STAT GR5221 Project

Runfeng Tian (rt2755)

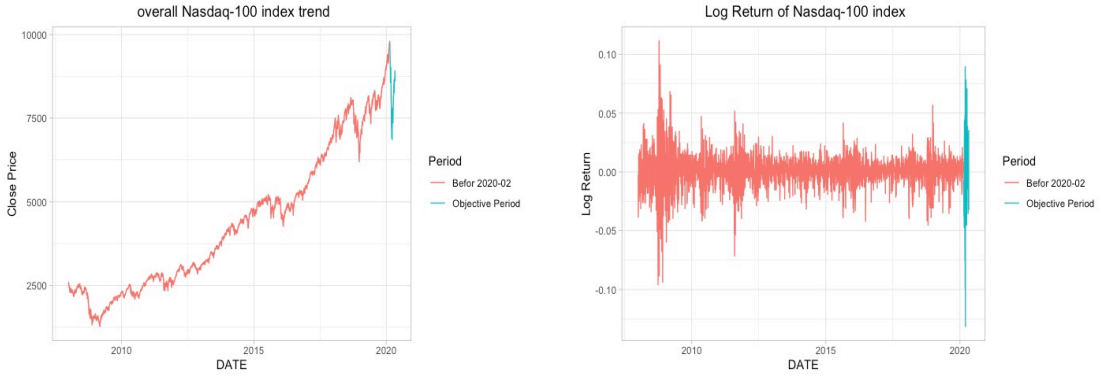
1 Introduction

1.1 Data description

Here we have a dataset containinng Nasdaq-100 index from 2008-01-02 to 2020-05-01. Nasdaq-100 index is a stock market index made up of 103 equity securities issued by 100 of the largest non-financial American and international companies such as Amazon, Apple inc. and Baidu. It is an important indication of the trend of American stock market.

1.2 Objective

In this project, whether time series model ARMA and GARCH can perform consistently when volatility changes abruptly in a very short time will be checked.



(a) Nasdaq-100 index trend

(b) Log return trend of Nasdaq-100 trend

Figure 1: Overall trend plot

Beginning in 2020-02, the pandemic of COVID-19 continued to impact the american stock market. From Figure1, we can find that the blue part in plot (a) and plot(b) have a high volatitlity different from former period. Such uncertainty in stock market is both a challenge and opportunity for investment. In this project, I will utilize time series models ARMA and GARCH trained on former data, and then test the model performances through forecasting on the latest data observations (check the accurracy of forecast in such a period with high volatility).

2 Exploratory data analysis and modeling

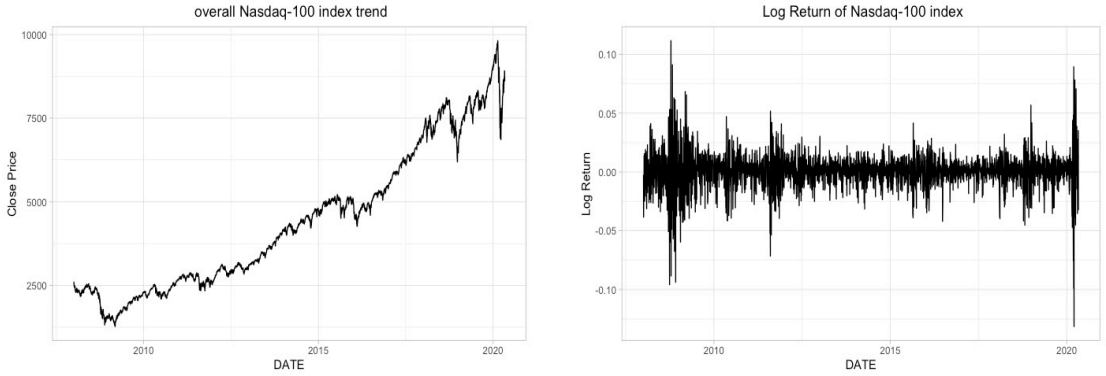
2.1 Data clean and transofrmation

This dataset containing total 3105 observations of Nasdaq-100 index from 2008-01-02 to 2020-05-01. After checking, I find that there are 3 missing values at time nodes:2016-12-14, 2017-09-06 amd 2020-04-02.

To deal with missing values, one can implement several methods such as deleting the observations, forward filling, backward filling and etc. Here, since the missing values are not neighbour to each other, it is easy to substitute the missing values with mean of the former and next non-missing values cloest to that data points.

In other words, let the P_t denote the original Nasdaq-100 index. Missing data points P_{t_i} can be represented as:

$$P_{t_i} = \frac{P_{t_i+1} + P_{t_i-1}}{2}$$



(a) Orginal P_t trend

(b) R_t trend

Figure 2: Data Transformation Plot

In Figure 2 part(a), there is an obvious increase trend, which means that the original Nasdaq-100 index data is non-stationary. In order to apply time series model such as ARMA or Garch on the series. Data transformation is necessary.
Let r_t denote the log return of the original price data P_t

$$r_t = \frac{P_t}{P_{t-1}} \quad t = 2, 3, .., 3105$$

Here, due to several beneficiary properites of log return, such as stationarity, gaussain property(if logPt is gaussain, then $r_t = \frac{P_t}{P_{t-1}}$is also gaussain. Hence, the compound log return from $t1$ to $t2$ is $log\frac{P_{t2}}{P_{t1}} = r_{t2} + r_{t2-1} + ... + r_{t1+1})$, it is convenient and meaningful to do time series analysis on squence r_t instead of P_t.

In Figue2(b), there is no obvious increase and decrease trend in r_t. So we can generally consider the sequence is stationary, but the volatilties vareis from periods to periods. Theoritically, GARCH model will have higher performance in such data than ARMA do.

2.2 Modeling and Forecast Evaluation

2.2.1 Modeling on the whole data set

In this section, the whole dataset with 3105 observations is split into training data before 2020-02-20 and test data from 2020-02-20 to 2020-05-01. Then ARMA and ARMA-GARCH models are trained seperately on training set.
First, train Model 1 ARAMA(4,5). The model odrder is selected based on AIC.

$$\begin{aligned} \textbf{Model 1}\ \text{ARMA}(4,5):\ \phi(B)r_t = \phi_0 + \theta(B)w_t, \quad w_t \sim N(0,\sigma_w^2) \\ where \quad \phi(z) = 1 - \phi_1 z - \phi_2 z^2 - \phi_3 z^3 - \phi_4 z^4, \\ \theta(z) = 1 + \theta_1 z + \theta_2 z^2 + \theta_3 z^3 + \theta_4 z^4 + \theta_5 z^5 \\ \phi_0\ is\ intercept \end{aligned} \tag{1}$$

The estimations of Model 1 are following:

```
Coefficients:
         ar1       ar2       ar3       ar4     ma1     ma2     ma3     ma4
      -0.7325  -0.6665  -0.9797  -0.2389  0.6592  0.5667  0.9200  0.1372
s.e.   0.3170   0.1762   0.1857   0.2911  0.3166  0.1572  0.1767  0.2850
         ma5     intercept
      -0.0873       4e-04
s.e.   0.0272       2e-04

sigma^2 estimated as 0.0001757:  log likelihood = 8867.43,  aic = -17712.85
```

Second, implement ARMA(4,5)-GARCH(1,1) on training data.

$$\begin{aligned} \textbf{Model 2}\ \text{ARMA}(4,5)-\text{GARCH}(1,1):\ \phi(B)r_t = \phi_0 + \theta(B)w_t, \quad w_t \sim N(0,\sigma_w^2) \\ where \quad \phi(z) = 1 - \phi_1 z - \phi_2 z^2 - \phi_3 z^3 - \phi_4 z^4, \\ \theta(z) = 1 + \theta_1 z + \theta_2 z^2 + \theta_3 z^3 + \theta_4 z^4 + \theta_5 z^5 \\ \phi_0\ is\ intercept \end{aligned} \tag{2}$$

$$\begin{aligned} w_t = sigma_t \epsilon_t,\ \epsilon_t \sim N(0,1) \\ \sigma_t^2 = \alpha_0 + \alpha_1 w_{t-1}^2 + \beta_1 \sigma_{t-1}^2 \end{aligned}$$

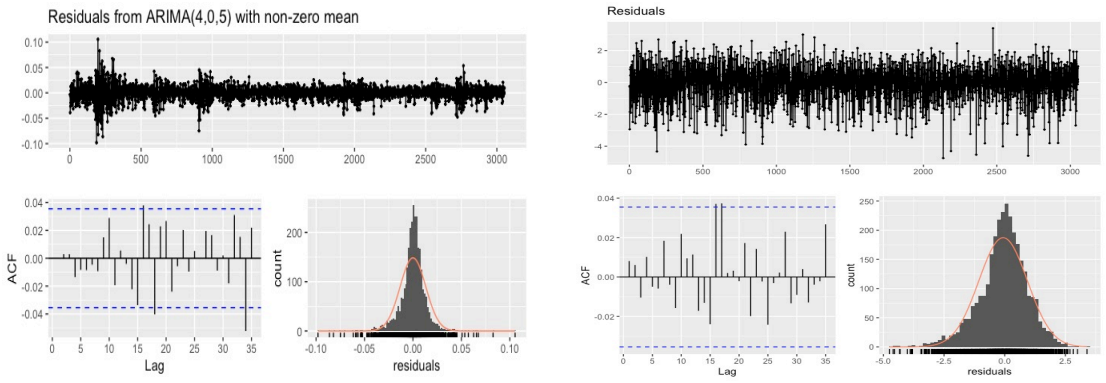
The estimations of Model 2 are following:

```
Coefficients:
        ar1       ar2      ar3      ar4     ma1     ma2      ma3      ma4
     -0.1038  -0.2323   0.7867   0.4298  0.0582  0.2099  -0.8096  -0.4091
s.e.  0.0001   0.0239   0.0132   0.0122  0.0261  0.0200   0.0227   0.0001
        ma5    intercept  omega   alpha1    beta1
      0.0127   0.0009   0.0000   0.1229  0.8521
s.e.  0.0195    0.0001  0.0000   0.0116  0.0146
```

2.3 Model diagnostic and forecast on test set

2.3.1 Model diagnostic

After modeling on trainig set, a series of goodness of fit check processes such as Ling-jun box test are carried.



(a) ARMA(4,5) Residual Check

(b) ARMA(4,5)-GARCH(1,1) Residual Check

Figure 3: Goodness of Fit Check Plot

From Figure3 residual plots and ACF plots, the reisudals of both ARMA(4,5) and ARMA(4,5)-GARCH(1,1) are generally uncorrelated. In addition, the both Ling-jung Box tests fail to reject null hypothesis: the residuals are uncorrelated under the significance level 0.05. Therefore, the 2 models are considered well-fitted.

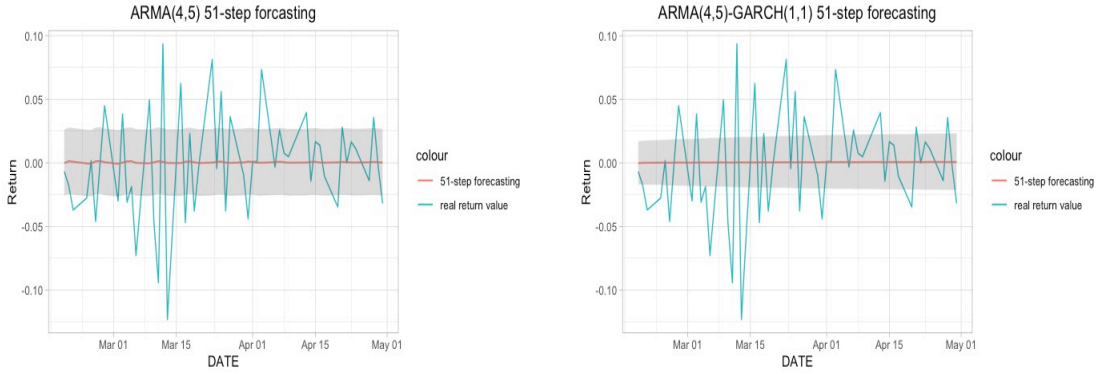
```
Lingjun-Box test:
ARMA(4,5): Q*= 5.9434, p-value=0.1144
ARMA(4,5)-GARCH(1,1): Q*=0.19952, p-value=0.6551
```

2.3.2 Model forecast on test set

After applying goodness of fit check to model 1 and model 2, in this part, the model performances on test set are evaluated which are the key point of this project. The test data containing 51 observations from 2020-02-20 to 2020-05-01 which is the period with much higher volatility than former period. 51-step forecast are conducted on both model 1 and model 2.
After forecaste, both forecaste and real log return values are tranformed into commoly used return. Let $\hat{r_t}$ and r_t respectively denote forecast value and real value of log return. Then return value R_t is expressed as:
$\hat{R_t} = e^{\hat{r_t}} - 1$, $R_t = e^{r_t} - 1$

In Figure 4, both forecasting values in 2 models(red line) are nearly 0 for each data point. The shadow dondary in the plots represent 95 percent confident interval for the real return value. But few of real value(blue points) fall into the confident interval. Such results mean that both ARMA(4,5) and ARMA(4,5)-GARCH(1,1) trained on whole dataset perform poorly on forecast.

The poor forecast can be caused by several possible factors: first, the models we choose are inapproriate for the struture of the data. Second, the way of model fitting is in approriate. It is



(a) ARMA(4,5) 51-step forecast

(b) ARMA(4,5)-GARCH(1,1) 51-step forecast

Figure 4: Forecast on test data

likely that the model trained on long history data can hardly size the abrupt change in vary short time. In this case, the models fail to explain the sudden volatility change leading to poor forecast.

2.4 Modeling on data from 2019-01-02 to 2020-05-01 and forecaste

In this section, modeling training the whole dataset, the same models in 2.2.1 are modeled on a subset of whole dataset containing relatively latest observations. The data containing 336 observations from 2019-01-02 to 2020-05-01 are split into training set including first 316 points and test set including last 20 points.

2.4.1 Modeling on smaller dataset

Model3: ARMA(4,5) trained on subset of data:

```
Coefficients:
         ar1      ar2      ar3      ar4     ma1     ma2     ma3     ma4
      -2.2287  -2.1512  -1.0017  -0.2166  2.0249  1.8683  1.0344  0.6069
s.e.   0.2055   0.4912   0.4685   0.1783  0.2019  0.4420  0.3808  0.1495
         ma5  intercept
      0.3515      3e-04
      0.0780      9e-04
sigma^2 estimated as 0.0002228:  log likelihood = 878.83,  aic = -1735.67
```

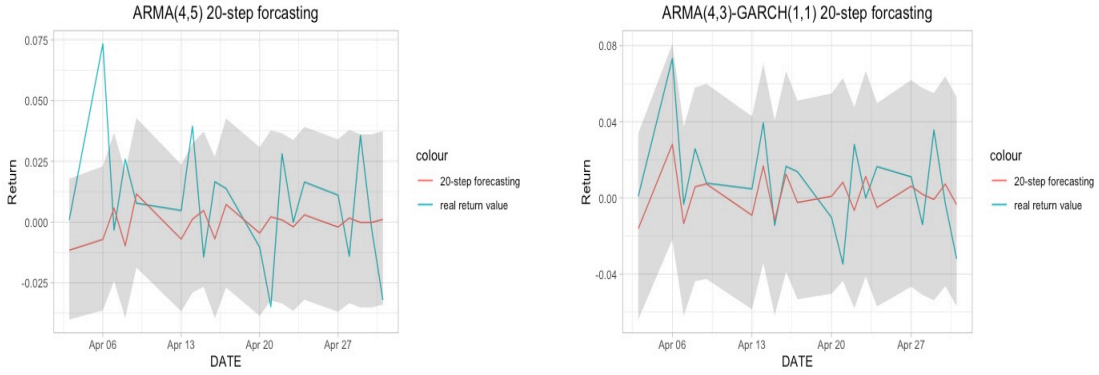
Model4: ARMA(4,3)-GARCH(1,1) (here the model order of q order of ARMA(4,5)-GARCH(1,1) has been lowered because for dataset with smaller number of observations, ARMA-GARCH is reasonable to be simpler):

```
Coefficients:
         ar1      ar2      ar3      ar4     ma1     ma2     ma3
      -1.7431  -1.2770  -0.4744  -0.1984  1.7002  1.1337  0.1999
s.e.   0.0402   0.1767   0.2020   0.0711  0.1373  0.1612  0.0597
```

```
      intercept   omega     alpha1     beta1
        0.0022    0.0000    0.4273    0.5717
s.e.    0.0003    0.0000    0.0846    0.0487
```

2.4.2 Model 3 and 4 evaluation based on forecast

In order to check wether smaller data set inluding latest observations can improve forecast when volatility changes abruply, similar forecast are carried on test data set(20 observations, 2020-04-03 to 2020-05-01).



(a) ARMA(4,5) 20-step forecast

(b) ARMA(4,5)-GARCH(1,1) 20-step forecast

Figure 5: Model 3 and Model 4 Forecast on test data from 2020-04-03 to 2020-05-01

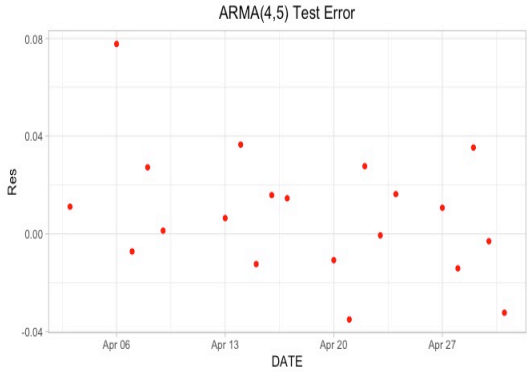
From Figure5, we can find that most real values of return fall into the 95% confidence interval, which means forecast performances of model 3 and model 4 are generally better than model 1 and 2 trained on whole history data. In addition, model 4 performs better on the test data than ARMA(4,5) does because in Figure5(b), the changing pattern of forecast values are similar to that of real values.

The residuals plot of model 3 and model 4 forecasts also show that ARMA(4,3)-GARCH(1,1) model forecast more accurately on the test set than ARMA(4,5) does, since the range of most residuals in Figure6(b) ([-0.025,0.025]) is smaller than that in Figure6(a)([-0.04,0.04]).

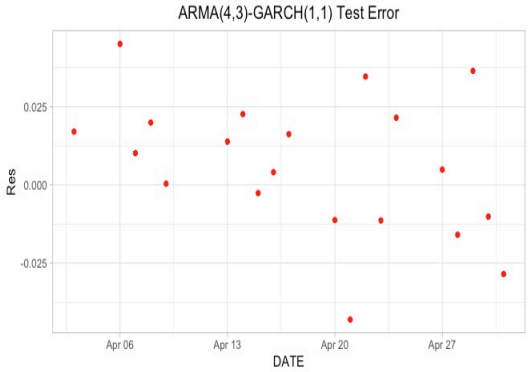
3 Conclusion

Based on the analysis and model comparisons in the former sections, it is concluded that time series models: ARMA and GARCH trained on relatively small dataset can perform better in forecasting than those trained on a dataset with certainly long duration in terms of forecast on a period during which the volatility changes abruptly . That is because for data with volitility varies from period to period, the former observations in the sequence may contain too much useless information would lead to incorrect model fitting. On the other hand, latest data is more useful in correctly desribing current and future market situation.

Furthermore, compared to ARMA model, ARMA-GARCH model taking volatility changes into



(a) ARMA(4,5) 20-step forecast residuals



(b) ARMA(4,5)-GARCH(1,1) 20-step forecast residuals

Figure 6: Model 3 and Model 4 test error

account is likely to achieve a better forecast on data with high volatility. In general, for short term forecast, GARCH model can be a very powerful model.

4 Extension and future research direction

In former discussion and analysis, wether ARMA and ARMA-GARCH model can perform well under the circumstances when volatilities vary in different periods is explored and one can utilize a smaller dataset containing latest data to improve forecast accuracy . Some researchers have provided other methods such as adding dummy variables respresenting the features of different volatilities to GARCH model. They argued that the volatility persistence has been overestimated by GARCH model and taking breakpoints for volatility differences into consideration can effectively correct the volatility persistence in GARCH([1]). Therefore, we can introduce the new model into modeling comparisons and evaluate models in term of forecast. In addition, stock arbitrage method can be designed based on the forecasting return if its accuracy is high enough.

References

[1] Ping Wang and Tomoe Moore. Sudden changes in volatility: The case of five central european stock markets. *Int. Fin. Markets, Inst. and Money*, 19:33–46, 2009.

Attachment

Runfeng Tian

5/5/2020

```
library(timeDate)
library(xts)
```

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

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

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

```
library(forecast)
```

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

```
## Registered S3 methods overwritten by 'forecast':
##   method             from
##   fitted.fracdiff    fracdiff
##   residuals.fracdiff fracdiff
```

```
Nasdaq_price<-read.csv('Nasdaq.csv')
Pt=xts(Nasdaq_price$Adj.Close,order.by=as.Date(Nasdaq_price$Date))
```

```
sum(is.na(Pt))
```

```
## [1] 3
```

```
#sapply(index(Pt[is.na(Pt)]),function(x){Pt[x]<-(Pt[x-1]+Pt[x+1])/2})
t<-list(index(Pt[is.na(Pt)]))
for(i in t){
  print(i)
  Pt[i]<-(as.numeric(Pt[i-1])+as.numeric(Pt[i+1]))/2
}
```

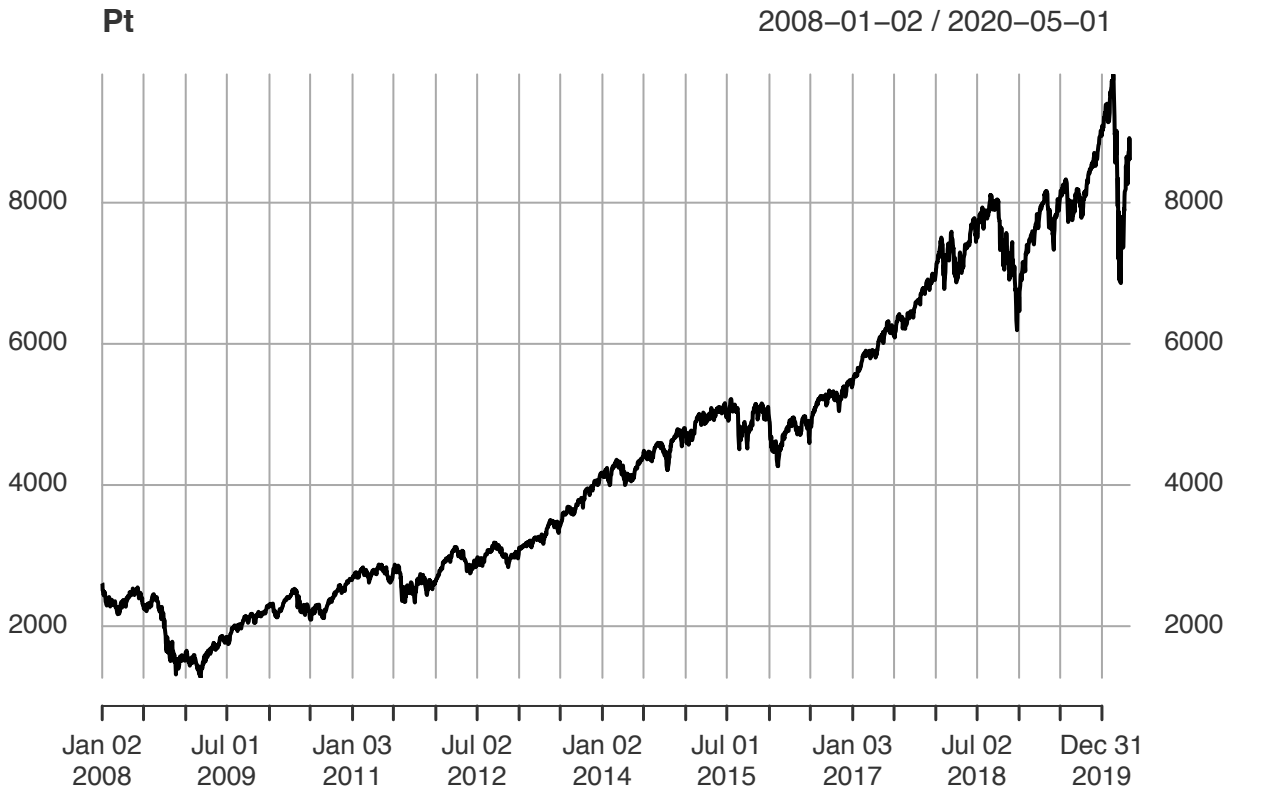
```
## [1] "2016-12-14" "2017-09-06" "2020-04-02"
```

```
sum(is.na(Pt))
```

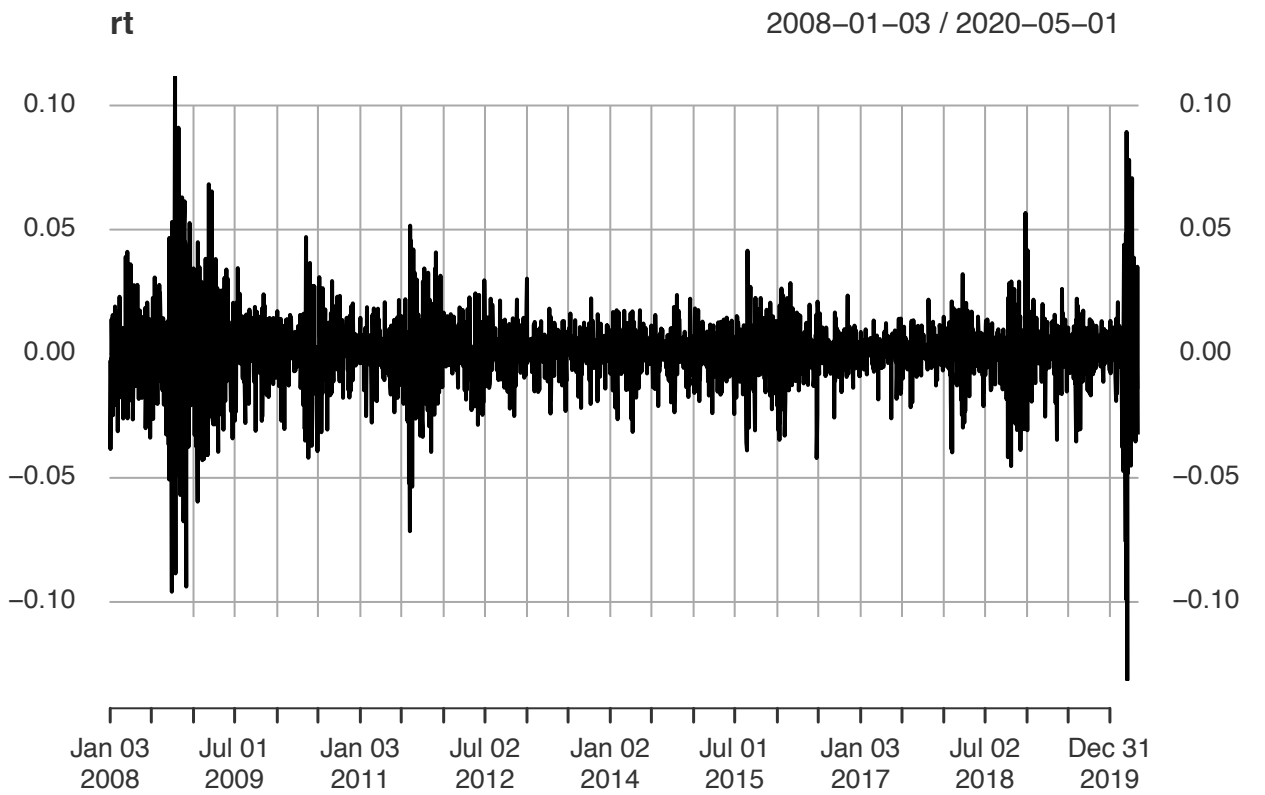
```
## [1] 0
```

```
rt<-diff(log(Pt))[-1]
summary<-to.weekly(Pt, name="Nasdaq_price")
```

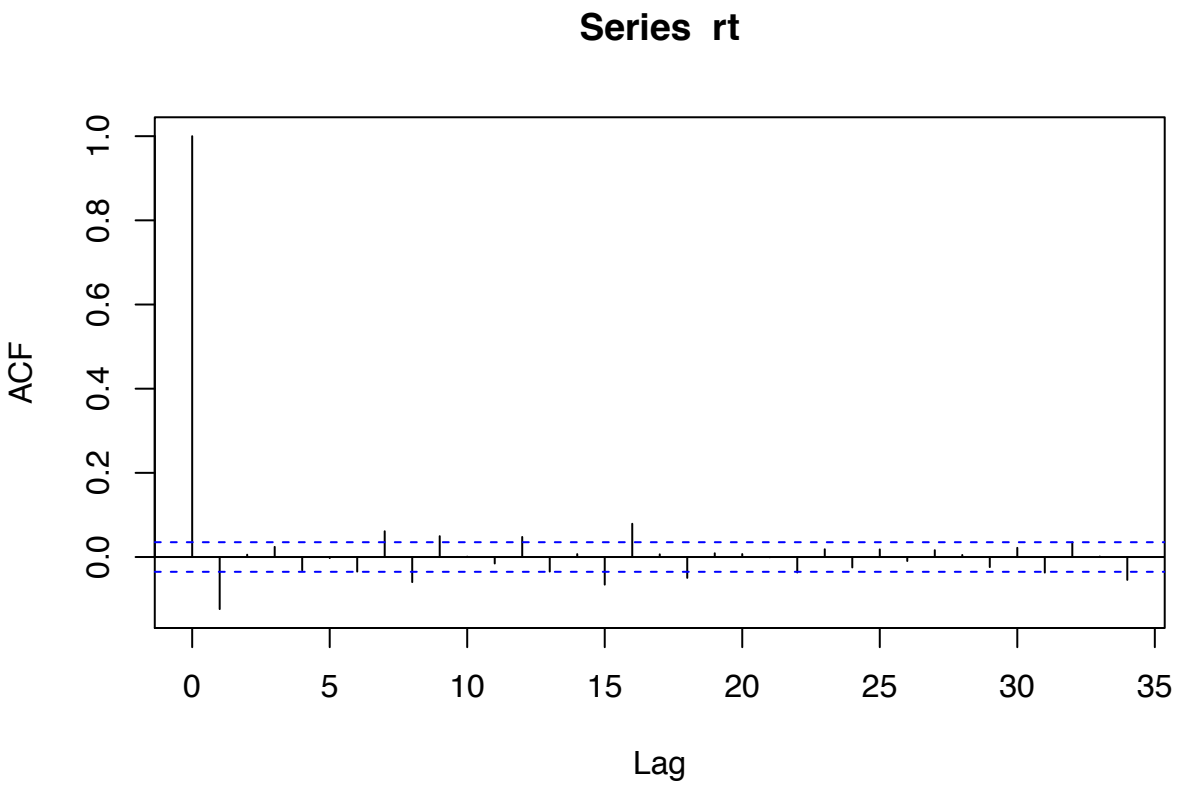
```
plot(Pt)
```


Pt
2008–01–02 / 2020–05–01
8000
6000
4000
2000
Jan 02 2008
Jul 01 2009
Jan 03 2011
Jul 02 2012
Jan 02 2014
Jul 01 2015
Jan 03 2017
Jul 02 2018
Dec 31 2019

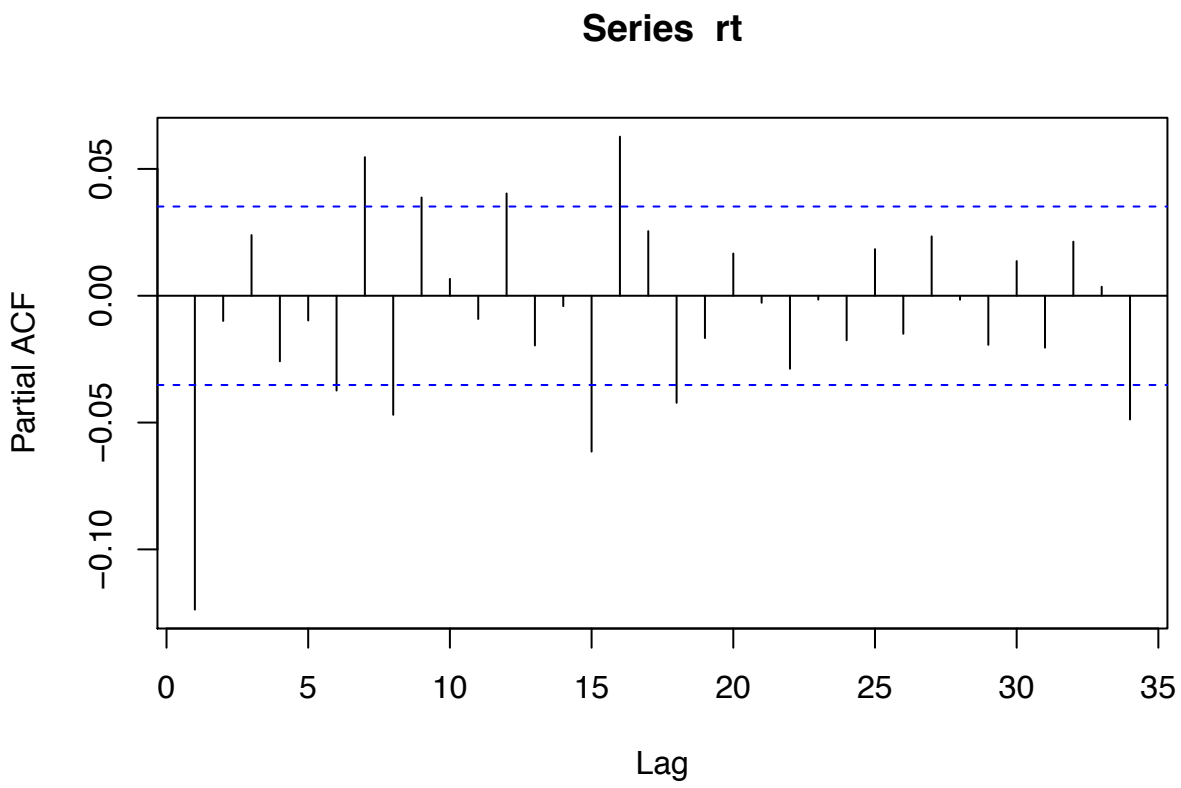
```
plot(rt)
```


rt
2008–01–03 / 2020–05–01
0.10
0.05
0.00
–0.05
–0.10
Jan 03 2008
Jul 01 2009
Jan 03 2011
Jul 02 2012
Jan 02 2014
Jul 01 2015
Jan 03 2017
Jul 02 2018
Dec 31 2019

```r
acf(rt,na.action = na.pass)
```




```r
pacf(rt,na.action = na.pass)
```



```
library(tibble)
library(tidyverse)
```

```
##   Attaching packages                                           tidyverse 1.3.0
```

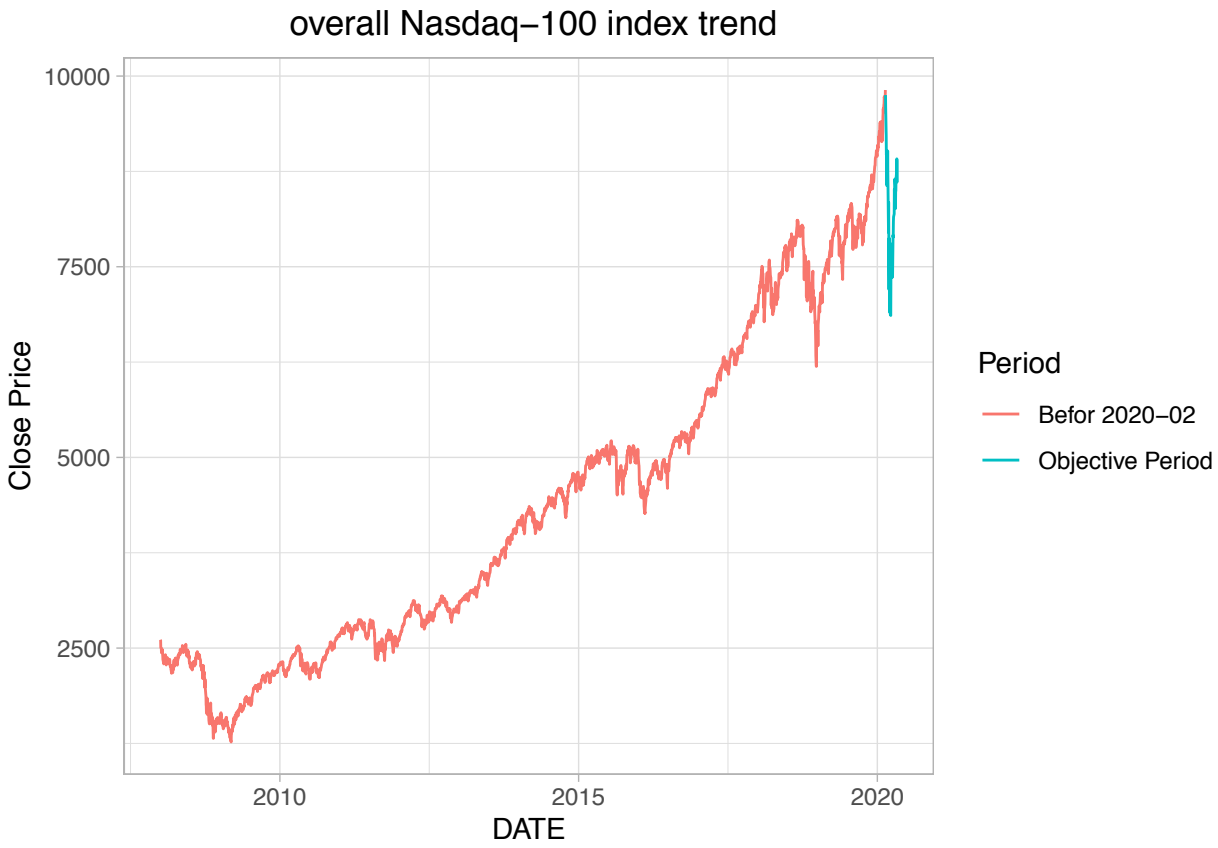
```
##   ggplot2 3.2.1      dplyr   0.8.3
##   tidyr   1.0.0      stringr 1.4.0
##   readr   1.3.1      forcats 0.4.0
##   purrr   0.3.3
```

```
##   Conflicts                                                tidyverse_conflicts()
## x dplyr::filter() masks stats::filter()
## x dplyr::first()  masks xts::first()
## x dplyr::lag()    masks stats::lag()
## x dplyr::last()   masks xts::last()
```

```
library(ggplot2)
d<-index(Pt)
all_data<-as_tibble(cbind(Pt,rt))%>%
mutate(date=d,Period=ifelse(date<='2020-2-19','Befor 2020-02','Objective Period'))
```

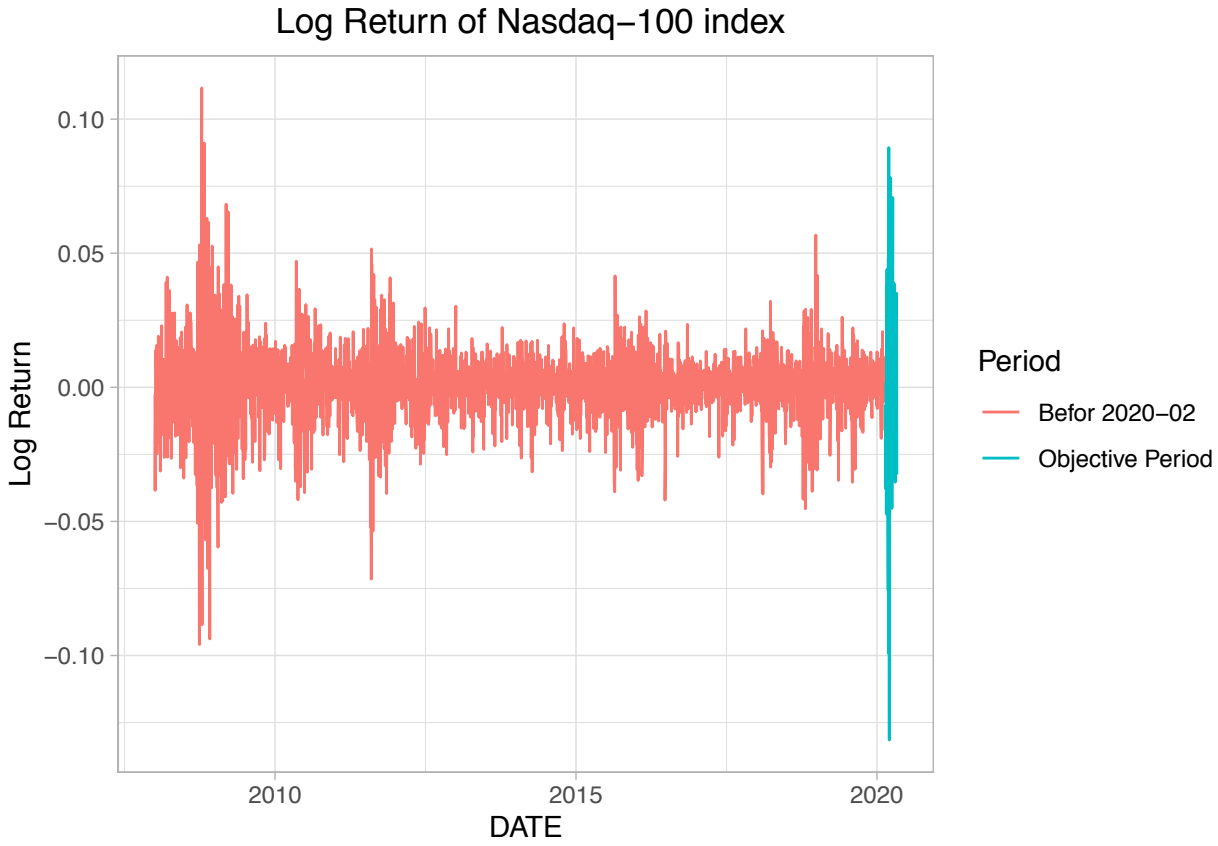
```
## Warning: Calling `as_tibble()` on a vector is discouraged, because the behavior is likely to change i
## This warning is displayed once per session.
```

```
ggplot(aes(date,Pt),data=all_data)+
 geom_line(aes(col=Period))+
  labs(title=" overall Nasdaq-100 index trend",
       x="DATE",'Close Price',y='Close Price')+
 theme_light()+
 theme(plot.title = element_text(hjust = 0.5), plot.subtitle = element_text(hjust = 0.5))
```

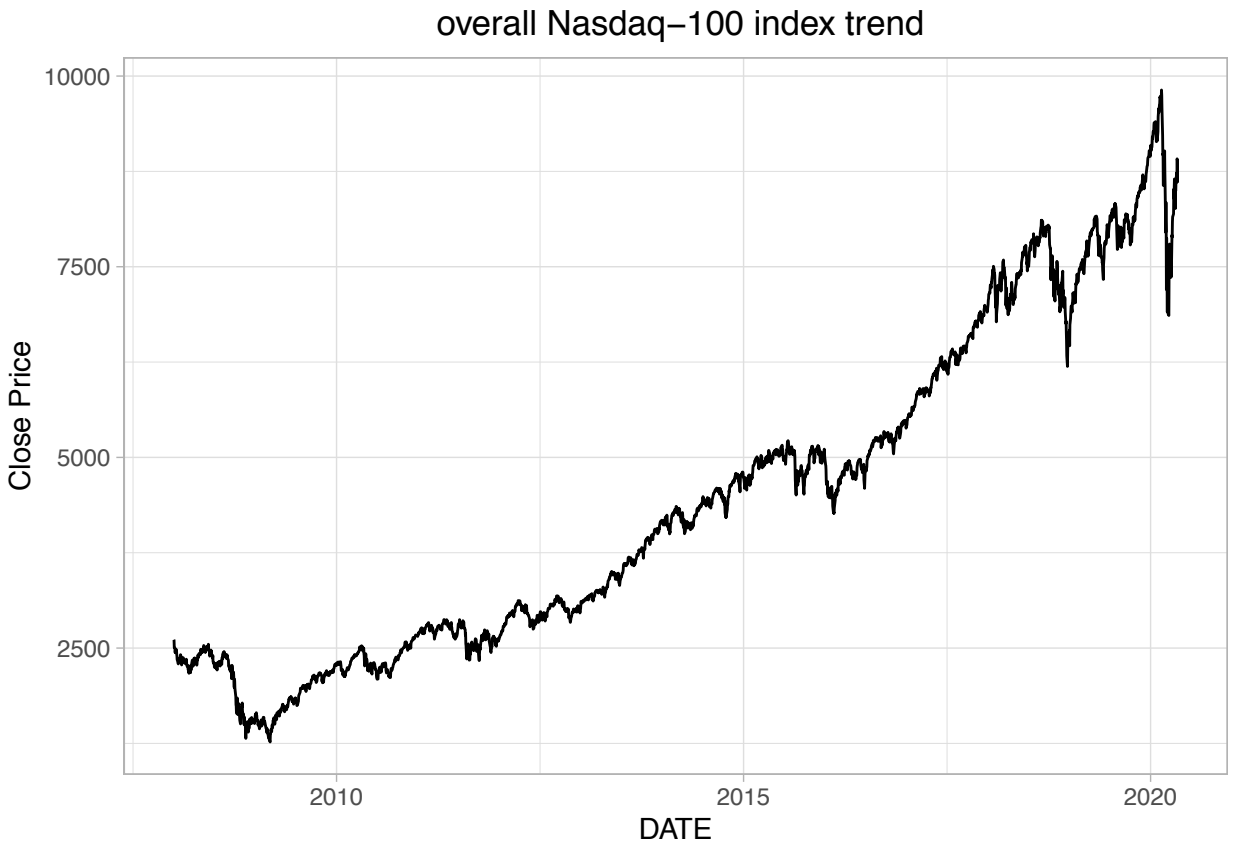



```
ggplot(aes(date,rt),data=all_data)+
 geom_line(aes(col=Period))+
   labs(title=" Log Return of Nasdaq-100 index",
        x="DATE",'Log Return',y='Log Return')+
 theme_light()+
theme(plot.title = element_text(hjust = 0.5), plot.subtitle = element_text(hjust = 0.5))
```

```
## Warning: Removed 1 rows containing missing values (geom_path).
```

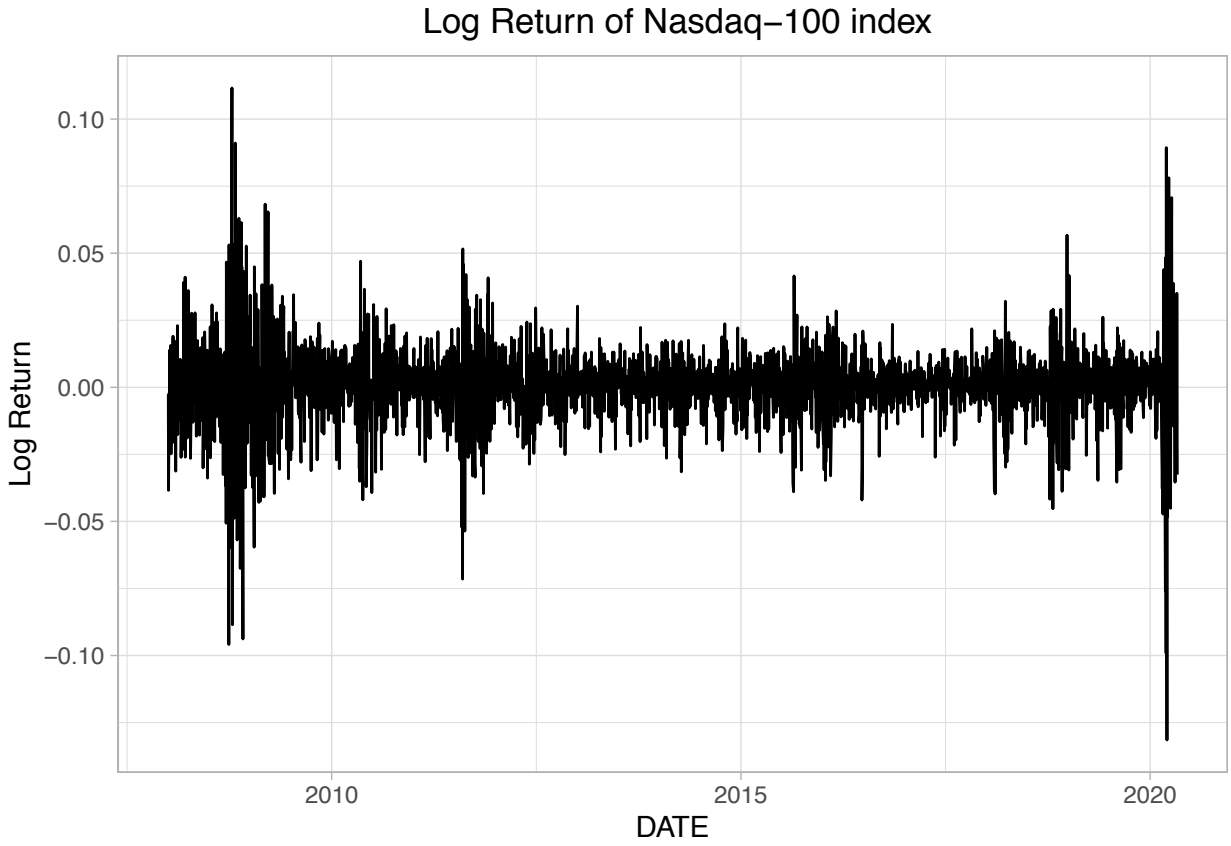



```
ggplot(aes(date,Pt),data=all_data)+
 geom_line()+
  labs(title=" overall Nasdaq-100 index trend",
       x="DATE",'Close Price',y='Close Price')+
 theme_light()+
 theme(plot.title = element_text(hjust = 0.5), plot.subtitle = element_text(hjust = 0.5))
```




```
ggplot(aes(date,rt),data=all_data)+
 geom_line()+
   labs(title=" Log Return of Nasdaq-100 index",
        x="DATE",'Log Return',y='Log Return')+
 theme_light()+
theme(plot.title = element_text(hjust = 0.5), plot.subtitle = element_text(hjust = 0.5))
```

```
## Warning: Removed 1 rows containing missing values (geom_path).
```




```
library(tseries)
adf.test(rt)
```

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  rt
## Dickey-Fuller = -15.251, Lag order = 14, p-value = 0.01
## alternative hypothesis: stationary
```

```
Pt[which.max(Pt)]
```

```
##                [,1]
## 2020-02-19 9817.18
```

```
rt_subset<-rt[index(rt)<='2020-02-19']
rt_test<-rt[index(rt)>'2020-02-19']
```

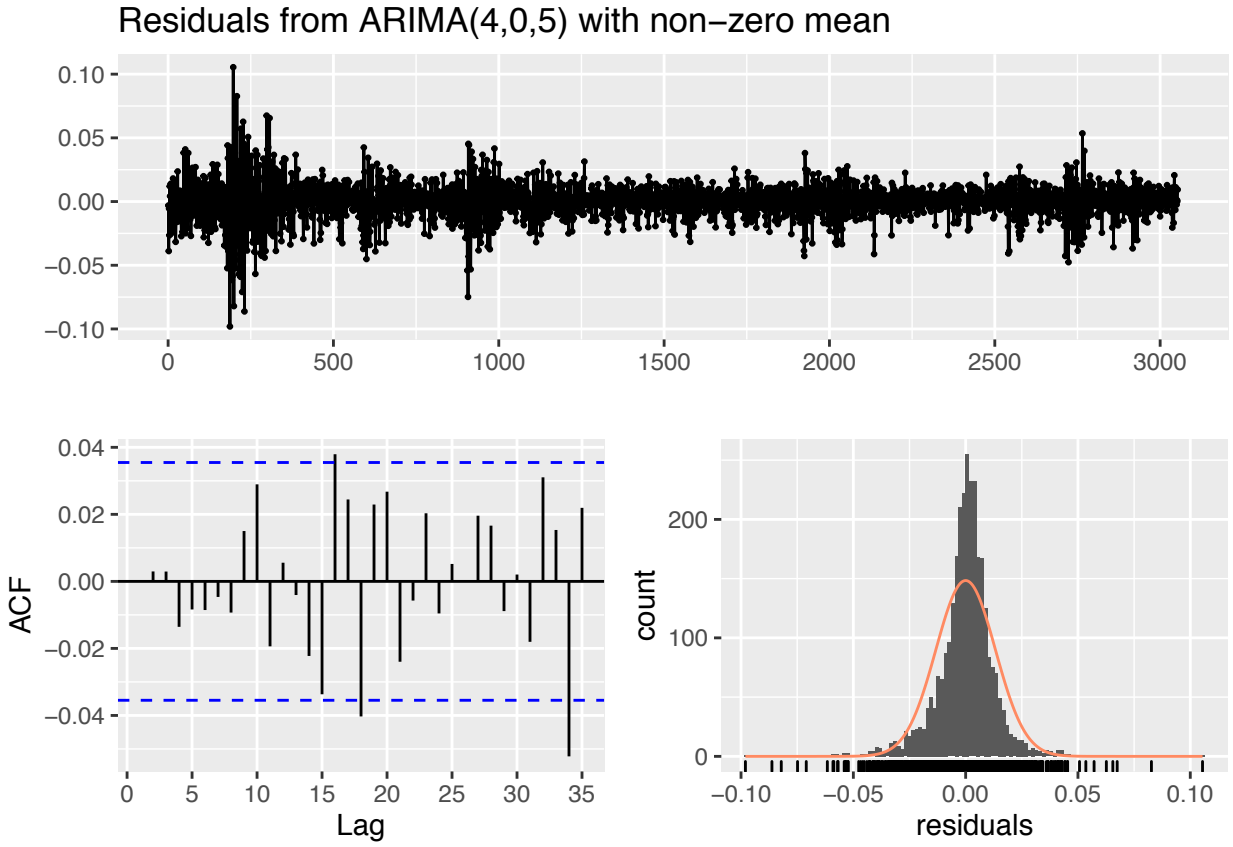
```
AIC_p_q_select<-matrix(NA,nrow=5,ncol=5)
for(p in 0:4){
  for(q in 0:4){
    model_tmp<-arima(rt_subset, order = c(p,0,q),include.mean = T)
    AIC_p_q_select[p+1,q+1]<-model_tmp$aic
  }
}
```

```
## Warning in arima(rt_subset, order = c(p, 0, q), include.mean = T): possible
## convergence problem: optim gave code = 1
```

```
which(AIC_p_q_select == min(AIC_p_q_select),arr.ind=T)-1
```

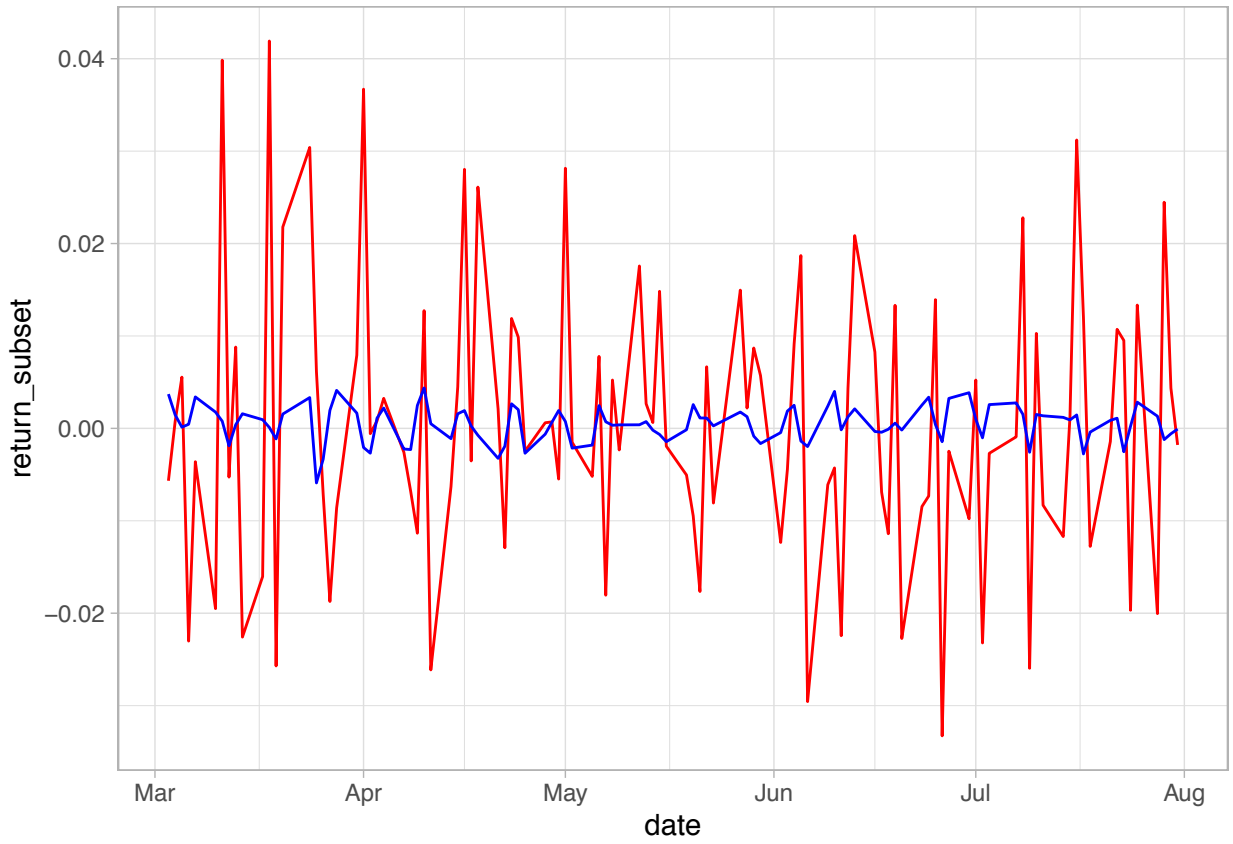
```
##      row col
## [1,]   4   3
```

```
Rt_subset.train<-arima(rt_subset, order = c(4,0,5),include.mean=T)
checkresiduals(Rt_subset.train)
```

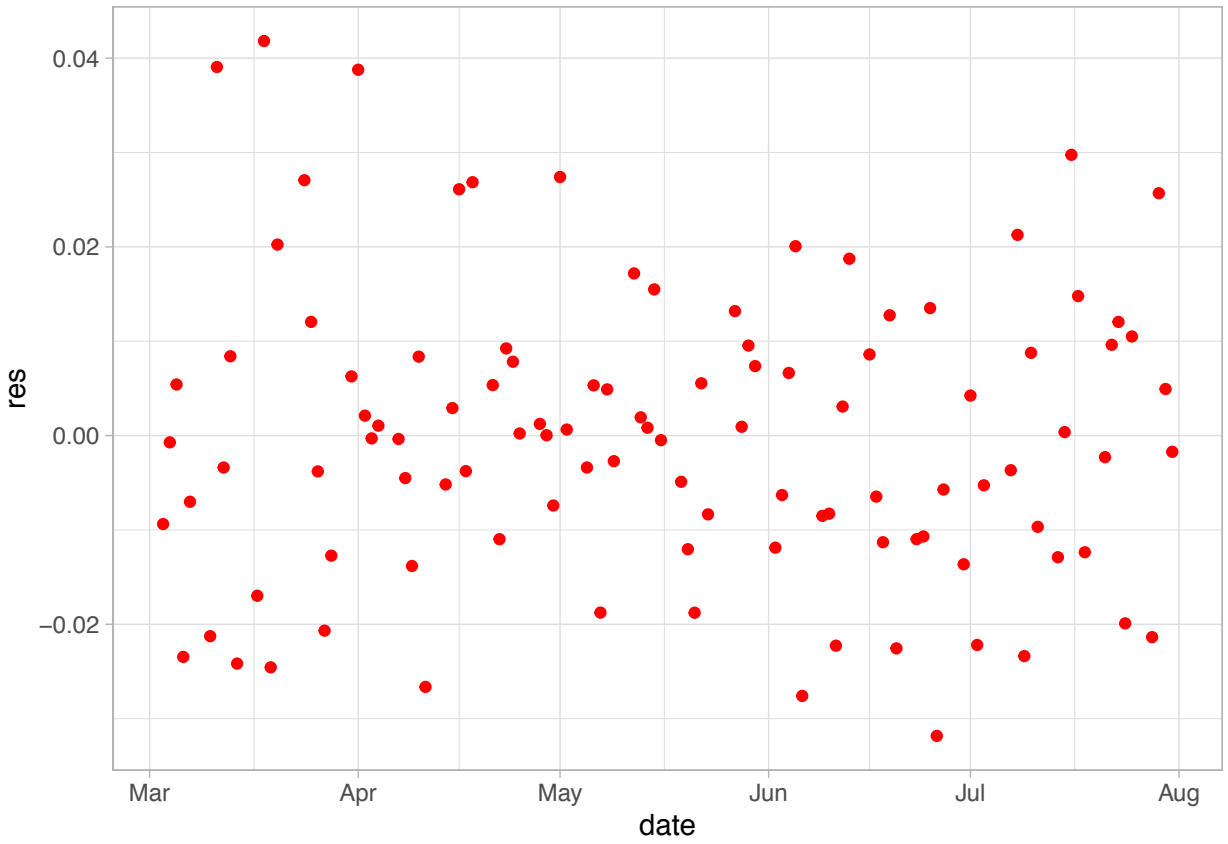



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(4,0,5) with non-zero mean
## Q* = 5.9434, df = 3, p-value = 0.1144
##
## Model df: 10.   Total lags used: 13
```

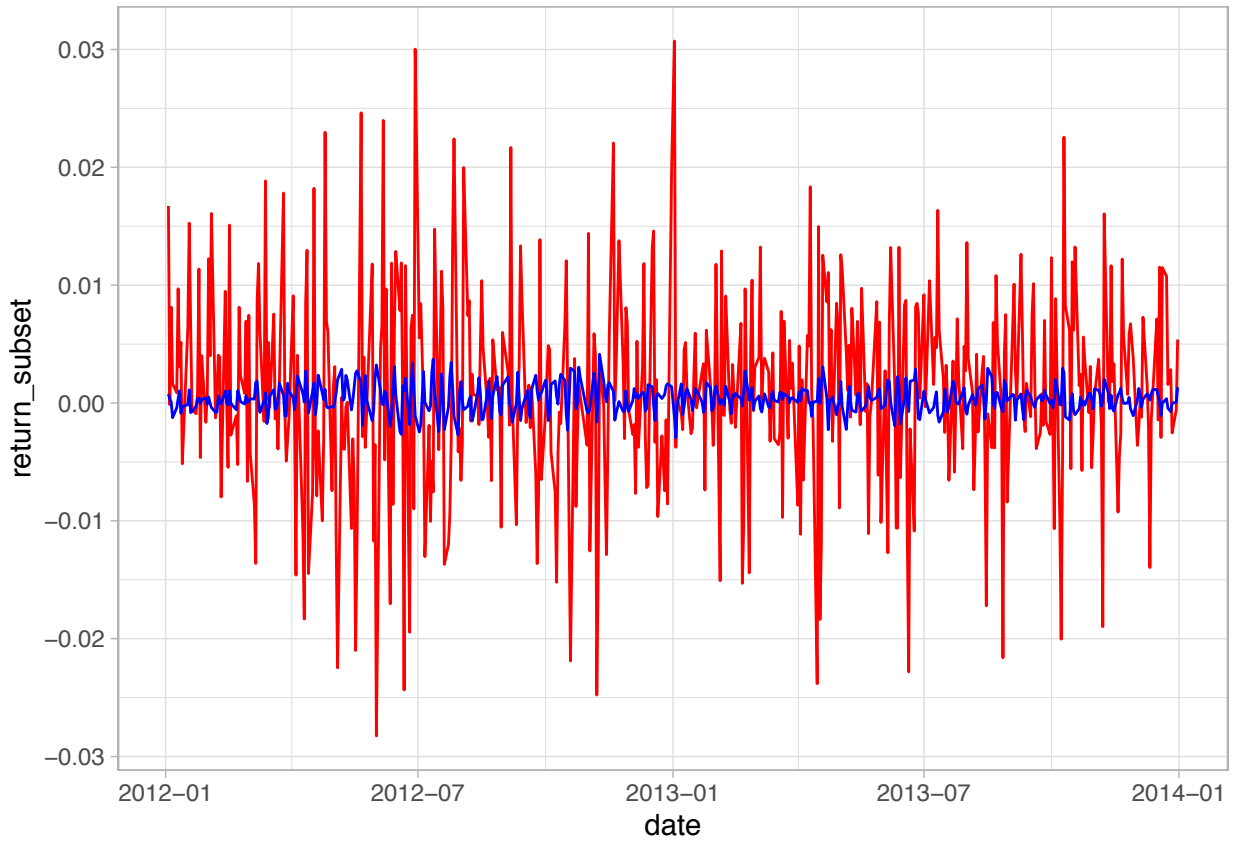
```
r_train_fit<-xts(fitted(Rt_subset.train),order.by=index(rt_subset))
d<-as.Date(index(r_train_fit))
return_train_fit<-exp(r_train_fit)-1
return_subset<-exp(rt_subset)-1

ARMA_summary_train<-as_tibble(cbind(return_subset,return_train_fit))%>%
 mutate(date=d,res=return_subset-return_train_fit)

ggplot(aes(date,return_subset),data=ARMA_summary_train[ARMA_summary_train$date>'2008-03-01'&ARMA_summary
 geom_line(col='red')+
 geom_line(aes(date,return_train_fit),col='blue')+
 theme_light()
```



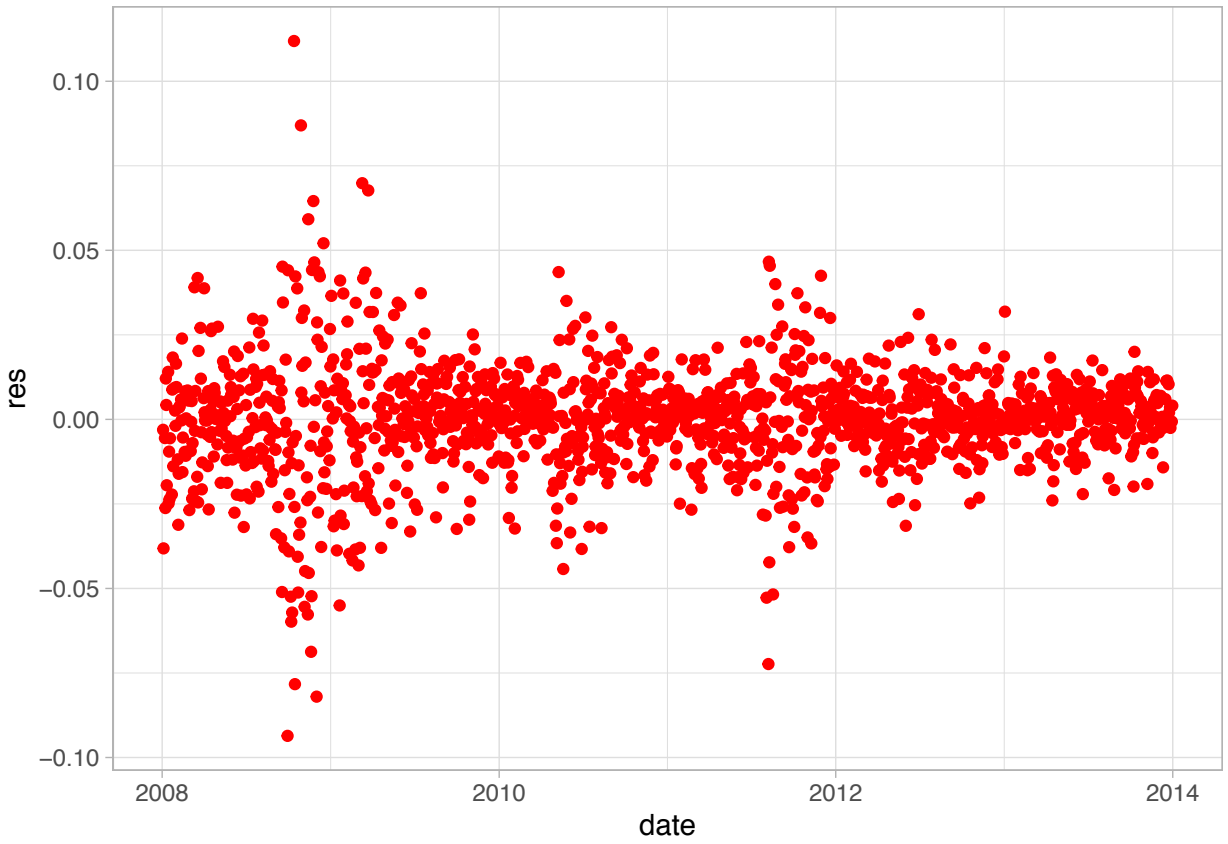

```
ggplot(aes(date,res),data=ARMA_summary_train[ARMA_summary_train$date>'2008-03-01'&ARMA_summary_train$da
 geom_point(col='red')+
 theme_light()
```




```
ggplot(aes(date,return_subset),data=ARMA_summary_train[ARMA_summary_train$date>'2012-01-01'&ARMA_summary
geom_line(col='red')+
geom_line(aes(date,return_train_fit),col='blue')+
theme_light()
```




```
ggplot(aes(date,res),data=ARMA_summary_train[ARMA_summary_train$date>'2002-01-02'&ARMA_summary_train$da
 geom_point(col='red')+
 theme_light()
```

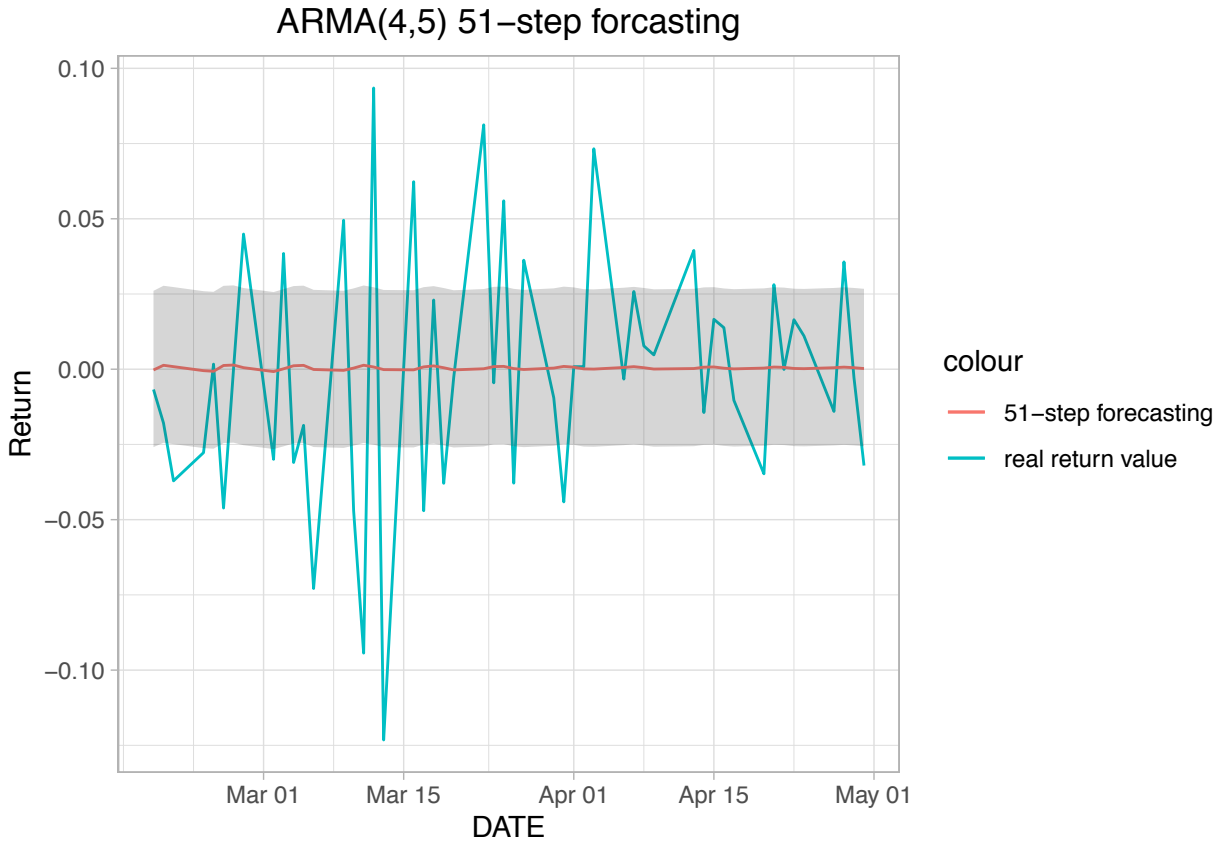



```
library(tidyverse)
library(tibble)
library(ggplot2)
d<-as.Date(index(Pt[(length(Pt)-51):(length(Pt)-1)]))
for_arma<-forecast(Rt_subset.train,h=51)
return_predict<-exp(for_arma$mean)-1
return_predict_high_bound<-exp(for_arma$upper[,2])-1
return_predict_lower_bound<-exp(for_arma$lower[,2])-1
return_test<-exp(rt_test)-1


ARMA_summary_predict<-as_tibble(cbind(return_test,return_predict))%>%
 mutate(date=d,res=return_test-return_predict,lower=return_predict_lower_bound,upper=return_predict_high
```

```
## Warning: `as_tibble.matrix()` requires a matrix with column names or a `.name_repair` argument. Using
## This warning is displayed once per session.
```

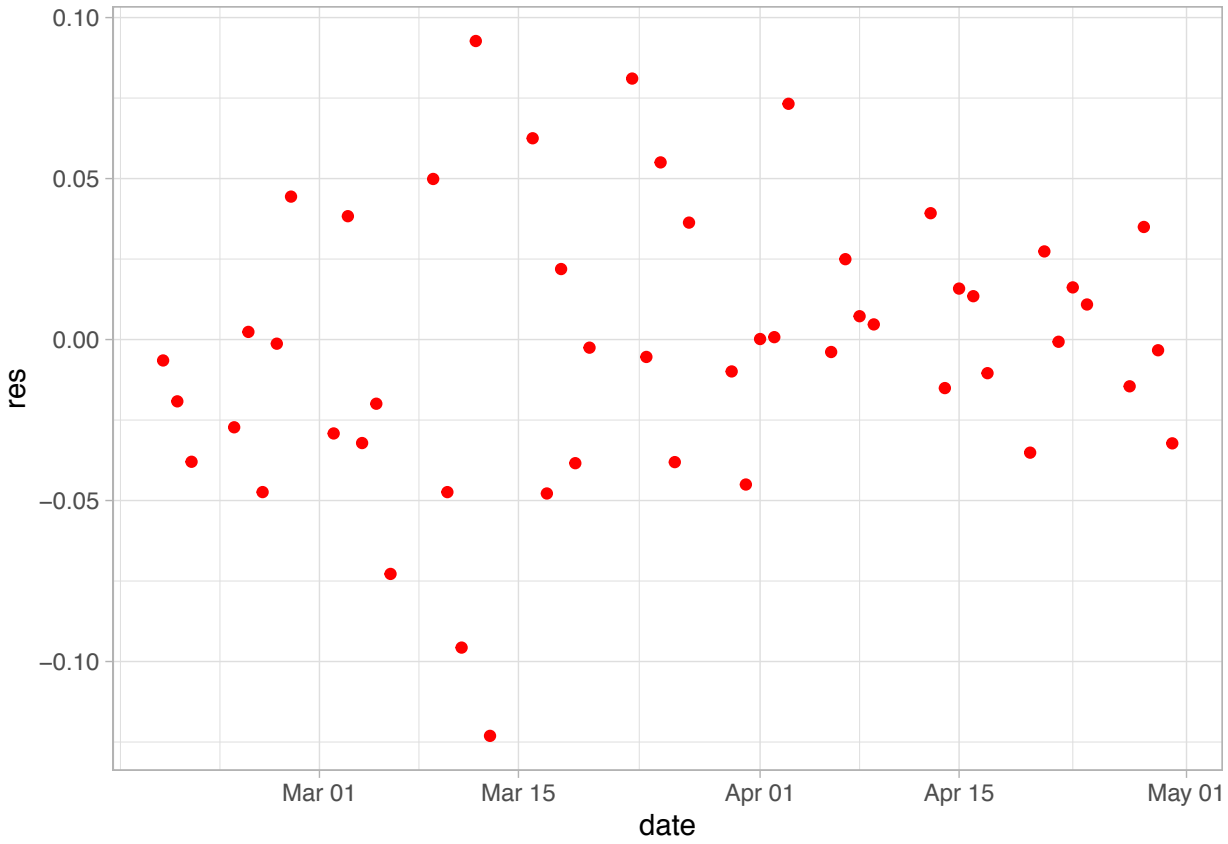
```
ggplot(aes(date,return_predict),data=ARMA_summary_predict)+
  geom_line(aes(date,return_test,color='real return value'))+
  geom_line(aes(col='51-step forecasting'))+
 geom_ribbon(aes(ymin=lower, ymax=upper), alpha=0.2)+
  labs(title="ARMA(4,5) 51-step forcasting",
  x="DATE",y='Return')+
 theme_light()+
theme(plot.title = element_text(hjust = 0.5), plot.subtitle = element_text(hjust = 0.5))
```

```
## Don't know how to automatically pick scale for object of type xts/zoo. Defaulting to continuous.
```




```
ggplot(aes(date,res),data=ARMA_summary_predict)+
 geom_point(col='red')+
 theme_light()
```

```
## Don't know how to automatically pick scale for object of type xts/zoo. Defaulting to continuous.
```




```
library(rugarch)
```

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

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

```
## The following object is masked from 'package:purrr':
##
##     reduce
```

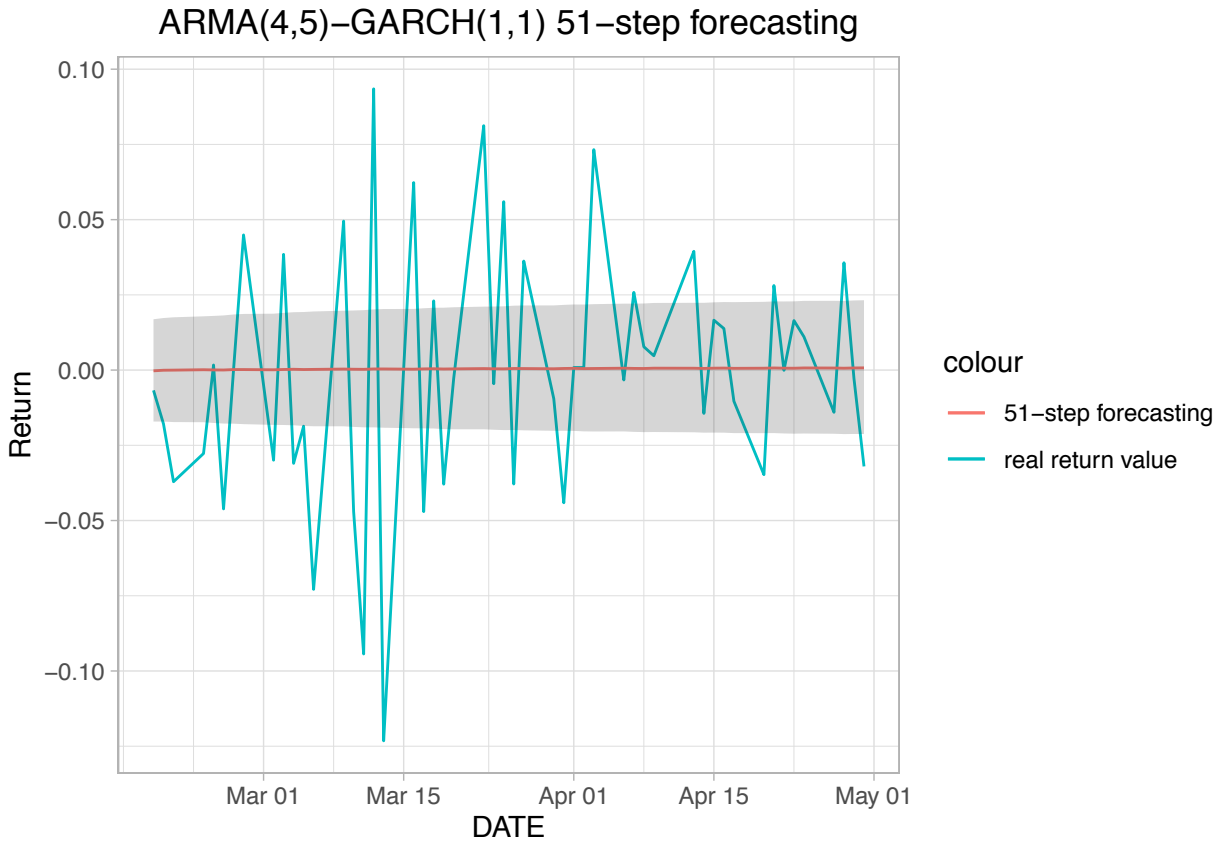
```
## The following object is masked from 'package:stats':
##
##     sigma
```

```
model.garch = ugarchspec(mean.model=list(armaOrder=c(4,5),include.mean=T, archm = FALSE, archpow = 1, ar
variance.model=list(model='sGARCH',garchOrder=c(1,1), submodel = NULL, external.regressors = NULL, varia
distribution.model = "norm" )
model.garch.fit = ugarchfit(data=rt, spec=model.garch, out.sample=51, solver = 'solnp')
forc=ugarchforecast(model.garch.fit,n.ahead=51)
test_prec_g<-forc@forecast$seriesFor
for_low<-test_prec_g-1.96*forc@forecast$sigmaFor
for_up<-test_prec_g+1.96*forc@forecast$sigmaFor
```

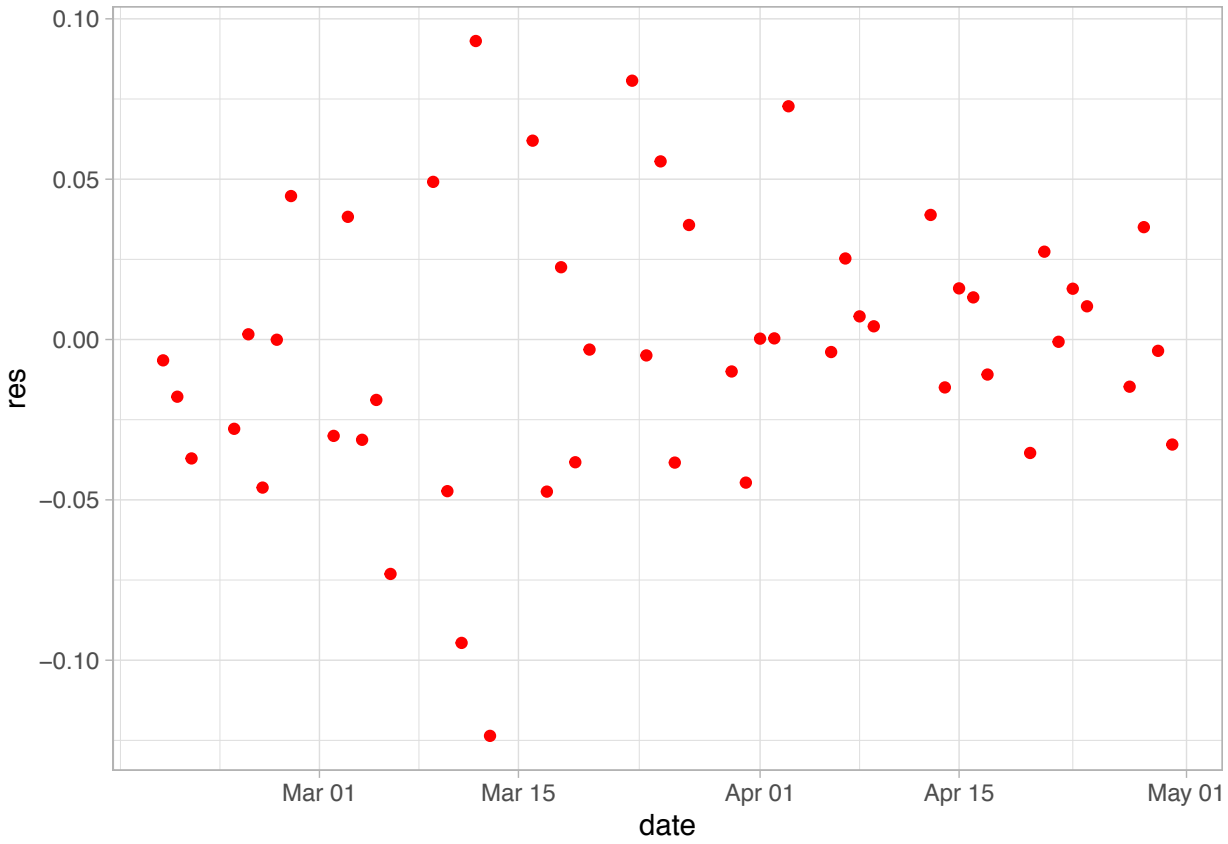
```
d<-as.Date(index(Pt[(length(Pt)-51):(length(Pt)-1)]))

return_predict_g<-exp(test_prec_g)-1
return_test<-exp(rt_test)-1
return_test_upper<-exp(for_low)-1
```

```
return_test_lower<-exp(for_up)-1
```

```
Garch_summary_predict<-as_tibble(cbind(return_test,return_predict_g))%>%
 mutate(date=d,res=return_test-return_predict_g,upper=return_test_upper,lower=return_test_lower)

ggplot(aes(date,return_test),data=Garch_summary_predict)+
 geom_line(aes(col='real return value'))+
 geom_line(aes(date,return_predict_g,col='51-step forecasting'))+
 geom_ribbon(aes(ymin=lower, ymax=upper), alpha=0.2)+
 labs(title="ARMA(4,5)-GARCH(1,1) 51-step forecasting",
  x="DATE",y='Return')+
 theme_light()+
  theme(plot.title = element_text(hjust = 0.5), plot.subtitle = element_text(hjust = 0.5))
```

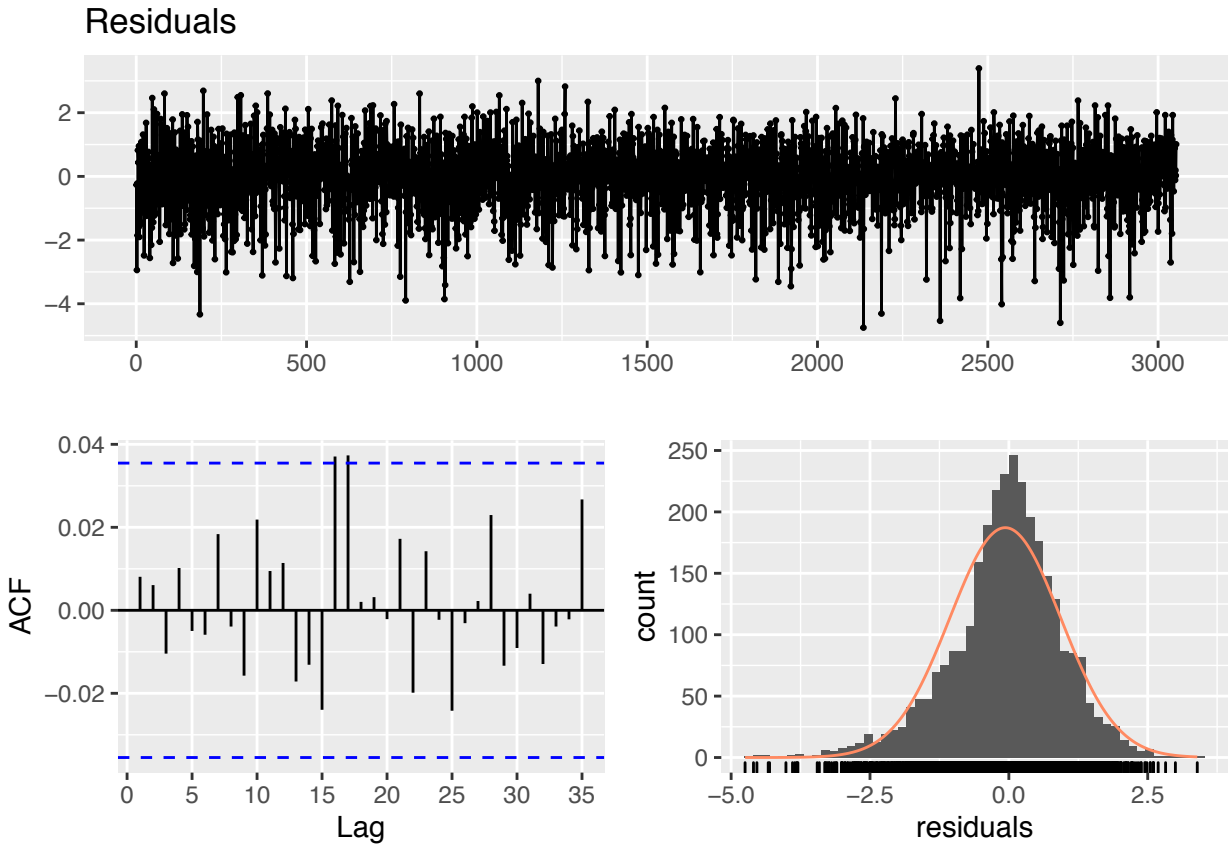



```
ggplot(aes(date,res),data=Garch_summary_predict)+
 geom_point(col='red')+
 theme_light()
```




```
std_residual_Garch<-model.garch.fit@fit$residuals/model.garch.fit@fit$sigma
checkresiduals(std_residual_Garch)
```

```
## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```




```
model.garch.fit@fit$coef
```

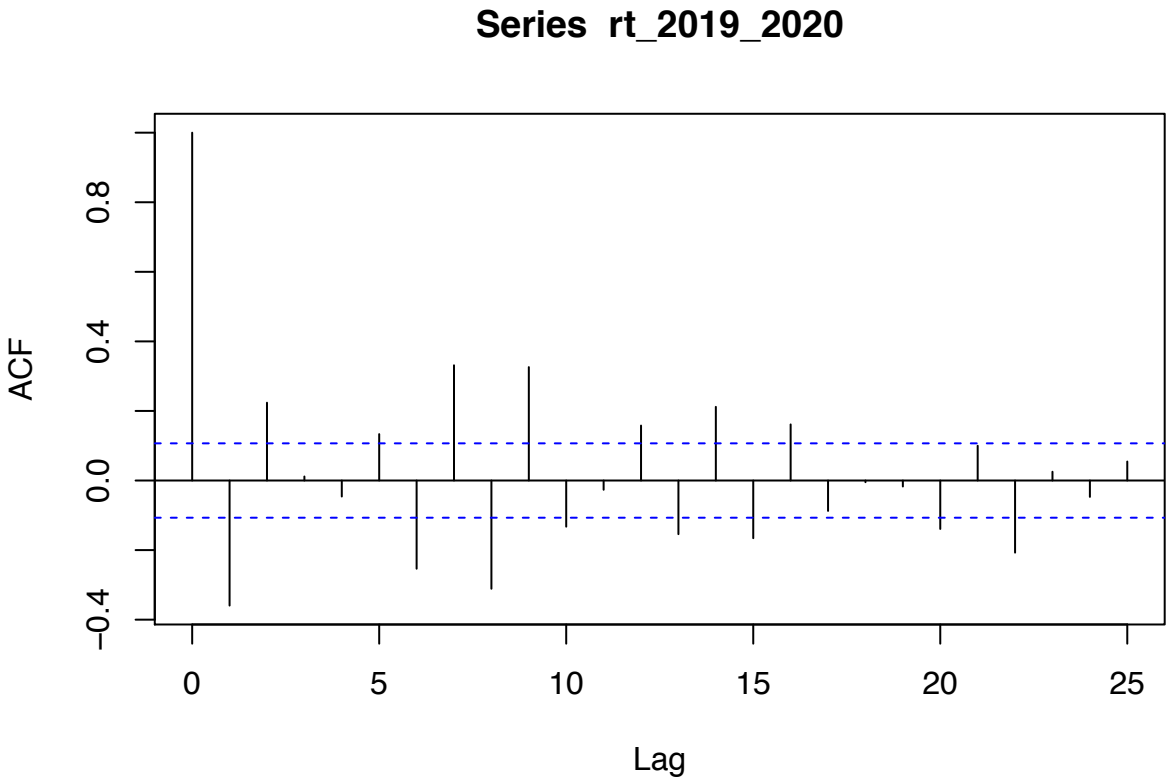
```
##            mu           ar1           ar2           ar3           ar4
##  8.753221e-04 -1.038117e-01 -2.323414e-01  7.866501e-01  4.298095e-01
##           ma1           ma2           ma3           ma4           ma5
##  5.815240e-02  2.099079e-01 -8.096319e-01 -4.091069e-01  1.271138e-02
##         omega        alpha1         beta1
##  3.736652e-06  1.228615e-01  8.521216e-01
```

```
Box.test(model.garch.fit@fit$residuals/model.garch.fit@fit$sigma, lag = 1, type = 'Ljung-Box', fitdf = 0
```

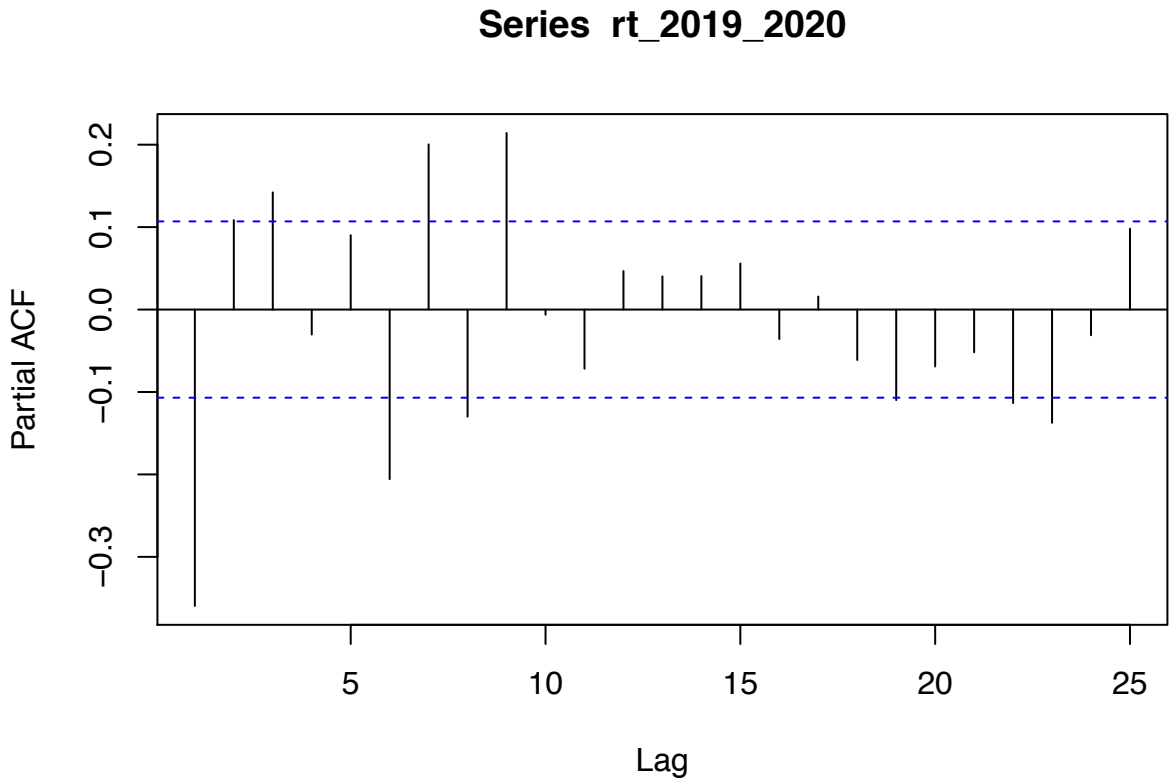
```
##
##  Box-Ljung test
##
## data:  model.garch.fit@fit$residuals/model.garch.fit@fit$sigma
## X-squared = 0.19952, df = 1, p-value = 0.6551
```

```
rt_2019_2020<-rt[index(rt)>='2019-01-01']
rt_2019_2020_train<-rt_2019_2020[1:(length(rt_2019_2020)-20)]
rt_2019_2020_test<-rt_2019_2020[(length(rt_2019_2020)-19):length(rt_2019_2020)]
```

```
acf(rt_2019_2020)
```




```
pacf(rt_2019_2020)
```




```
AIC_p_q_select<-matrix(NA,nrow=6,ncol=6)
for(p in 0:5){
```

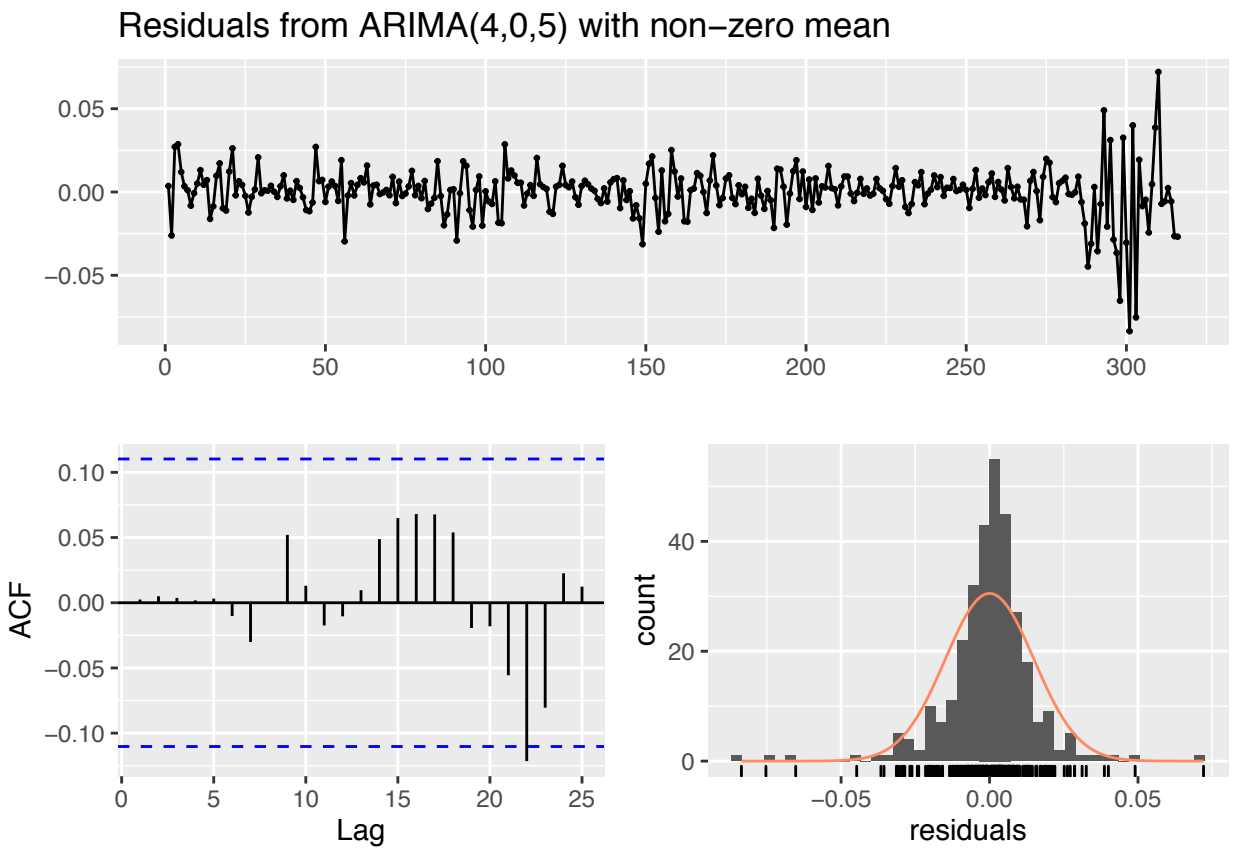
```
  for(q in 0:5){
    model_tmp<-arima(rt_2019_2020_train, order = c(p,0,q),include.mean = T)
    AIC_p_q_select[p+1,q+1]<-model_tmp$aic
  }
}
```

```
## Warning in arima(rt_2019_2020_train, order = c(p, 0, q), include.mean = T):
## possible convergence problem: optim gave code = 1
```

```
which(AIC_p_q_select == min(AIC_p_q_select),arr.ind=T)-1
```

```
##      row col
## [1,]   4   5
```

```
Rt_2019_2020.train<-arima(rt_2019_2020_train, order = c(4,0,5),include.mean=T)
checkresiduals(Rt_2019_2020.train)
```



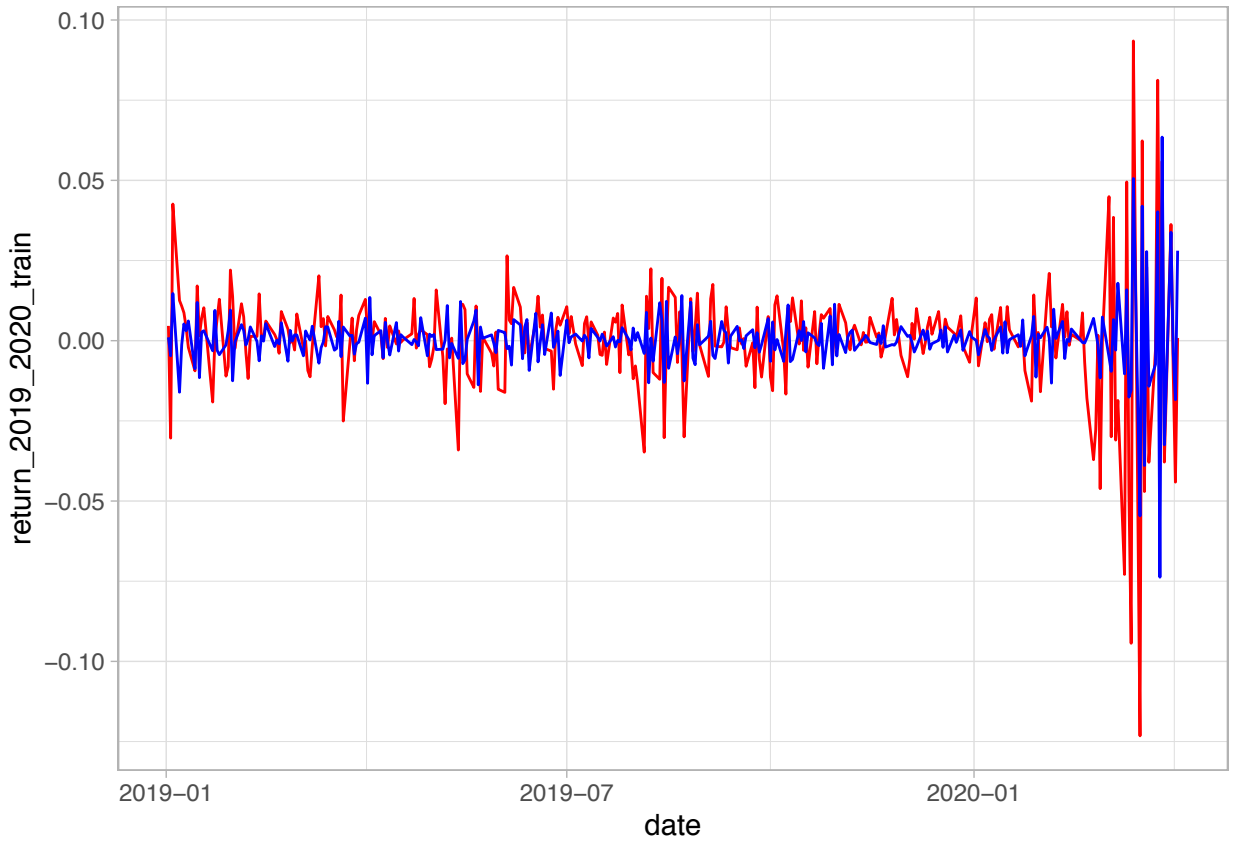

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(4,0,5) with non-zero mean
## Q* = 1.4581, df = 3, p-value = 0.692
##
## Model df: 10.   Total lags used: 13
```

```
Rt_2019_2020.train$coef
```

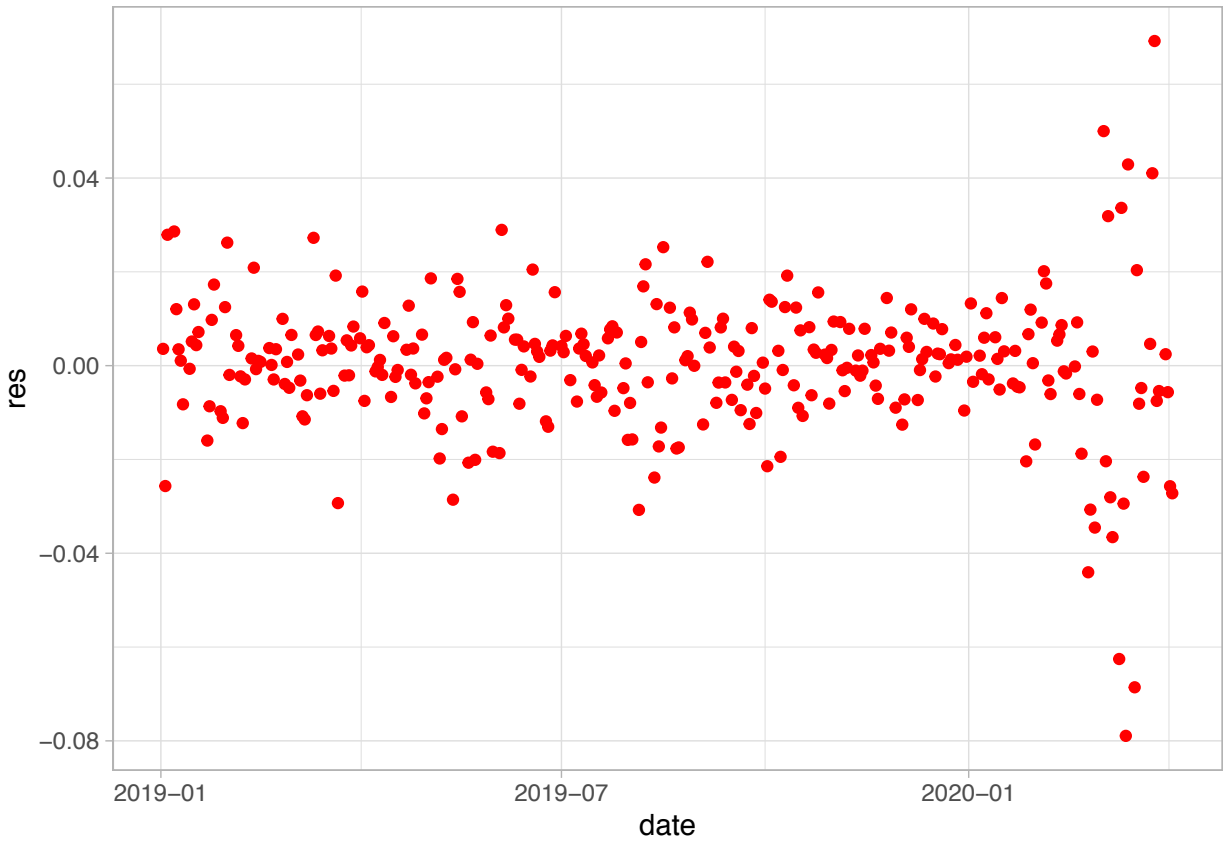
```
##          ar1          ar2          ar3          ar4          ma1          ma2
## -2.228668330 -2.151161733 -1.001687707 -0.216598464  2.024932610  1.868255777
```

```
##          ma3          ma4          ma5    intercept
##  1.034392676  0.606894228  0.351497907  0.000258584
```

```
r_train_2019_2020_fit<-xts(fitted(Rt_2019_2020.train),order.by=index(rt_2019_2020_train))
d<-as.Date(index(r_train_2019_2020_fit))
return_2019_2020_train_fit<-exp(r_train_2019_2020_fit)-1
return_2019_2020_train<-exp(rt_2019_2020_train)-1

ARMA_2019_2020_summary_train<-as_tibble(cbind(return_2019_2020_train,return_2019_2020_train_fit))%>%
 mutate(date=d,res=return_2019_2020_train-return_2019_2020_train_fit)

ggplot(aes(date,return_2019_2020_train),data=ARMA_2019_2020_summary_train)+
 geom_line(col='red')+
 geom_line(aes(date,return_2019_2020_train_fit),col='blue')+
 theme_light()
```

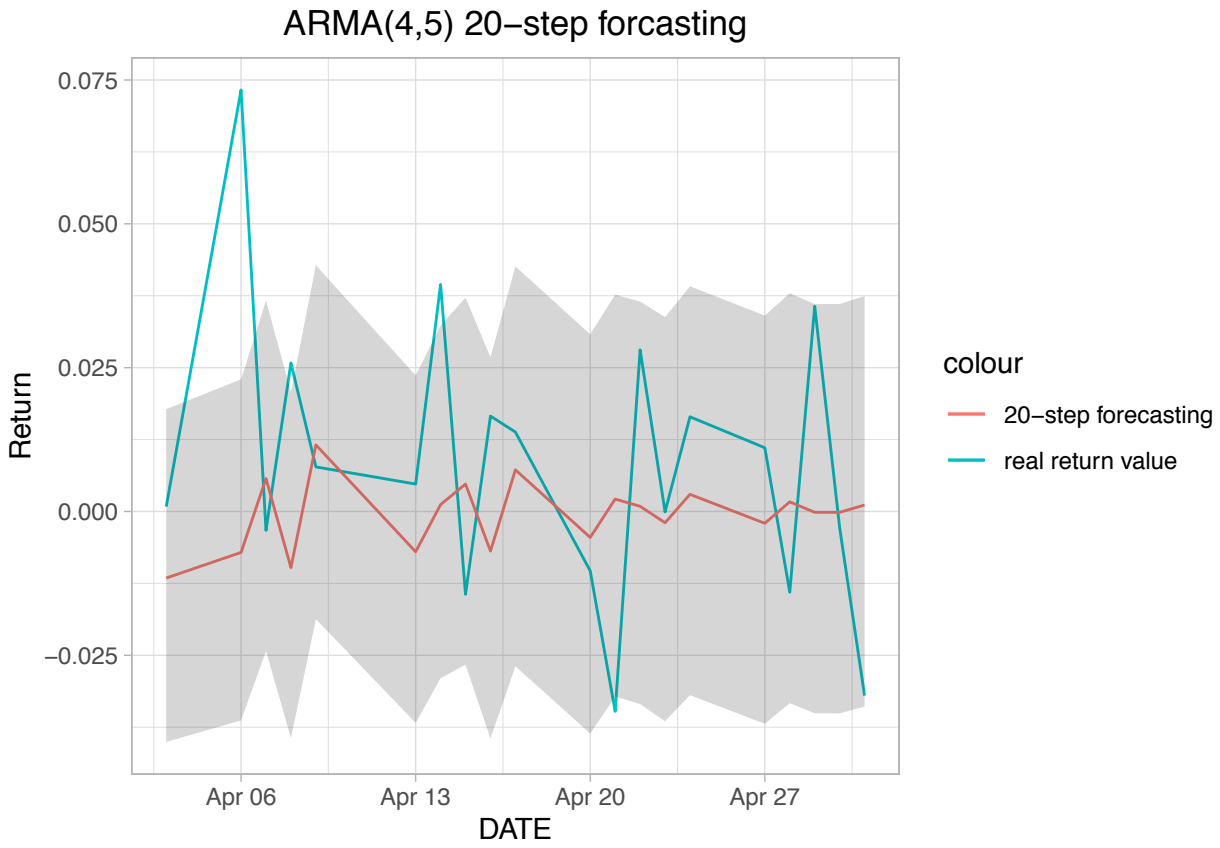



```
ggplot(aes(date,res),data=ARMA_2019_2020_summary_train)+
 geom_point(col='red')+
 theme_light()
```



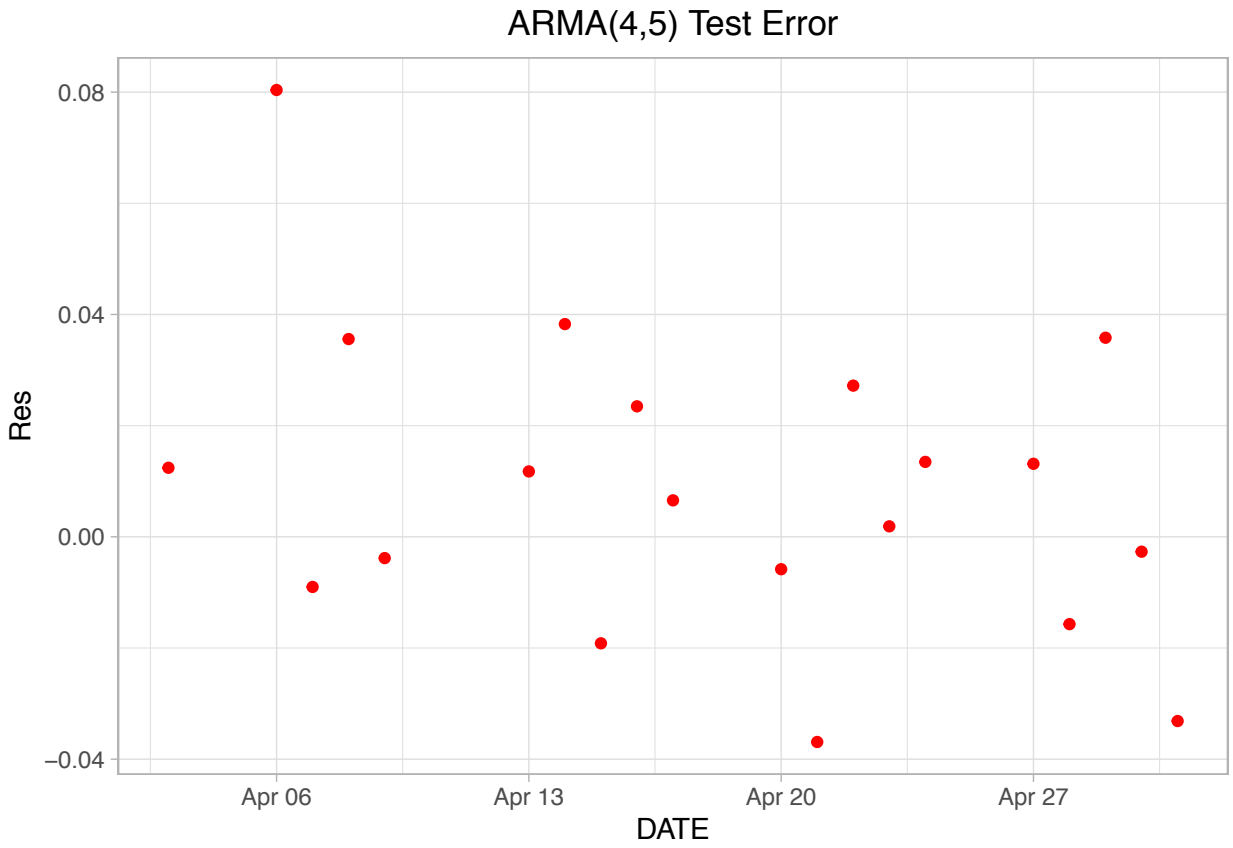

```
d<-as.Date(index(rt_2019_2020_test))
for_arma_2019_2020<-forecast(Rt_2019_2020.train,h=20)
test_2019_2020_predict<-for_arma_2019_2020$mean
return_2019_2020_predict<-exp(test_2019_2020_predict)-1
return_2019_2020_predict_high_bound<-exp(for_arma_2019_2020$upper[,2])-1
return_2019_2020_predict_lower_bound<-exp(for_arma_2019_2020$lower[,2])-1
return_test_2019_2020<-exp(rt_2019_2020_test)-1

ARMA_summary_2019_2020_predict<-as_tibble(cbind(return_test_2019_2020,return_2019_2020_predict,upper=re
 mutate(date=d,res=return_test_2019_2020-return_2019_2020_predict)

ggplot(aes(date,return_test_2019_2020),data=ARMA_summary_2019_2020_predict)+
 geom_line(aes(col='real return value'))+
 geom_line(aes(date,return_2019_2020_predict,col='20-step forecasting'))+
geom_ribbon(aes(ymin=lower, ymax=upper), alpha=0.2)+
labs(title="ARMA(4,5) 20-step forcasting",
  x="DATE",y='Return')+
 theme_light()+
 theme(plot.title = element_text(hjust = 0.5), plot.subtitle = element_text(hjust = 0.5))
```

```
## Don't know how to automatically pick scale for object of type xts/zoo. Defaulting to continuous.
```




```
ggplot(aes(date,res),data=ARMA_summary_2019_2020_predict)+
 geom_point(col='red')+
 labs(title="ARMA(4,5) Test Error",
 x="DATE",y='Res')+
 theme_light()+
 theme(plot.title = element_text(hjust = 0.5), plot.subtitle = element_text(hjust = 0.5))
```

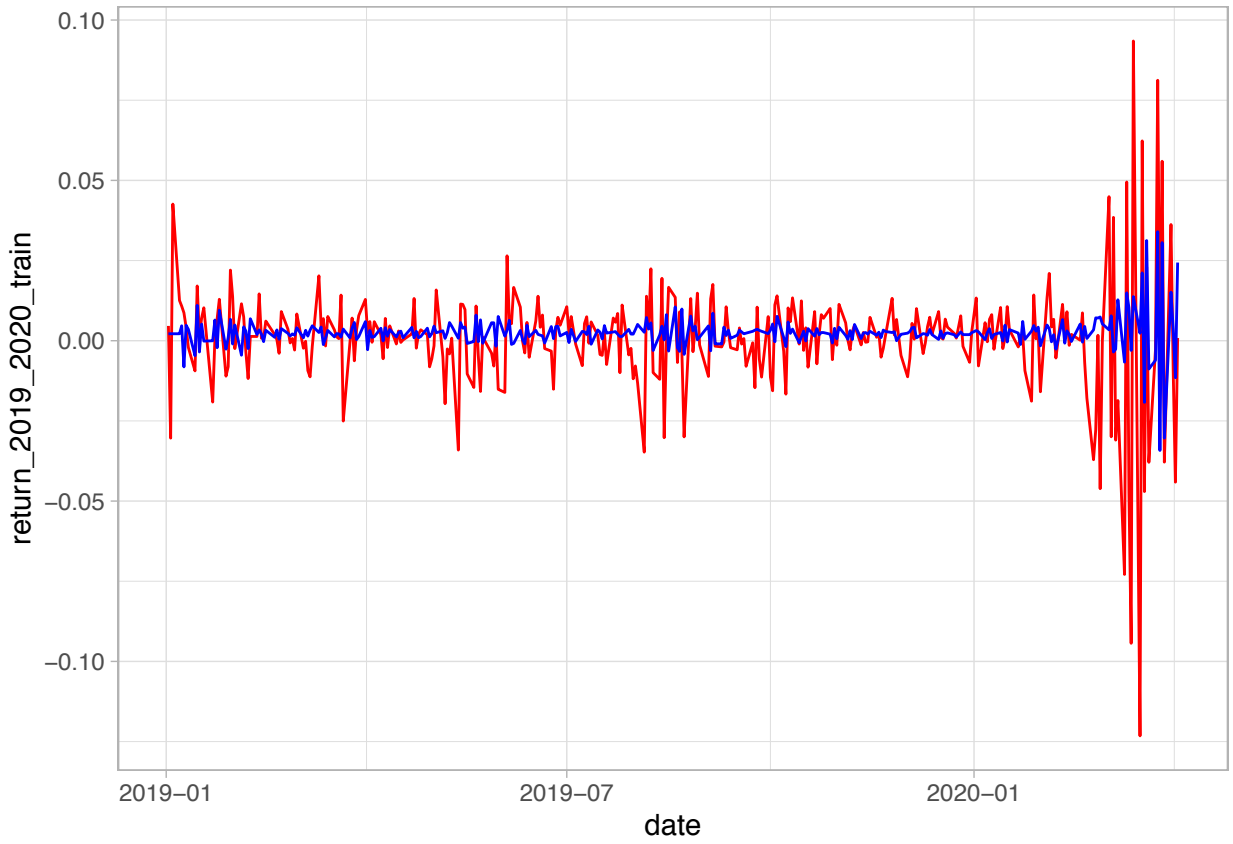
```
## Don't know how to automatically pick scale for object of type xts/zoo. Defaulting to continuous.
```



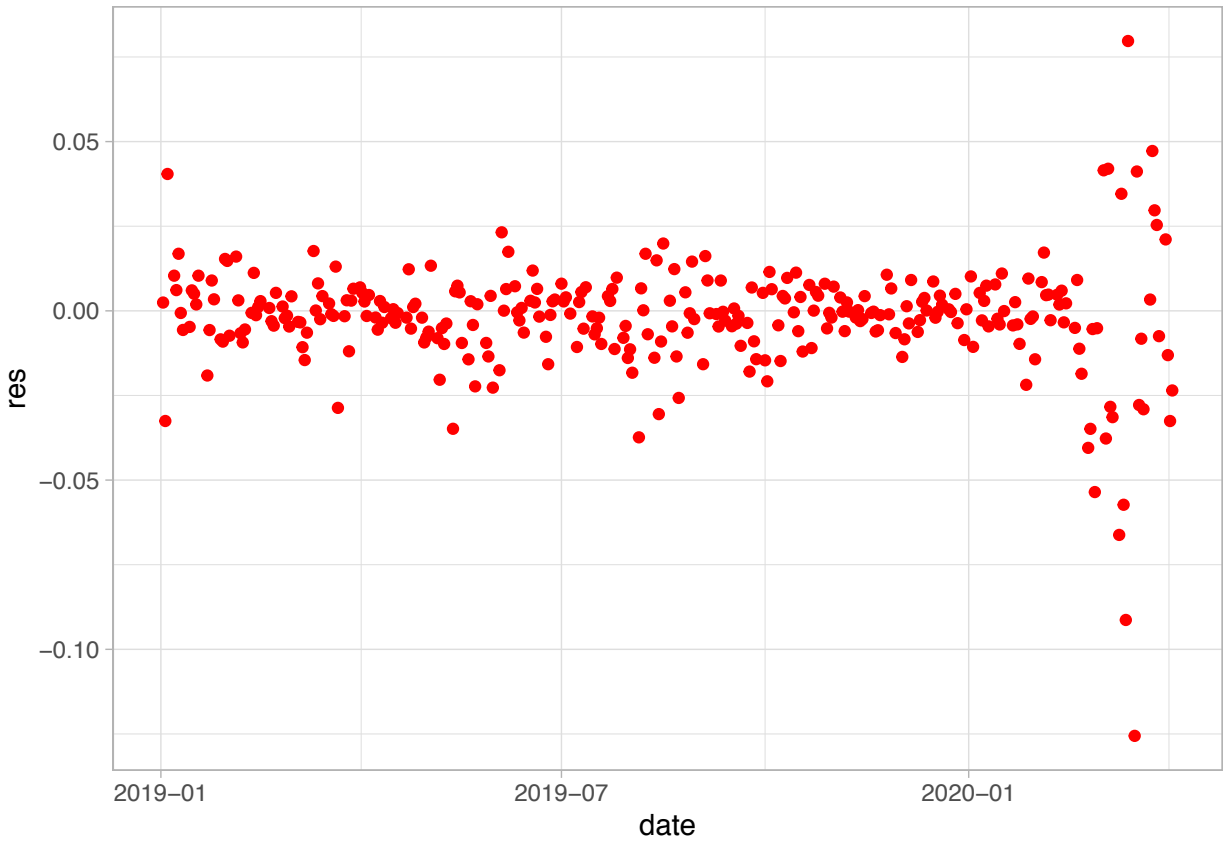

```
model.garch = ugarchspec(mean.model=list(armaOrder=c(4,3),include.mean=T, archm = FALSE, archpow = 1, ar
variance.model=list(model='sGARCH',garchOrder=c(1,1), submodel = NULL, external.regressors = NULL, varia
distribution.model = "norm" )
model.garch.fit = ugarchfit(data=rt_2019_2020, spec=model.garch, out.sample=20, solver = 'solnp')
```

Garch fitted plot

```
r_train_2019_2020_fit_g<-xts( model.garch.fit@fit$fitted.values,order.by=index(rt_2019_2020_train))
d<-as.Date(index(r_train_2019_2020_fit))
return_2019_2020_train_fit<-exp(r_train_2019_2020_fit_g)-1
return_2019_2020_train<-exp(rt_2019_2020_train)-1

Garch_2019_2020_summary_train<-as_tibble(cbind(return_2019_2020_train,return_2019_2020_train_fit))%>%
 mutate(date=d,res=return_2019_2020_train-return_2019_2020_train_fit)

ggplot(aes(date,return_2019_2020_train),data=Garch_2019_2020_summary_train)+
 geom_line(col='red')+
 geom_line(aes(date,return_2019_2020_train_fit),col='blue')+
 theme_light()
```

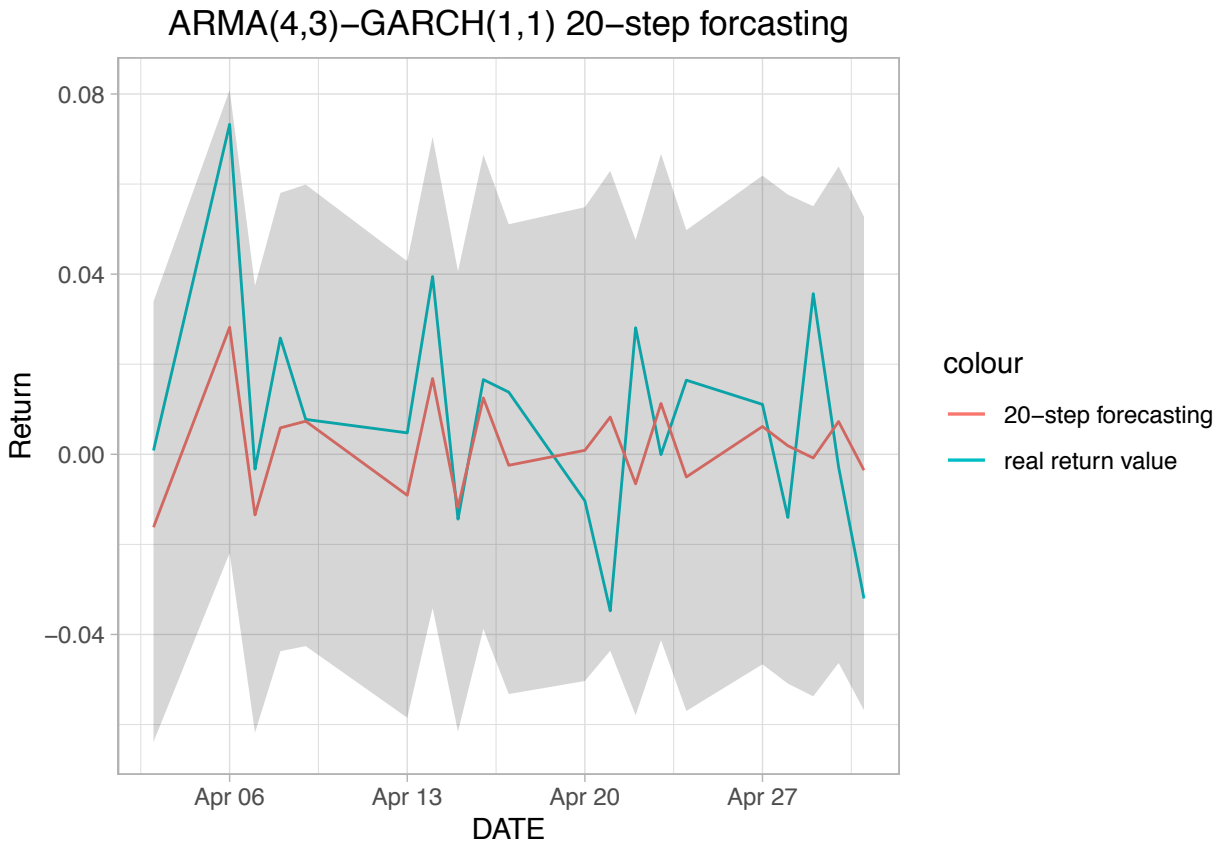



```
ggplot(aes(date,res),data=Garch_2019_2020_summary_train)+
 geom_point(col='red')+
 theme_light()
```

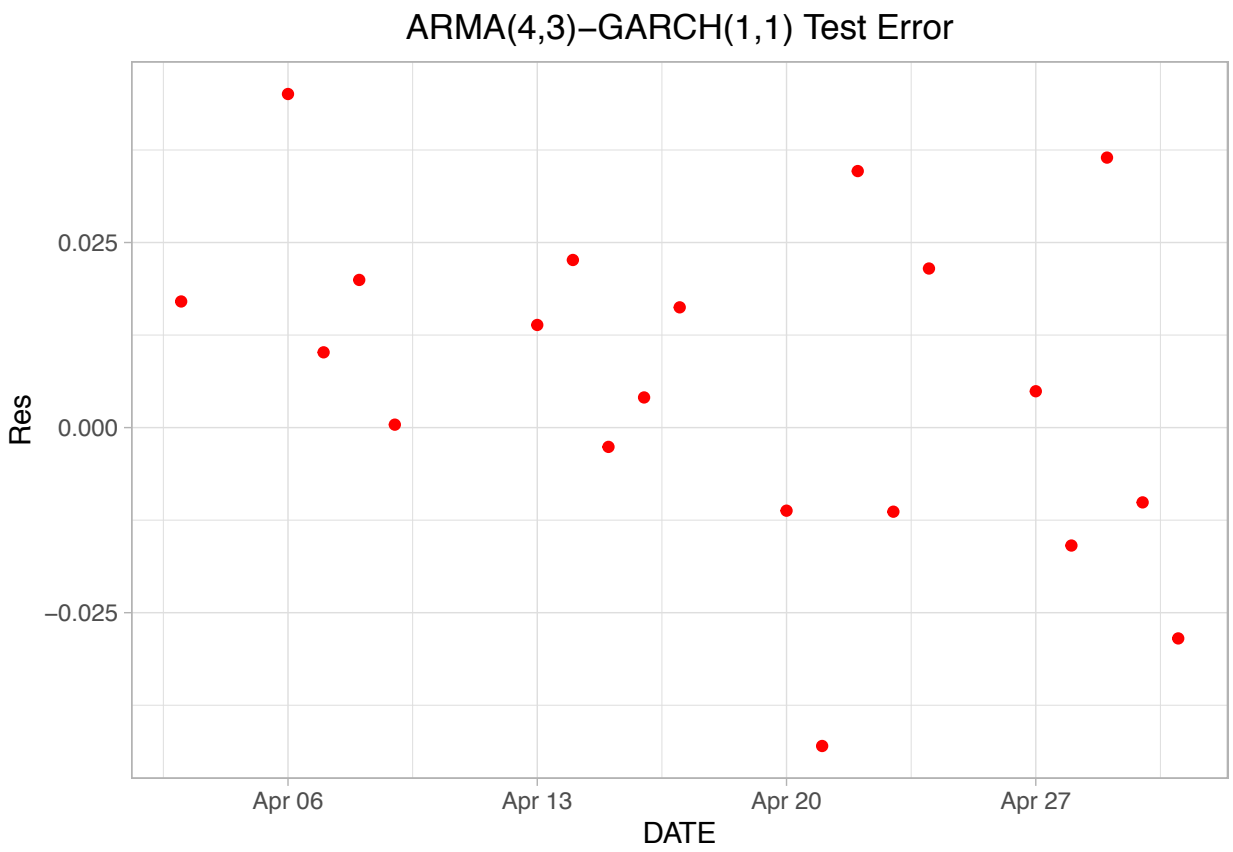


Garch forecast

```
d<-as.Date(index(rt_2019_2020_test))

forc=ugarchforecast(model.garch.fit,n.ahead=20)
test_prec_2019_2020_g<-forc@forecast$seriesFor
for_low_2019_2020_g<-test_prec_2019_2020_g-1.96*forc@forecast$sigmaFor
for_up_2019_2020_g<-test_prec_2019_2020_g+1.96*forc@forecast$sigmaFor


return_test_2019_2020_g<-exp(test_prec_2019_2020_g)-1
return_test_2019_2020_g_upper<-exp(for_up_2019_2020_g)-1
return_test_2019_2020_g_lower<-exp(for_low_2019_2020_g)-1


Garch_summary_2019_2020_predict<-as_tibble(cbind(return_test_2019_2020,return_test_2019_2020_g))%>%
 mutate(date=d,res=return_test_2019_2020-return_test_2019_2020_g,upper=return_test_2019_2020_g_upper,lo

ggplot(aes(date,return_test_2019_2020),data=Garch_summary_2019_2020_predict)+
 geom_line(aes(col='real return value'))+
 geom_line(aes(date,return_test_2019_2020_g,col='20-step forecasting'))+
geom_ribbon(aes(ymin=lower, ymax=upper), alpha=0.2)+
  labs(title="ARMA(4,3)-GARCH(1,1) 20-step forcasting",
  x="DATE",y='Return')+
 theme_light()+
 theme(plot.title = element_text(hjust = 0.5), plot.subtitle = element_text(hjust = 0.5))
```




```
ggplot(aes(date,res),data=Garch_summary_2019_2020_predict)+
geom_point(col='red')+
labs(title="ARMA(4,3)-GARCH(1,1) Test Error",
 x="DATE",y='Res')+
 theme_light()+
 theme(plot.title = element_text(hjust = 0.5), plot.subtitle = element_text(hjust = 0.5))
```


ARMA(4,3)−GARCH(1,1) Test Error
Res
0.025
0.000
−0.025
Apr 06
Apr 13
Apr 20
Apr 27
DATE