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
}
```

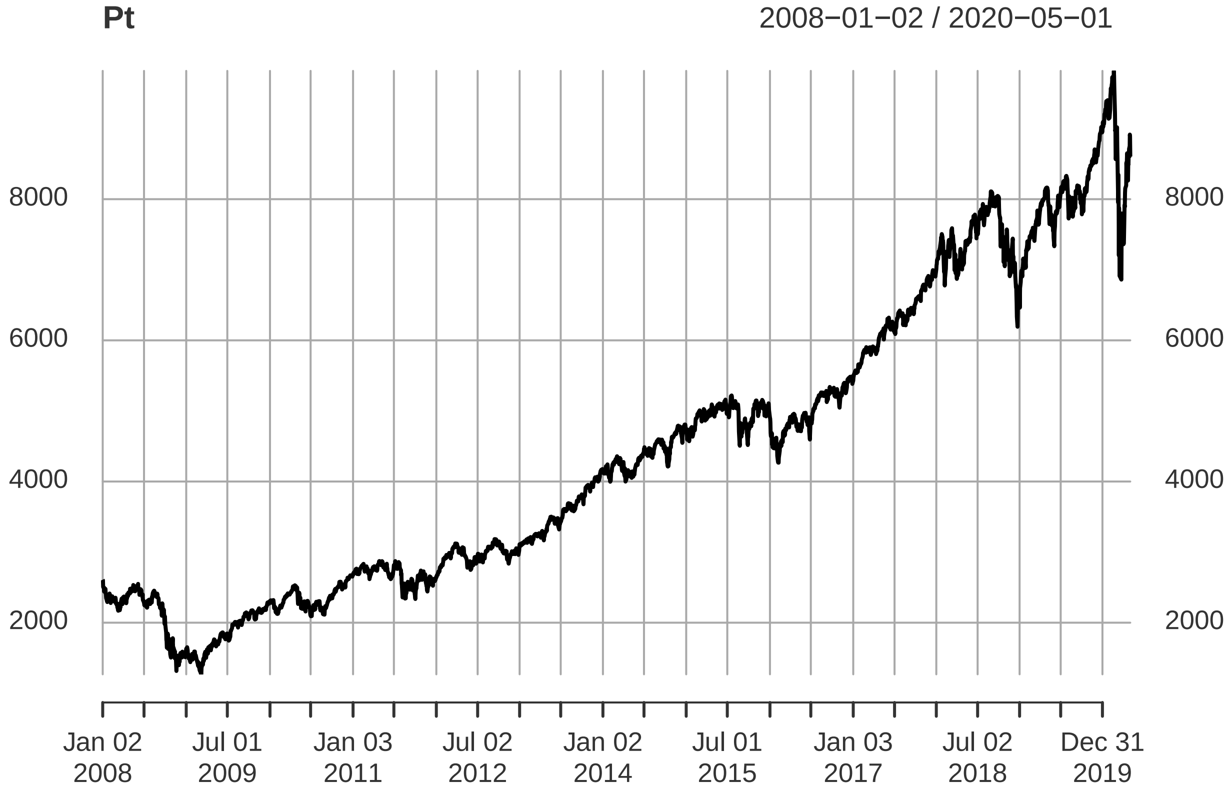
```
## [1] "2016-12-14" "2017-09-06" "2020-04-02"
```

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

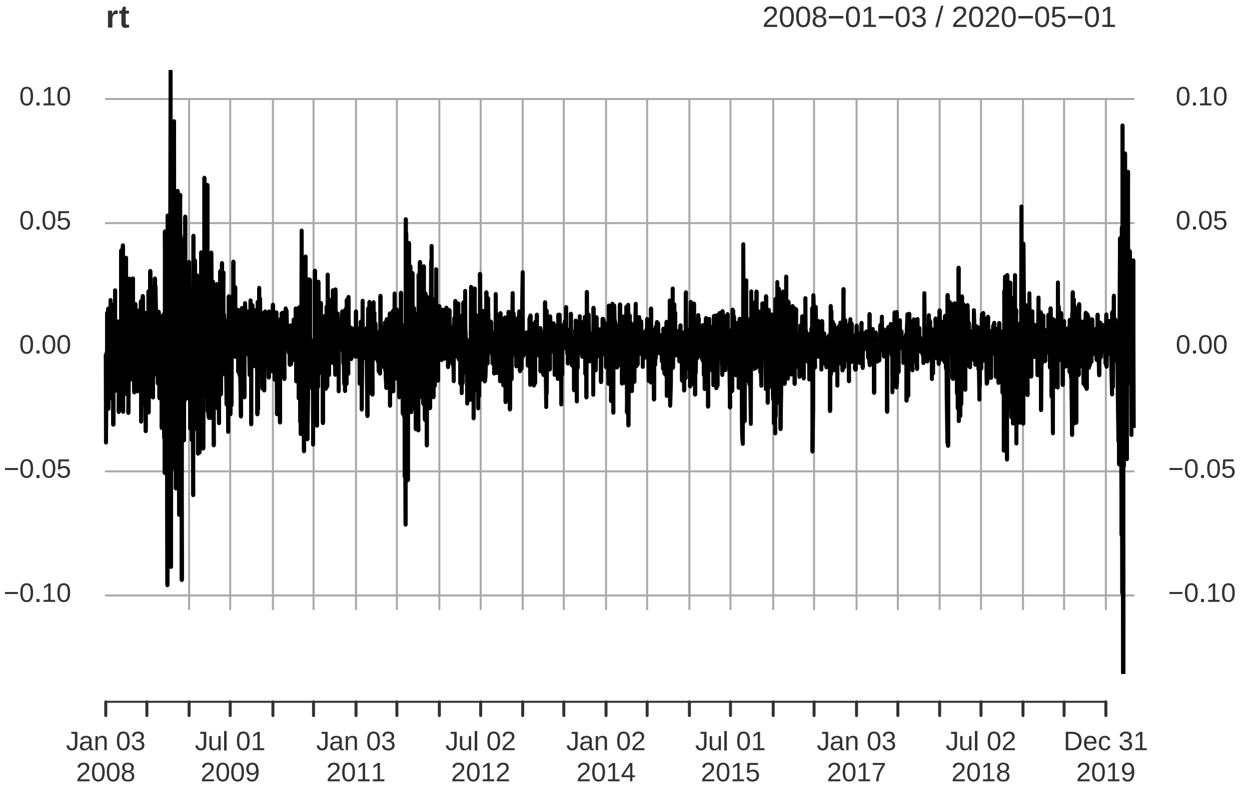
```
## [1] 0
```

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

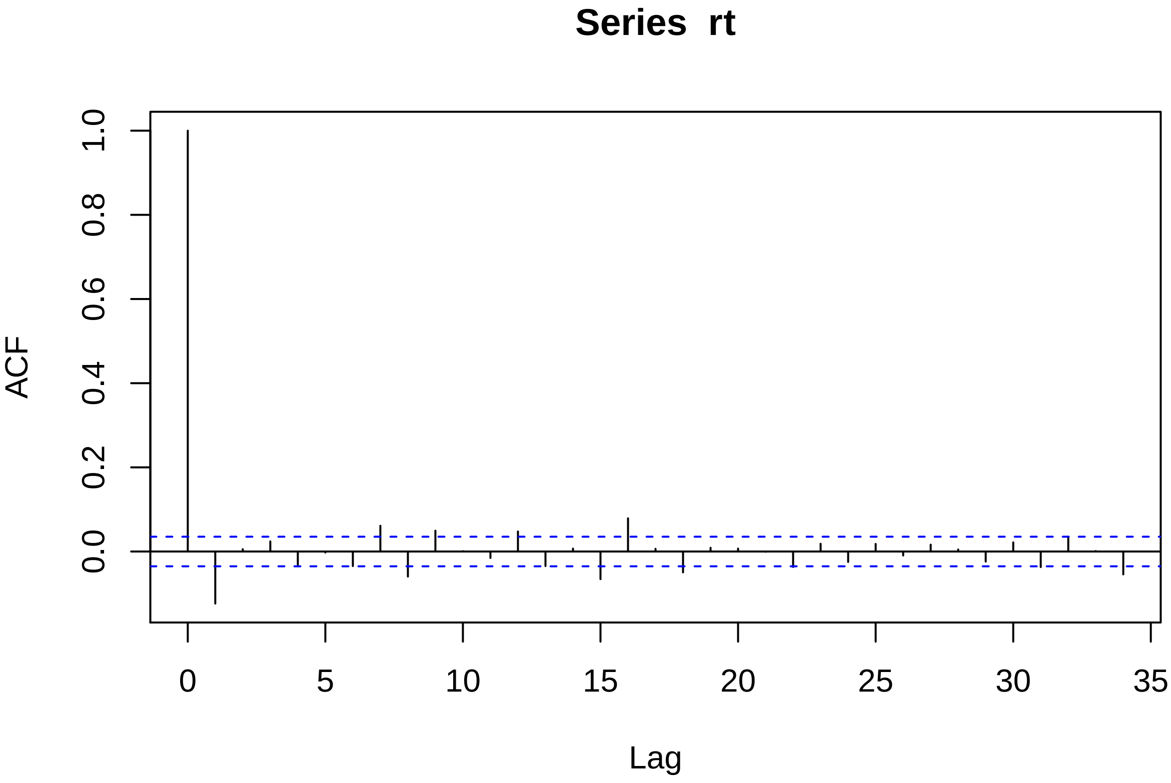
```
plot(Pt)
```


Pt
2008−01−02 / 2020−05−01
8000
6000
4000
2000
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

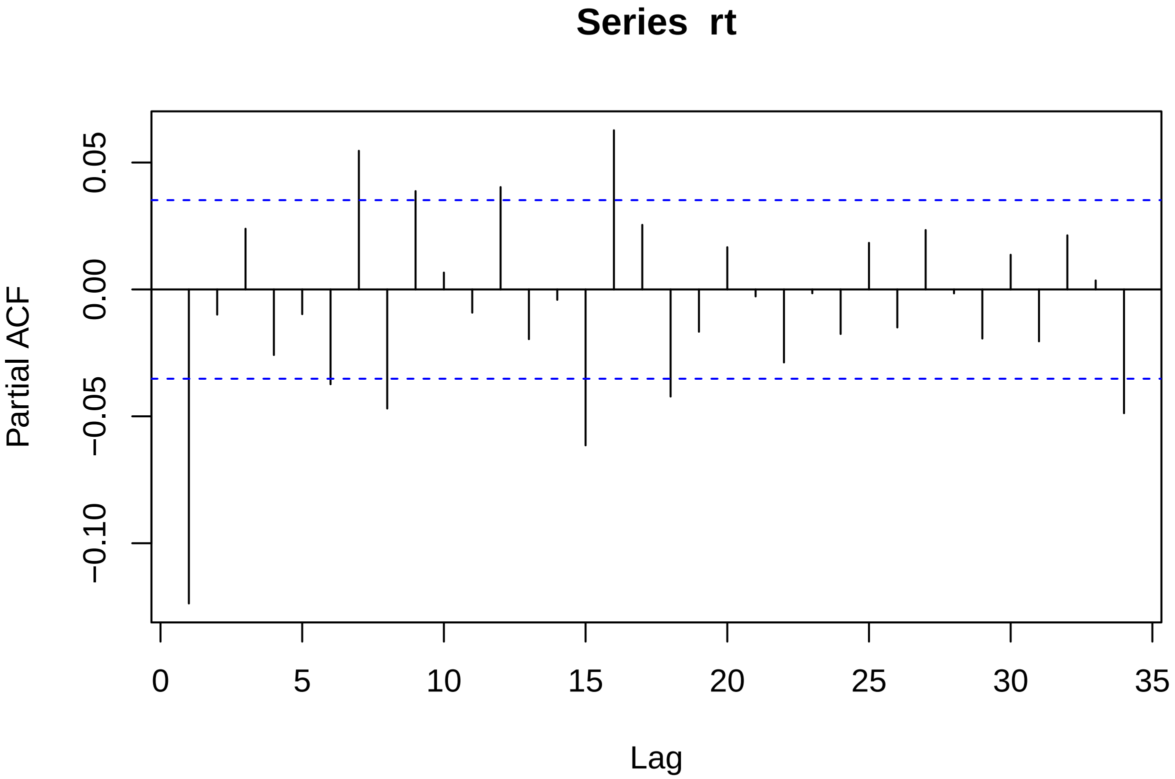
```
plot(rt)
```


rt
2008−01−03 / 2020−05−01
0.10
0.05
0.00
−0.05
−0.10
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

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



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



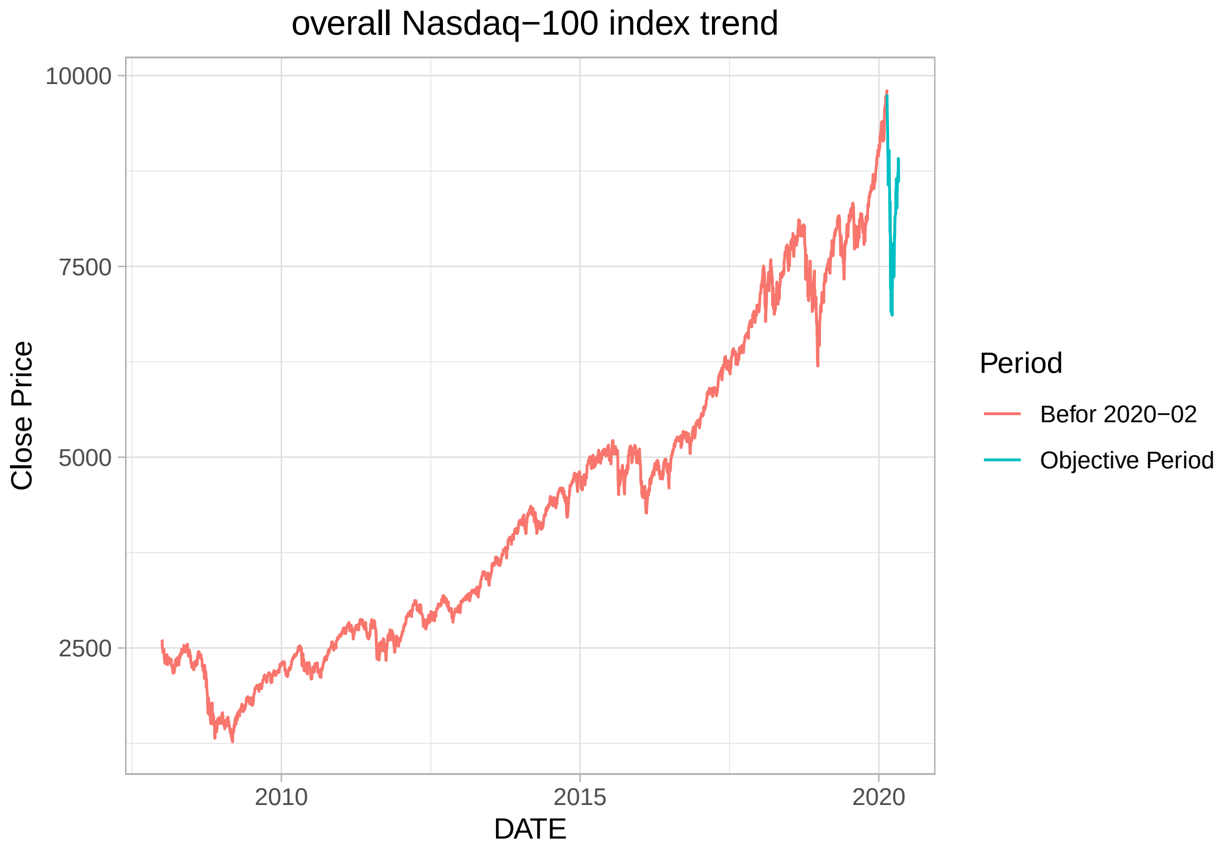
```r
library(tibble)
library(tidyverse)
```

```
##   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()
```

```r
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'))
```

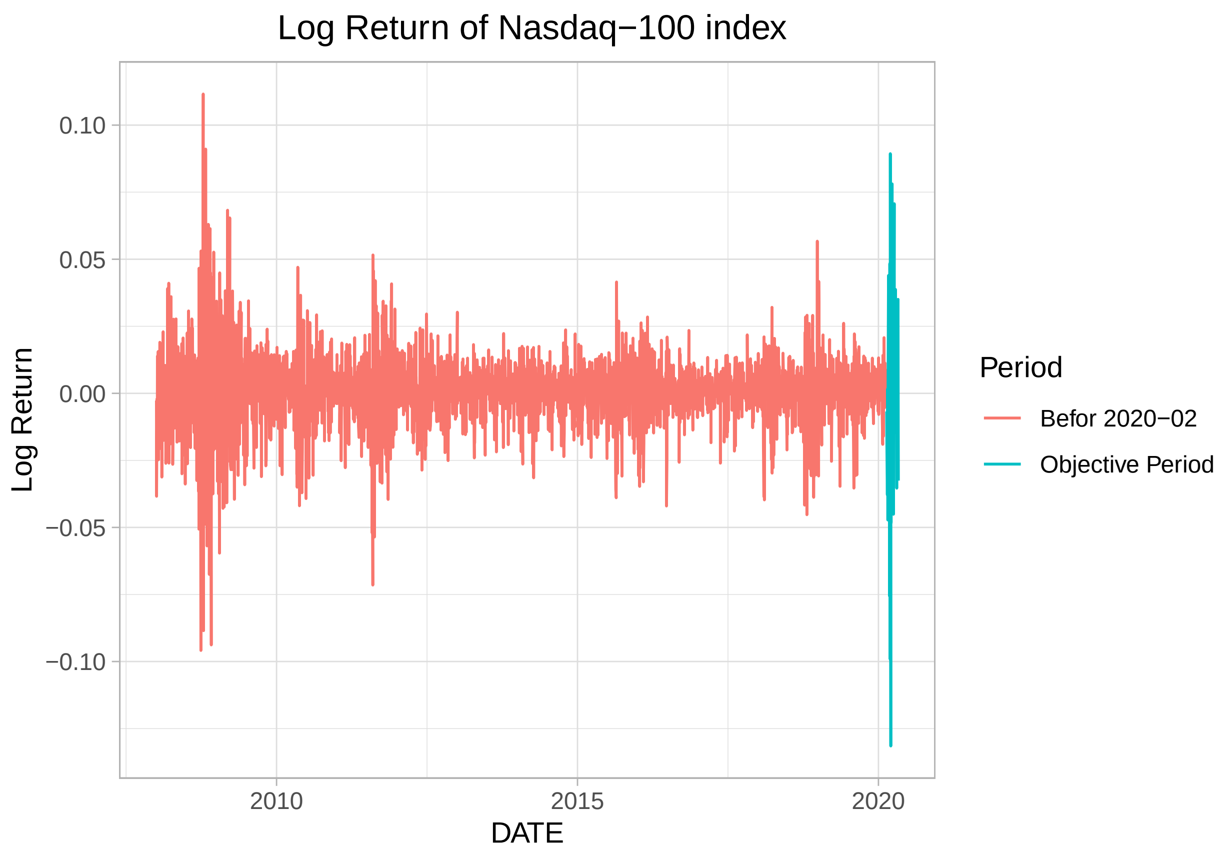
```
## Warning: Calling `as_tibble()` on a vector is discouraged, because the behavior is likely to change
## This warning is displayed once per session.
```

```r
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))
```

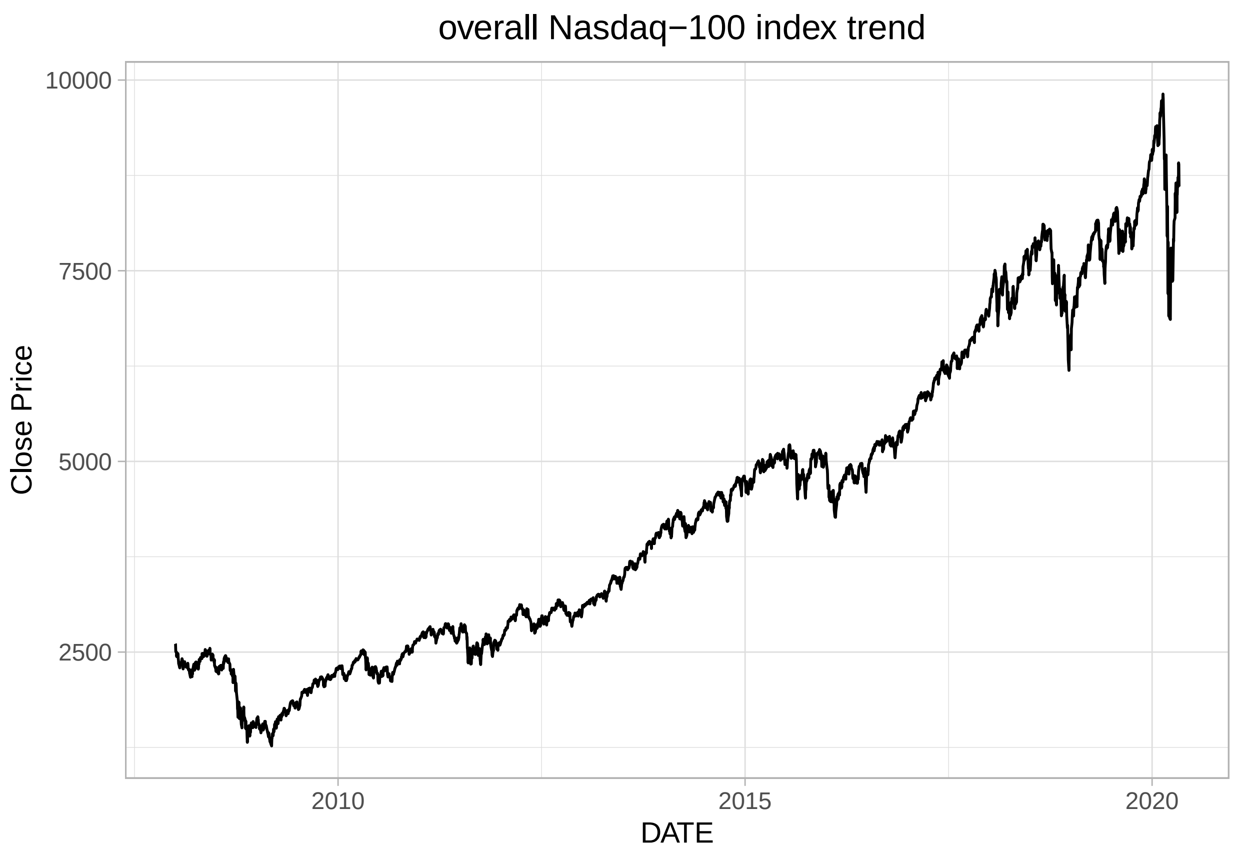


```
ggplot(aes(date,rt),data=all_data)+
 geom_line(aes(col=Period))+
   labs(title=" Log Return of Nasdaq-100 index",
       x="DATE",'Log Return',y='Log Return')+
 theme_light()+
theme(plot.title = element_text(hjust = 0.5), plot.subtitle = element_text(hjust = 0.5))
```

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

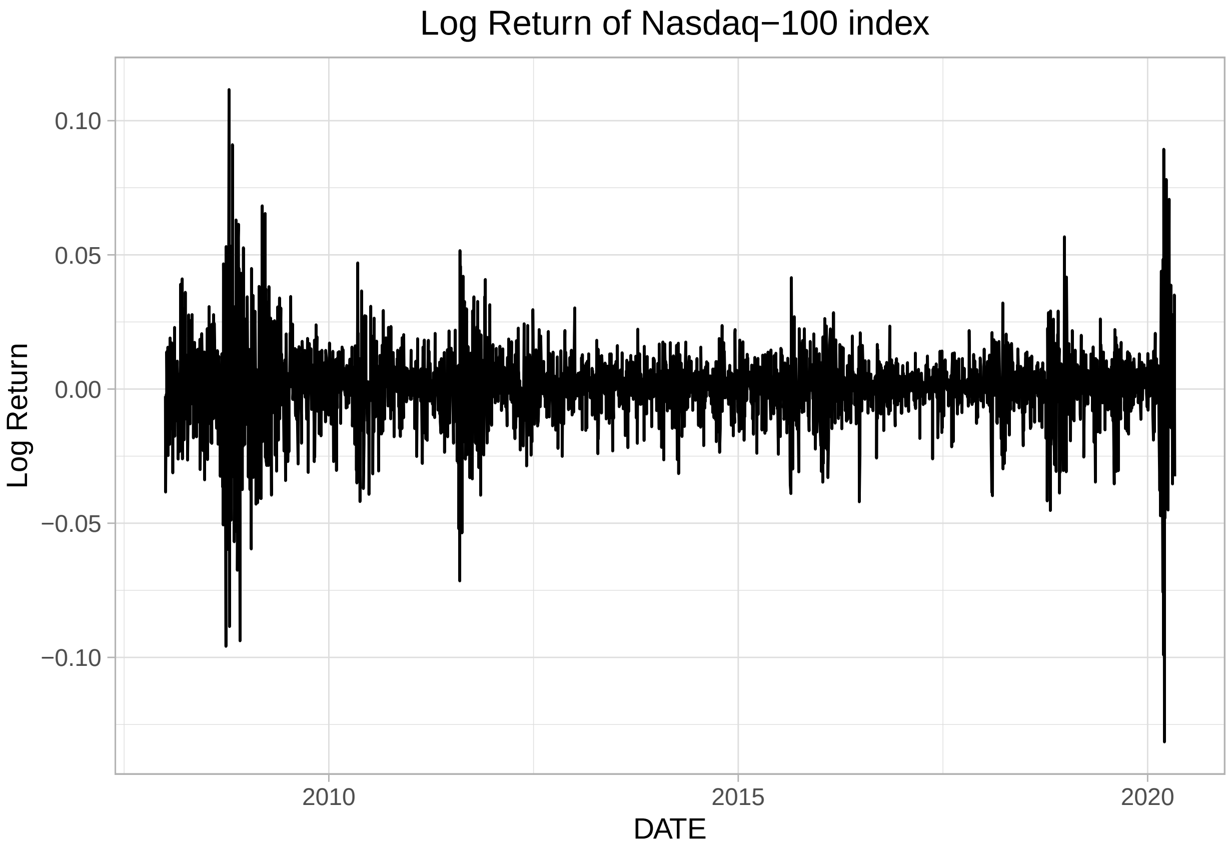


```
ggplot(aes(date,Pt),data=all_data)+
 geom_line()+
  labs(title=" overall Nasdaq-100 index trend",
       x="DATE",'Close Price',y='Close Price')+
 theme_light()+
 theme(plot.title = element_text(hjust = 0.5), plot.subtitle = element_text(hjust = 0.5))
```



```
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']
```

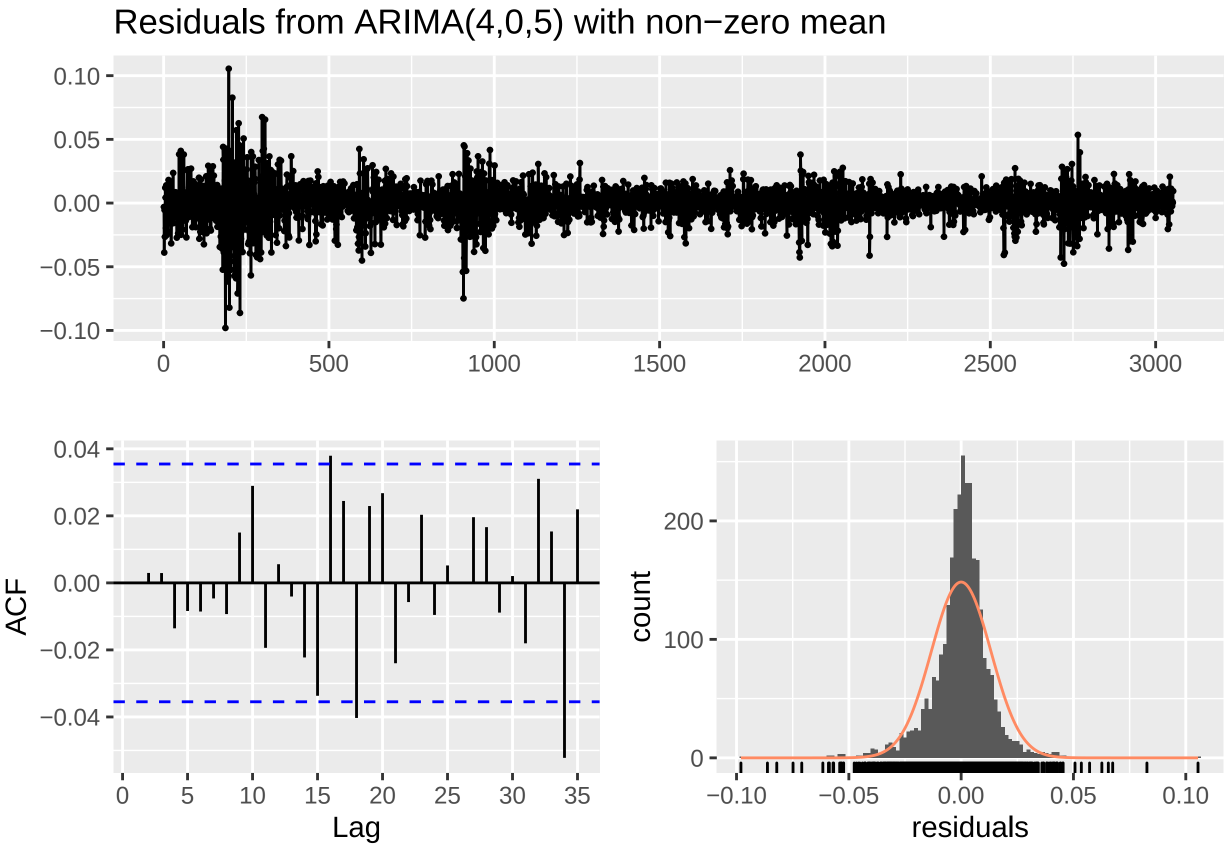
```
AIC_p_q_select<-matrix(NA,nrow=5,ncol=5)
for(p in 0:4){
  for(q in 0:4){
    model_tmp<-arima(rt_subset, order = c(p,0,q),include.mean = T)
    AIC_p_q_select[p+1,q+1]<-model_tmp$aic
  }
}
```

```
## 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
```

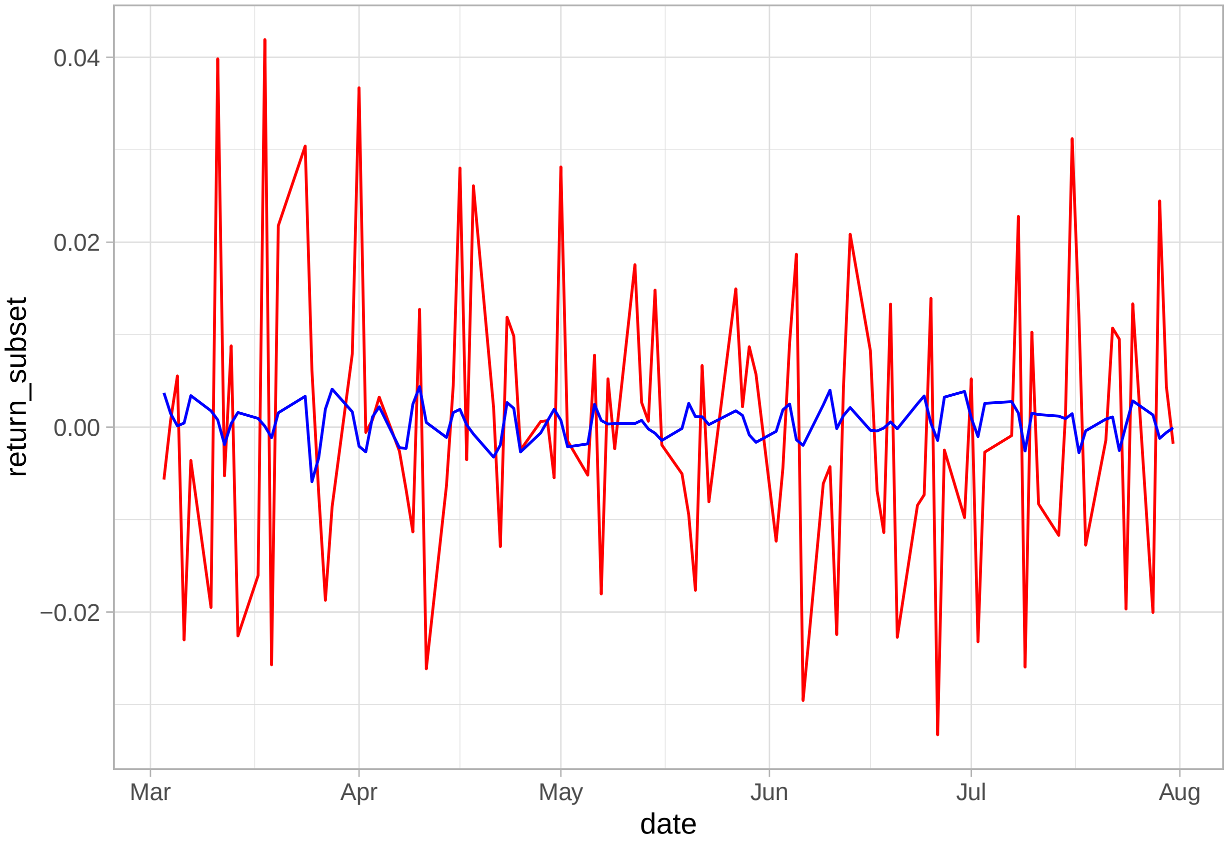
```
##      row col
## [1,]   4   3
```

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

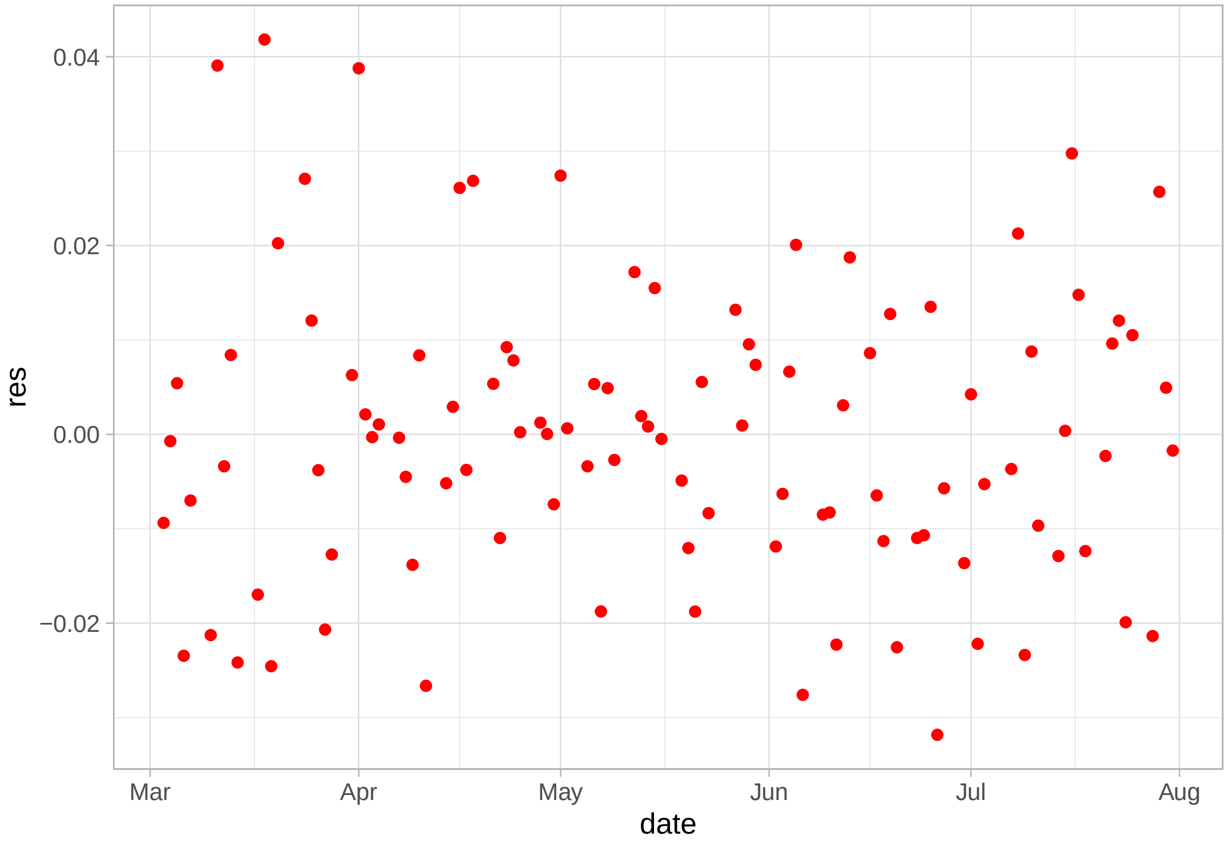


```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(4,0,5) with non-zero mean
## Q* = 5.9434, df = 3, p-value = 0.1144
##
## Model df: 10.   Total lags used: 13
```

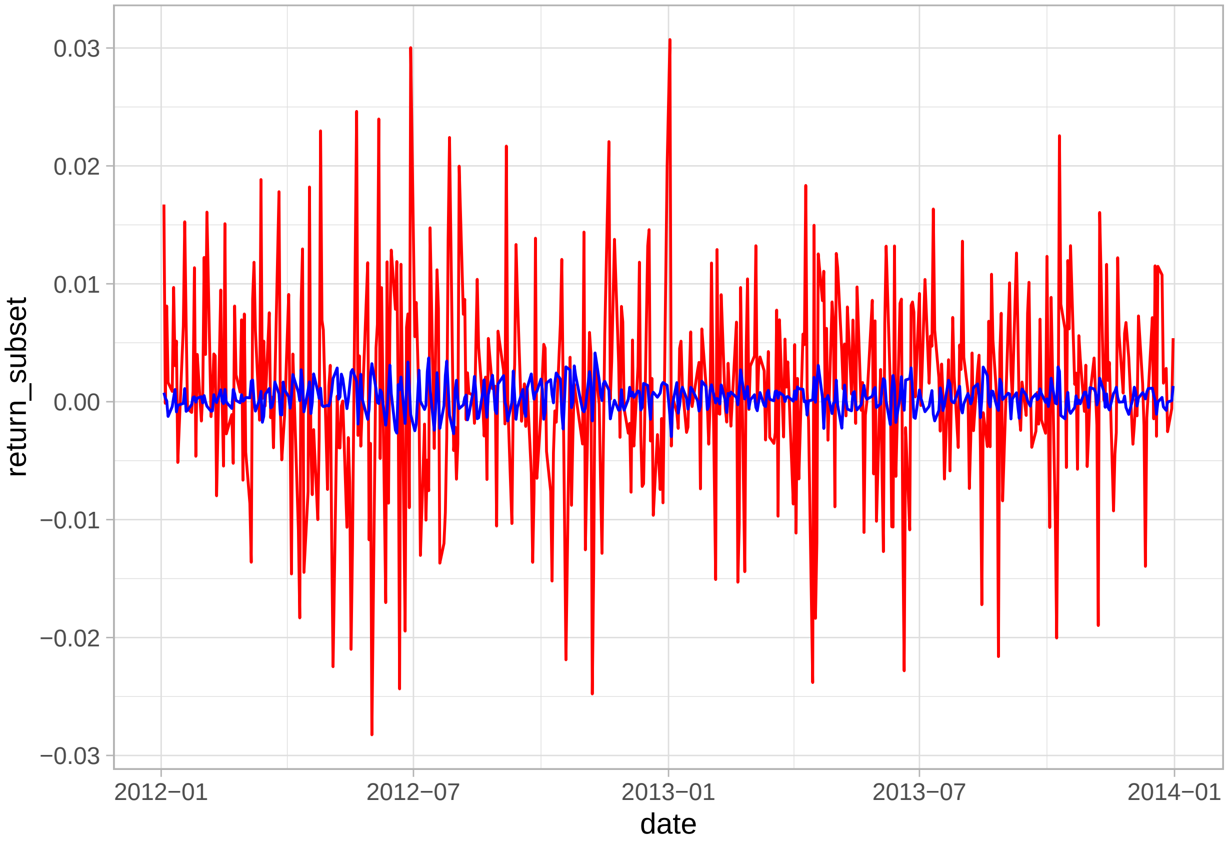
```
r_train_fit<-xts(fitted(Rt_subset.train),order.by=index(rt_subset))
d<-as.Date(index(r_train_fit))
return_train_fit<-exp(r_train_fit)-1
return_subset<-exp(rt_subset)-1

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()
```



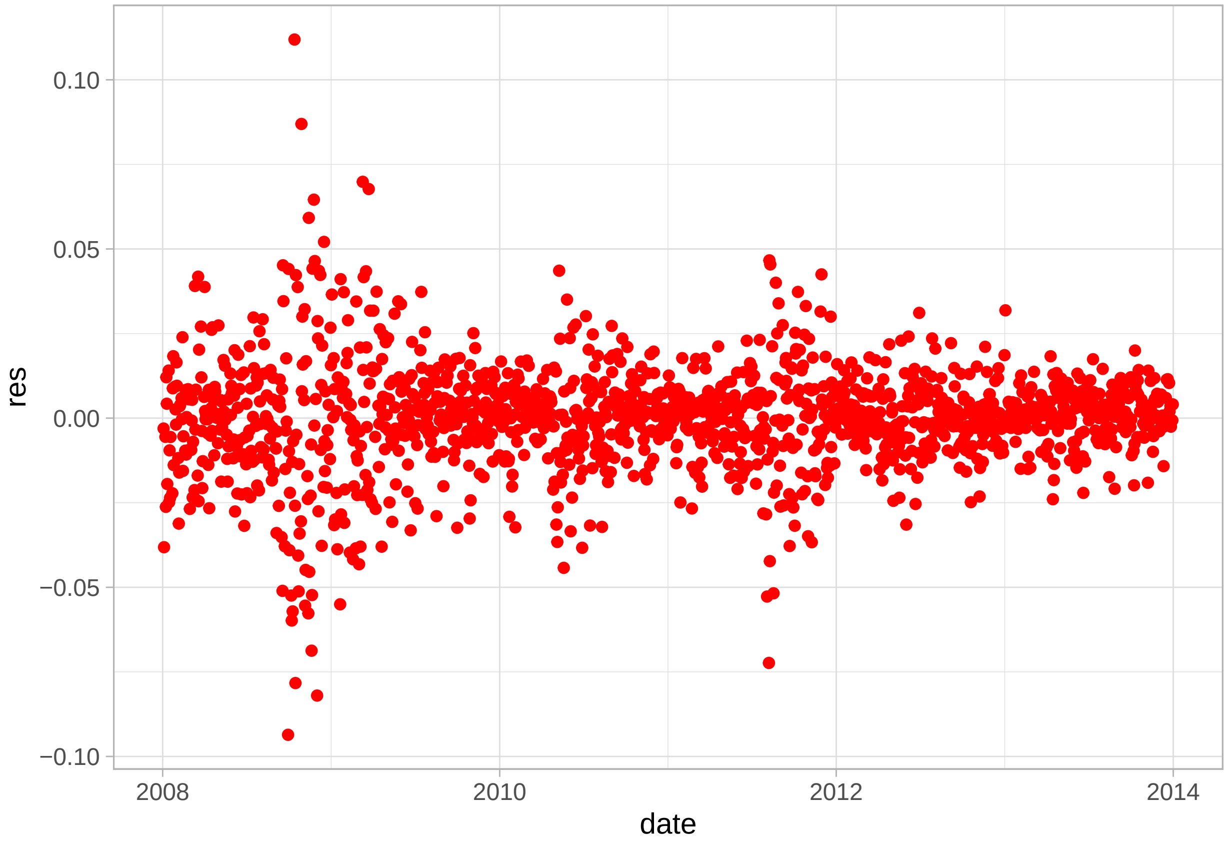
```
ggplot(aes(date,res),data=ARMA_summary_train[ARMA_summary_train$date>'2008-03-01'&ARMA_summary_train$dat
 geom_point(col='red')+
 theme_light()
```



```
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()
```

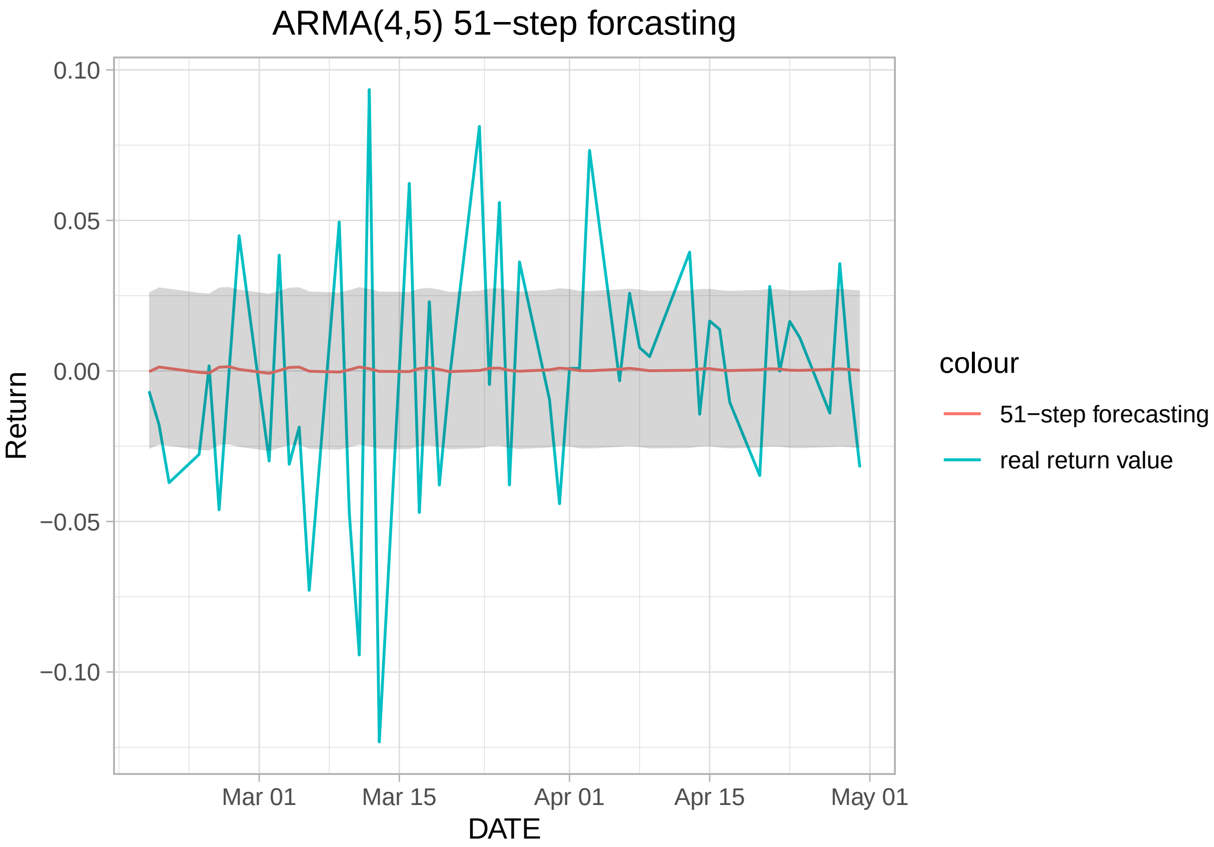


```
library(tidyverse)
library(tibble)
library(ggplot2)
d<-as.Date(index(Pt[(length(Pt)-51):(length(Pt)-1)]))
for_arma<-forecast(Rt_subset.train,h=51)
return_predict<-exp(for_arma$mean)-1
return_predict_high_bound<-exp(for_arma$upper[,2])-1
return_predict_lower_bound<-exp(for_arma$lower[,2])-1
return_test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

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

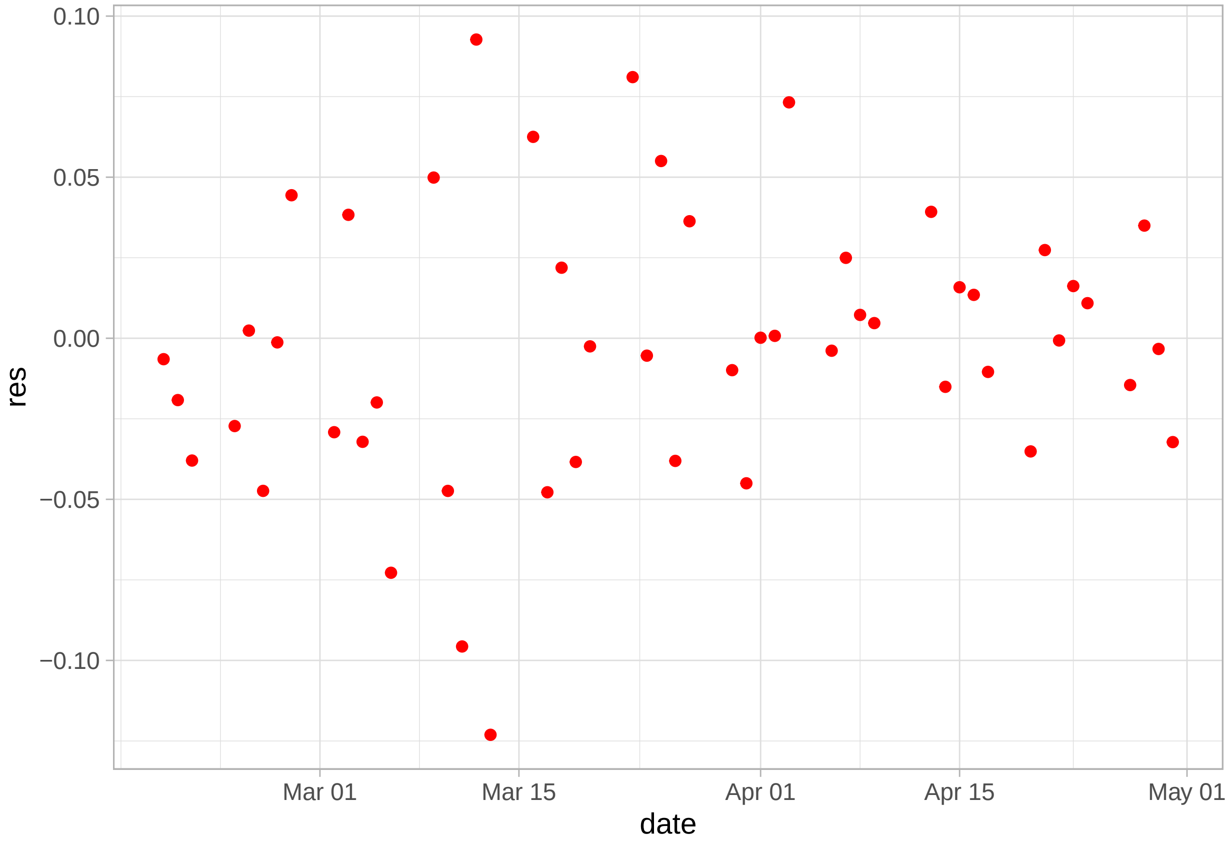
```
ggplot(aes(date,return_predict),data=ARMA_summary_predict)+
  geom_line(aes(date,return_test,color='real return value'))+
  geom_line(aes(col='51-step forecasting'))+
 geom_ribbon(aes(ymin=lower, ymax=upper), alpha=0.2)+
  labs(title="ARMA(4,5) 51-step forcasting",
  x="DATE",y='Return')+
 theme_light()+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



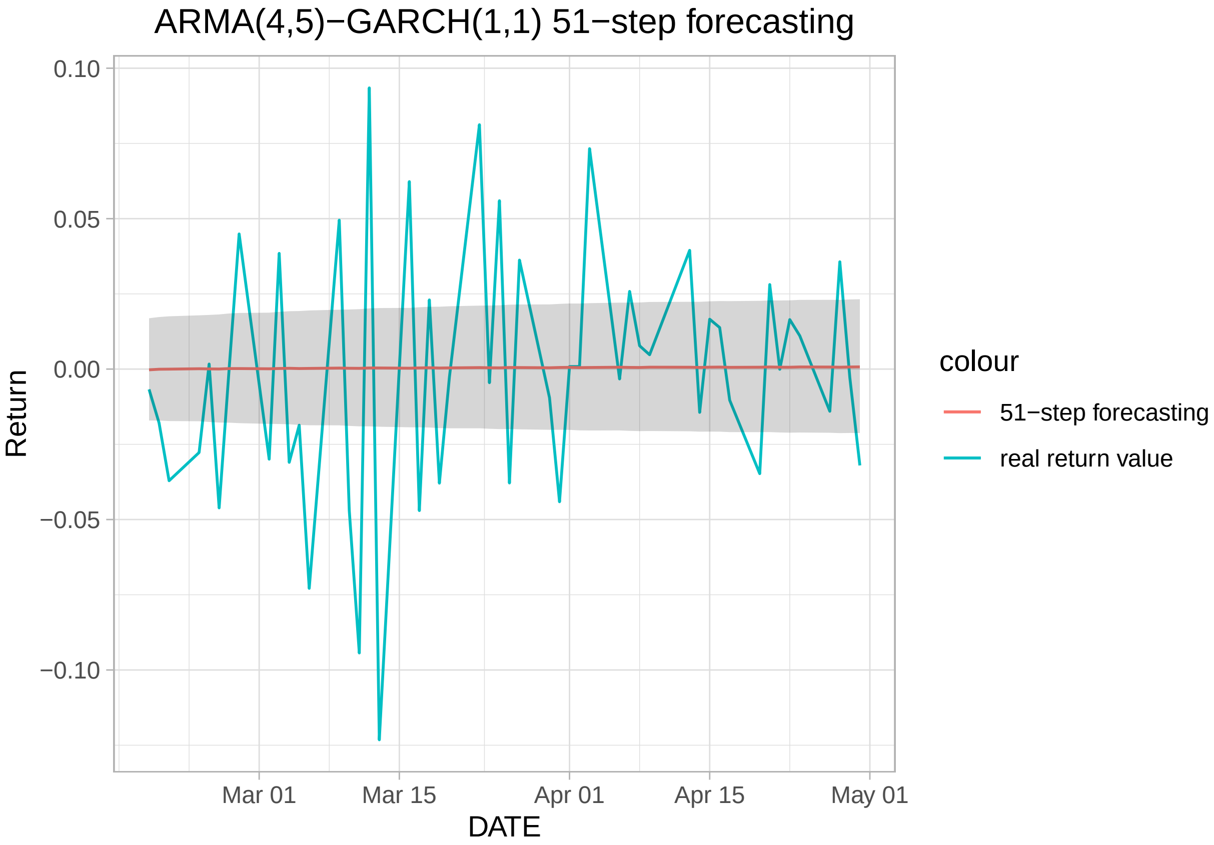
```r
library(rugarch)
```

```
## 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
```

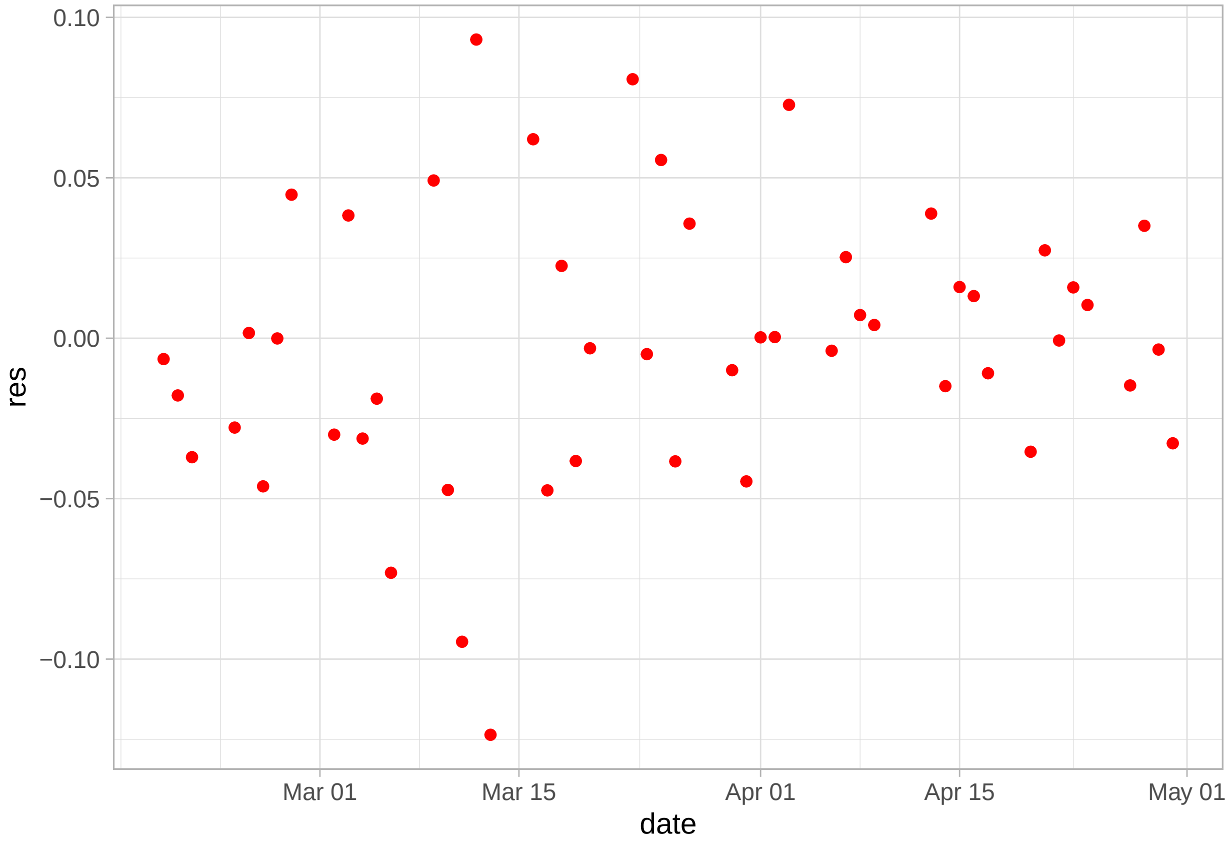
```r
model.garch = ugarchspec(mean.model=list(armaOrder=c(4,5),include.mean=T, archm = FALSE, archpow = 1, a
variance.model=list(model='sGARCH',garchOrder=c(1,1), submodel = NULL, external.regressors = NULL, varia
distribution.model = "norm" )
model.garch.fit = ugarchfit(data=rt, spec=model.garch, out.sample=51, 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
```

```r
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
```

```r
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))
```

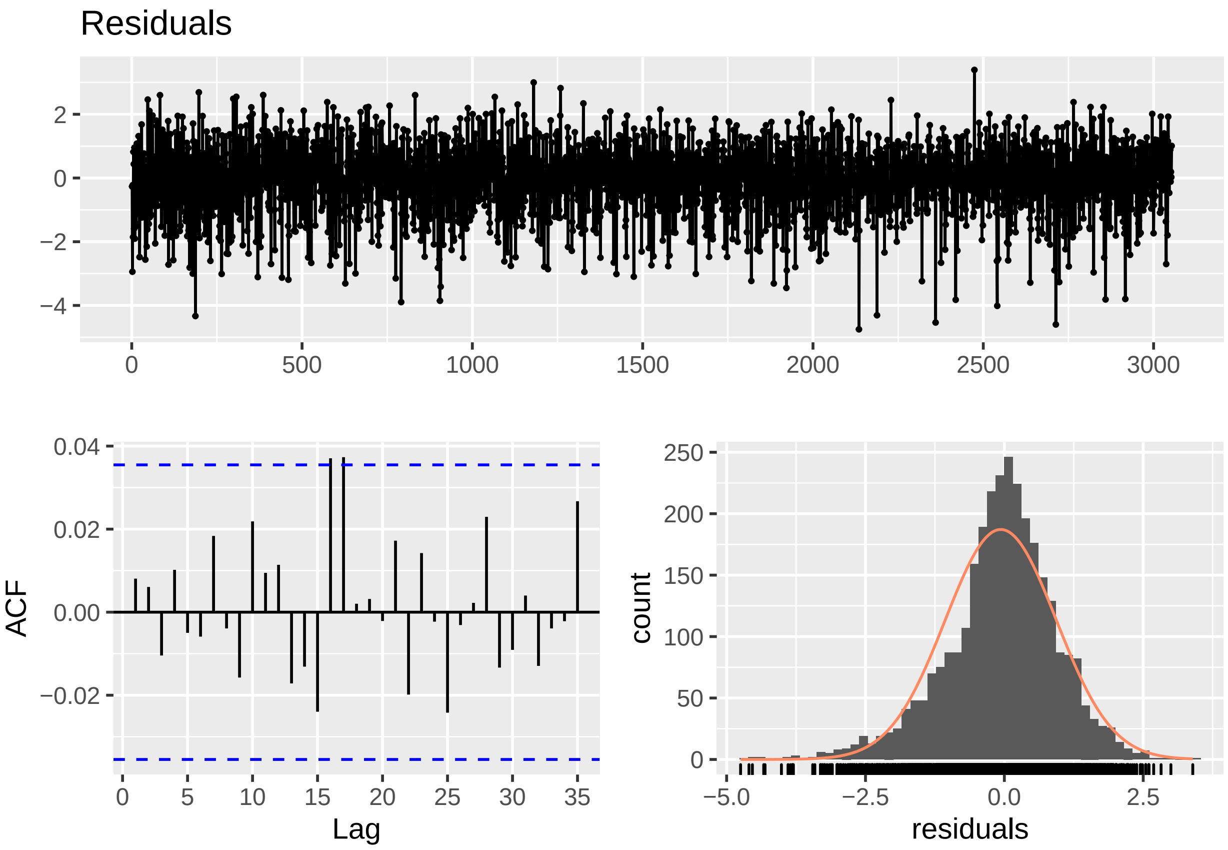


```r
ggplot(aes(date,res),data=Garch_summary_predict)+
 geom_point(col='red')+
 theme_light()
```



```
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
```

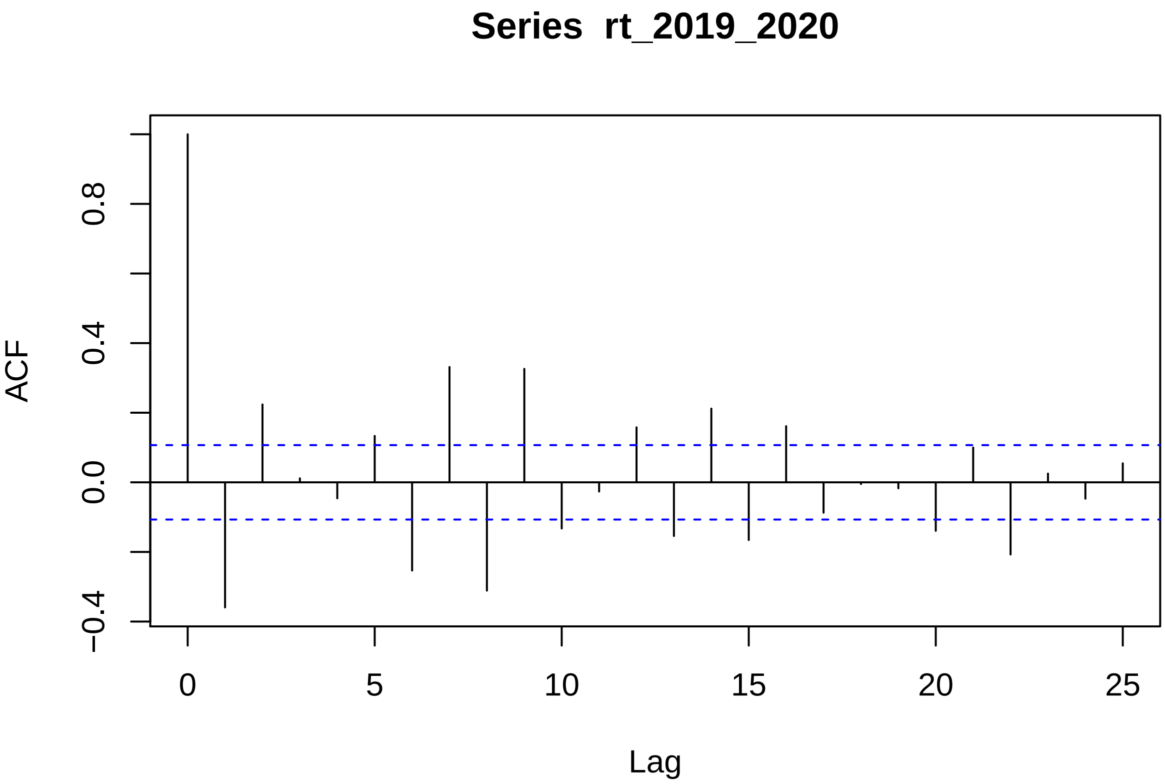
```
##            mu           ar1           ar2           ar3           ar4
##  8.753221e-04 -1.038117e-01 -2.323414e-01  7.866501e-01  4.298095e-01
##           ma1           ma2           ma3           ma4           ma5
##  5.815240e-02  2.099079e-01 -8.096319e-01 -4.091069e-01  1.271138e-02
##         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 =
```

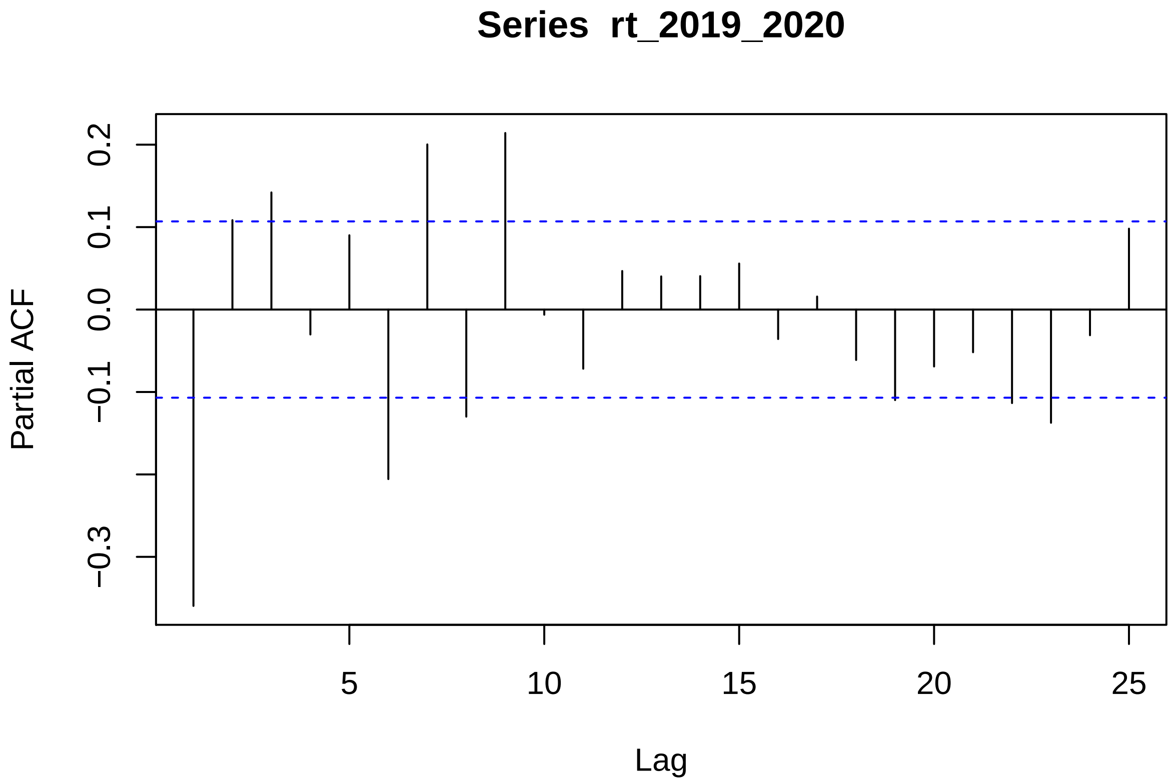
```
##
##  Box-Ljung test
##
## data:  model.garch.fit@fit$residuals/model.garch.fit@fit$sigma
## X-squared = 0.19952, df = 1, p-value = 0.6551
```

```
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)
```



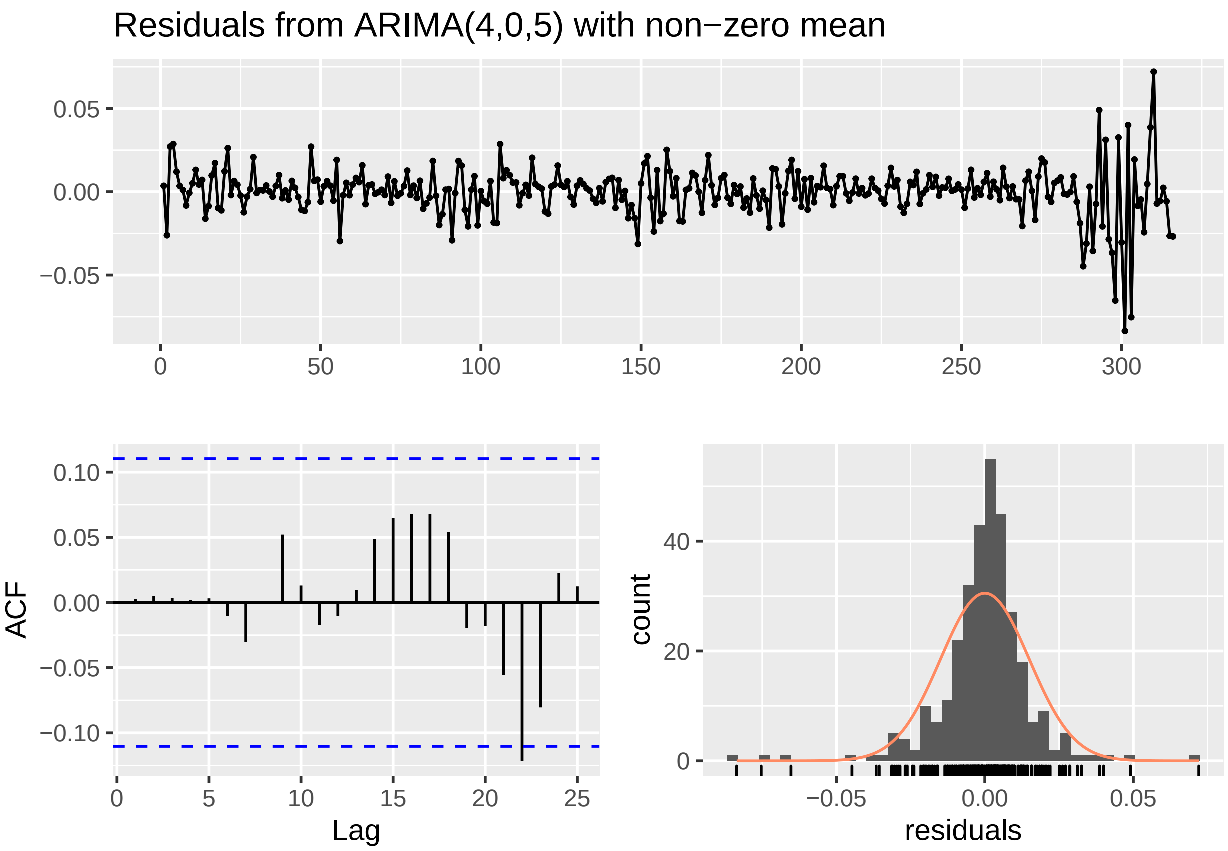
```
AIC_p_q_select<-matrix(NA,nrow=6,ncol=6)
for(p in 0:5){
  for(q in 0:5){
    model_tmp<-arima(rt_2019_2020_train, order = c(p,0,q),include.mean = T)
    AIC_p_q_select[p+1,q+1]<-model_tmp$aic
  }
}
```

```
## 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)
```

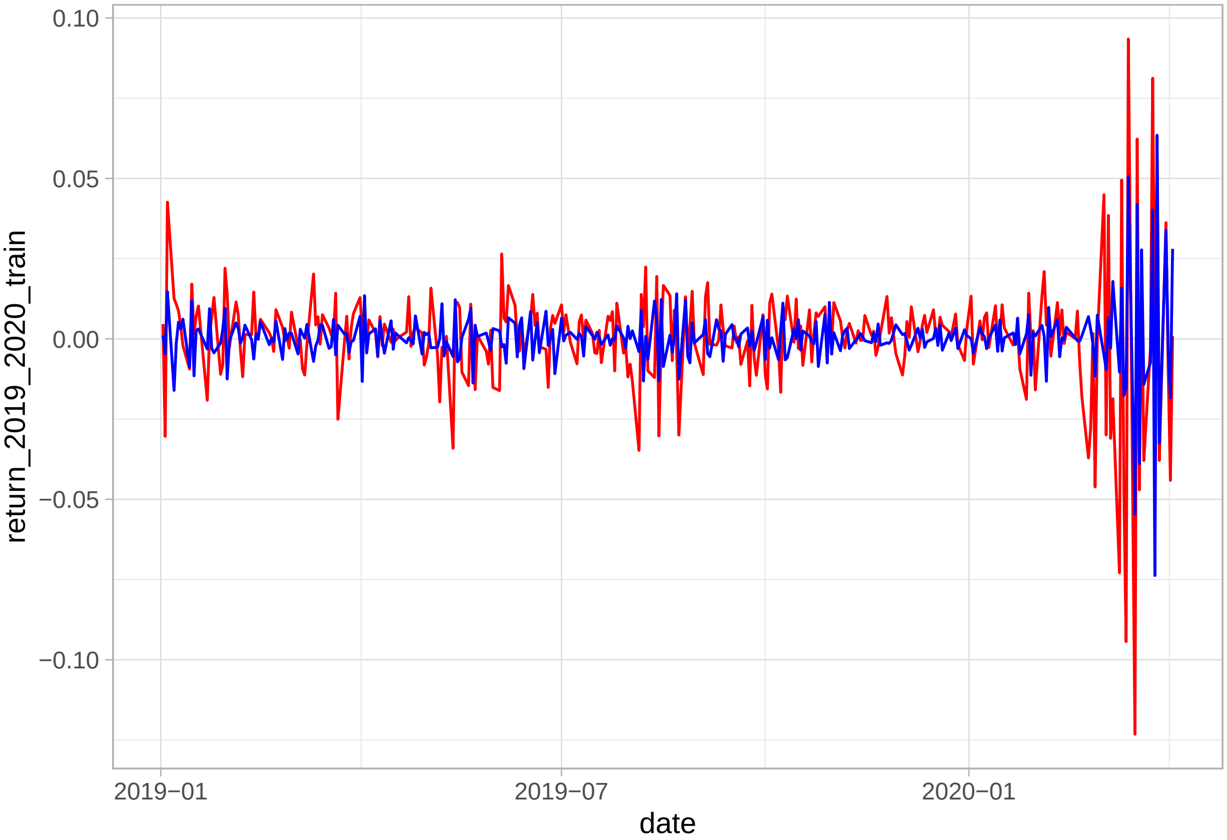


```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(4,0,5) with non-zero mean
## Q* = 1.4581, df = 3, p-value = 0.692
##
## Model df: 10.   Total lags used: 13
```

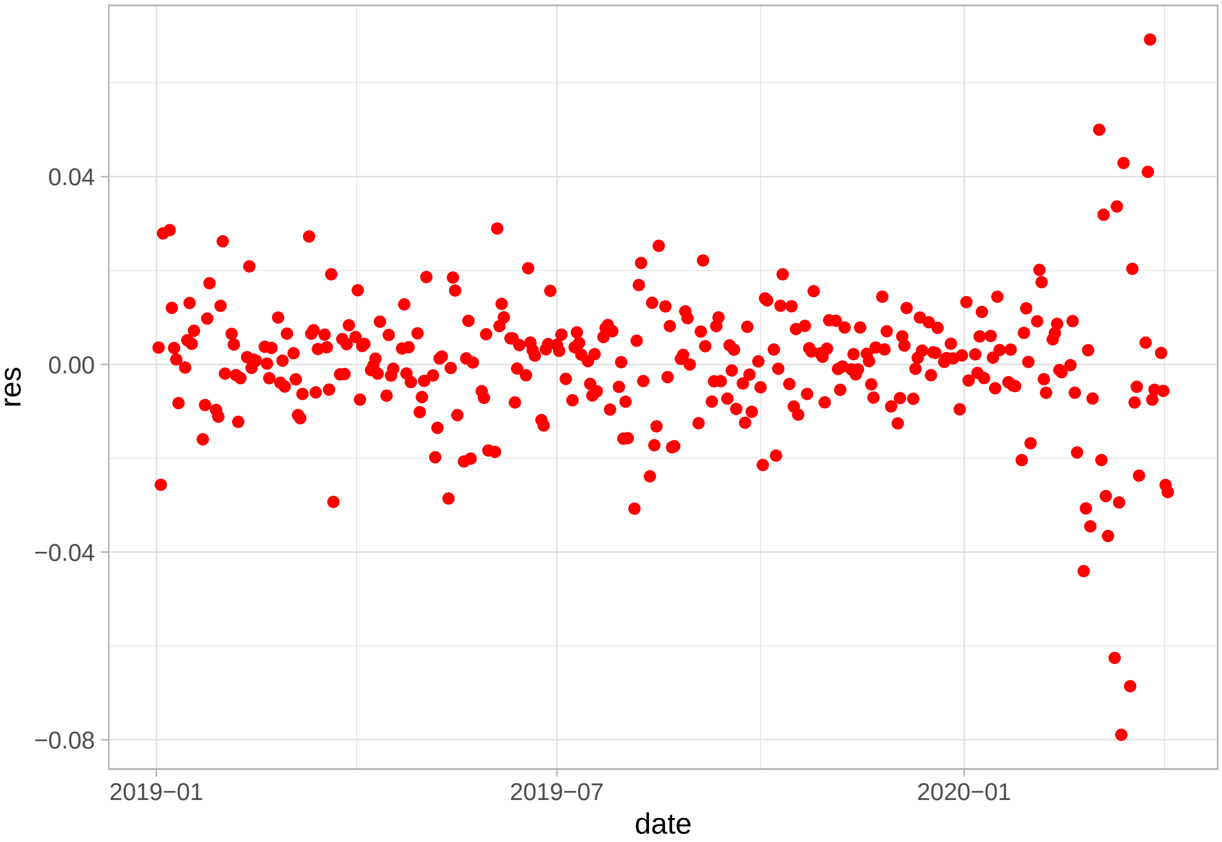
```
Rt_2019_2020.train$coef
```

```
##          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()
```

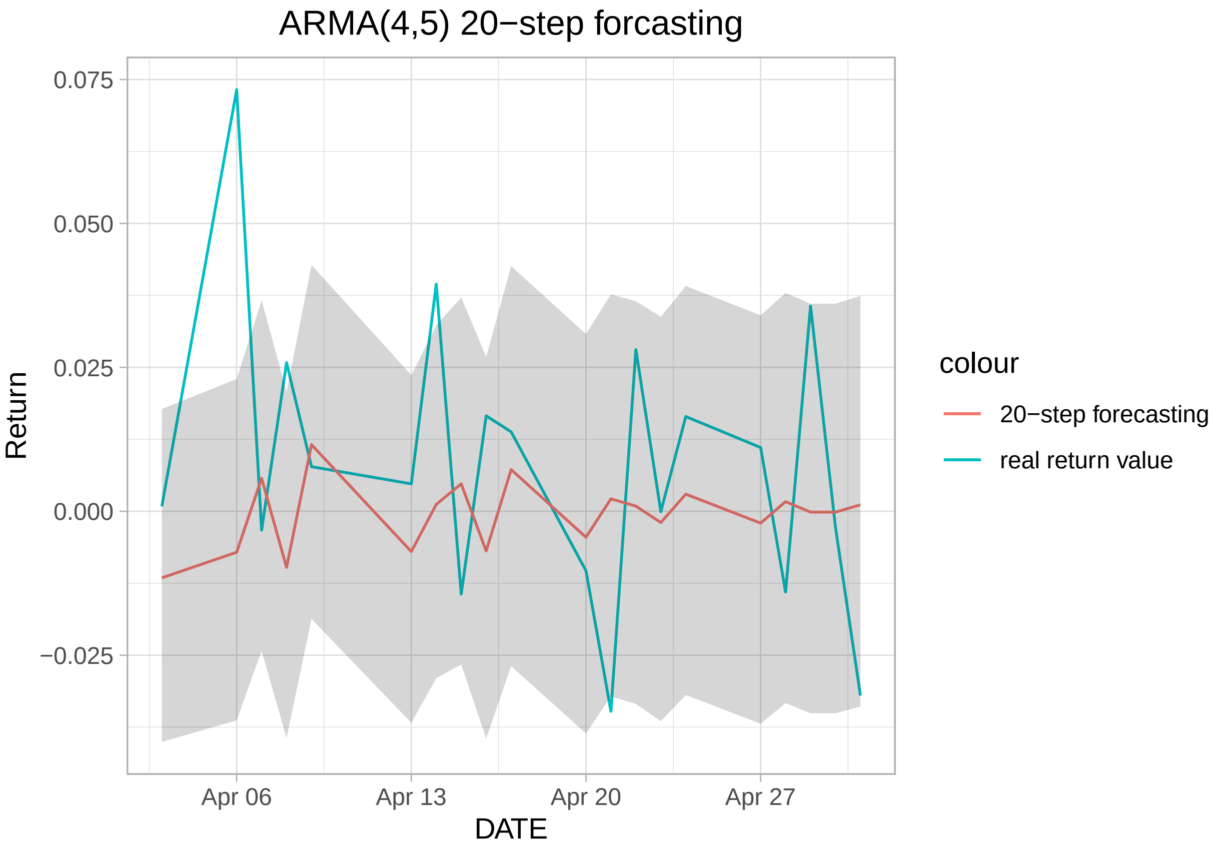


```
ggplot(aes(date,res),data=ARMA_2019_2020_summary_train)+
 geom_point(col='red')+
 theme_light()
```



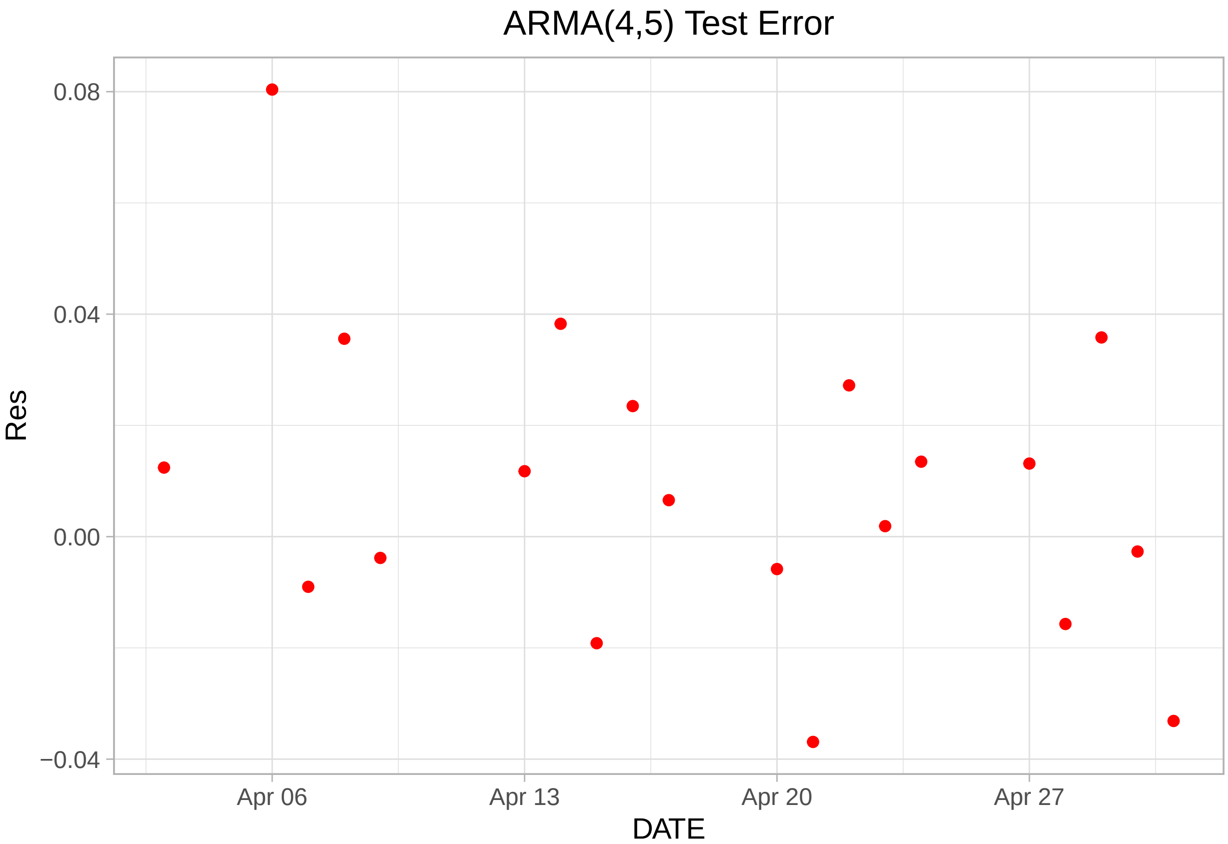
```
d<-as.Date(index(rt_2019_2020_test))
for_arma_2019_2020<-forecast(Rt_2019_2020.train,h=20)
test_2019_2020_predict<-for_arma_2019_2020$mean
return_2019_2020_predict<-exp(test_2019_2020_predict)-1
return_2019_2020_predict_high_bound<-exp(for_arma_2019_2020$upper[,2])-1
return_2019_2020_predict_lower_bound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

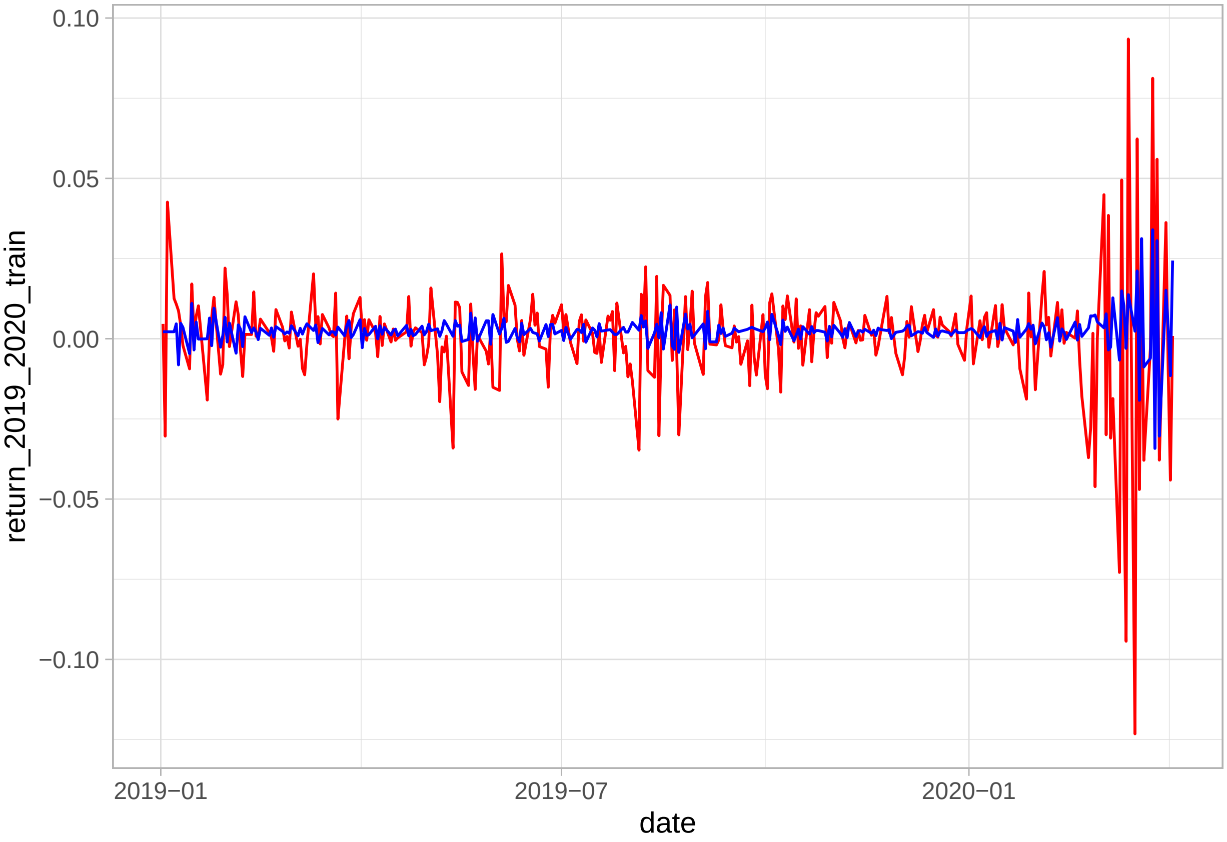
```
## Don't know how to automatically pick scale for object of type xts/zoo. Defaulting to continuous.
```



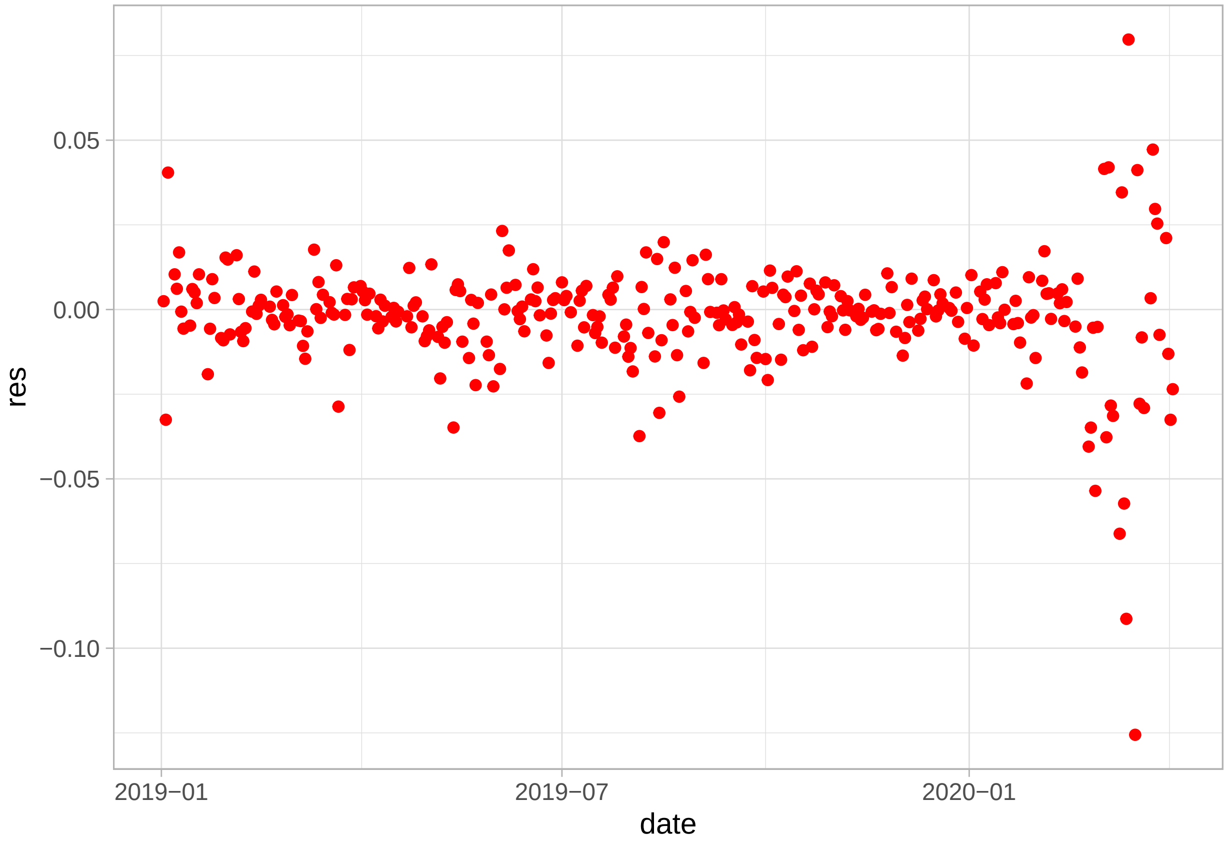
```
model.garch = ugarchspec(mean.model=list(armaOrder=c(4,3),include.mean=T, archm = FALSE, archpow = 1, a
variance.model=list(model='sGARCH',garchOrder=c(1,1), submodel = NULL, external.regressors = NULL, vari
distribution.model = "norm" )
model.garch.fit = ugarchfit(data=rt_2019_2020, spec=model.garch,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

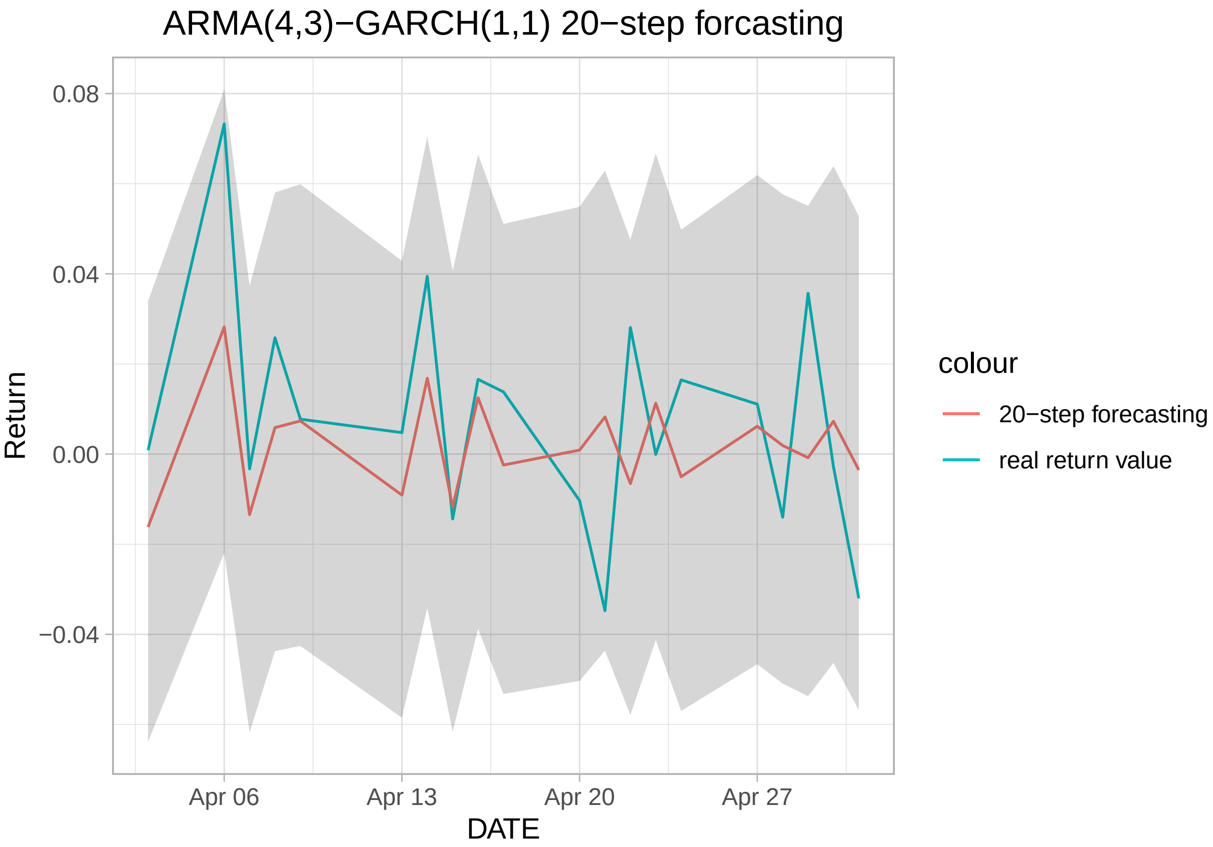


```
ggplot(aes(date,res),data=Garch_2019_2020_summary_train)+
 geom_point(col='red')+
 theme_light()
```

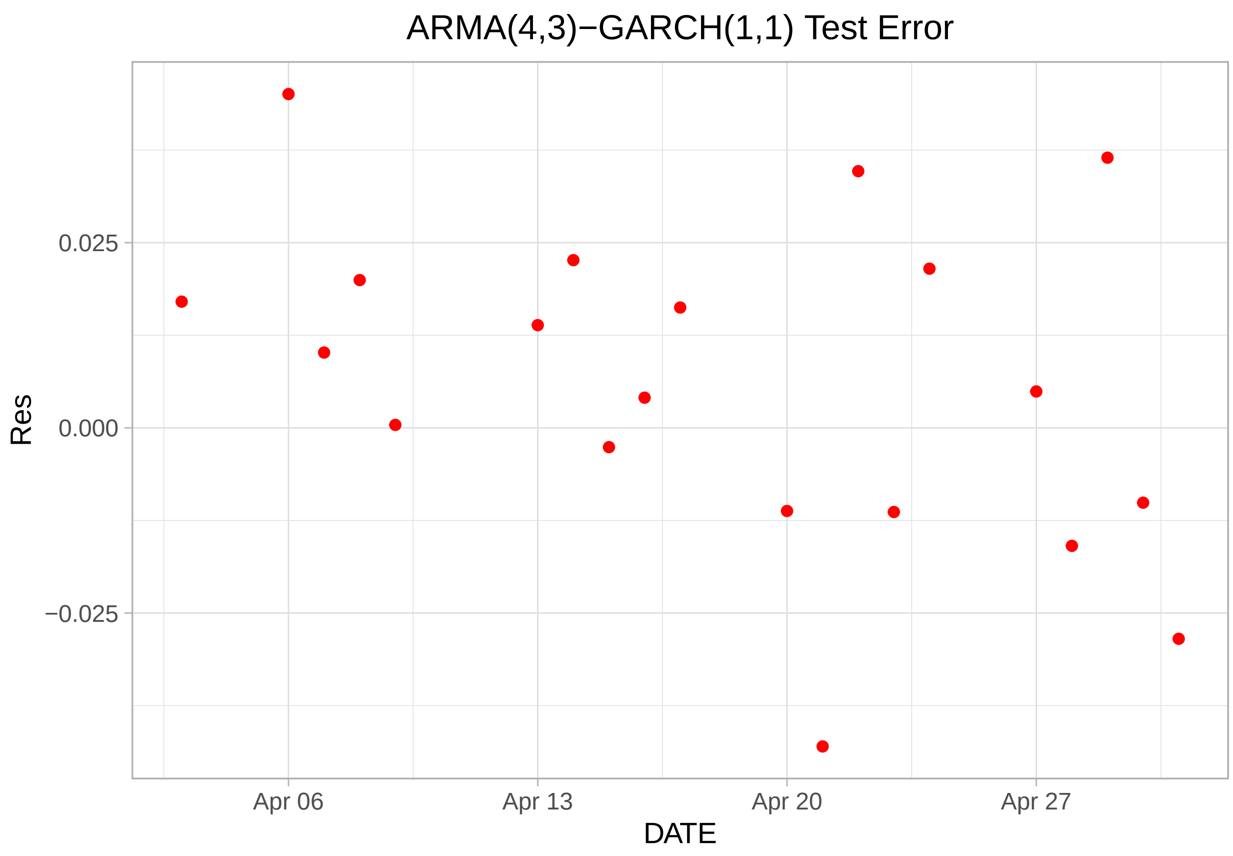


Garch forecast

```
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