Untitled

2024-04-22

Extract data

Draw the Amazon stock data from 2023-01-01 to 2023-12-31 as train set

```r
library(quantmod)
```

```
## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

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

## Loading required package: TTR

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
getSymbols("AMZN",src='yahoo',return.class='ts', from = '2023-01-01', to = '2023-12-31')
```

```
## [1] "AMZN"
```

```r
train <- AMZN[,4]
length(train)
```

```
## [1] 250
```

Draw the Amazon stock data from 2024-01-01 to 2024-01-31 as test set

```r
getSymbols("AMZN",src='yahoo',return.class='ts', from = '2024-01-01', to = '2024-01-31')
```

```
## [1] "AMZN"
```

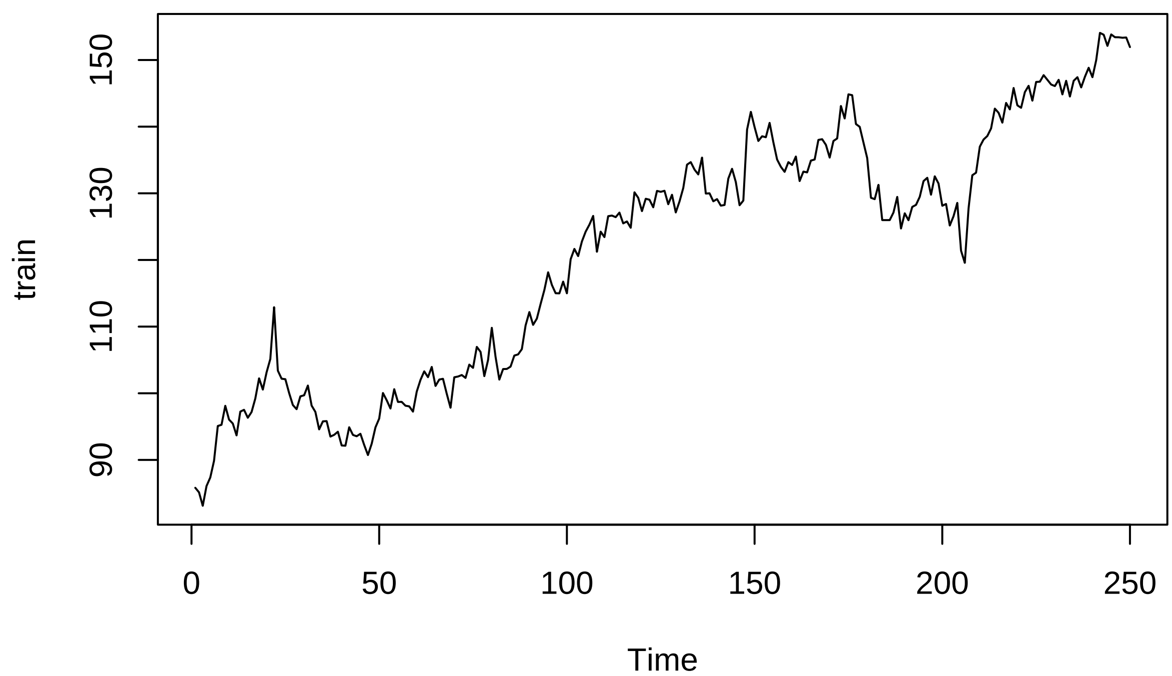
```r
test <- AMZN[,4]
length(test)
```

```
## [1] 20
```

EDA

we can first observe the time series plot of the data. And we can see that the scale(10^2) is large, and there is trend on the long term.

```r
ts.plot(train)
```



So we can reduce the scale by log transformation and fit a trend model for the data.

```r
train_set <- log(train)
test_set <- log(test)
tfit <- time(train_set)
```

Remove the Deterministic trend

linear trend model

```r
mlr.lin <- lm(train_set ~ tfit)

summary(mlr.lin)
```

```
##
## Call:
## lm(formula = train_set ~ tfit)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.168507 -0.044383 -0.005994  0.049328  0.153878
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.527e+00  7.544e-03   600.1   <2e-16 ***
## tfit        2.064e-03  5.211e-05    39.6   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05946 on 248 degrees of freedom
## Multiple R-squared:  0.8635, Adjusted R-squared:  0.8629
## F-statistic:  1568 on 1 and 248 DF,  p-value: < 2.2e-16
```

quadratic trend model

```r
tsqfit <- tfit^2/factorial(2)
mlr.quad <- lm(train_set ~ tfit + tsqfit)

summary(mlr.quad)
```

```
##
## Call:
## lm(formula = train_set ~ tfit + tsqfit)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.162385 -0.049980  0.009945  0.043060  0.180370
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.477e+00  1.055e-02 424.422  < 2e-16 ***
## tfit         3.272e-03  1.940e-04  16.861  < 2e-16 ***
## tsqfit      -9.627e-06  1.497e-06  -6.429 6.59e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.05515 on 247 degrees of freedom
## Multiple R-squared:  0.883,  Adjusted R-squared:  0.8821
## F-statistic: 932.3 on 2 and 247 DF,  p-value: < 2.2e-16
```

cubic trend model

```r
tcubfit <- tfit^3/factorial(3)
mlr.cub <- lm(train_set ~ tfit + tsqfit + tcubfit)

summary(mlr.cub)
```

```
##
## Call:
## lm(formula = train_set ~ tfit + tsqfit + tcubfit)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.16760 -0.04541  0.01161  0.04416  0.17727
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.494e+00  1.409e-02 319.025  < 2e-16 ***
## tfit         2.424e-03  4.851e-04   4.997  1.1e-06 ***
## tsqfit       7.221e-06  8.973e-06   0.805   0.4217
## tcubfit     -1.342e-07  7.050e-08  -1.904   0.0581 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05486 on 246 degrees of freedom
## Multiple R-squared:  0.8847, Adjusted R-squared:  0.8833
## F-statistic: 629.3 on 3 and 246 DF,  p-value: < 2.2e-16
```

quartic trend model

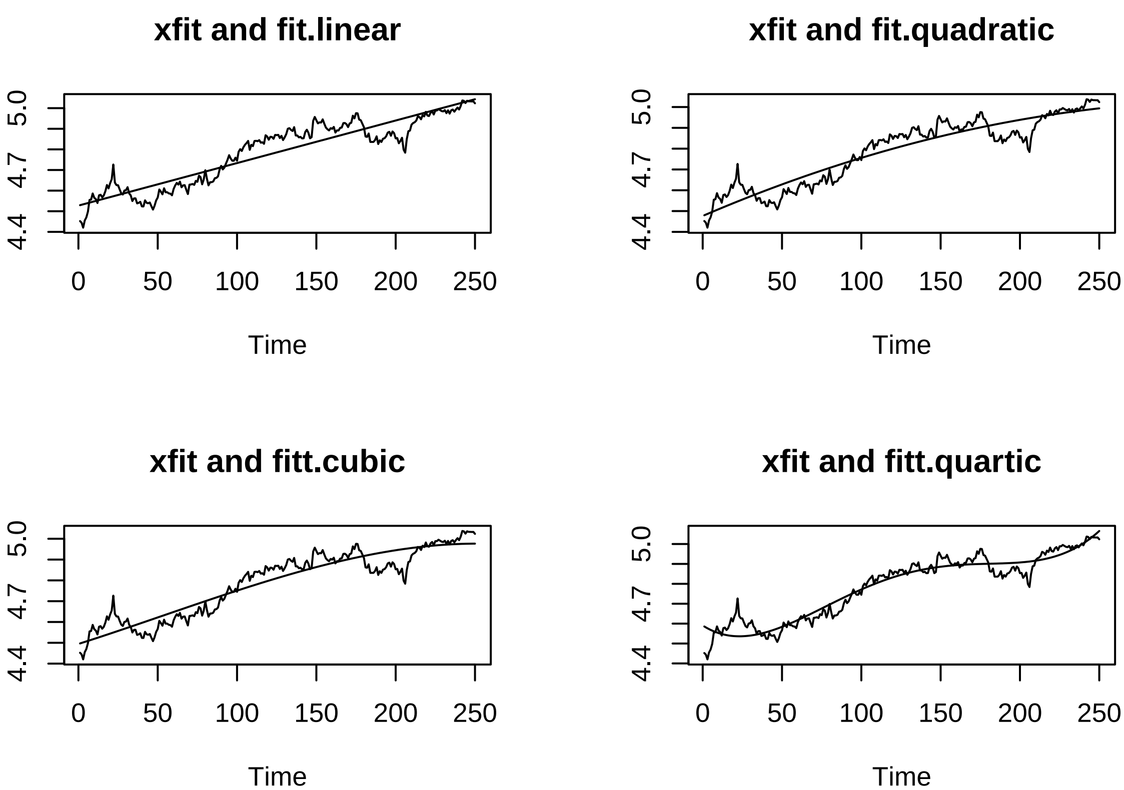
```r
tquarfit <- tfit^4/factorial(4)
mlr.quar <- lm(train_set ~ tfit + tsqfit + tcubfit + tquarfit)

summary(mlr.quar)
```

```
##
## Call:
## lm(formula = train_set ~ tfit + tsqfit + tcubfit + tquarfit)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.156319 -0.025418  0.003902  0.028243  0.190171
##
## Coefficients:
```

```
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.591e+00  1.468e-02 312.807  < 2e-16 ***
## tfit        -5.109e-03  8.067e-04  -6.332 1.14e-09 ***
## tsqfit       2.761e-04  2.607e-05  10.591  < 2e-16 ***
## tcubfit     -5.126e-06  4.677e-07 -10.961  < 2e-16 ***
## tquarfit     3.977e-08  3.697e-09  10.758  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0453 on 245 degrees of freedom
## Multiple R-squared:  0.9217, Adjusted R-squared:  0.9204
## F-statistic: 721.1 on 4 and 245 DF,  p-value: < 2.2e-16
```

trend model selection

```r
par(mfrow=c(2,2))
plin=cbind(train_set,mlr.lin$fitted)
ts.plot(plin,main="xfit and fit.linear")
pquad=cbind(train_set,mlr.quad$fitted)
ts.plot(pquad,main="xfit and fit.quadratic")
pcub=cbind(train_set,mlr.cub$fitted)
ts.plot(pcub,main="xfit and fitt.cubic")
pquar=cbind(train_set,mlr.quar$fitted)
ts.plot(pquar,main="xfit and fitt.quartic")
```



in-sample metric

we can compare the trend models by AIC

```
nfit <- length(train_set)
```

```
AIC.lin <-  AIC(mlr.lin)/nfit
AIC.quad <- AIC(mlr.quad)/nfit
AIC.cub <- AIC(mlr.cub)/nfit
AIC.quar <- AIC(mlr.quar)/nfit
data.frame(
  model = c("lin", "quad", "cub", "quar"),
  AIC = c(AIC.lin, AIC.quad, AIC.cub, AIC.quar)
)
```

```
##   model       AIC
## 1   lin -2.791029
## 2  quad -2.937741
## 3   cub -2.944371
## 4  quar -3.323230
```

By AIC, we can see that quartic trend model has the best effect on the combination of fitting and complexity.

out of sample metric

we can calculate the MAPE of each model.

```
new <- data.frame(tfit=c(378:397))
pfore.lin <- predict(mlr.lin,new,se.fit = TRUE)
efore.lin <- test_set - pfore.lin$fit
```

```
tfit <- c(378:397)
tsqfit <- tfit^2/factorial(2)
mat <- matrix(c(tfit,tsqfit),nrow=20,ncol=2,dimnames = list(c(),c("tfit","tsqfit")))
newnq <- data.frame(mat)
pfore.quad <- predict(mlr.quad,newnq,se.fit = TRUE)
efore.quad <- test_set - pfore.quad$fit
```

```
tfit <- c(378:397)
tcubfit <- tfit^3/factorial(3)
mat <- matrix(c(tfit,tsqfit,tcubfit),nrow=20,ncol=3, dimnames = list(c(),c("tfit","tsqfit","tcubfit")))
newnc <- data.frame(mat)
pfore.cub <- predict(mlr.cub,newnc,se.fit = TRUE)
efore.cub <- test_set - pfore.cub$fit
```

```
tfit <- c(378:397)
tquarfit <- tfit^4/factorial(4)
mat <- matrix(c(tfit,tsqfit,tcubfit,tquarfit),nrow=20,ncol=4,
              dimnames = list(c(),c("tfit","tsqfit","tcubfit","tquarfit")))
newnc <- data.frame(mat)
pfore.quar <- predict(mlr.quar,newnc,se.fit = TRUE)
efore.quar <- test_set - pfore.quar$fit
```

```r
mape.lin <- 100*(mean(abs((efore.lin)/test_set)))
mape.quad <- 100*(mean(abs((efore.quad)/test_set)))
mape.cub <- 100*(mean(abs((efore.cub)/test_set)))
mape.quar <- 100*(mean(abs((efore.quar)/test_set)))
```

```r
data.frame(
  model = c("lin", "quad", "cub", "quar"),
  MAPE = c(mape.lin, mape.quad, mape.cub, mape.quar)
)
```

```
##   model        MAPE
## 1   lin   5.8314833
## 2  quad   0.5580105
## 3   cub   7.1480535
## 4  quar 118.7957671
```

We can see that the quadratic model has the smallest MAPE, which means that it has the best predictive performance. Although the quartic model do well in AIC, but it is a overfitted model, it has too large MAPE.
Considering the AIC and MAPE, I decide to use quadratic model to remove the deterministic trend for further analyze.

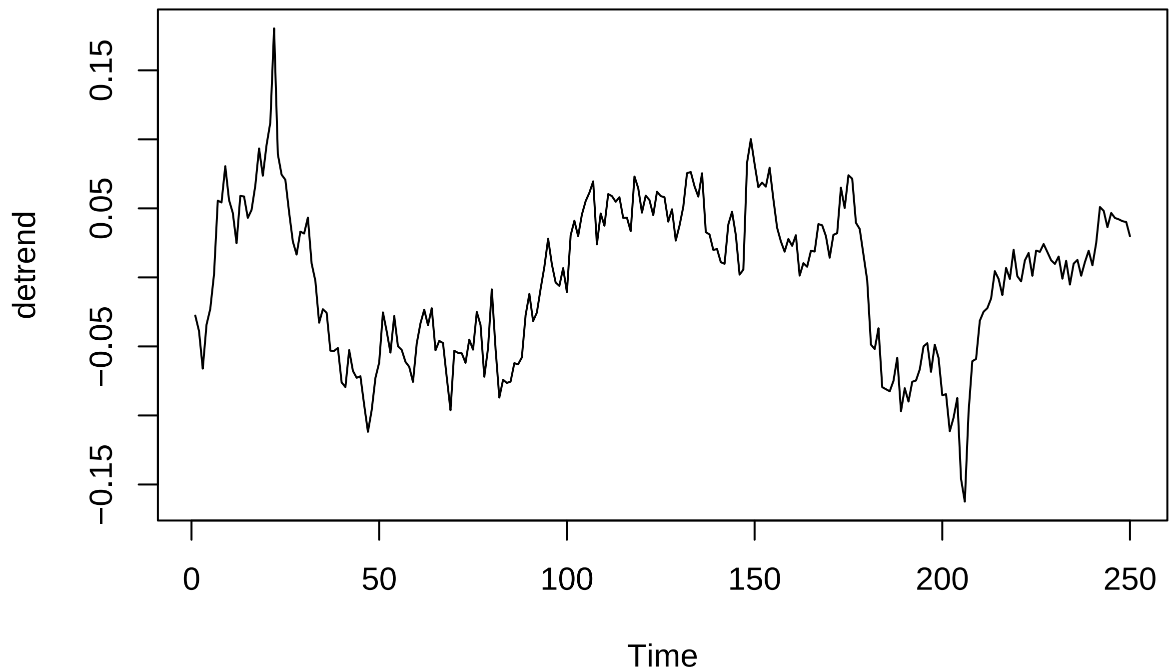
Stochastic trend

After removing the deterministic trend, we need to explore the stochastic trend of the data to help us better forecast.

```r
detrend <- mlr.quad$resid
detrend_test <- efore.quad
```

let's see the time series plot of the data we need to analyze now.

```r
ts.plot(detrend)
```



Well, the mean is around 0, and the values volatile between around -0.15 and 0.15.

Sationarity check And let's check whether the data is statioanary or not.

```
library(fUnitRoots)
adfTest(detrend, lags=10, type = "c")
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 10
##   STATISTIC:
##     Dickey-Fuller: -2.3845
##   P VALUE:
##     0.169
##
## Description:
##  Mon Apr 22 23:53:18 2024 by user:
```

From the Augmented Dickey-Fuller Test, we can know its p-value is significantly large, so we need to difference the data to make it statioanry.

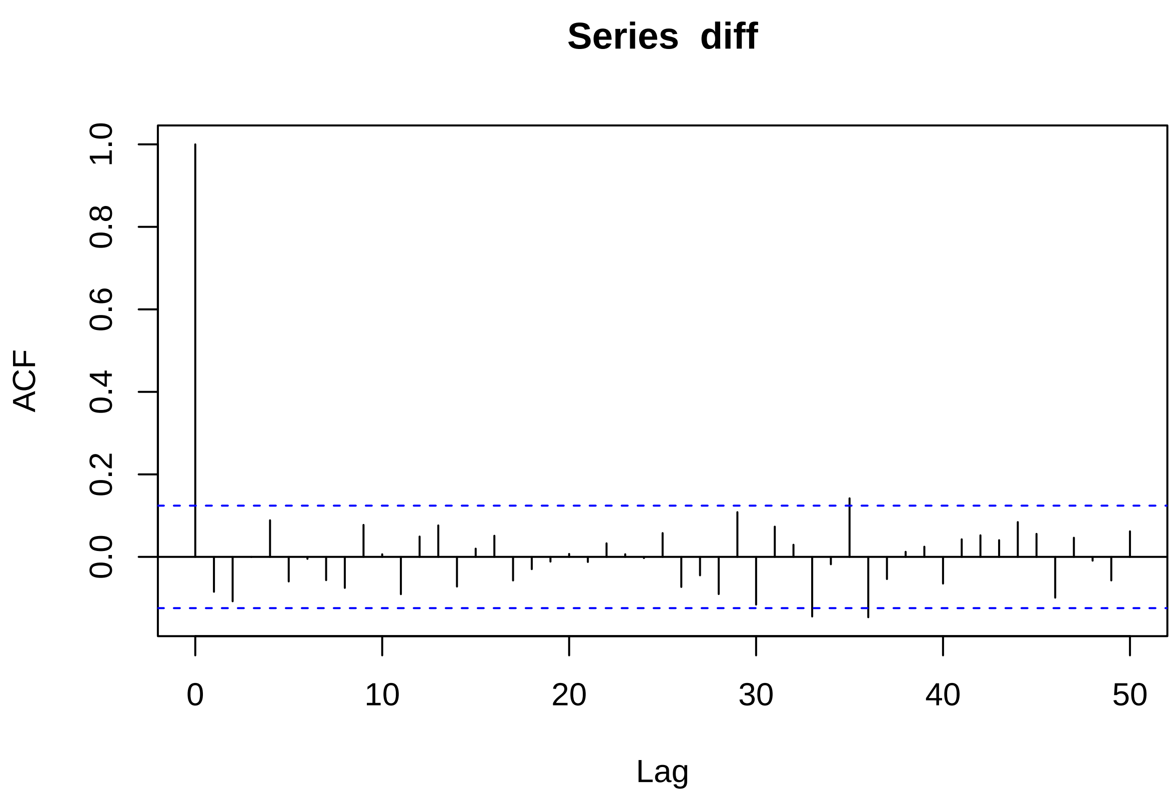
```r
diff <- diff(detrend)
adfTest(diff, lags=10, type = "c")
```

```
## Warning in adfTest(diff, lags = 10, type = "c"): p-value smaller than printed
## p-value
```

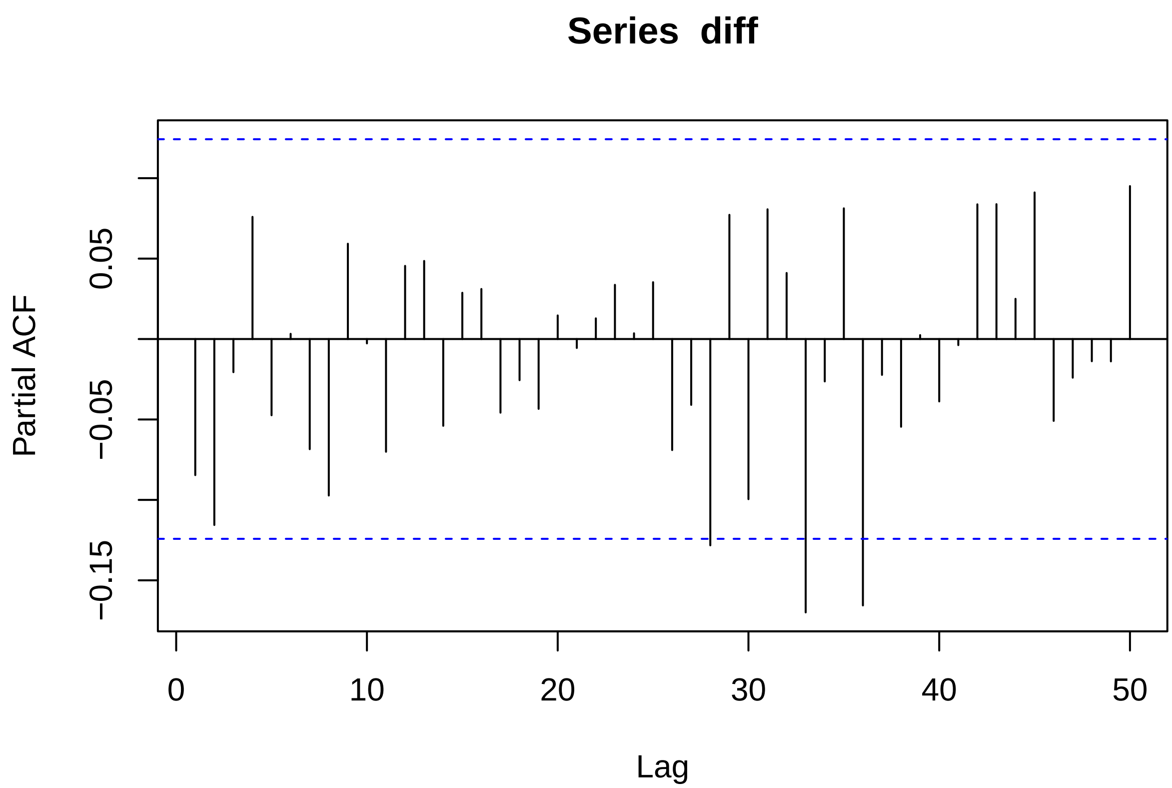
```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 10
##   STATISTIC:
##     Dickey-Fuller: -5.3002
##   P VALUE:
##     0.01
##
## Description:
##  Mon Apr 22 23:53:18 2024 by user:
```

After differencing, we can see that the data is stationary now.

```r
acf(diff, lag.max = 50)
```



```r
pacf(diff, lag.max = 50)
```



From the acf and pacf plot, we can barely observe any autocorrelation.

```r
ar(x = diff)
```

```
##
## Call:
## ar(x = diff)
##
## Coefficients:
##       1        2
## -0.0944  -0.1156
##
## Order selected 2  sigma^2 estimated as  0.0004243
```

But I still want to fit an arima model.
And the ar() function recommend me to fit AR(2).

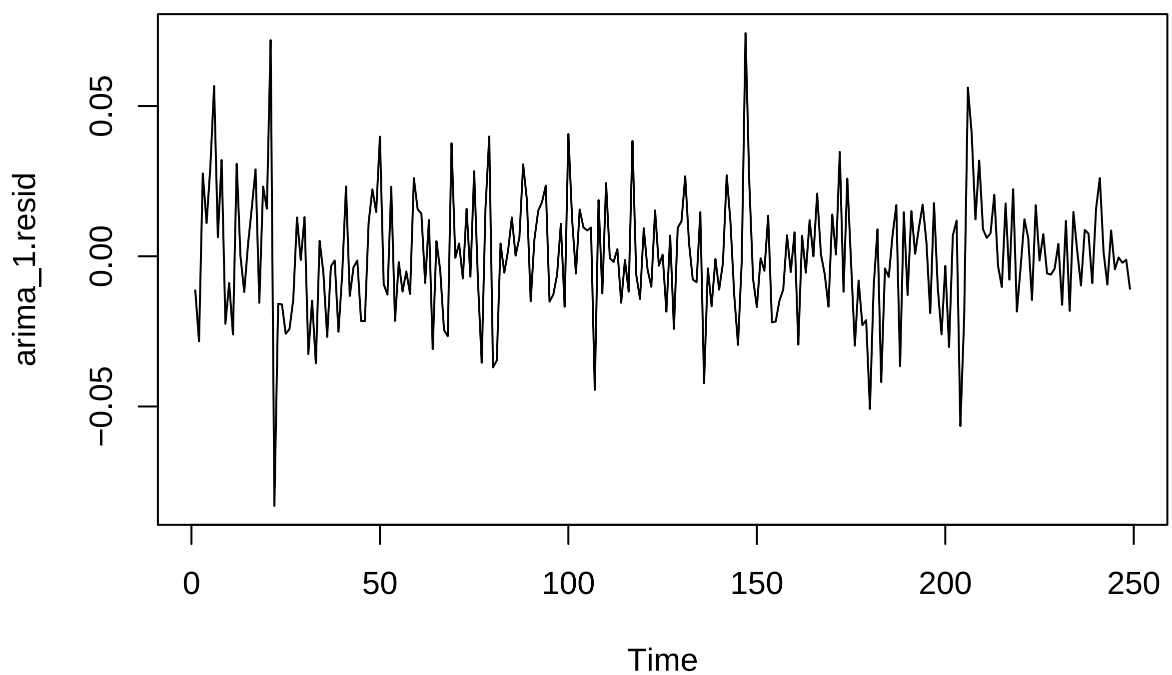
ARIMA(2,1,0)

```r
arima_1 <- arima(diff, order=c(2,0,0), include.mean = TRUE)

arima_1
```

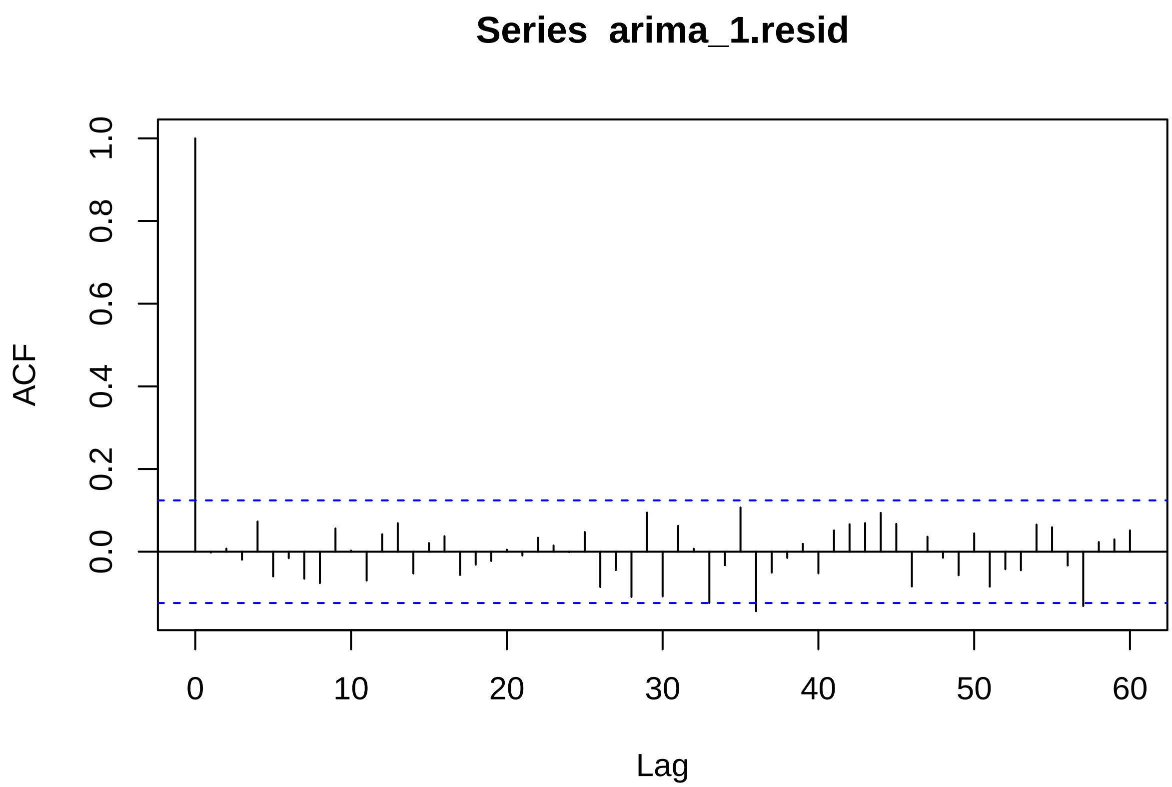
```
##
## Call:
## arima(x = diff, order = c(2, 0, 0), include.mean = TRUE)
##
## Coefficients:
##           ar1      ar2  intercept
##       -0.0947  -0.1161     0.0003
## s.e.   0.0629   0.0630     0.0011
##
## sigma^2 estimated as 0.0004191:  log likelihood = 614.97,  aic = -1221.93
```

let's see check the residuals of the arima model

```r
arima_1.resid <- arima_1$residuals
ts.plot(arima_1.resid)
```




```r
acf(arima_1.resid, lag.max = 60)
```



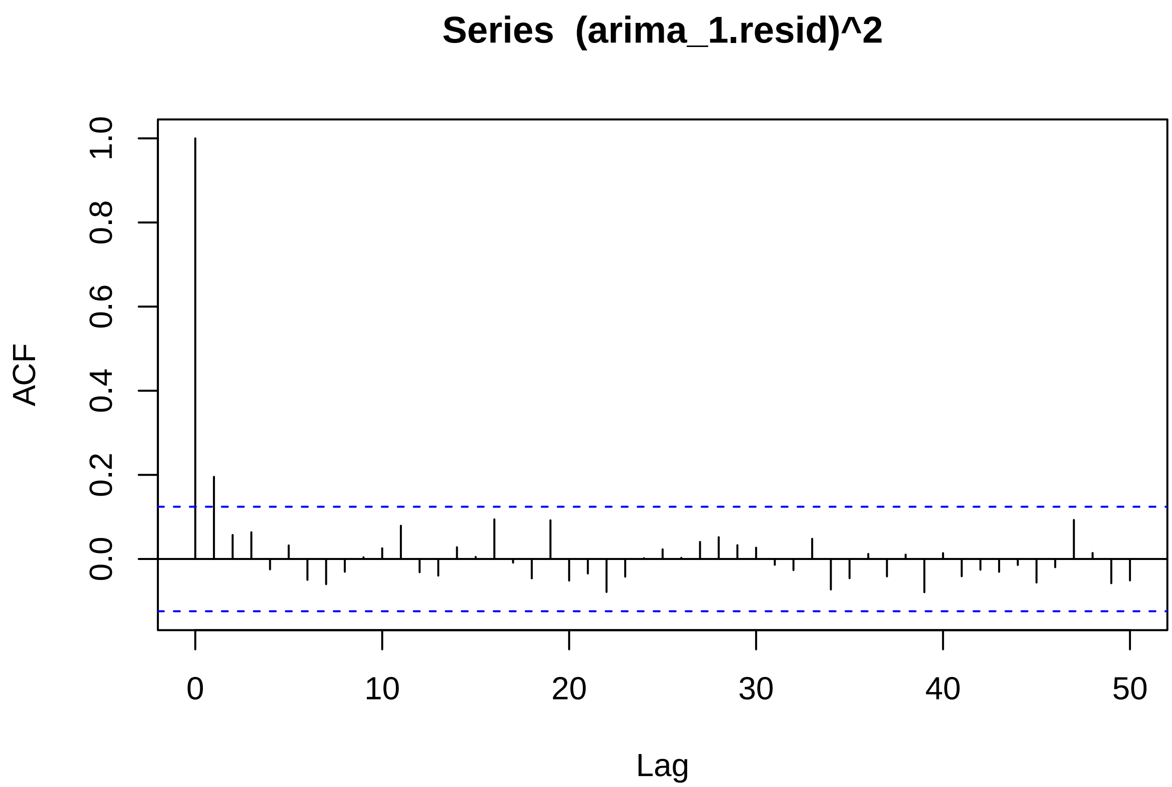

```r
Box.test(arima_1.resid, lag=30, fitdf=1, type = c("Ljung-Box"))
```

```
##
##  Box-Ljung test
##
## data:  arima_1.resid
## X-squared = 24.332, df = 29, p-value = 0.7124
```

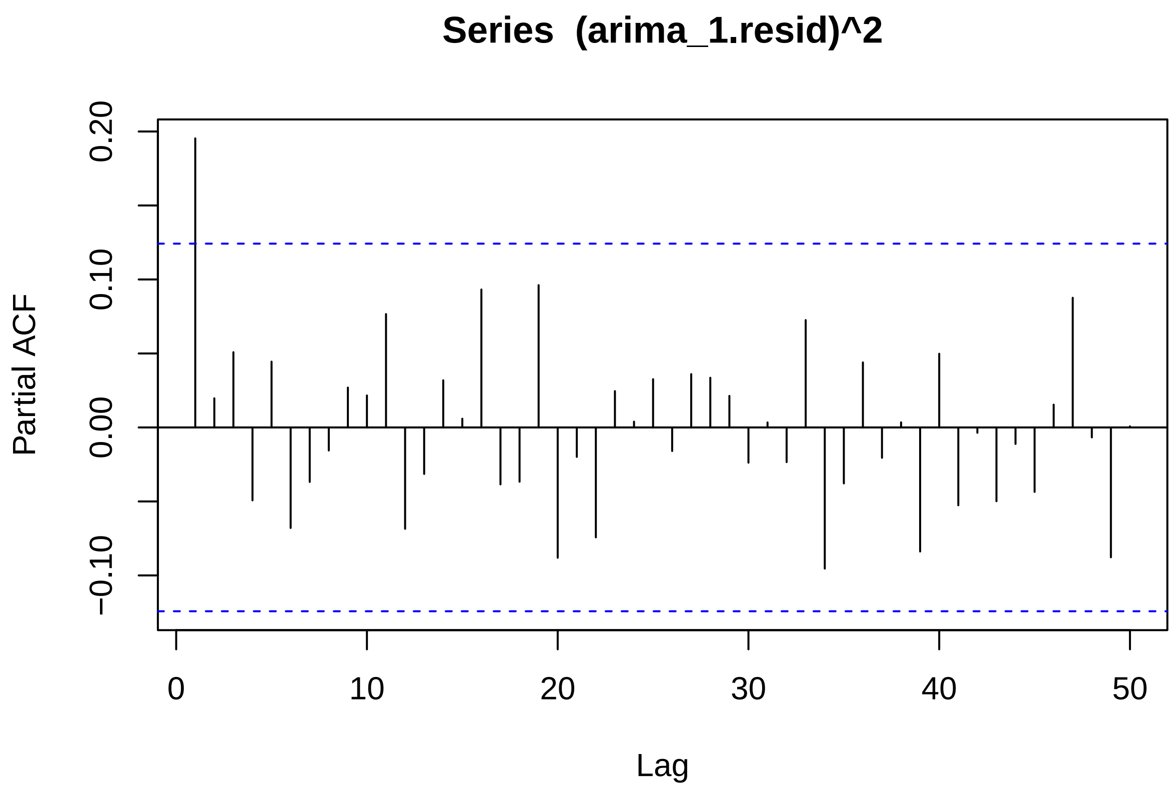
From the acf plot and the result of Box-Ljung test, we can know that the mean model is adequate. And let's check whether there is some ARCH effect.

Heteroscedasticity check

```r
acf((arima_1.resid)^2, lag.max = 50)
```



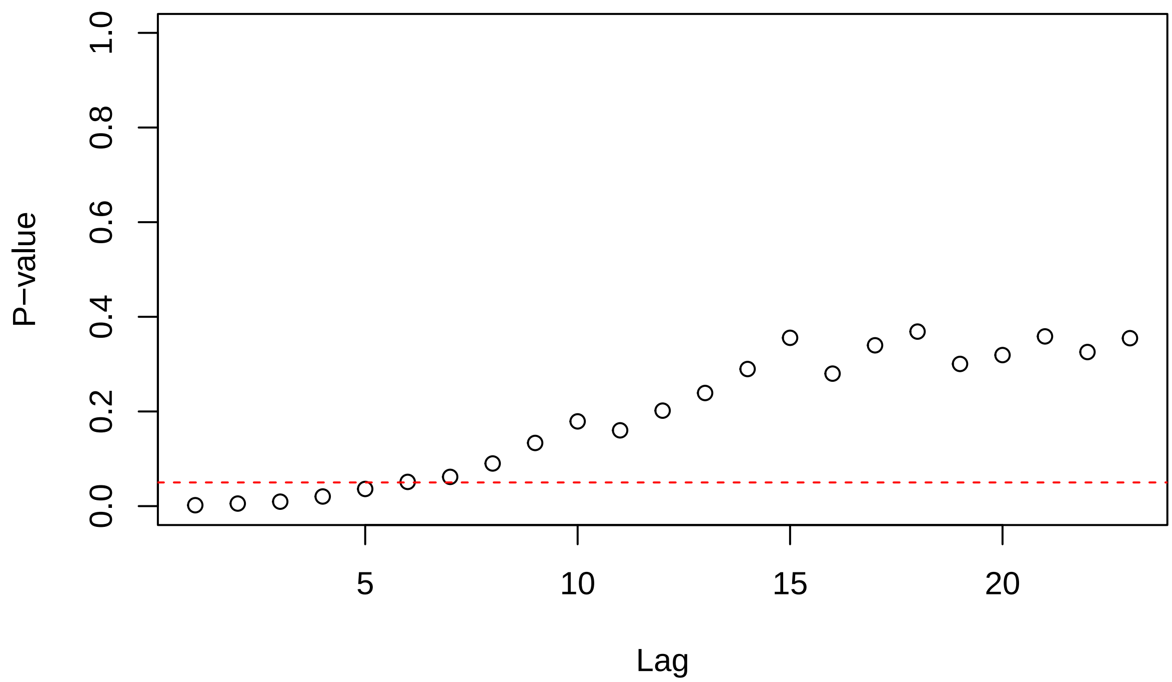

```
pacf((arima_1.resid)^2, lag.max = 50)
```




```r
library(TSA)
```

```
##
## Attaching package: 'TSA'

## The following objects are masked from 'package:stats':
##
##     acf, arima

## The following object is masked from 'package:utils':
##
##     tar
```

```r
McLeod.Li.test(y = arima_1.resid)
```



From the acf and pacf plot of the square of the residuals, we can see there is heteroscedasticity, and we need to fit some ARCH model to improve it.

Heteroscedastic Model

ARCH(1)

we can first try ARCH(1) model.

```
library(fGarch)
```

```
## NOTE: Packages 'fBasics', 'timeDate', and 'timeSeries' are no longer
## attached to the search() path when 'fGarch' is attached.
##
## If needed attach them yourself in your R script by e.g.,
##         require("timeSeries")
```

```
##
## Attaching package: 'fGarch'
```

```
## The following object is masked from 'package:TTR':
##
##     volatility
```

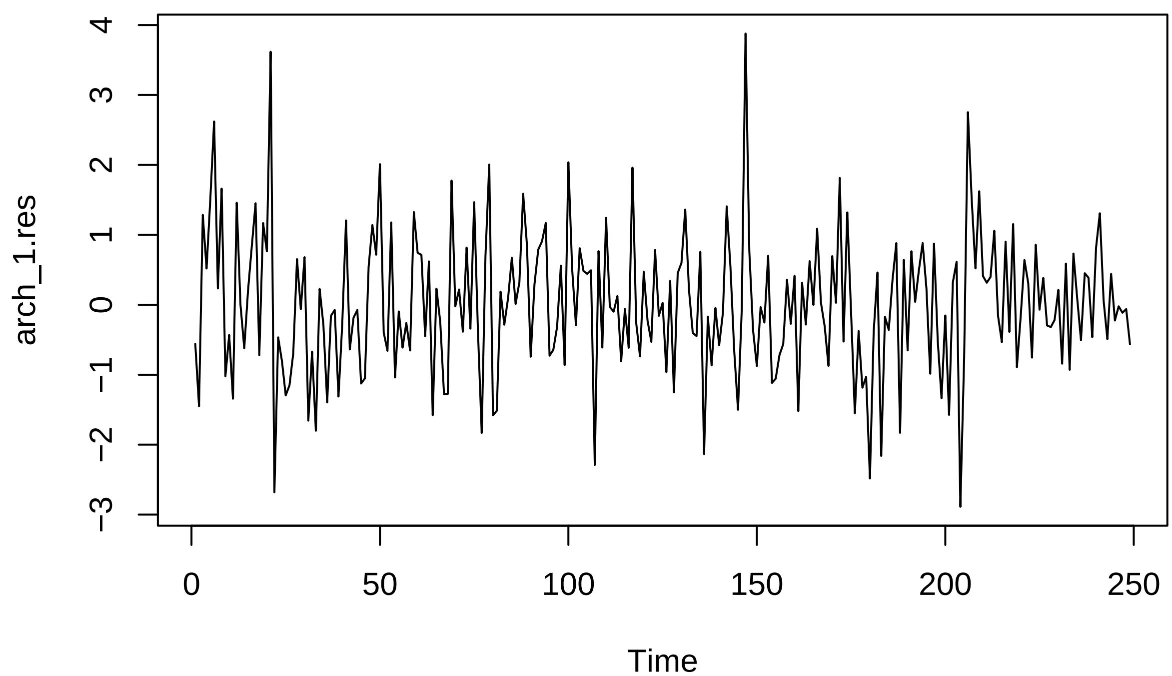
```r
arch_1 <- garchFit(~garch(1,0), data = arima_1.resid, trace=FALSE,
                   cond.dist=c("norm"), include.mean=FALSE)

summary(arch_1)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~garch(1, 0), data = arima_1.resid, cond.dist = c("norm"),
##     include.mean = FALSE, trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ garch(1, 0)
## <environment: 0x1145a3dd0>
##  [data = arima_1.resid]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##     omega     alpha1
## 0.0003667  0.1147555
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## omega  3.667e-04   3.938e-05    9.312   <2e-16 ***
## alpha1 1.148e-01   6.808e-02    1.686   0.0919 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  618.222    normalized:  2.482819
##
## Description:
##  Mon Apr 22 23:53:19 2024 by user:
##
##
## Standardised Residuals Tests:
##                                 Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  18.3431389 0.0001039532
##  Shapiro-Wilk Test  R    W       0.9858868 0.0147014066
##  Ljung-Box Test     R    Q(10)   4.8716325 0.8995863699
##  Ljung-Box Test     R    Q(15)   9.8557127 0.8287164482
##  Ljung-Box Test     R    Q(20)  11.9929702 0.9163177318
##  Ljung-Box Test     R^2  Q(10)   4.8915148 0.8983060284
##  Ljung-Box Test     R^2  Q(15)   7.1105536 0.9545090009
##  Ljung-Box Test     R^2  Q(20)  12.8840517 0.8823032312
##  LM Arch Test       R    TR^2    6.7969287 0.8707363449
```

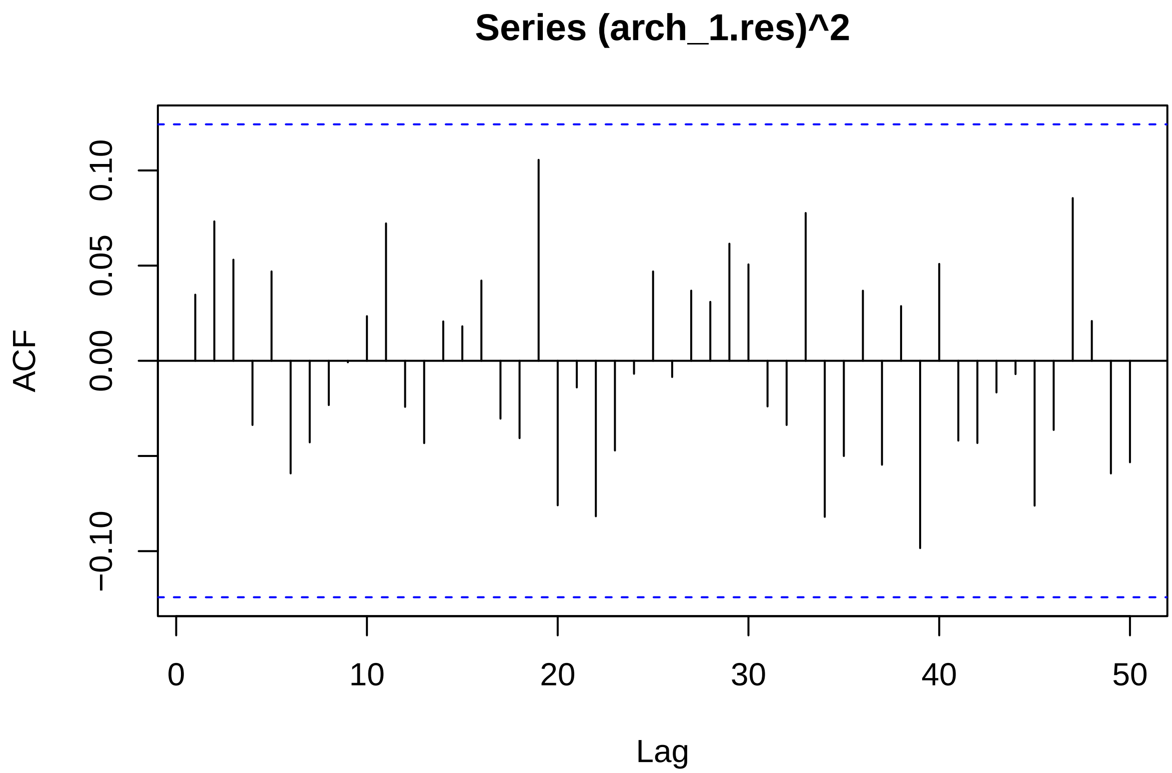
```
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -4.949574 -4.921322 -4.949702 -4.938202
```

Let's check the residuals of the model.

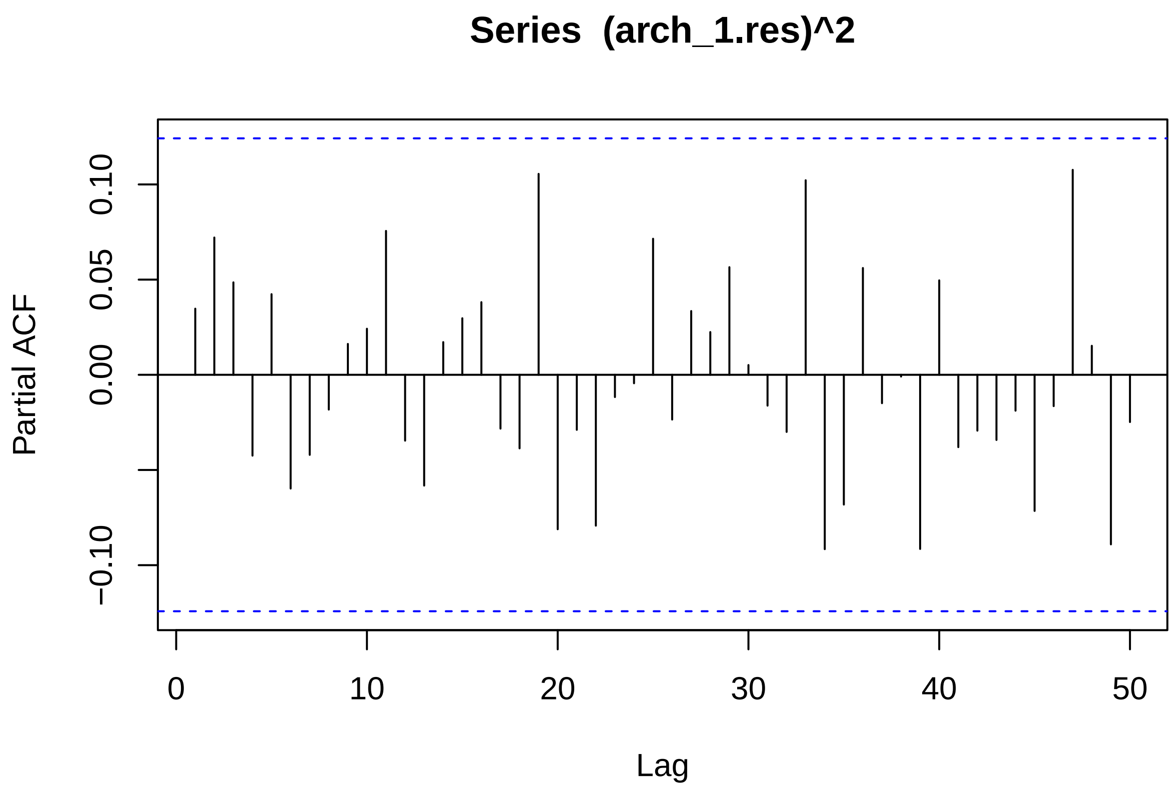
```
arch_1.res <- residuals(arch_1, standardize=TRUE)
ts.plot(arch_1.res)
```



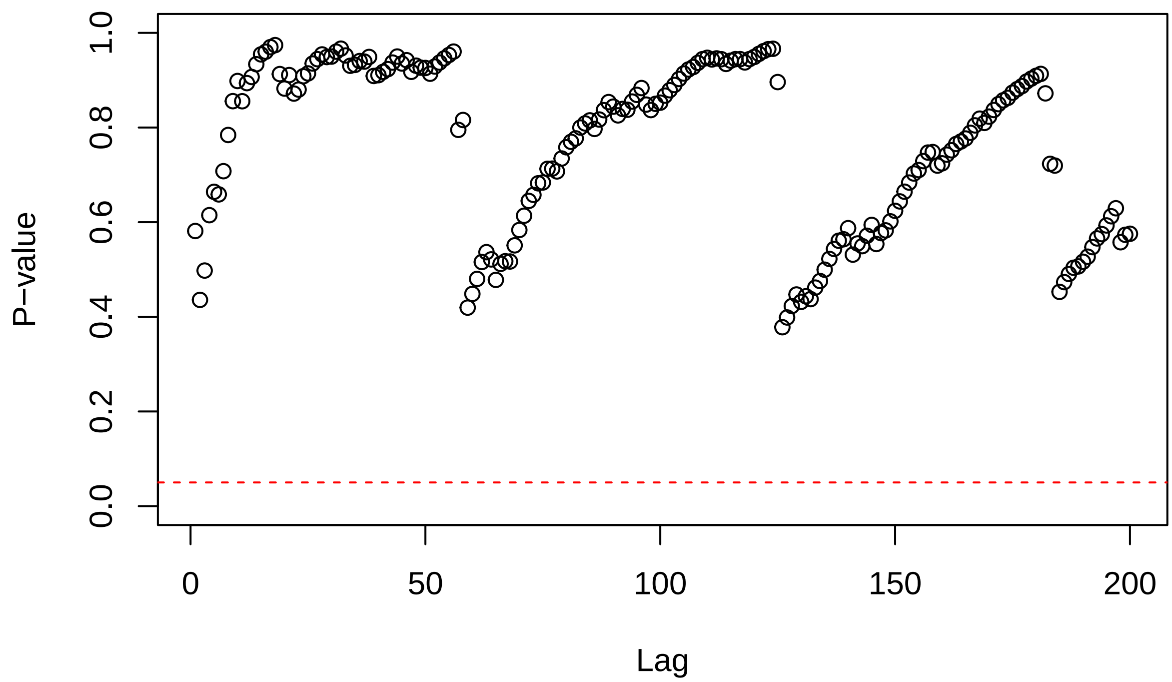

```
acf((arch_1.res)^2, lag.max = 50)
```




```
pacf((arch_1.res)^2, lag.max = 50)
```




```
McLeod.Li.test(y = arch_1.res, gof.lag = 200)
```



The acf and pacf plot shows there is no further ARCH effect, and McLeod-Li test shows that the model is adequate. But it shows some abnormal pattern that I cannot explain. Let's check the normality of the residuals.

```
shapiro.test(arch_1.res)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  arch_1.res
## W = 0.98589, p-value = 0.0147
```

The Shapiro-Wilk Normality Test shows that the residuals is not a white noise under the significant level $\alpha = 0.05$.
So we need to fit other models or greater orders.

GARCH(1,1)

```
garch_11 <- garchFit(~arma(2,0)+garch(1,1), data=arima_1.resid, trace=FALSE, cond.dist=c("norm"), inclu

summary(garch_11)
```

```
##
```

```
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~arma(2, 0) + garch(1, 1), data = arima_1.resid,
##     cond.dist = c("norm"), include.mean = FALSE, trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ arma(2, 0) + garch(1, 1)
## <environment: 0x113910860>
##  [data = arima_1.resid]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##         ar1          ar2        omega       alpha1        beta1
## -0.06446519   0.04390722   0.00017196   0.16109034   0.42513185
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##          Estimate  Std. Error  t value Pr(>|t|)
## ar1    -6.447e-02   7.435e-02   -0.867   0.3859
## ar2     4.391e-02   6.872e-02    0.639   0.5228
## omega   1.720e-04   7.783e-05    2.209   0.0271 *
## alpha1  1.611e-01   8.418e-02    1.914   0.0557 .
## beta1   4.251e-01   2.056e-01    2.068   0.0386 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  621.2061    normalized:  2.494804
##
## Description:
##  Mon Apr 22 23:53:19 2024 by user:
##
##
## Standardised Residuals Tests:
##                                 Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  15.4862157 0.0004337215
##  Shapiro-Wilk Test  R    W       0.9881126 0.0376363158
##  Ljung-Box Test     R    Q(10)   5.6074356 0.8470965898
##  Ljung-Box Test     R    Q(15)  10.6279659 0.7784938891
##  Ljung-Box Test     R    Q(20)  12.6848340 0.8904932510
##  Ljung-Box Test     R^2  Q(10)   3.3491555 0.9719736875
##  Ljung-Box Test     R^2  Q(15)   6.9962619 0.9577533939
##  Ljung-Box Test     R^2  Q(20)  13.4698595 0.8563270952
##  LM Arch Test       R    TR^2    5.9906542 0.9165524389
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -4.949447 -4.878815 -4.950232 -4.921017
```

```r
shapiro.test(residuals(garch_11, standardize=TRUE))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(garch_11, standardize = TRUE)
## W = 0.98811, p-value = 0.03764
```

We can see that the data is still not normally distributed under the significant level $\alpha = 0.05$.

APARCH(1,1)

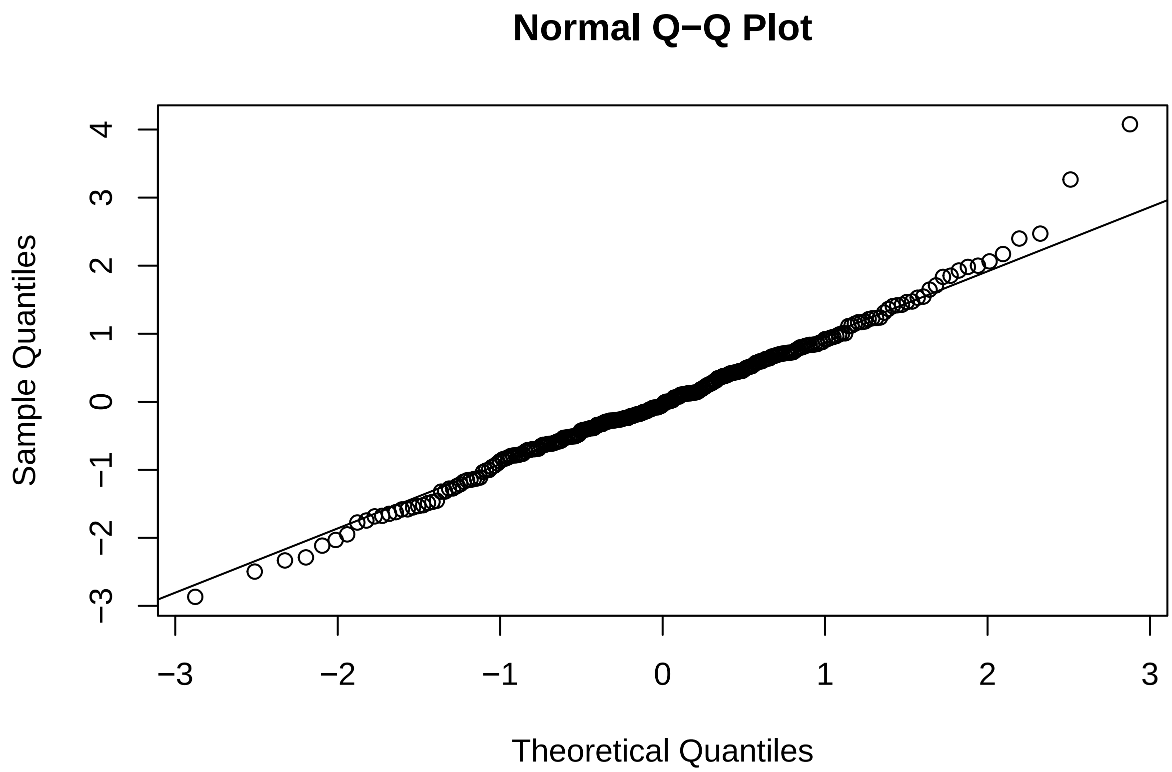
Since I observed that in the time series plot, the absolute value of positive peaks(~4) is larger than the absolute value of the negative peaks(~3), so I think that this might be the leverage effect, and I fit a APARCH/TGARCH model.

```r
aparch_11 <- garchFit(~arma(2,0)+aparch(1,1), data=arima_1.resid,
                      trace=FALSE, cond.dist=c("norm"),
                      delta=1,include.delta=F, include.mean=FALSE)

summary(aparch_11)
```

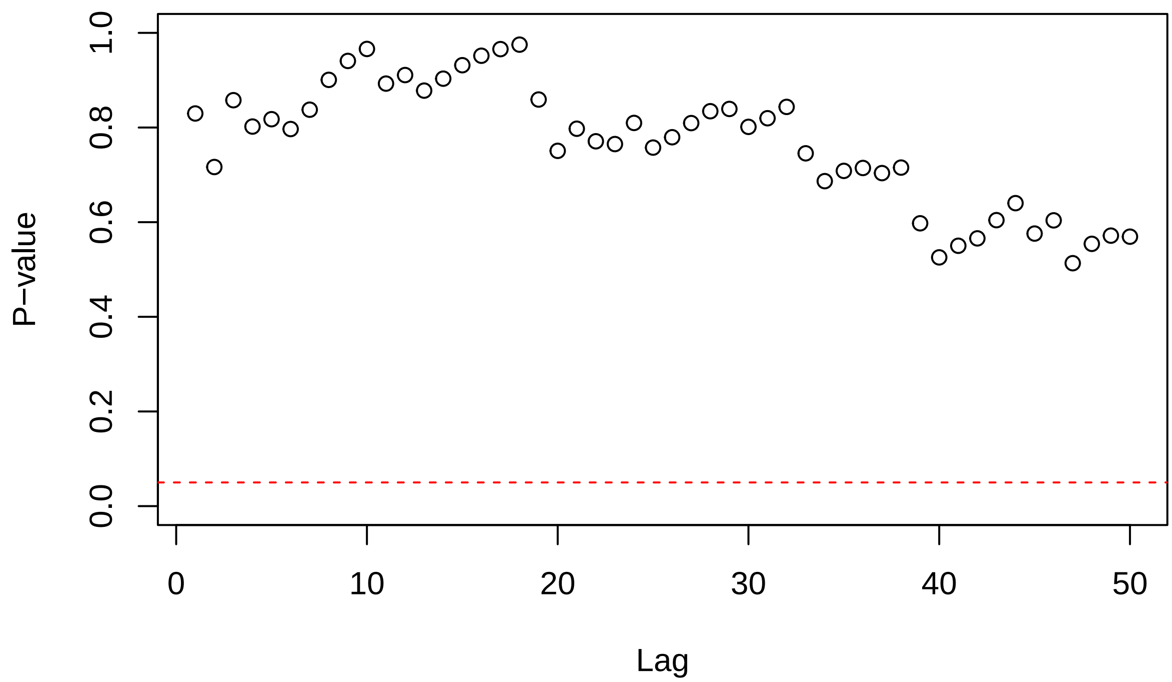
```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~arma(2, 0) + aparch(1, 1), data = arima_1.resid,
##     delta = 1, cond.dist = c("norm"), include.mean = FALSE, include.delta = F,
##     trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ arma(2, 0) + aparch(1, 1)
## <environment: 0x116438a08>
##  [data = arima_1.resid]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##        ar1         ar2       omega      alpha1      gamma1       beta1
## -0.0787227   0.0327566   0.0079085   0.1851578  -0.1844504   0.4634344
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## ar1    -0.078723    0.076237   -1.033   0.3018
## ar2     0.032757    0.066171    0.495   0.6206
## omega   0.007909    0.004308    1.836   0.0664 .
```

```
## alpha1  0.185158    0.076463    2.422   0.0155 *
## gamma1 -0.184450    0.255376   -0.722   0.4701
## beta1   0.463434    0.226398    2.047   0.0407 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  622.924    normalized:  2.501703
##
## Description:
##  Mon Apr 22 23:53:19 2024 by user:
##
##
## Standardised Residuals Tests:
##                                 Statistic     p-Value
##  Jarque-Bera Test   R    Chi^2  13.347759 0.001263487
##  Shapiro-Wilk Test  R    W       0.989754 0.075907476
##  Ljung-Box Test     R    Q(10)   5.982691 0.816715344
##  Ljung-Box Test     R    Q(15)  11.152999 0.741674999
##  Ljung-Box Test     R    Q(20)  12.893123 0.881922391
##  Ljung-Box Test     R^2  Q(10)   3.531179 0.966029079
##  Ljung-Box Test     R^2  Q(15)   7.796317 0.931688603
##  Ljung-Box Test     R^2  Q(20)  15.440130 0.750693918
##  LM Arch Test       R    TR^2    5.942989 0.918928583
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -4.955213 -4.870455 -4.956338 -4.921097
```

```r
aparch_11.resid <- residuals(aparch_11, standardize=TRUE)
qqnorm(aparch_11.resid)
qqline(aparch_11.resid)
```




```
McLeod.Li.test(y = aparch_11.resid, gof.lag = 50)
```




```r
shapiro.test(aparch_11.resid)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  aparch_11.resid
## W = 0.98975, p-value = 0.07591
```

We can see that the residuals is normally distributed now, and the model is adequate.

Final Model

According to the analyze above, we can get a model with the following expression:
$X_t = \mu + 3.272 \times 10^{-3}t - 4.8135 \times 10^{-6}t^2 + Y_t$
$Y_t - Y_{t-1} = Z_t$
$Z_t = -0.0787Z_{t-1} + 0.0328Z_{t-2} + e_t$
$e_t = \sigma_t \epsilon_t$
$\sigma_t^2 = 0.0079 + 0.185e_{t-1} - 0.184e_{t-1}I\{e_{t-1} < 0\} + 0.4634\sigma_{t-1}$

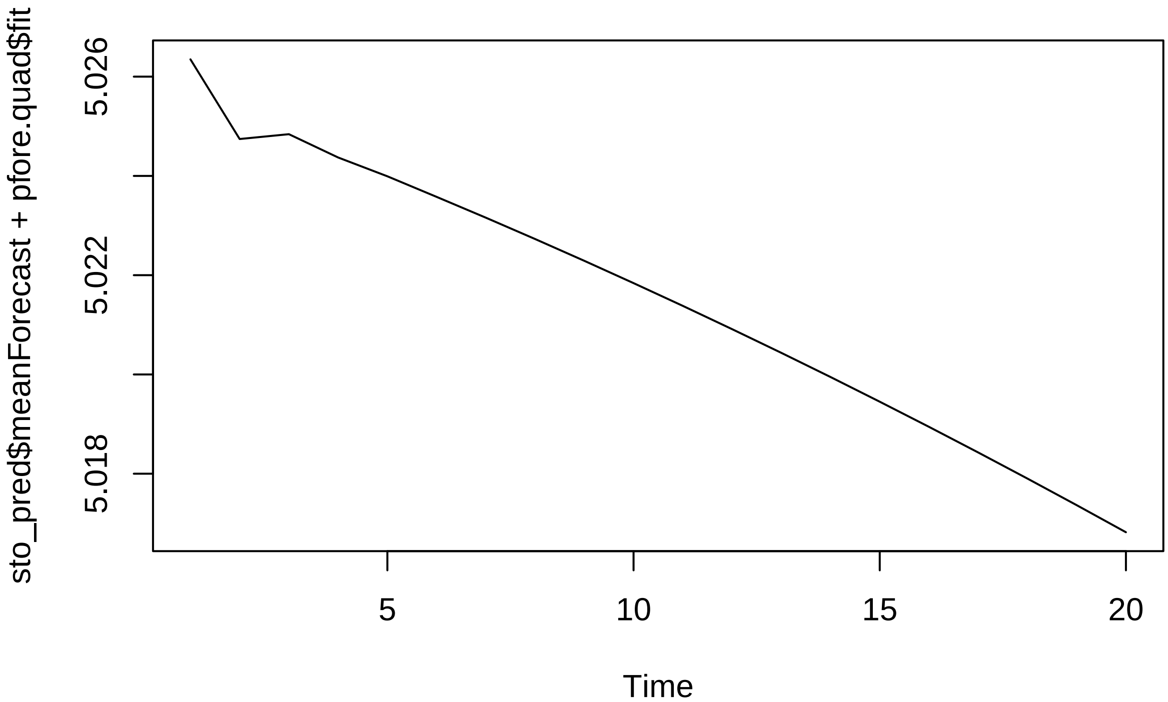
Forecast

We have already forecasted the deterministic trend, so what we need to do now is to forecast the stochastic trend and combine them.

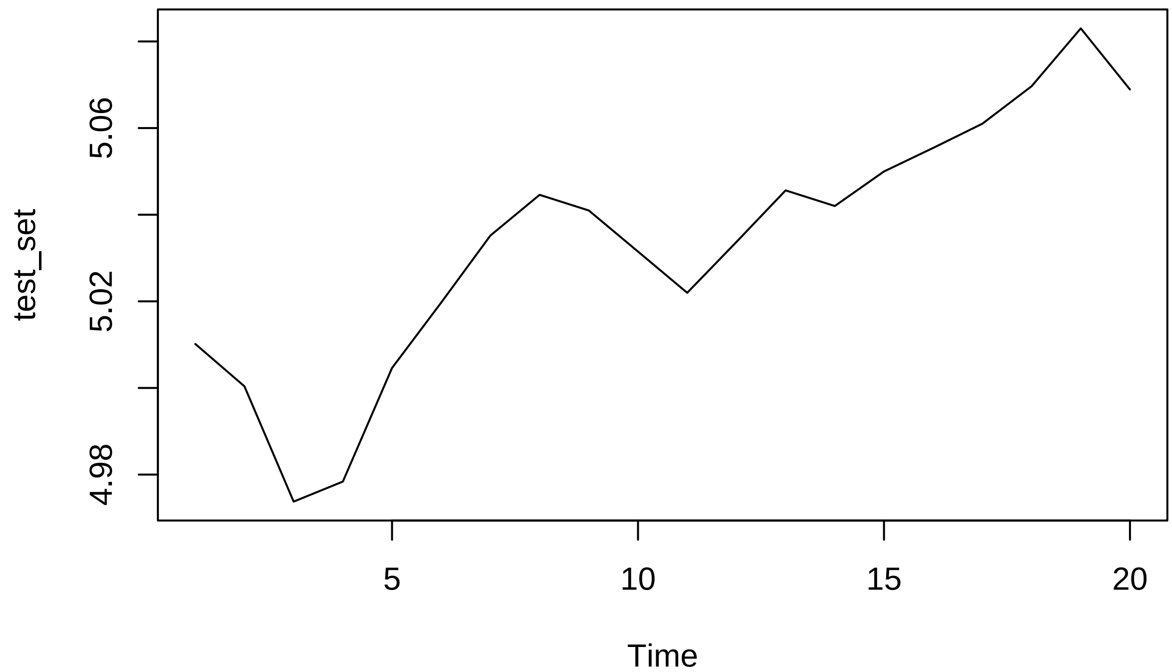
Forecast the ARIMA(2,1,0)+TGARCH(1,1)

```
sto_pred <- predict(aparch_11, n.ahead=20, plot=FALSE, conf=.95, nx=100)
```

```
ts.plot(sto_pred$meanForecast + pfore.quad$fit)
```




```
ts.plot(test_set)
```



Let's calculate the MAPE of the whole model(deterministic trend + stochastic trend model)

```
mape_log <- 100*(mean(abs((test_set - (sto_pred$meanForecast + pfore.quad$fit))/test_set)))
cat("The MAPE of the log transformation process is:",mape_log)
```

```
## The MAPE of the log transformation process is: 0.5584481
```

```
mape <- 100*(mean(abs((test - exp(sto_pred$meanForecast + pfore.quad$fit))/test)))
cat("The MAPE of the whole model is:",mape)
```

```
## The MAPE of the whole model is: 2.789165
```