- c(1:20)
sapply(lags, FUN = function(lag) {
                    test <- Box.test(logret.fit$residuals, lag = lag, type = c("Box-Pierce", "Ljung-Box
                    test$p.value
})
```

```
##  [1] 0.9833876 0.9997333 0.9998227 0.9999916 0.9999995 0.9999986 0.9999996
##  [8] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## [15] 0.9999998 0.9999999 1.0000000 0.9999980 0.9999988 0.9999952
```

P-values are very high => the residual is white noise.

ARCH effect on residuals

```
par(mfrow=c(2,2))
acf(logret.fit$residuals, lag.max = 20)
pacf(logret.fit$residuals, lag.max = 20)
acf(logret.fit$residuals^2, lag.max = 20)
pacf(logret.fit$residuals^2, lag.max = 20)
```



ACF and PACF of residuals clearly show conditional heteroscedasticity

Let us do LB test on residuals^2

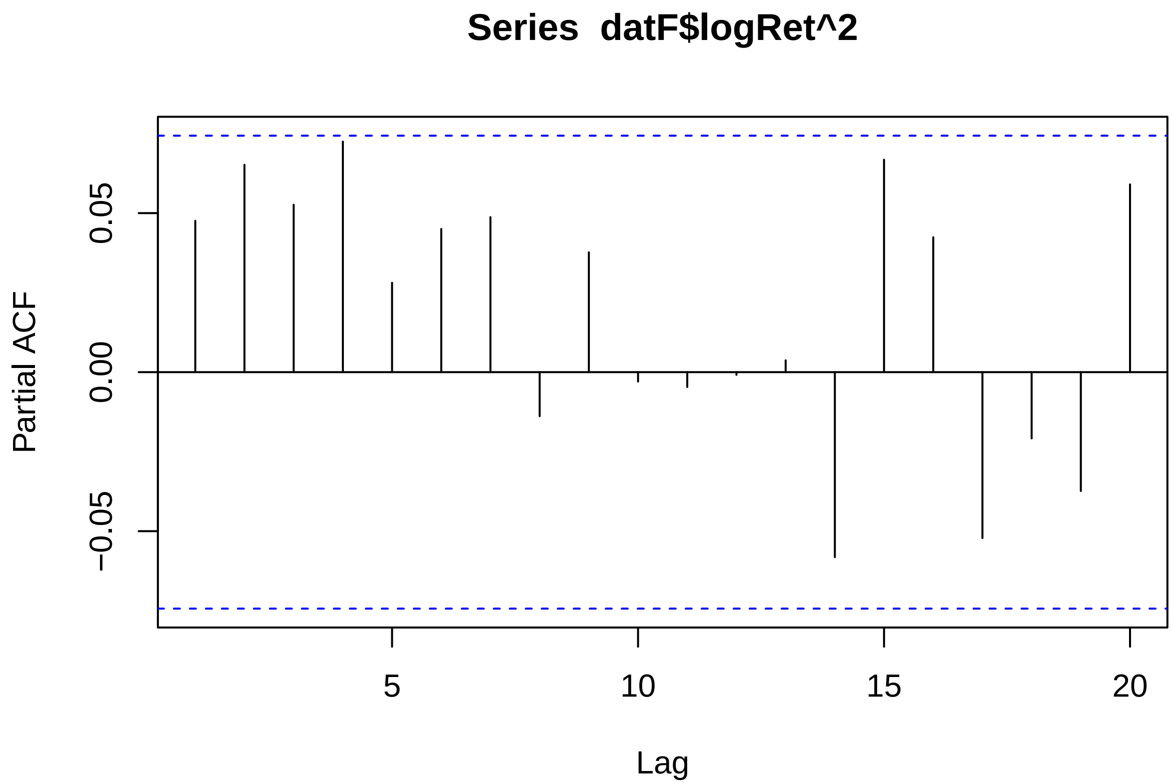
```
lags <- c(1:20)
sapply(lags, FUN = function(lag) {
                    test <- Box.test(logret.fit$residuals^2, lag = lag, type = c("Box-Pierce", "Ljung-B
                    test$p.value
})
```

```
##  [1] 0.0117694631 0.0003430990 0.0003504318 0.0002464178 0.0004326448
##  [6] 0.0007452725 0.0005524597 0.0010513383 0.0004738406 0.0006691216
## [11] 0.0012090605 0.0019856451 0.0028282093 0.0031847927 0.0010630363
## [16] 0.0013212098 0.0017427494 0.0026919060 0.0041035567 0.0019481185
```

P-values are less than 0.05 => means that shocks are not independent. There is ARCH effect in almost all the lags.

PACF of squared log returns

```
pacf(datF$logRet^2, lag.max = 20)
```



Seems and ARCH(2) model can be fitted

$r_t = \mu + a_t$

$a_t = \sigma_t \epsilon_t$

$\sigma_t^2 = \alpha_0 + \alpha_1 a_{t-1}^2 + \alpha_2 a_{t-2}^2$

Fit the Model

```
library(fGarch)
garch2.fit <- garchFit(~garch(2,0), data = datF$logRet, trace = FALSE)
garch2.fit
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~garch(2, 0), data = datF$logRet, trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ garch(2, 0)
## <environment: 0x7fdf8c991580>
##  [data = datF$logRet]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##         mu       omega      alpha1      alpha2
## -0.0169129   0.0035752   0.1125097   0.0985089
```

```
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu     -0.0169129   0.0024141   -7.006 2.46e-12 ***
## omega   0.0035752   0.0003063   11.672  < 2e-16 ***
## alpha1  0.1125097   0.0533135    2.110   0.0348 *
## alpha2  0.0985089   0.0511087    1.927   0.0539 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  898.8208    normalized:  1.293267
##
## Description:
##  Tue Sep 11 23:47:21 2018 by user:
```

$\mu = -0.0169129$

$\alpha_0 = \omega = 0.0035752$

$\alpha_1 = 0.1125097$

$\alpha_2 = 0.0985089$

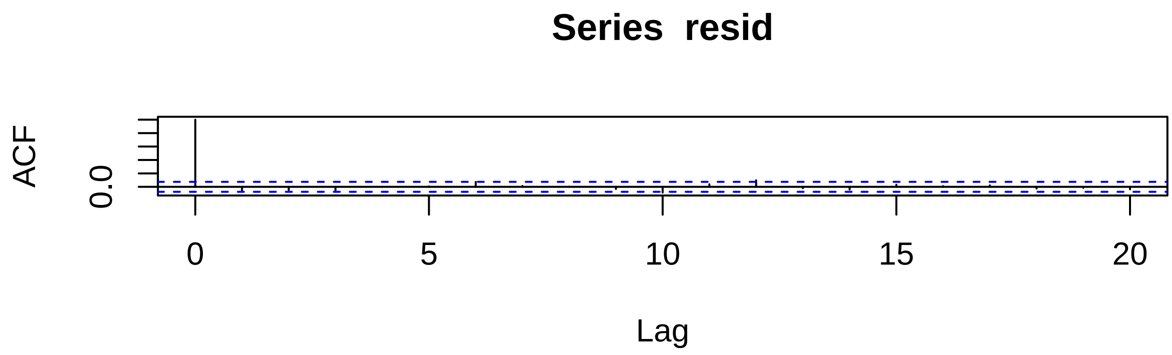
So the Model is:

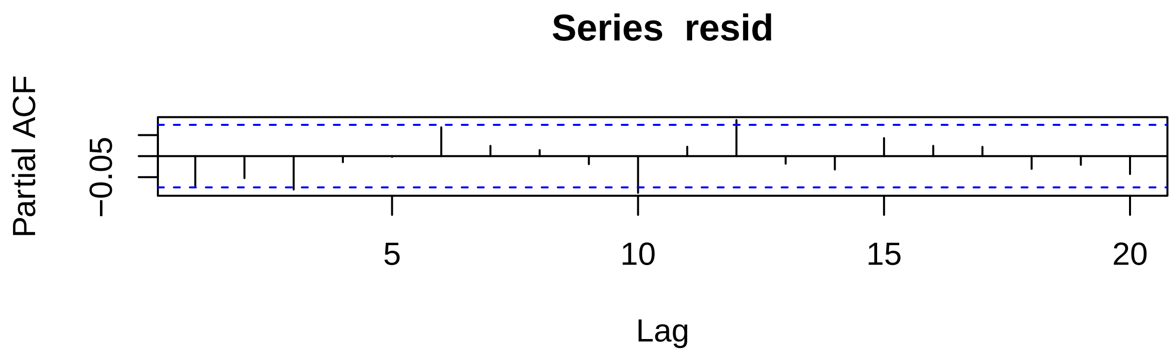
$r_t = -0.0169129 + a_t$

$\sigma_t^2 = 0.0035752 + 0.1125097a_{t-1}^2 + 0.0985089a_{t-2}^2$

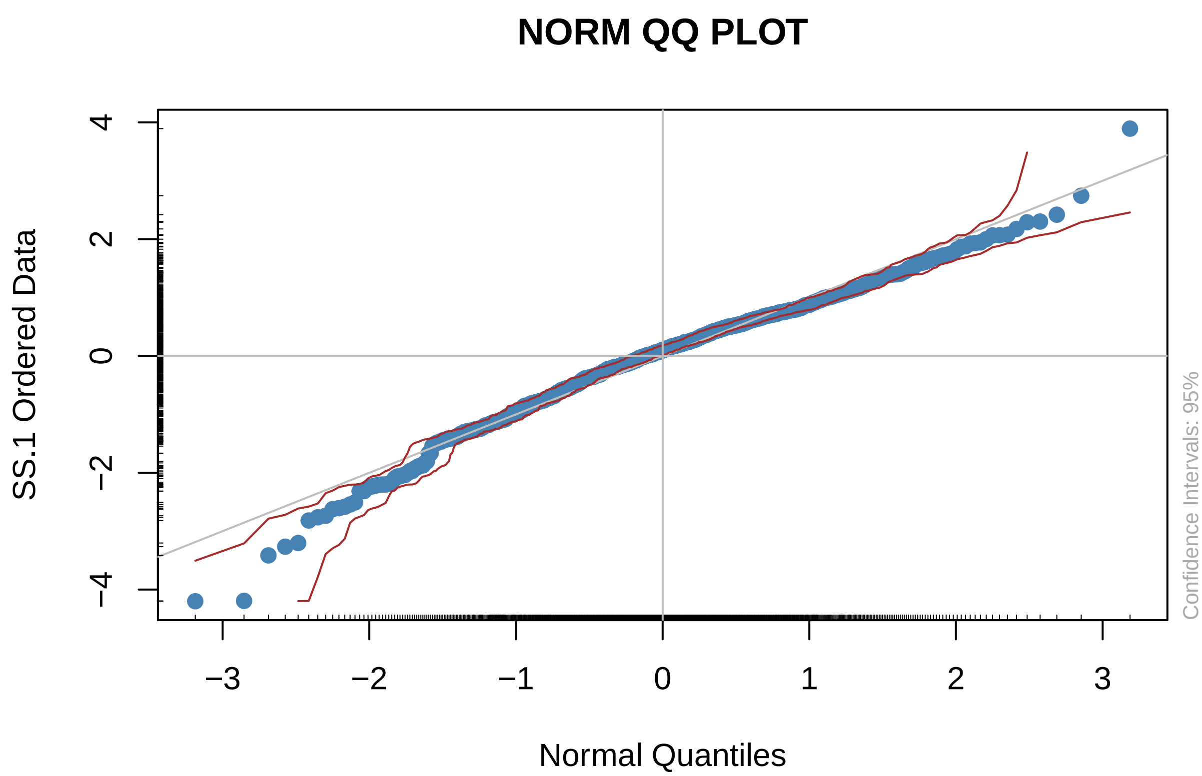
$a_t = \sigma_t \epsilon_t$

```r
resid <- residuals(garch2.fit)
par(mfrow=c(2,1))
acf(resid, lag.max = 20)
pacf(resid, lag.max = 20)
```



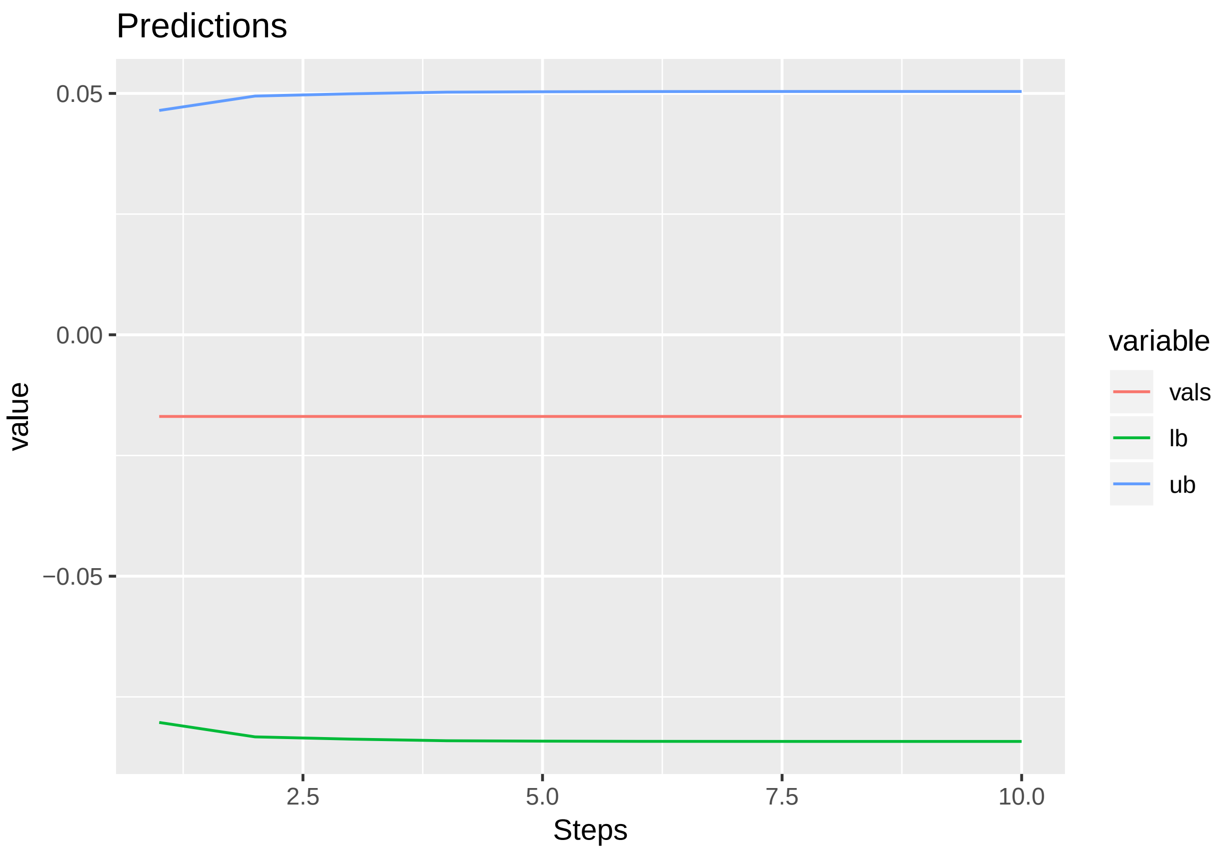



```
qqnormPlot(resid)
```

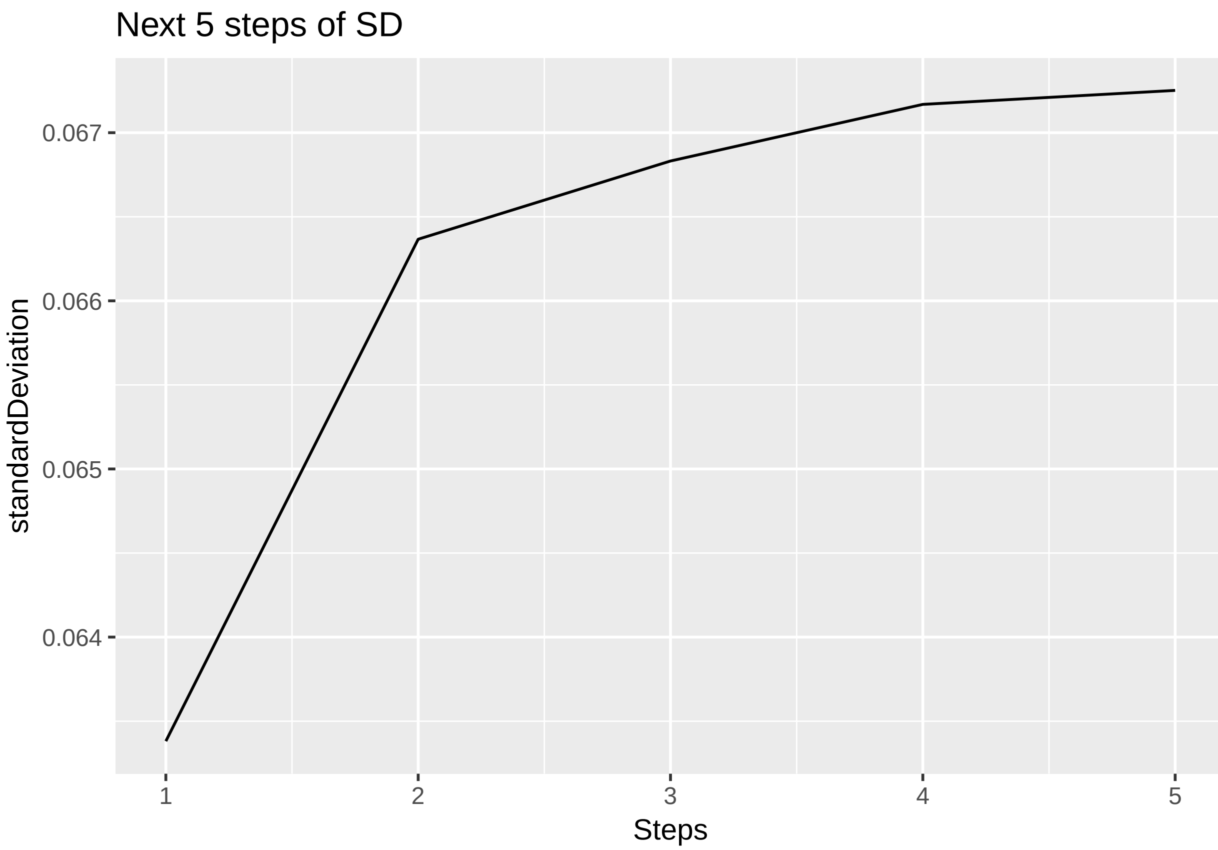


Prediction

```
#predic.val <- predict(garch2.fit, 5)
predVal = fGarch::predict(garch2.fit, 10)

temp.dat <- data.frame("steps" = c(1:dim(predVal)[1]), vals = predVal$meanForecast,
                       lb = predVal$meanForecast - predVal$standardDeviation,
                       ub = predVal$meanForecast + predVal$standardDeviation)
temp.dat.melt = melt(temp.dat, id.vars = "steps")

ggplot(data=temp.dat.melt, aes(x=steps, y=value, group=variable, colour=variable)) +
    geom_line()+
   ggtitle("Predictions")+
  xlab("Steps")
```




```
#Predict Next 5 steps of SD
ggplot(data=predVal[1:5,], aes(x =  c(1:dim(predVal)[1])[1:5], y=standardDeviation)) +
    geom_line() +
    ggtitle("Next 5 steps of SD") +
    xlab("Steps")
```



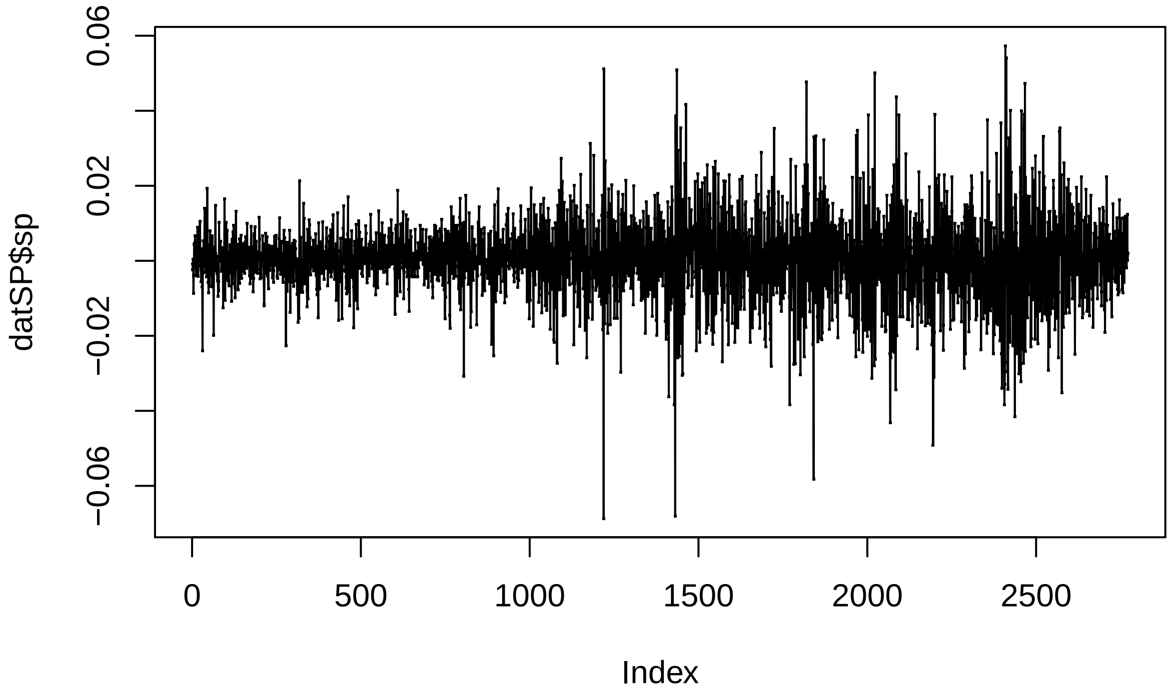
Bulding ARCH/GARCH model on S&P composite index

```
ret.what  <- list(date=numeric(), gm=numeric(), sp=numeric())
ret.widths <- c(12, 25-13, 34-25)
strip.white <- c(TRUE, TRUE, TRUE)

datSP = scan(file = "./data/d-gmsp9303.txt", what = ret.what, strip.white = strip.white)
datSP = as.data.frame(datSP)

head(datSP)
```

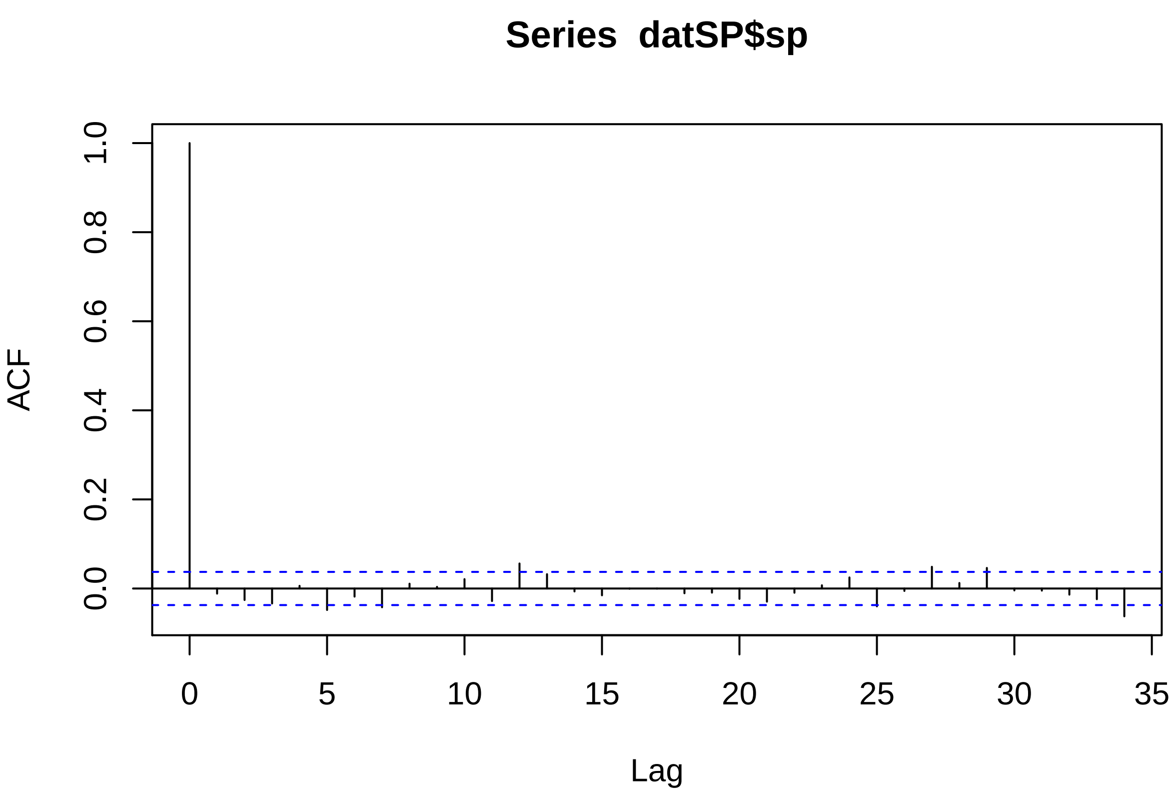
```
##       date       gm        sp
## 1 19930104  0.01938 -0.000757
## 2 19930105  0.01141 -0.002389
## 3 19930106  0.02256  0.000414
## 4 19930107 -0.01838 -0.008722
## 5 19930108  0.00000 -0.003900
## 6 19930111  0.02622  0.004428
```

Graphically look for ARCH effect

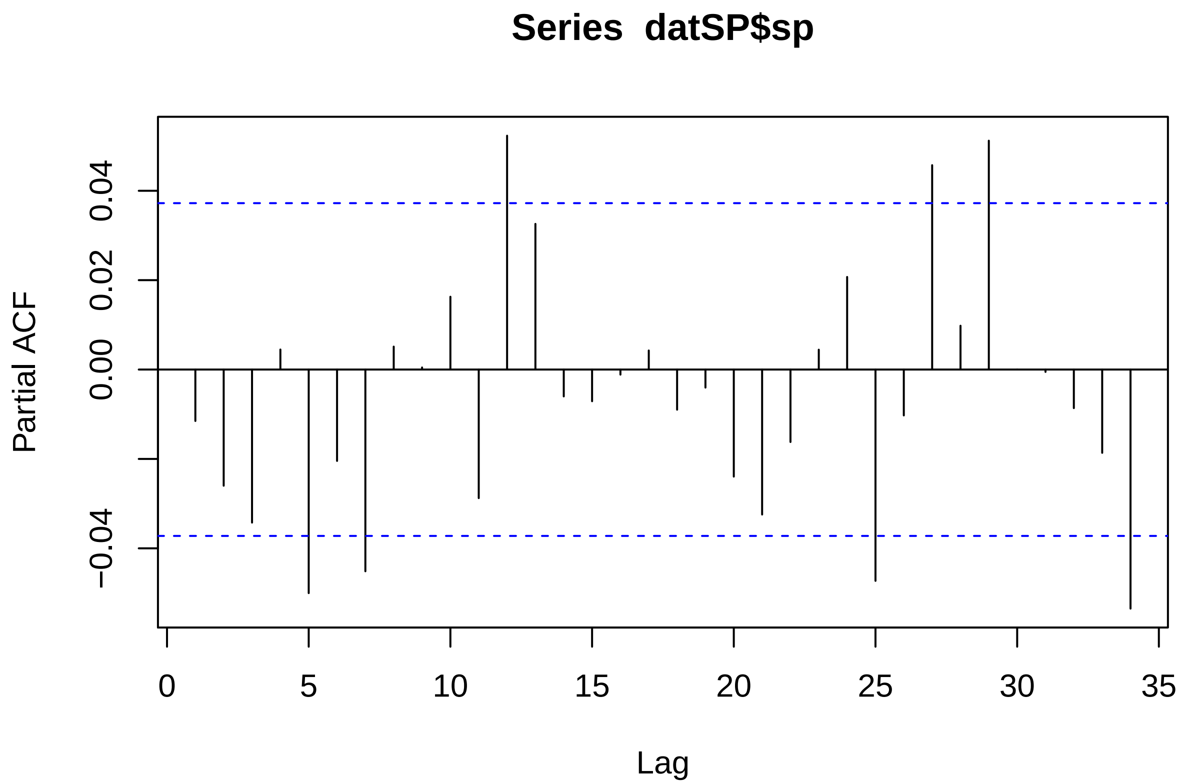
```
plot(datSP$sp, type="o", pch=".")
```



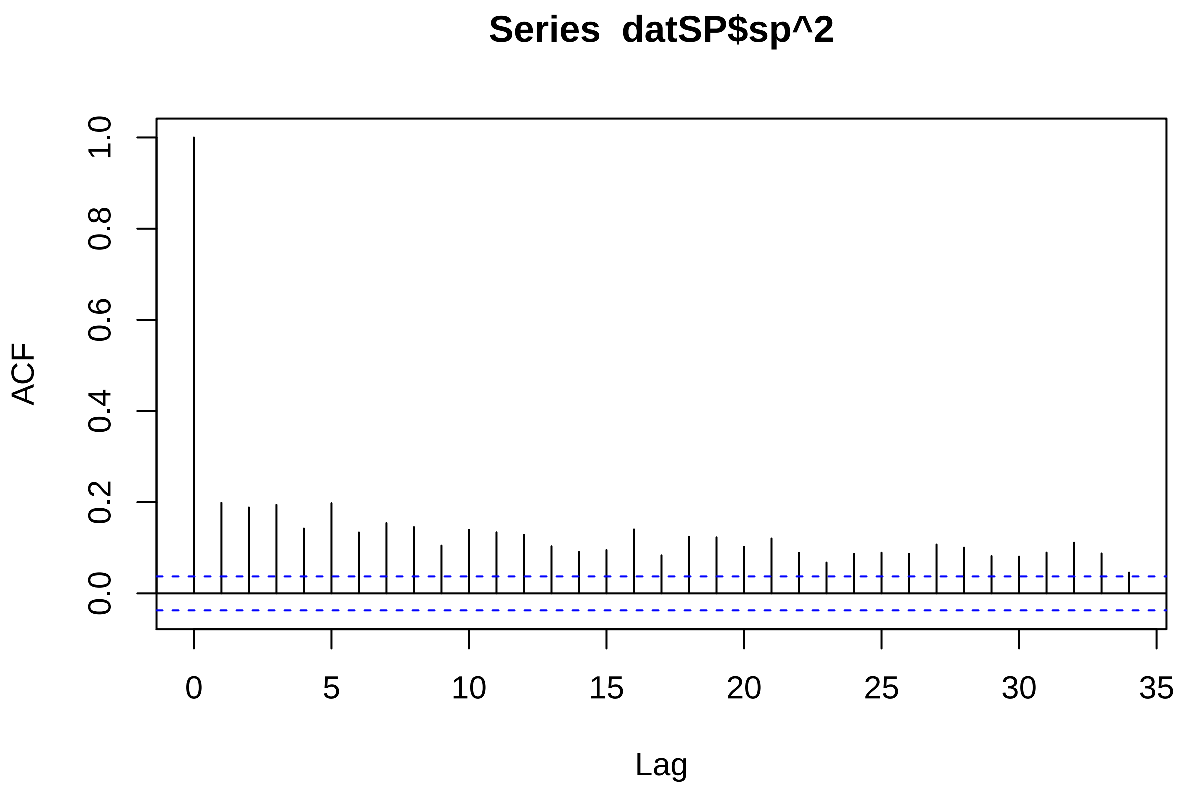

```
acf(datSP$sp)
```



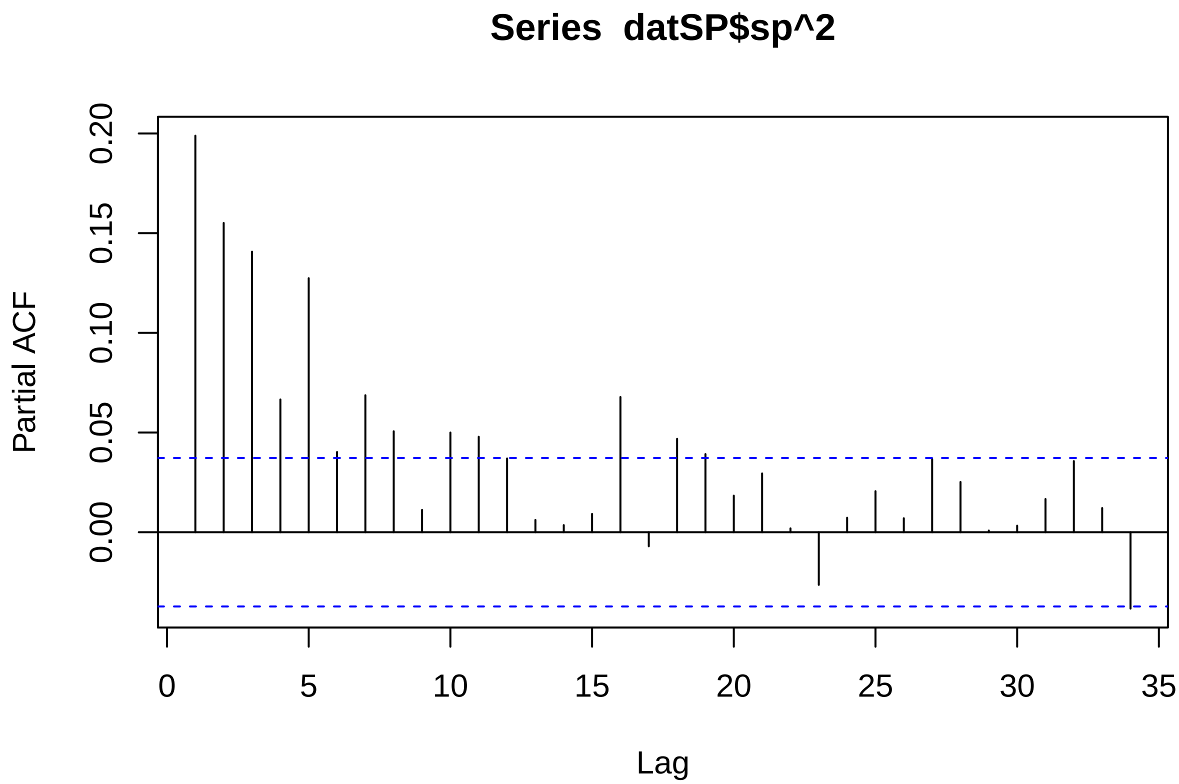

```
pacf(datSP$sp)
```




```r
acf(datSP$sp^2)
```




```r
pacf(datSP$sp^2)
```



The ACF plot shows almost no serial correlations but the squared series show significant autocorrelations.

Let us do auto correlations tests on few lags.

```
lags <- c(1:20)
sapply(lags, FUN = function(lag) {
                    test <- Box.test(datSP$sp^2, lag = lag, type = "Ljung-Box", fitdf = 0)
                    test$p.value
})
```

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

The p-values are almost 0, this means we can reject the Null hypothesis that series are uncorrelated or independent.

Since serial correlations are present we need to fit the mean equation first and then apply ARCH test of residues.

```
#ARMA
arima6.6.fit = arima(x = datSP$sp, order = c(6,0,6))
arima6.6.fit
```

```
##
## Call:
## arima(x = datSP$sp, order = c(6, 0, 6))
##
## Coefficients:
```

```
## Warning in sqrt(diag(x$var.coef)): NaNs produced
```

```
##          ar1     ar2      ar3     ar4      ar5      ar6      ma1     ma2
##       0.3976  0.4411  -0.7451  0.5780  -0.0107  -0.6592  -0.4127  -0.459
## s.e.     NaN     NaN   0.1770  0.1557      NaN      NaN      NaN     NaN
```

```
##          ma3      ma4      ma5     ma6  intercept
##       0.7362  -0.5566  -0.0285  0.6448      4e-04
## s.e.  0.2148   0.1533   0.1008     NaN      2e-04
##
## sigma^2 estimated as 0.0001197:  log likelihood = 8583.27,  aic = -17138.54
```

```
arima5.5.fit = arima(x = datSP$sp, order = c(5,0,5))
arima5.5.fit
```

```
##
## Call:
## arima(x = datSP$sp, order = c(5, 0, 5))
##
## Coefficients:
##          ar1     ar2      ar3      ar4      ar5      ma1      ma2     ma3
##       0.6955  0.6782  -0.6275  -0.1178  -0.0894  -0.7125  -0.6990  0.6303
## s.e.  0.4136  0.8117   0.6045   0.6350   0.6036   0.4146   0.8152  0.6185
##          ma4     ma5  intercept
##       0.1591  0.0403      4e-04
## s.e.  0.6197  0.5935      2e-04
##
## sigma^2 estimated as 0.0001198:  log likelihood = 8581.18,  aic = -17138.35
```

```
#aic = -17138.35

arima4.4.fit = arima(x = datSP$sp, order = c(4,0,4))
arima4.4.fit
```

```
##
## Call:
## arima(x = datSP$sp, order = c(4, 0, 4))
##
## Coefficients:

## Warning in sqrt(diag(x$var.coef)): NaNs produced

##           ar1     ar2     ar3     ar4     ma1      ma2      ma3     ma4
##       -0.1362  0.2585  0.0087  0.3602  0.1160  -0.2906  -0.0536  -0.347
## s.e.   0.2767     NaN  0.1610     NaN  0.2735      NaN   0.1673     NaN
##       intercept
##           4e-04
## s.e.      2e-04
##
## sigma^2 estimated as 0.0001206:  log likelihood = 8572.39,  aic = -17124.79
```

```
#Fails with Nan

arima2.2.fit = arima(x = datSP$sp, order = c(2,0,2))
arima2.2.fit
```

```
##
## Call:
## arima(x = datSP$sp, order = c(2, 0, 2))
##
## Coefficients:
##          ar1      ar2      ma1      ma2  intercept
##       0.7499  -0.0148  -0.7646  -0.0114      4e-04
## s.e.  0.8820   0.7469   0.8735   0.7599      2e-04
```

```
##
## sigma^2 estimated as 0.0001208:  log likelihood = 8569.87,  aic = -17127.73
```

```r
#aic = -17127.73

ar6.fit = arima(x = datSP$sp, order = c(6,0,0))
ar6.fit
```

```
##
## Call:
## arima(x = datSP$sp, order = c(6, 0, 0))
##
## Coefficients:
##           ar1      ar2      ar3     ar4      ar5      ar6  intercept
##       -0.0133  -0.0279  -0.0362  0.0033  -0.0502  -0.0204      4e-04
## s.e.   0.0190   0.0190   0.0190  0.0190   0.0190   0.0190      2e-04
##
## sigma^2 estimated as 0.0001206:  log likelihood = 8572.24,  aic = -17128.48
```

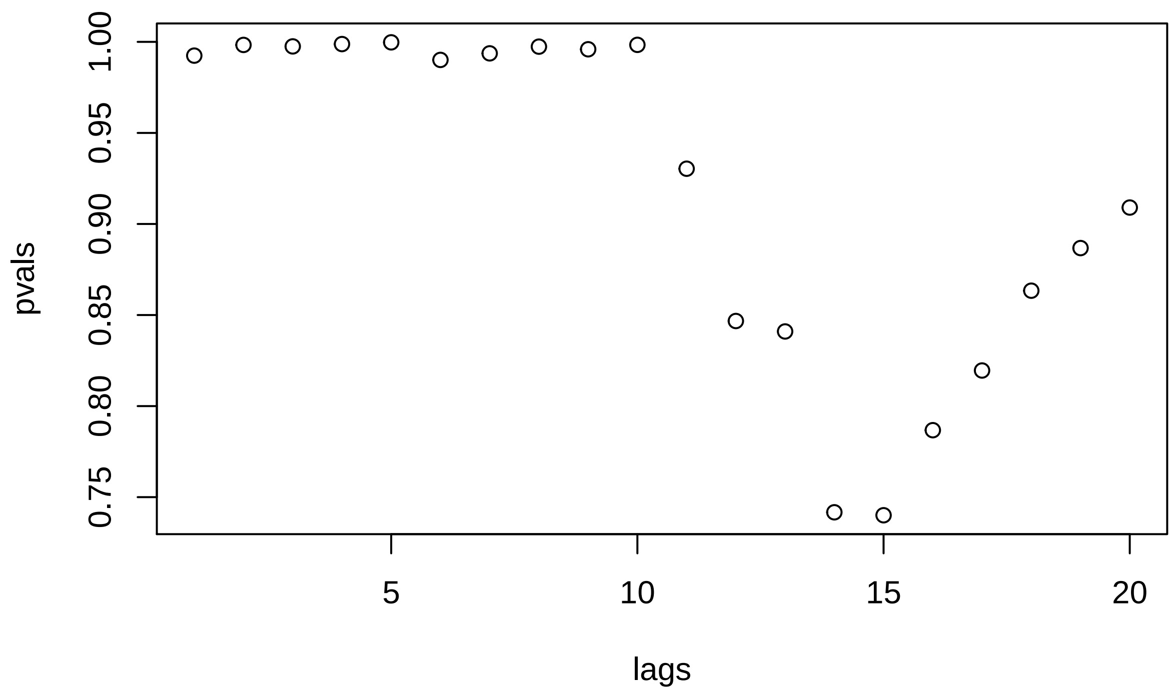
```r
#aic = -17128.48

ar5.fit = arima(x = datSP$sp, order = c(5,0,0))
ar5.fit
```

```
##
## Call:
## arima(x = datSP$sp, order = c(5, 0, 0))
##
## Coefficients:
##           ar1     ar2      ar3     ar4     ar5  intercept
##       -0.0123  -0.028  -0.0355  0.0039  -0.050      4e-04
## s.e.   0.0190   0.019   0.0190  0.0190   0.019      2e-04
##
## sigma^2 estimated as 0.0001207:  log likelihood = 8571.66,  aic = -17129.32
```

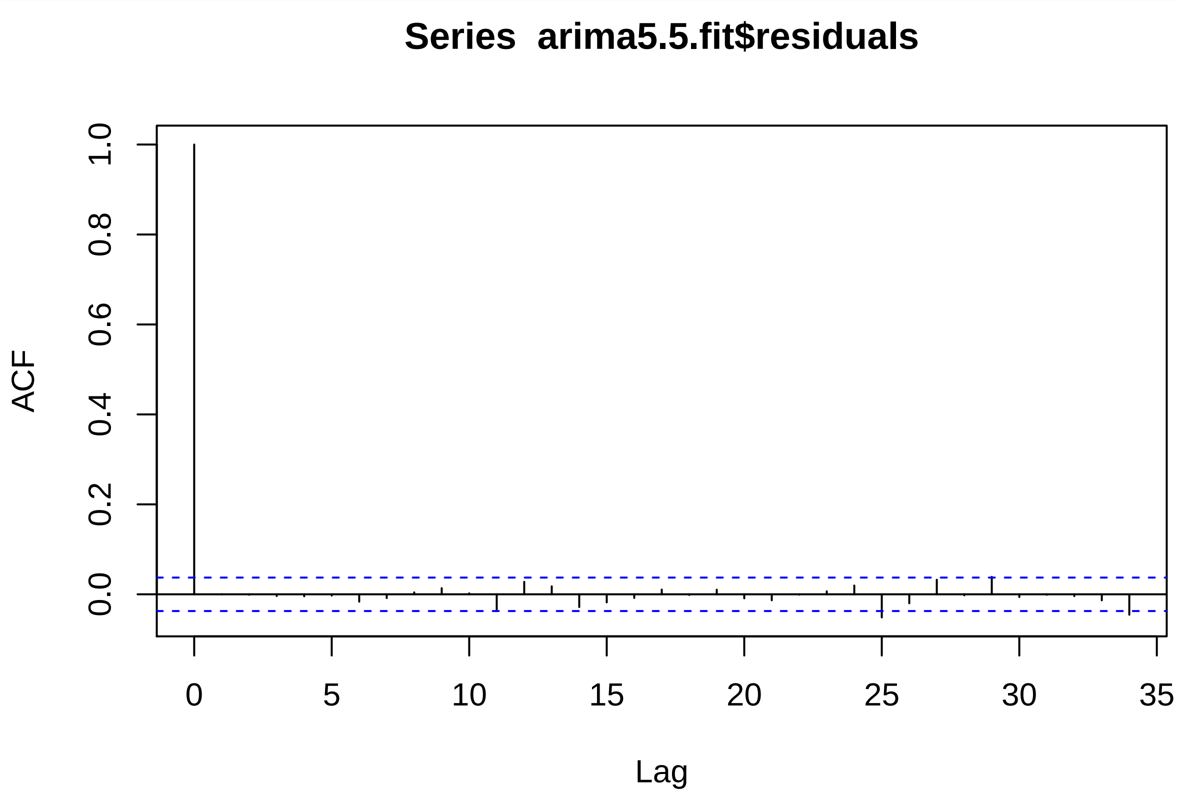
```r
#aic = -17129.32
```

We pick ARMA(5,5) model for lowest AIC values

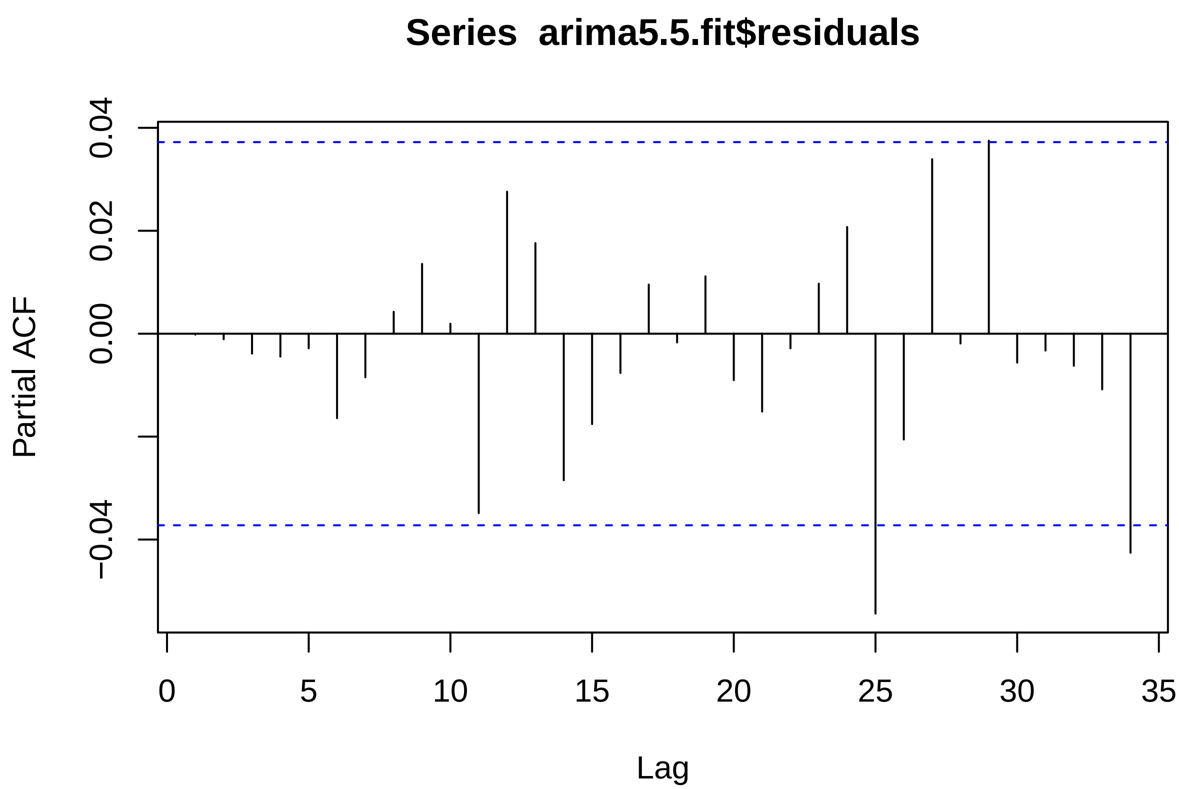
```r
arima5.5.fit = arima(x = datSP$sp, order = c(5,0,5))

lags <- c(1:20)
pvals <- sapply(lags, FUN = function(lag) {
                    test <- Box.test(arima5.5.fit$residuals, lag = lag, type =  "Ljung-Box", fitdf = 0)
                    test$p.value
})
plot(x=lags, y = pvals)
```




```r
#ACF/PACF
acf(arima5.5.fit$residuals)
```



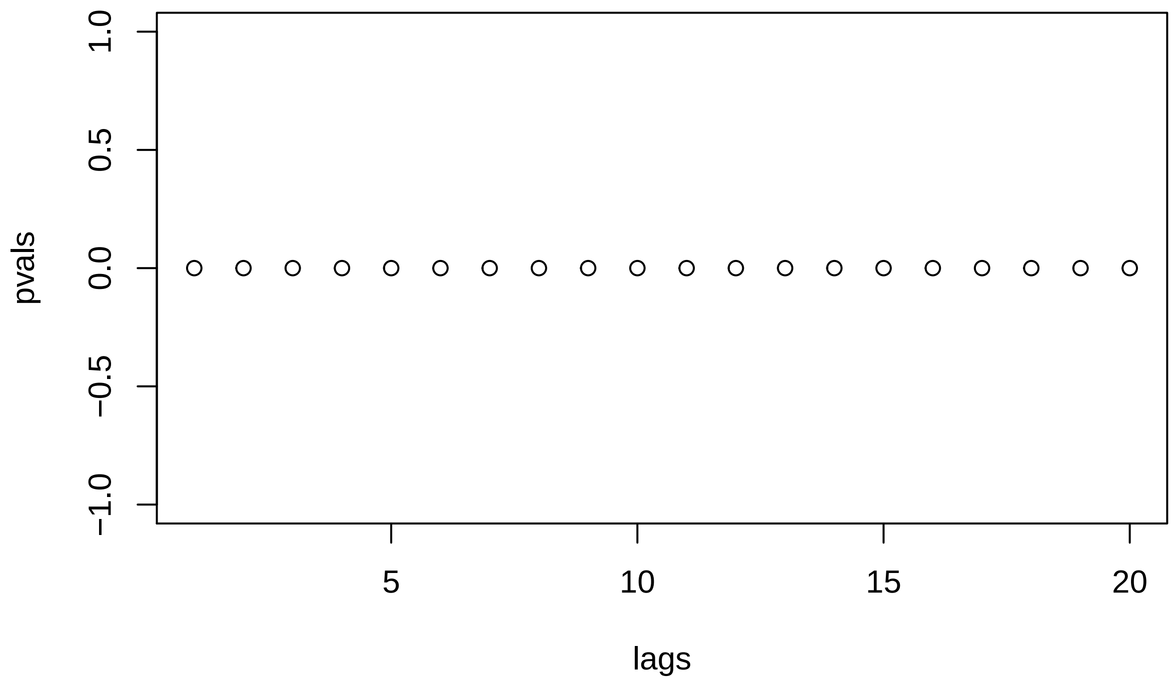

```r
pacf(arima5.5.fit$residuals)
```



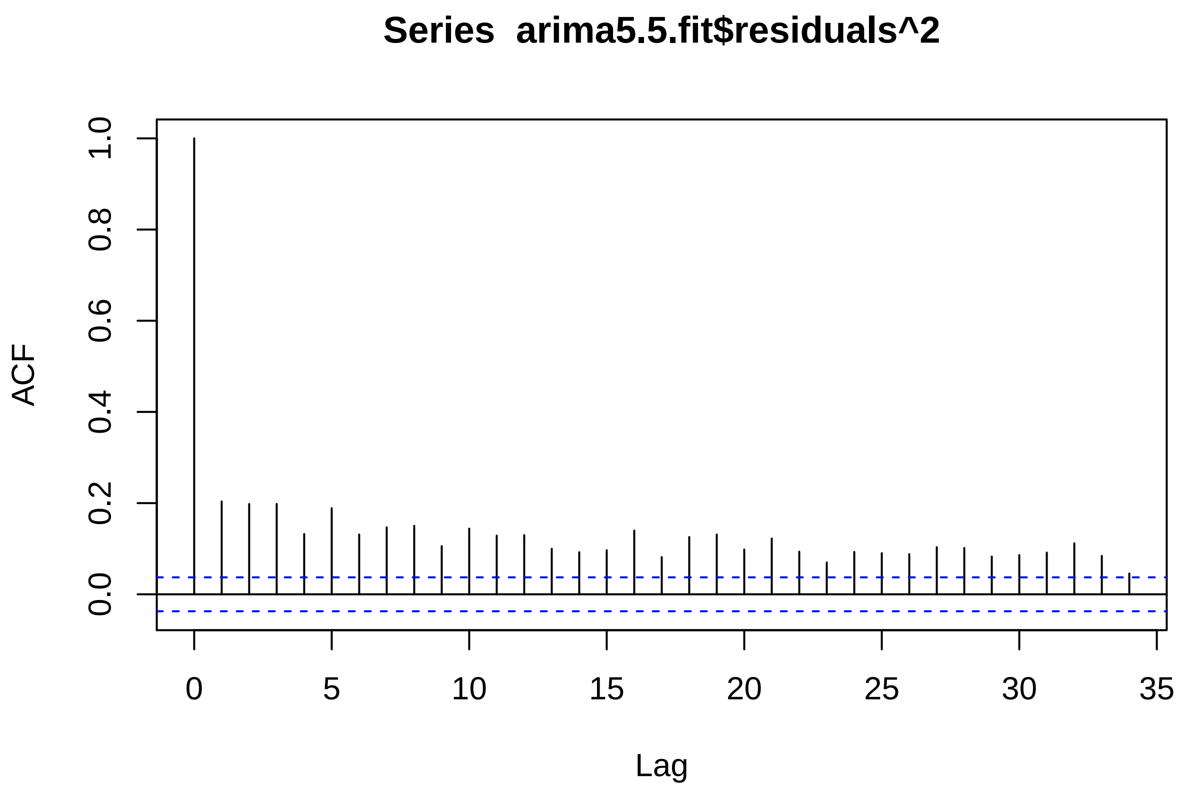
So the residuals are not autocorrelated.

ARCH test on residual squared:

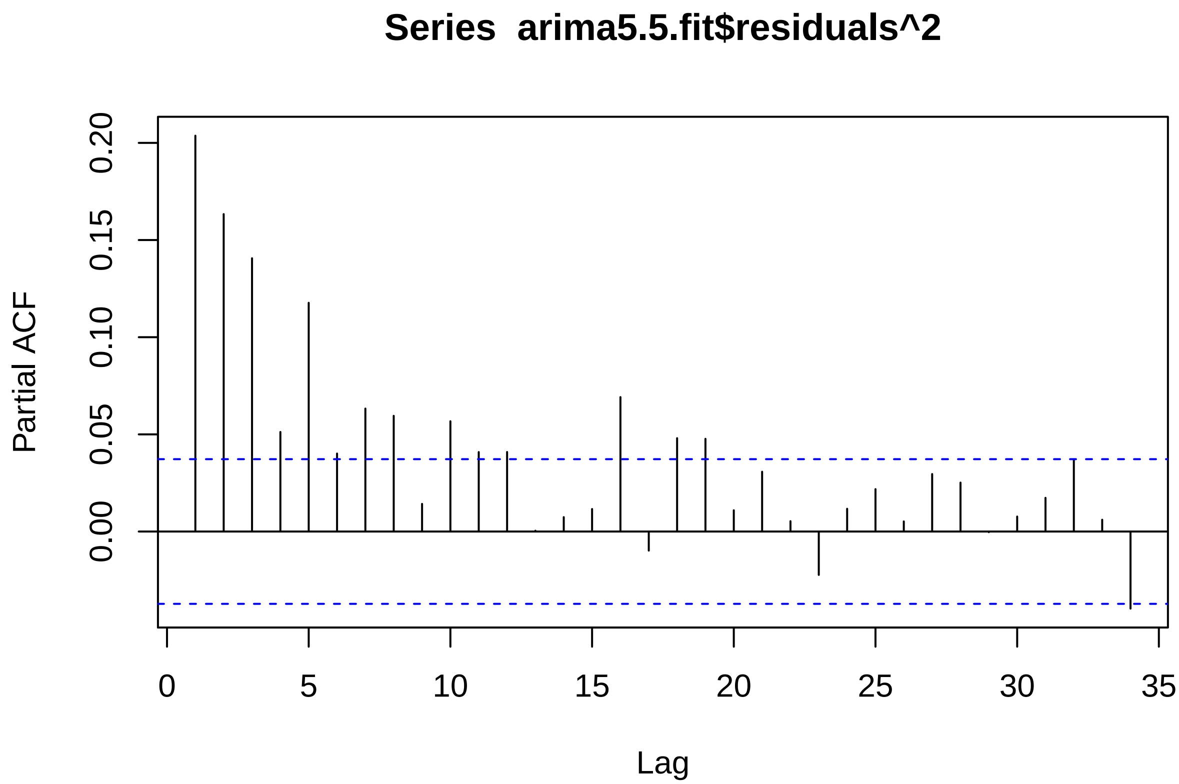
```
lags <- c(1:20)
pvals <- sapply(lags, FUN = function(lag) {
                    test <- Box.test(arima5.5.fit$residuals^2, lag = lag, type =  "Ljung-Box", fitdf = (
                    test$p.value
})
plot(x=lags, y = pvals)
```




```
#ACF/PACF
acf(arima5.5.fit$residuals^2)
```




```
pacf(arima5.5.fit$residuals^2)
```



So there is autocorrelation in residues.

To build a garch model , we do a joint max likelohood estimation with AR(5). Not that AARMA(5,5) does not converge/gives NaNs

```
#garch55.fit <- garchFit(~garch(5,5), data = datSP$sp, trace = FALSE)
#garch55.fit

library(rugarch)
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'rugarch'
```

```
## The following object is masked from 'package:stats':
##
##     sigma
```

```
spec <- ugarchspec(variance.model = list(model = "sGARCH",
                                         garchOrder = c(1, 1),
                                         submodel = NULL,
                                         external.regressors = NULL,
                                         variance.targeting = FALSE),

                   mean.model     = list(armaOrder = c(5, 0),
                                         external.regressors = NULL,
                                         distribution.model = "norm",
                                         start.pars = list(),
                                         fixed.pars = list()))
```

```
## Warning: unidentified option(s) in mean.model:
```

```
##  distribution.model start.pars fixed.pars
```

```
garch.ar5.fit <- ugarchfit(spec = spec, data = datSP$sp, solver.control = list(trace=0))

#garch.ar5.fit <- garchFit(formula = ~garch(2,1), data = datSP$sp, trace = FALSE)
garch.ar5.fit
```

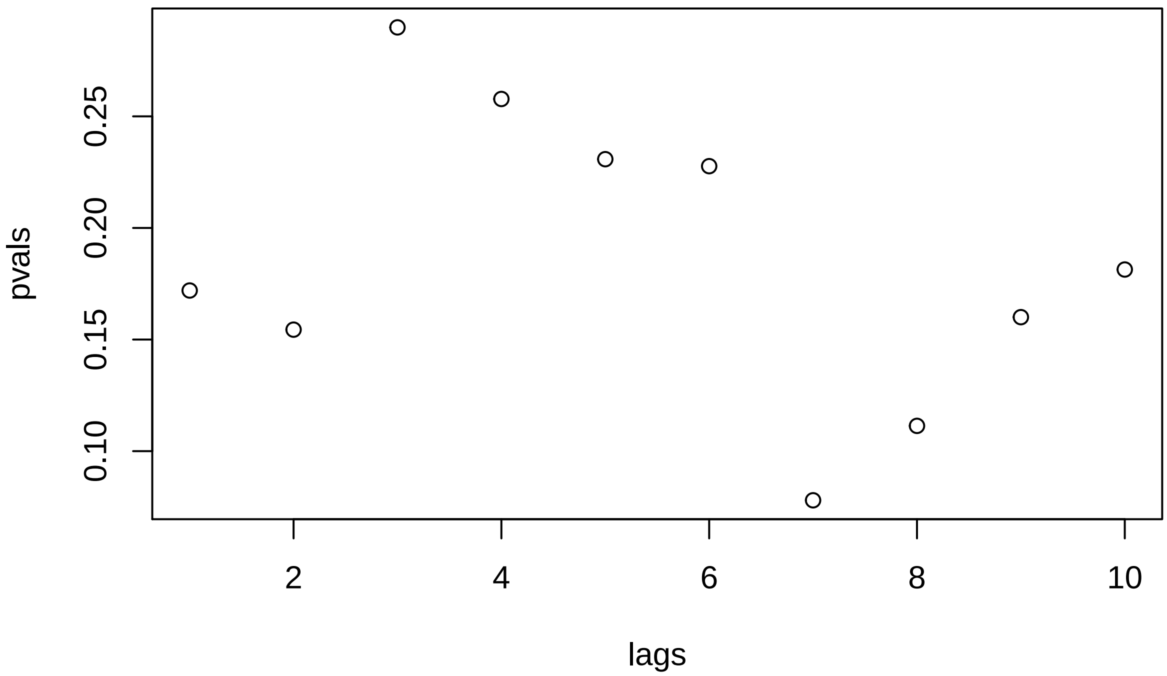
```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(5,0,0)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000640    0.000134  4.78344 0.000002
## ar1     0.013345    0.019855  0.67214 0.501493
## ar2    -0.005858    0.019653 -0.29807 0.765648
## ar3    -0.039025    0.019652 -1.98588 0.047047
## ar4    -0.020464    0.019767 -1.03524 0.300556
## ar5    -0.072067    0.019518 -3.69233 0.000222
## omega   0.000001    0.000001  0.89343 0.371630
## alpha1  0.069731    0.010754  6.48415 0.000000
## beta1   0.928146    0.010498 88.41024 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.000640    0.000190  3.36860 0.000756
## ar1     0.013345    0.021742  0.61381 0.539339
## ar2    -0.005858    0.021299 -0.27504 0.783288
## ar3    -0.039025    0.019726 -1.97836 0.047888
## ar4    -0.020464    0.020303 -1.00795 0.313477
## ar5    -0.072067    0.018984 -3.79612 0.000147
## omega   0.000001    0.000004  0.13064 0.896060
## alpha1  0.069731    0.094012  0.74173 0.458253
## beta1   0.928146    0.088974 10.43170 0.000000
##
## LogLikelihood : 8977.043
##
## Information Criteria
## ------------------------------------
##
## Akaike       -6.4704
## Bayes        -6.4512
## Shibata      -6.4705
## Hannan-Quinn -6.4635
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
```

```
##                          statistic p-value
## Lag[1]                       0.147  0.7014
## Lag[2*(p+q)+(p+q)-1][14]     5.249  1.0000
## Lag[4*(p+q)+(p+q)-1][24]    11.923  0.5598
## d.o.f=5
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                    0.02539  0.8734
## Lag[2*(p+q)+(p+q)-1][5]   3.10970  0.3874
## Lag[4*(p+q)+(p+q)-1][9]   3.64911  0.6488
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]   0.07269 0.500 2.000  0.7875
## ARCH Lag[5]   0.17421 1.440 1.667  0.9715
## ARCH Lag[7]   0.26453 2.315 1.543  0.9945
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  457.439
## Individual Statistics:
## mu      0.18426
## ar1     0.67833
## ar2     0.06316
## ar3     0.12635
## ar4     0.32792
## ar5     0.02737
## omega  97.02699
## alpha1  0.29524
## beta1   0.23932
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          2.1 2.32 2.82
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value      prob sig
## Sign Bias            1.970 4.892e-02  **
## Negative Sign Bias   0.555 5.789e-01
## Positive Sign Bias   2.292 2.200e-02  **
## Joint Effect        29.492 1.765e-06 ***
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     55.85    1.721e-05
## 2    30     67.59    6.434e-05
```

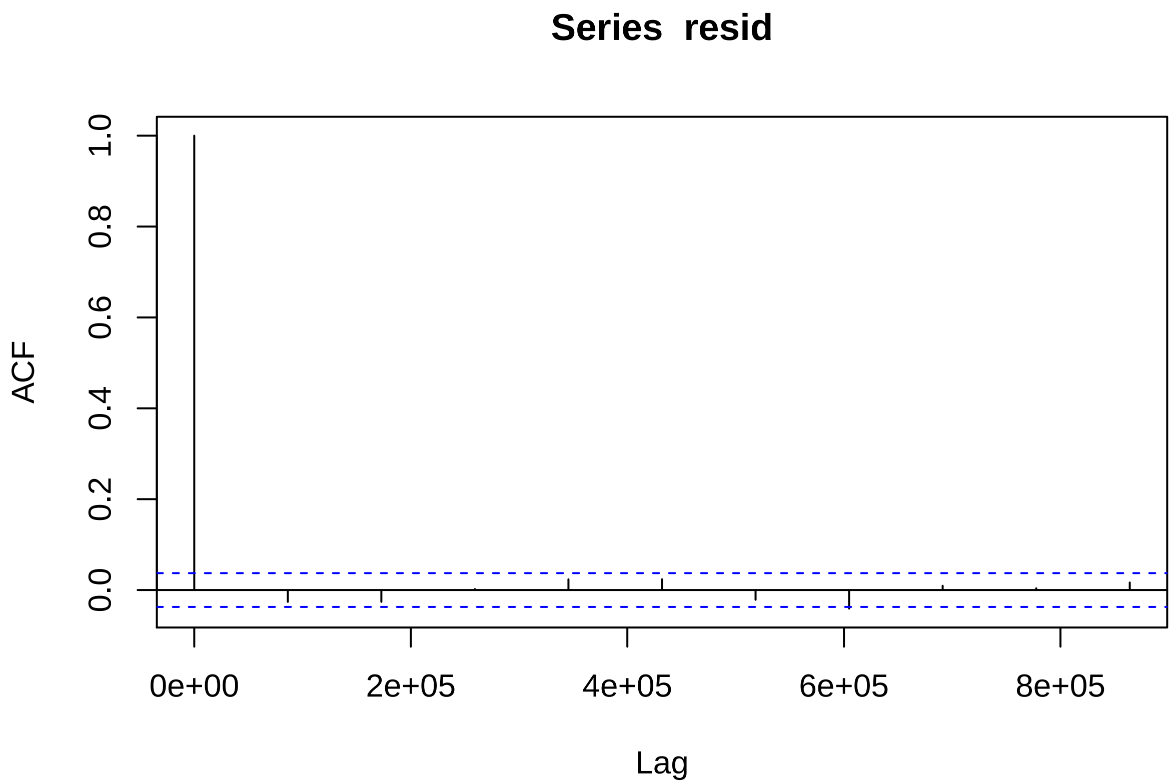
```
## 3    40      90.25    6.099e-06
## 4    50      96.51    6.053e-05
##
##
## Elapsed time : 0.501334
```

All coefficients are significantly different from 0

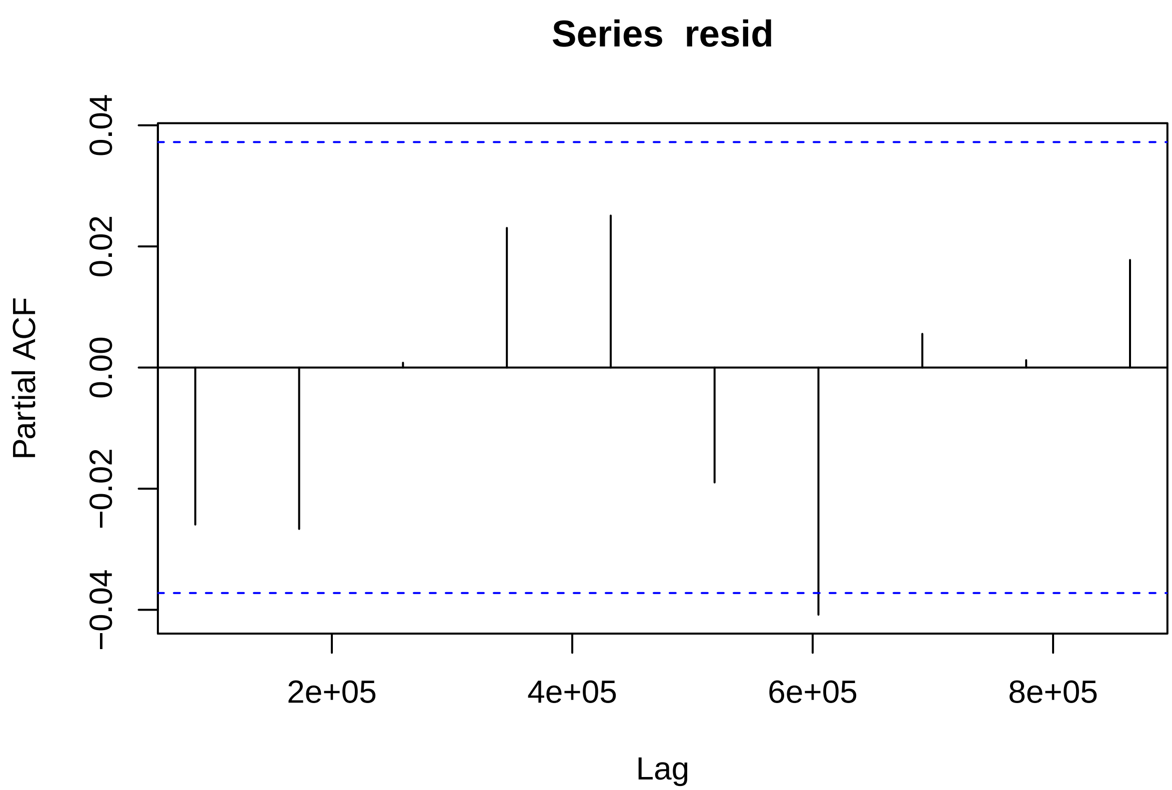
```
resid <- residuals(garch.ar5.fit)
lags <- c(1:10)
pvals <- sapply(lags, FUN = function(lag) {
                    test <- Box.test(resid, lag = lag, type =  "Ljung-Box", fitdf = 0)
                    test$p.value
})
plot(x=lags, y = pvals)
abline(h = 0.05)
```




```
#ACF/PACF
acf(resid, lag.max = 10)
```




```
pacf(resid, lag.max = 10)
```




```
predVal = ugarchforecast(garch.ar5.fit,n.ahead=10,data=datSP$sp)
```

```r
#predVal@forecast$seriesFor
#predVal@forecast$sigmaFor

oldPlusForeCast <- c(datSP$sp, predVal@forecast$seriesFor)
ub              <- c(datSP$sp, predVal@forecast$seriesFor + predVal@forecast$sigmaFor)
lb              <- c(datSP$sp, predVal@forecast$seriesFor - predVal@forecast$sigmaFor)
temp.dat <- data.frame(x = 1:length(oldPlusForeCast) , vals = oldPlusForeCast,  lb = lb, ub=ub)

#Last 30 + 10 projected means and theor bounds
temp.dat.melt = melt(tail(temp.dat,50), id.vars = "x")

ggplot(data=temp.dat.melt, aes(x=x, y=value, group=variable, colour=variable)) +
    geom_line()+
   ggtitle("Current + Pred")+
  xlab("x")
```