ORIE5550_HW6_Markdown

Luis Alonso Cendra Villalobos (lc2234)

2024-03-18

```
#install.packages("tidyverse")

library(astsa)
```

```
## Warning: package 'astsa' was built under R version 4.3.2
```

```
library(perARMA)
```

```
## Warning: package 'perARMA' was built under R version 4.3.3
```

```
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.3.2
```

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

```
##
## Attaching package: 'forecast'
```

```
## The following object is masked from 'package:astsa':
##
##     gas
```

```
library(urca)
```

```
## Warning: package 'urca' was built under R version 4.3.2
```

```
library(xts)
```

```
## Warning: package 'xts' was built under R version 4.3.2
```

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

```
## Warning: package 'zoo' was built under R version 4.3.2
```

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

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

```
library(fGarch)
```

```
## Warning: package 'fGarch' was built under R version 4.3.2

## 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")
```

```
library(tseries)
```

```
## Warning: package 'tseries' was built under R version 4.3.2
```

```
library(FinTS)
```

```
## Warning: package 'FinTS' was built under R version 4.3.3

##
## Attaching package: 'FinTS'

## The following object is masked from 'package:forecast':
##
##     Acf
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.3.2

## Warning: package 'ggplot2' was built under R version 4.3.3

## Warning: package 'tibble' was built under R version 4.3.2

## Warning: package 'tidyr' was built under R version 4.3.2

## Warning: package 'readr' was built under R version 4.3.2

## Warning: package 'purrr' was built under R version 4.3.2

## Warning: package 'dplyr' was built under R version 4.3.2
```

```
## Warning: package 'stringr' was built under R version 4.3.2

## Warning: package 'forcats' was built under R version 4.3.2

## Warning: package 'lubridate' was built under R version 4.3.2

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2

## -- 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()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(forecast)
```

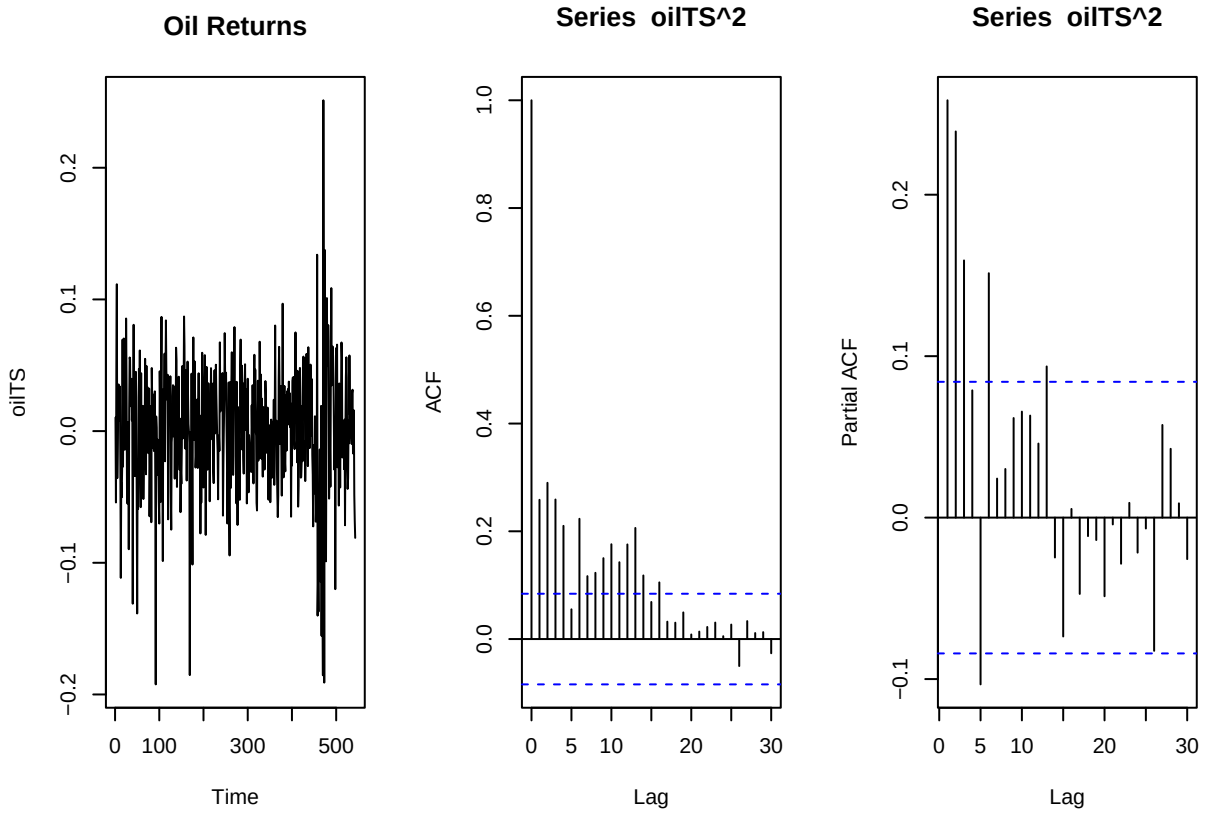
Problem 3

Consider weekly West Texas Intermediate (WTI) crude oil spot price oil in the R package astsa. Do the following.

(a) Compute log return of the price; Produce a time plot and sample ACF and PACF of the squared log return; Fit ARMA(p ,q) and use the residual to perform the ARCH effect test and normality test;

```r
oilTS <- diff(log(oil))[-1]

par(mfrow=c(1,3))
plot.ts(oilTS, main="Oil Returns")
acf(oilTS^2,lag.max=30)
pacf(oilTS^2,lag.max=30)
```




```
fit1 <- auto.arima(oilTS,max.p=5,max.q=5,
                   allowdrift=TRUE,allowmean=TRUE,ic="aic")
fit1
```

```
## Series: oilTS
## ARIMA(1,0,1) with zero mean
##
## Coefficients:
##           ar1     ma1
##       -0.5223  0.7116
## s.e.   0.0879  0.0693
##
## sigma^2 = 0.002116:  log likelihood = 902.44
## AIC=-1798.87   AICc=-1798.83   BIC=-1785.98
```

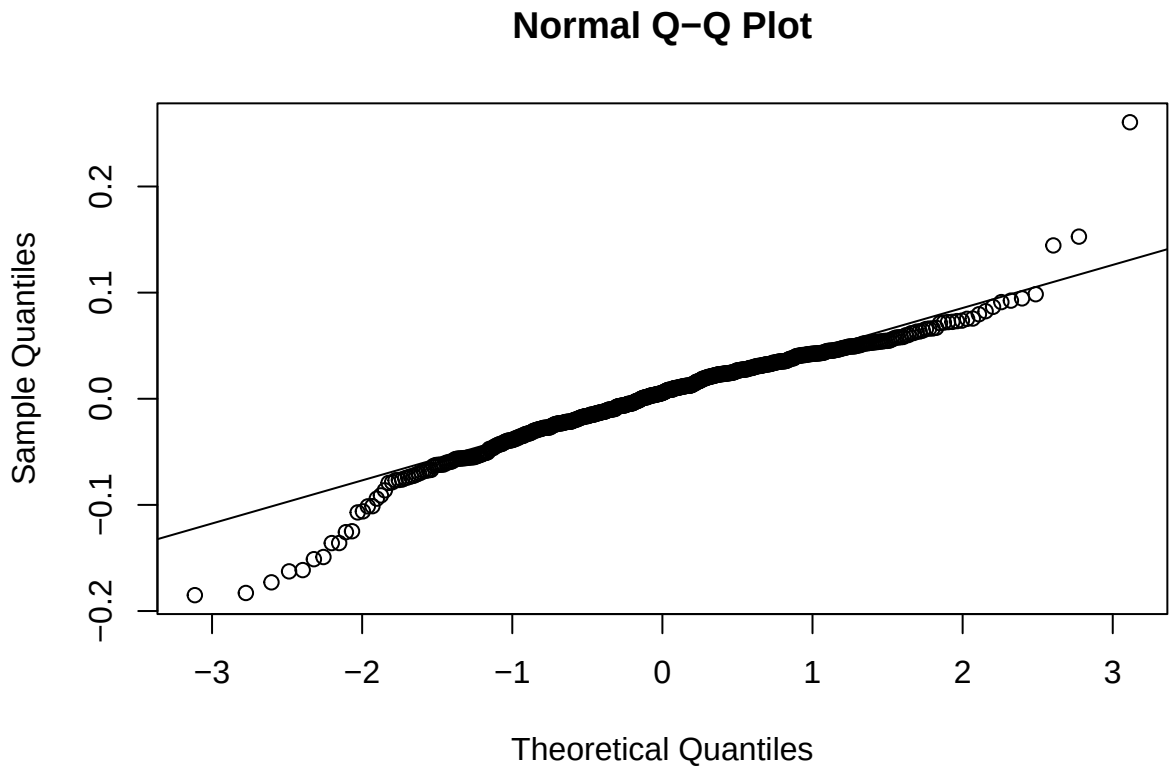
```
summary(fit1)
```

```
## Series: oilTS
## ARIMA(1,0,1) with zero mean
##
## Coefficients:
##           ar1     ma1
##       -0.5223  0.7116
## s.e.   0.0879  0.0693
##
```

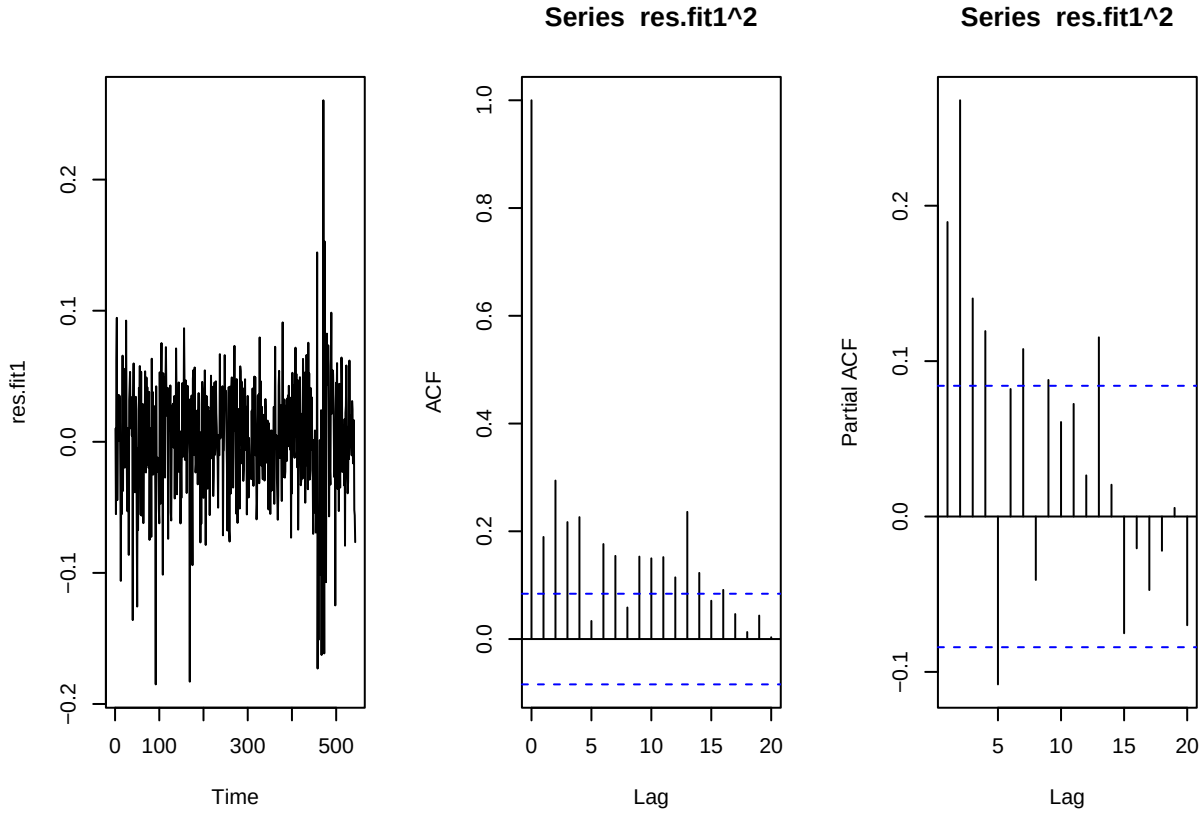
```
## sigma^2 = 0.002116:  log likelihood = 902.44
## AIC=-1798.87   AICc=-1798.83   BIC=-1785.98
##
## Training set error measures:
##                       ME       RMSE        MAE MPE MAPE      MASE        ACF1
## Training set 0.001584466 0.04591465 0.03463751 Inf  Inf 0.7640771 -0.01836772
```

```
res.fit1 <- fit1$residuals

par(mfrow=c(1,1))
qqnorm(res.fit1)
qqline(res.fit1)
```




```
par(mfrow=c(1,3))
plot.ts(res.fit1)
acf(res.fit1^2,lag.max=20)
pacf(res.fit1^2,lag.max=20)
```




```
#  Use Ljung-Box test for r_t^2
Box.test(res.fit1^2,lag=20,type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  res.fit1^2
## X-squared = 248.74, df = 20, p-value < 2.2e-16
```

```
# ARCH effect on residuals
ArchTest(res.fit1,lag=20)
```

```
##
##  ARCH LM-test; Null hypothesis: no ARCH effects
##
## data:  res.fit1
## Chi-squared = 106.45, df = 20, p-value = 8.677e-14
```

From the ARCH-effect and Box-Ljung tests, we can reject the null hypothesis of no ARCH effect, thus the series does present an autocorrelation structure in its residuals up to lag 20, this can also be seen from the ACF and PACF. Thus, normality is also rejected.

(b) Fit ARCH(p) models with p = 1,...,4 for the residual; Choose the best model from aic or bic;

```
arch1 <- garchFit(~garch(1,0),data=oilTS,include.mean=FALSE,trace=FALSE)
arch1@formula
```

```
## data ~ garch(1, 0)
## attr(,"data")
## [1] "data = oilTS"
## <environment: 0x0000024a31b18520>
```

```
arch1@fit$ics
```

```
##       AIC       BIC       SIC      HQIC
## -3.329109 -3.313282 -3.329136 -3.322921
```

```
arch1@fit$llh
```

```
## LogLikelihood
##     -905.8532
```

```
arch2 <- garchFit(~garch(2,0),data=oilTS,include.mean=FALSE,trace=FALSE)
arch2@formula
```

```
## data ~ garch(2, 0)
## attr(,"data")
## [1] "data = oilTS"
## <environment: 0x0000024a389987d0>
```

```
arch2@fit$ics
```

```
##       AIC       BIC       SIC      HQIC
## -3.358427 -3.334686 -3.358488 -3.349144
```

```
arch2@fit$llh
```

```
## LogLikelihood
##     -914.8129
```

```
arch3 <- garchFit(~garch(3,0),data=oilTS,include.mean=FALSE,trace=FALSE)
arch3@formula
```

```
## data ~ garch(3, 0)
## attr(,"data")
## [1] "data = oilTS"
## <environment: 0x0000024a39055030>
```

```
arch3@fit$ics
```

```
##       AIC       BIC       SIC      HQIC
## -3.360996 -3.329342 -3.361104 -3.348619
```

```
arch3@fit$llh
```

```
## LogLikelihood
##     -916.5105
```

```
arch4 <- garchFit(~garch(4,0),data=oilTS,include.mean=FALSE,trace=FALSE)
arch4@formula
```

```
## data ~ garch(4, 0)
## attr(,"data")
## [1] "data = oilTS"
## <environment: 0x0000024a39908840>
```

```
arch4@fit$ics
```

```
##       AIC       BIC       SIC      HQIC
## -3.359388 -3.319820 -3.359556 -3.343917
```

```
arch4@fit$llh
```

```
## LogLikelihood
##     -917.0739
```

All models show have very similar information criteria, the largest log-likelihood is $ARCH(1)$, but this model also has a relatively lower AIC and BIC criteria coefficients. Guiding ourselves with the AIC and BIC criteria, we choose the models $ARCH(2)$ and $ARCH(3)$, with the former having a slightly larger log-likelihood. Thus, we choose the model $ARCH(2)$. We can further support this decision in terms of model parsimony.

(c) Find the best GARCH(p,q) model among (p,q) = (1,1),(1,2),(2,1),(2,2), chosen by using aic or bic; Compare with the ARCH(p) model chosen in (b) to determine the final model in terms of the aic or bic; Conclude whether GARCH(p,q) seems the better;

```
garch1 <- garchFit(~garch(1,1),data=oilTS,include.mean=FALSE,trace=FALSE)
garch1@formula
```

```
## data ~ garch(1, 1)
## attr(,"data")
## [1] "data = oilTS"
## <environment: 0x0000024a3b406700>
```

```
garch1@fit$ics
```

```
##       AIC       BIC       SIC      HQIC
## -3.391073 -3.367333 -3.391134 -3.381791
```

```
garch1@fit$llh
```

```
## LogLikelihood
##     -923.6764
```

```
garch2 <- garchFit(~garch(1,2),data=oilTS,include.mean=FALSE,trace=FALSE)
garch2@formula
```

```
## data ~ garch(1, 2)
## attr(,"data")
## [1] "data = oilTS"
## <environment: 0x0000024a36774818>
```

```
garch2@fit$ics
```

```
##       AIC       BIC       SIC      HQIC
## -3.387840 -3.356186 -3.387948 -3.375463
```

```
garch2@fit$llh
```

```
## LogLikelihood
##     -923.7986
```

```
garch3 <- garchFit(~garch(2,1),data=oilTS,include.mean=FALSE,trace=FALSE)
garch3@formula
```

```
## data ~ garch(2, 1)
## attr(,"data")
## [1] "data = oilTS"
## <environment: 0x0000024a359040b0>
```

```
garch3@fit$ics
```

```
##       AIC       BIC       SIC      HQIC
## -3.387835 -3.356181 -3.387943 -3.375458
```

```
garch3@fit$llh
```

```
## LogLikelihood
##     -923.7973
```

```
garch4 <- garchFit(~garch(2,2),data=oilTS,include.mean=FALSE,trace=FALSE)
garch4@formula
```

```
## data ~ garch(2, 2)
## attr(,"data")
## [1] "data = oilTS"
## <environment: 0x0000024a388bfd50>
```

```
garch4@fit$ics
```

```
##       AIC       BIC       SIC      HQIC
## -3.385504 -3.345935 -3.385671 -3.370032
```

```
garch4@fit$llh
```

```
## LogLikelihood
##     -924.1642
```

Again, all models show have very similar information criteria and log-likelihood. With these minimal differences at hand, the criteria slightly favour the model $Garch(1,1)$. And, as before, model parsimony supports this decision.

Comparing both conditional variance models, we choose $GARCH(1,1)$ based on AIC and BIC.

(d) Refit the data to the ARMA(p ,q)-GARCH(p,q) model for the log return; Perform the model diagnostic;

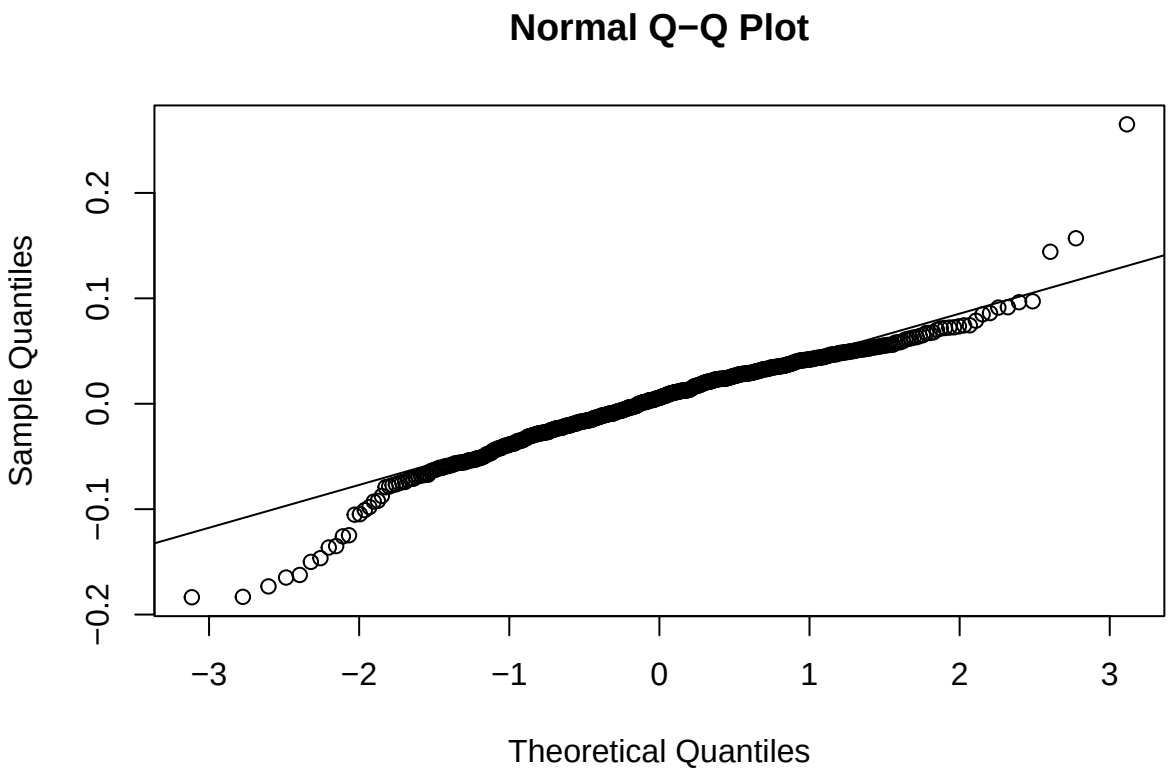
```
fit2 <- garchFit(~arma(1,1)+garch(1,1),data=oilTS,include.mean=FALSE,trace=FALSE)
summary(fit2)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~arma(1, 1) + garch(1, 1), data = oilTS, include.mean = FALSE,
##     trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ arma(1, 1) + garch(1, 1)
## <environment: 0x0000024a35eb2360>
##  [data = oilTS]
##
## Conditional Distribution:
##  norm
##
## Coefficient(s):
##         ar1          ma1        omega       alpha1        beta1
## -0.44876359   0.64680200   0.00010801   0.06295735   0.88087985
```

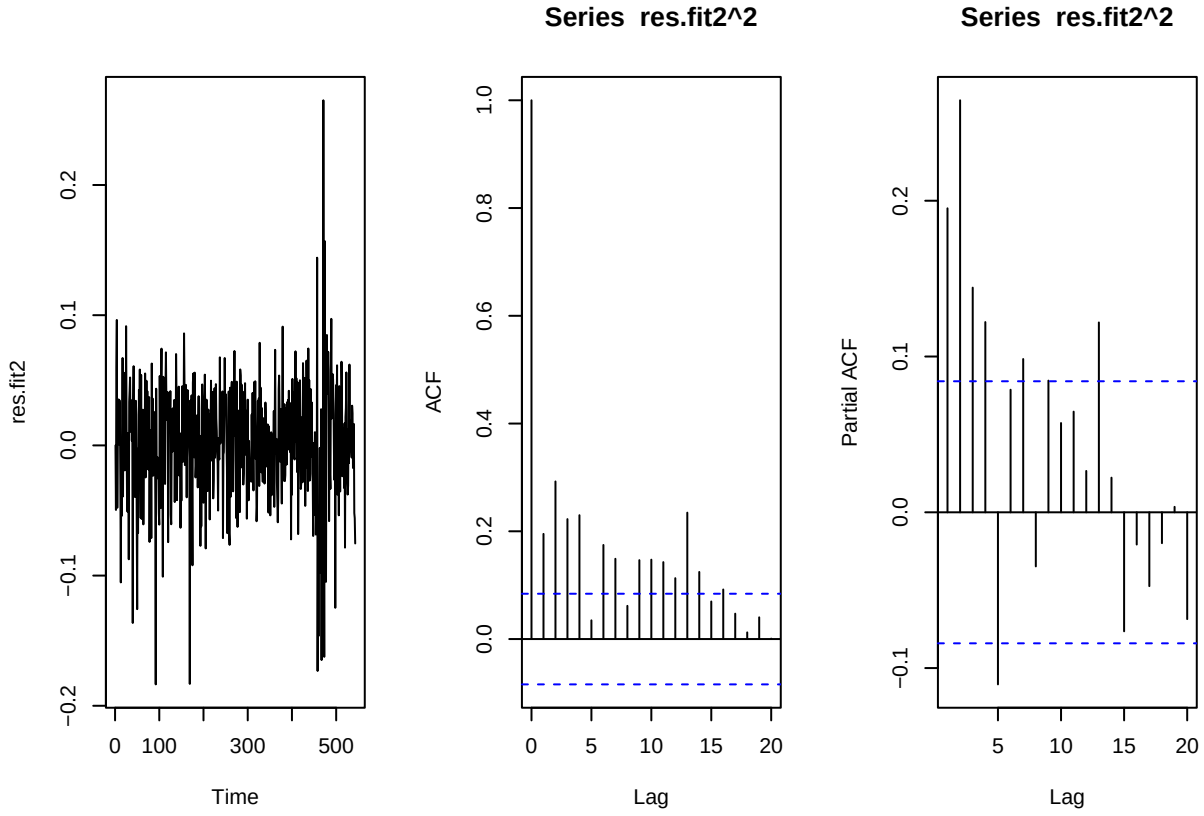
```
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##          Estimate  Std. Error  t value Pr(>|t|)
## ar1    -4.488e-01   1.085e-01   -4.137 3.51e-05 ***
## ma1     6.468e-01   9.109e-02    7.101 1.24e-12 ***
## omega   1.080e-04   4.878e-05    2.214 0.026826 *
## alpha1  6.296e-02   1.741e-02    3.616 0.000299 ***
## beta1   8.809e-01   3.452e-02   25.520  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  935.6933    normalized:  1.723192
##
## Description:
##  Sat Mar 30 16:20:57 2024 by user: alons
##
##
## Standardised Residuals Tests:
##                                 Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  119.7198110 0.000000e+00
##  Shapiro-Wilk Test  R    W        0.9718829 1.067562e-08
##  Ljung-Box Test     R    Q(10)    9.6204231 4.744015e-01
##  Ljung-Box Test     R    Q(15)   15.8147795 3.944673e-01
##  Ljung-Box Test     R    Q(20)   21.2258932 3.839455e-01
##  Ljung-Box Test     R^2  Q(10)    7.1739881 7.089237e-01
##  Ljung-Box Test     R^2  Q(15)   11.6401281 7.060389e-01
##  Ljung-Box Test     R^2  Q(20)   13.0453577 8.754295e-01
##  LM Arch Test       R    TR^2     7.2804036 8.385406e-01
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.427968 -3.388400 -3.428135 -3.412497
```

```
res.fit2 <- fit2@residuals

par(mfrow=c(1,1))
qqnorm(res.fit2)
qqline(res.fit2)
```




```
par(mfrow=c(1,3))
plot.ts(res.fit2)
acf(res.fit2^2,lag.max=20)
pacf(res.fit2^2,lag.max=20)
```




```
#  Use Ljung-Box test for r_t^2
Box.test(res.fit2^2,lag=20,type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  res.fit2^2
## X-squared = 247.17, df = 20, p-value < 2.2e-16
```

```
# ARCH effect on residuals
ArchTest(res.fit2,lag=20)
```

```
##
##  ARCH LM-test; Null hypothesis: no ARCH effects
##
## data:  res.fit2
## Chi-squared = 106.18, df = 20, p-value = 9.735e-14
```

(e) By using p ,q ,p,q decided above, fit ARMA(p ,q)-apARCH(p,q) model for the log return; Write down the exact model that was fit to the series; Conclude whether asymmetry with respect to negative and positive disturbances should be considered.

```r
ap.fit <- garchFit(~arma(1,1)+aparch(1,1), data=oilTS,cond.dist='std',trace=FALSE)
summary(ap.fit)
```

```
##
## Title:
##  GARCH Modelling
##
## Call:
##  garchFit(formula = ~arma(1, 1) + aparch(1, 1), data = oilTS,
##     cond.dist = "std", trace = FALSE)
##
## Mean and Variance Equation:
##  data ~ arma(1, 1) + aparch(1, 1)
## <environment: 0x0000024a39eb9d10>
##  [data = oilTS]
##
## Conditional Distribution:
##  std
##
## Coefficient(s):
##         mu         ar1         ma1       omega      alpha1      gamma1
##  0.0054917  -0.4649722   0.6573130   0.0013311   0.0426392   1.0000000
##      beta1       delta       shape
##  0.9025692   1.1884599   9.0509703
##
## Std. Errors:
##  based on Hessian
##
## Error Analysis:
##         Estimate  Std. Error  t value Pr(>|t|)
## mu      0.0054917   0.0029437    1.866 0.062100 .
## ar1    -0.4649722   0.1055031   -4.407 1.05e-05 ***
## ma1     0.6573130   0.0880603    7.464 8.37e-14 ***
## omega   0.0013311   0.0005399    2.465 0.013690 *
## alpha1  0.0426392   0.0176556    2.415 0.015733 *
## gamma1  1.0000000   0.0626540   15.961  < 2e-16 ***
## beta1   0.9025692   0.0318932   28.300  < 2e-16 ***
## delta   1.1884599   0.5549375    2.142 0.032225 *
## shape   9.0509703   2.7316136    3.313 0.000922 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Log Likelihood:
##  936.5509    normalized:  1.724771
##
## Description:
##  Sat Mar 30 16:20:57 2024 by user: alons
##
##
## Standardised Residuals Tests:
##                                 Statistic      p-Value
##  Jarque-Bera Test   R    Chi^2  289.5942411 0.000000e+00
##  Shapiro-Wilk Test  R    W        0.9540087 5.687783e-12
```

```
##  Ljung-Box Test     R    Q(10)    7.5082142 6.767516e-01
##  Ljung-Box Test     R    Q(15)   12.0297584 6.767752e-01
##  Ljung-Box Test     R    Q(20)   16.2452286 7.012989e-01
##  Ljung-Box Test     R^2  Q(10)   34.0263662 1.828084e-04
##  Ljung-Box Test     R^2  Q(15)   39.1882501 6.006290e-04
##  Ljung-Box Test     R^2  Q(20)   39.9603141 5.053435e-03
##  LM Arch Test       R    TR^2    19.0379000 8.761791e-02
##
## Information Criterion Statistics:
##       AIC       BIC       SIC      HQIC
## -3.416394 -3.345171 -3.416931 -3.388545
```

```
mu <- ap.fit@fit$coef[1]
ar1 <- ap.fit@fit$coef[2]
ma1 <- ap.fit@fit$coef[3]
omega <- ap.fit@fit$coef[4]
alpha1 <- ap.fit@fit$coef[5]
gamma1 <- ap.fit@fit$coef[6]
beta1 <- ap.fit@fit$coef[7]
delta <- ap.fit@fit$coef[8]
shape <-ap.fit@fit$coef[9]
```

The exact model is:

$$X_t - 0.005 = -0.465(X_{t-1} - 0.005) + h_t + 0.657h_{t-1} \text{ where } \{h_t : t \in \mathbb{Z}\} \sim GARCH(1,1)$$
$$h_t^{1.188/2} = 0.001 + 0.043(|r_{t-1}| - 1r_{t-1})^{1.188} + 0.903h_{t-1}^{1.188/2}$$
$$r_t = \sqrt{h_t}e_t \text{ where } \{e_t\} \overset{i.i.d.}{\sim} \mathcal{N}(0,1)$$

Given that the leverage parameter $\gamma_1 > 0$ then, past negative shocks have a deeper impact on current conditional variance than past positive shocks.

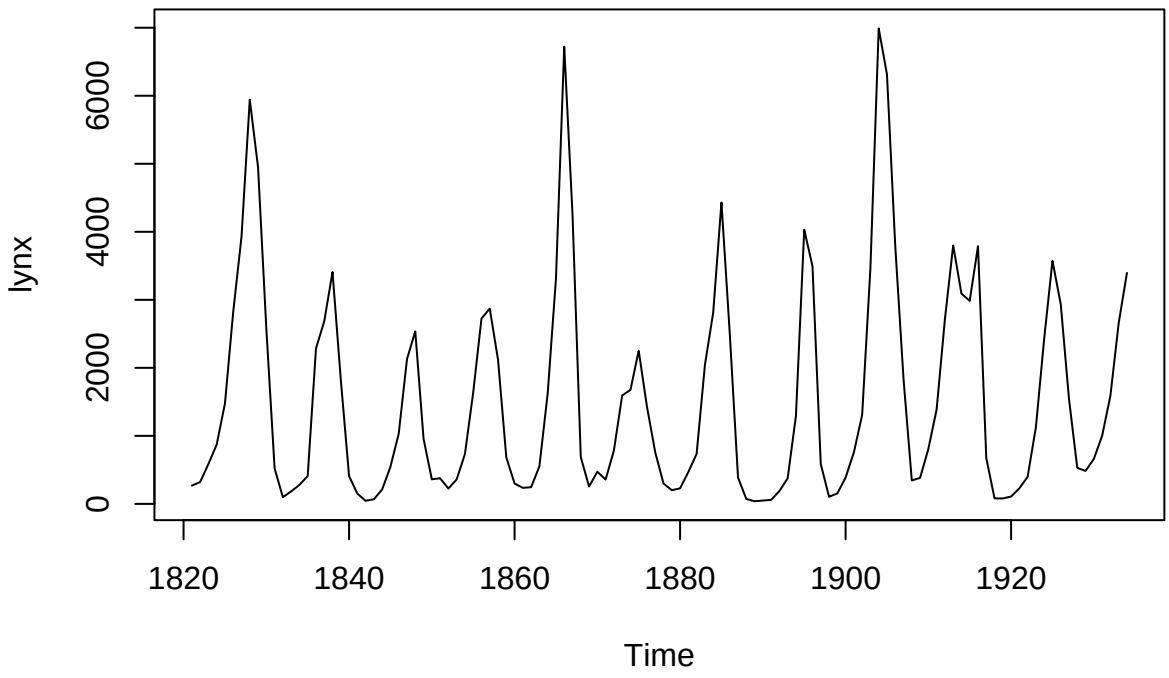
Problem 4

Consider annual numbers of lynx trappings for 1821–1934 in Canada lynx in the R package forecast. Do the following.

(a) Produce and include a time plot of the series; Hold out the last 14 observations for test data; For the remaining data, use set.seed(123) and fit the NNAR(p,k) model through nnetar (set P = 0);

```
set.seed(123)

plot(lynx,type="l")
```




```
# Leave out the last 14 observations as test data
training_data <- head(lynx,-14)
test_data <- tail(lynx, 14)

fit_lynx_a <- nnetar(training_data, P = 0) #Number of seasonal lags used as inputs
fit_lynx_a
```

```
## Series: training_data
## Model:  NNAR(8,4)
## Call:   nnetar(y = training_data, P = 0)
##
## Average of 20 networks, each of which is
## a 8-4-1 network with 41 weights
## options were - linear output units
##
## sigma^2 estimated as 76943
```

```
# RMSE for whole data should be better
```

(b) Compute the root mean square forecasting error for the one-step ahead using time series cross-validation for NNAR(p,k), p = 1,...,5, models (set P = 0); Conclude which of these models (i.e., which p) has the smallest error;

```r
set.seed(123)

#' param p Embedding dimension for non-seasonal time series. Number of ' non-seasonal lags used as inpu

for (i in c(1:5)){
  cat('Model: p = ', i, "\n")
  fcast_lynx <- function(x, h){forecast(nnetar(training_data, p = i, P = 0),h=1)}
  e1 <- tsCV(training_data, fcast_lynx, h=1)
  cat("RMSE: ", sqrt(mean(e1^2, na.rm=TRUE)), "\n") #
}
```

```
## Model: p =  1
## RMSE:  2049.446
## Model: p =  2
## RMSE:  2022.467
## Model: p =  3
## RMSE:  1931.996
## Model: p =  4
## RMSE:  1959.805
## Model: p =  5
## RMSE:  1970.548
```

```r
fit_lynx_b <- nnetar(training_data, p = 3, P = 0) # cross-validation tscv function three inputs time se
fit_lynx_b
```

```
## Series: training_data
## Model:  NNAR(3,2)
## Call:   nnetar(y = training_data, p = 3, P = 0)
##
## Average of 20 networks, each of which is
## a 3-2-1 network with 11 weights
## options were - linear output units
##
## sigma^2 estimated as 433038
```

The smallest RMSE for the one-step forecasts is achieved by setting $p = 3$ as the optimal number of seasonal lags used as inputs, resulting in a model $NNAR(3, 2)$.

(c) Compare the squared root of the mean squared error for the model from (a) and the model chosen from (b); Conclude which one has a larger estimation error; Draw the time series plot of lynx and overlap the two fitted lines with different colors;

```r
set.seed(123)

# training data vs fitted
sqrt(mean((training_data - fit_lynx_a$fitted)^2, na.rm=TRUE)) # model a
```

```
## [1] 277.3855
```

```r
sqrt(mean((training_data - fit_lynx_b$fitted)^2, na.rm=TRUE)) # model b
```

```
## [1] 658.0564
```

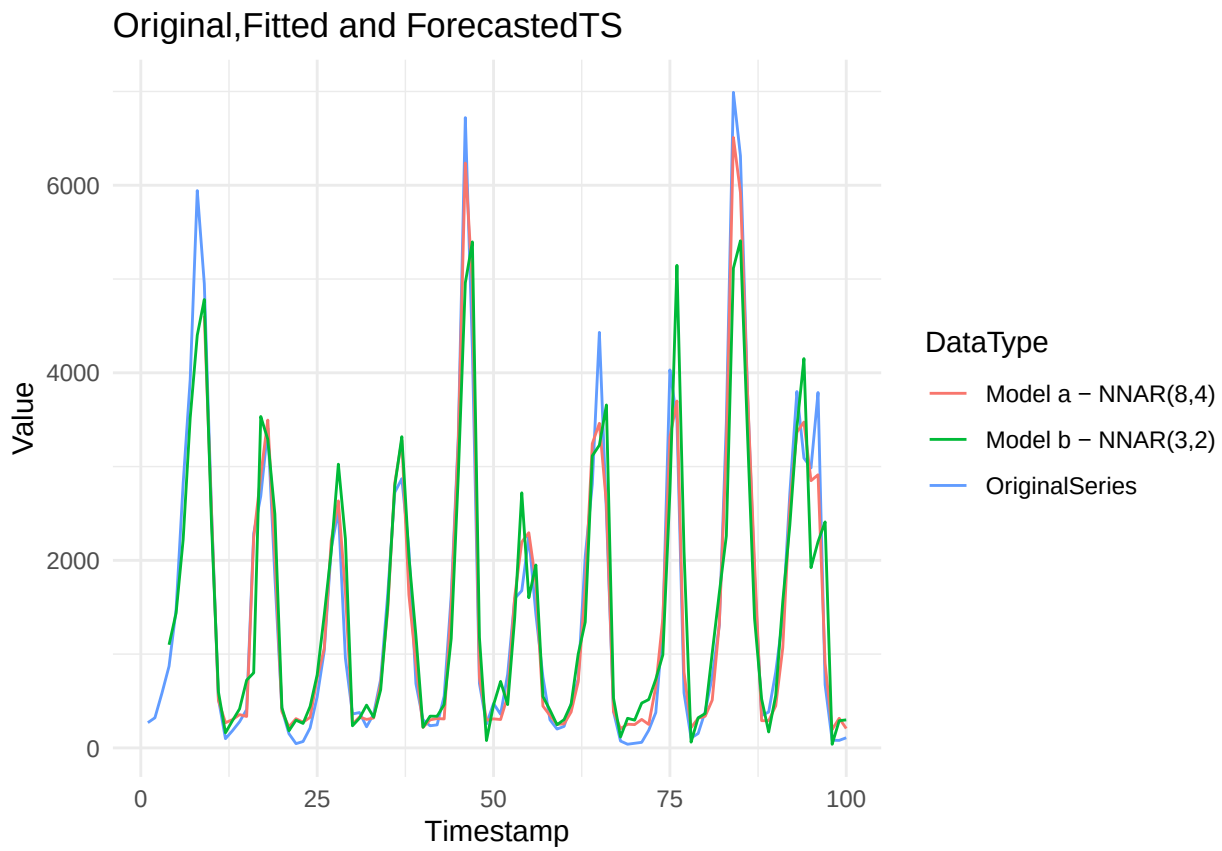
```r
library(tidyr)
df1 <-data.frame(original_series = training_data, timestamp= seq(1,100,1))
df2 <-data.frame(model_a = fit_lynx_a$fitted, timestamp= seq(1,100,1))
df3 <-data.frame(model_b = fit_lynx_b$fitted, timestamp= seq(1,100,1))

merged_df <-merge(df1,df2, by= "timestamp", all=TRUE)
merged_df <-merge(merged_df,df3, by= "timestamp",all= TRUE)

ggplot(merged_df, aes(x= timestamp)) +
  geom_line(aes(y= original_series, color= "OriginalSeries")) +
  geom_line(aes(y= model_a, color= "Model a - NNAR(8,4)")) +
  geom_line(aes(y= model_b, color= "Model b - NNAR(3,2)"))+
  labs(x= "Timestamp", y= "Value",color= "DataType")+
  ggtitle("Original,Fitted and ForecastedTS") +
  theme_minimal()
```

```
## Warning: Removed 8 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## Warning: Removed 3 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



The RMSE for the one-step forecasts are lower for the model chosen in (a), which is an $NNAR(8,4)$.

(d) For the test data, compare the square root of the mean forecasting error from (a) and the model chosen from (b); Conclude which one has a larger forecasting error; Draw the time series plot of lynx and overlap the two forecasts with different colors;

```
set.seed(123)


model_a <- forecast(fit_lynx_a,h=14,PI=FALSE)
model_b <- forecast(fit_lynx_b,h=14,PI=FALSE)

# Test data vs forecasted
sqrt(mean((test_data - model_a$mean)^2, na.rm=TRUE)) # model a
```

```
## [1] 1545.65
```

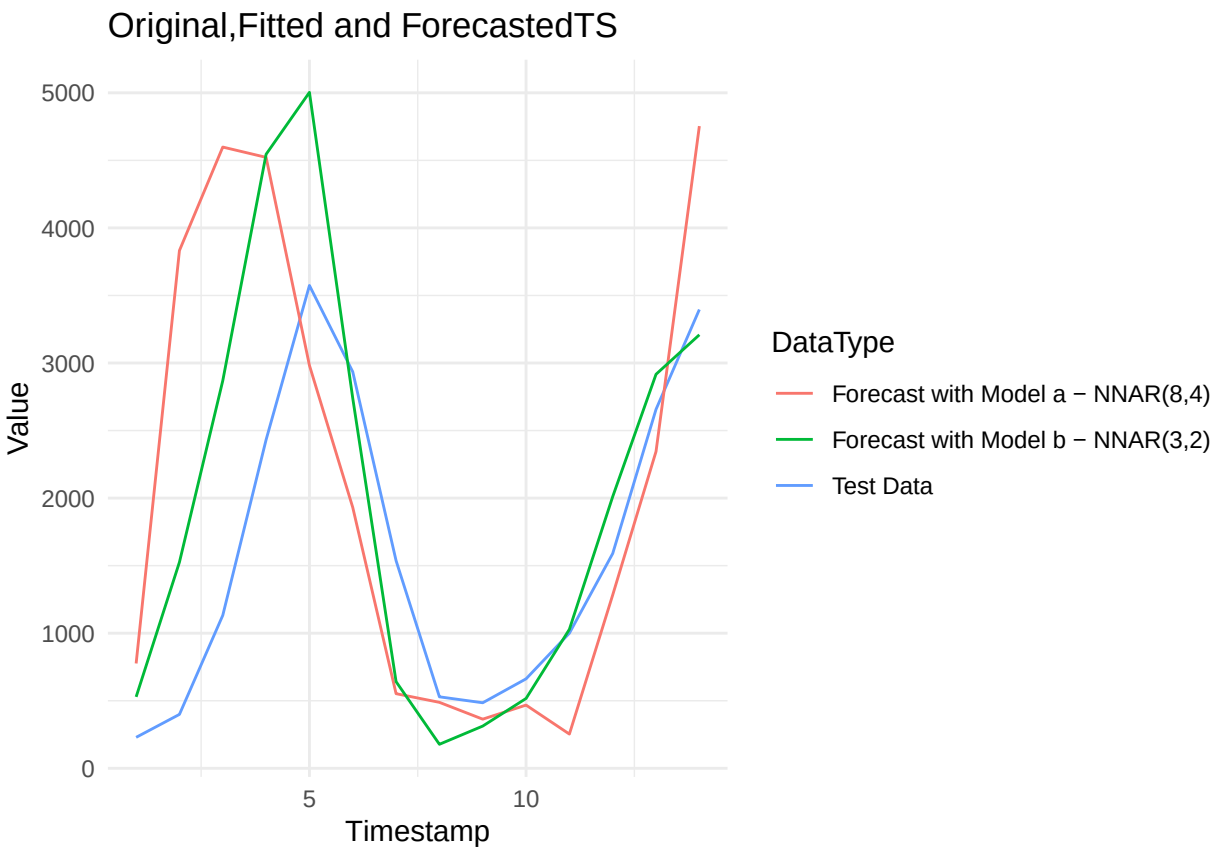
```
sqrt(mean((test_data - model_b$mean)^2, na.rm=TRUE)) # model b
```

```
## [1] 932.8031
```

```
library(tidyr)
df1 <-data.frame(original_series = test_data, timestamp= seq(1,14,1))
df2 <-data.frame(model_a = model_a$mean, timestamp= seq(1,14,1))
df3 <-data.frame(model_b = model_b$mean, timestamp= seq(1,14,1))

merged_df <-merge(df1,df2, by= "timestamp", all=TRUE)
merged_df <-merge(merged_df,df3, by= "timestamp",all= TRUE)

ggplot(merged_df, aes(x= timestamp)) +
  geom_line(aes(y= original_series, color= "Test Data")) +
  geom_line(aes(y= model_a, color= "Forecast with Model a - NNAR(8,4)")) +
  geom_line(aes(y= model_b, color= "Forecast with Model b - NNAR(3,2)"))+
  labs(x= "Timestamp", y= "Value",color= "DataType")+
  ggtitle("Original,Fitted and ForecastedTS") +
  theme_minimal()
```


Original,Fitted and ForecastedTS
5000
4000
3000
2000
1000
0
Value
5
10
Timestamp
DataType
Forecast with Model a − NNAR(8,4)
Forecast with Model b − NNAR(3,2)
Test Data