Econ144_hw5

Sijia Hua

6/3/2019

```r
library("readxl")
```

```
## Warning: package 'readxl' was built under R version 3.5.2
```

```r
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 3.5.2
```

```r
library(tseries)
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
library("fpp2")
```

```
## Loading required package: ggplot2

## Loading required package: fma

## Loading required package: expsmooth
```

```r
library(MLmetrics) # use RMSE
```

```
##
## Attaching package: 'MLmetrics'

## The following object is masked from 'package:base':
##
##     Recall
```

```r
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.5.2

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2)
library(AnalyzeTS)
```

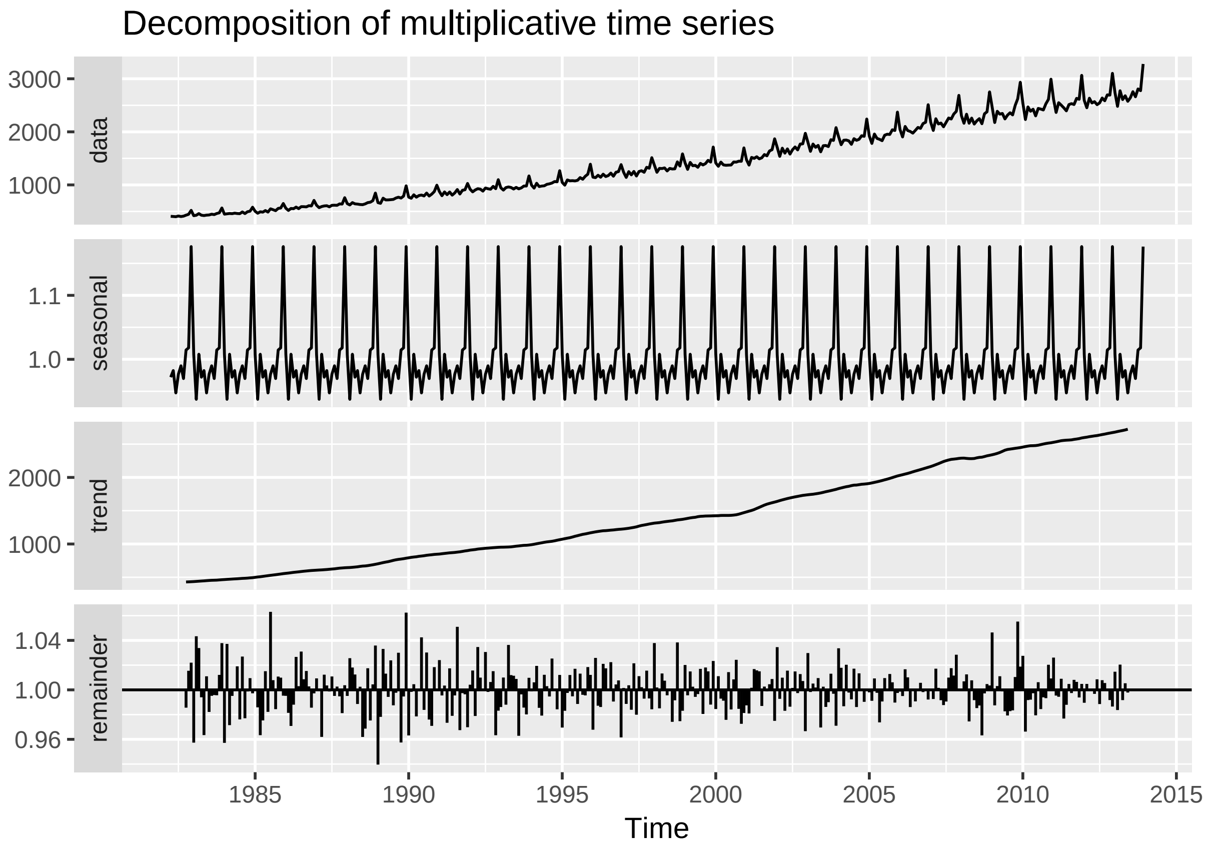
```
## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

## The following objects are masked from 'package:fma':
##
##     cement, housing, petrol

## Loading required package: TTR

## Loading required package: urca

##
## Attaching package: 'AnalyzeTS'

## The following object is masked from 'package:base':
##
##     pmax
```

```r
library(Hmisc)
```

```
## Warning: package 'Hmisc' was built under R version 3.5.2

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:dplyr':
##
##     src, summarize

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

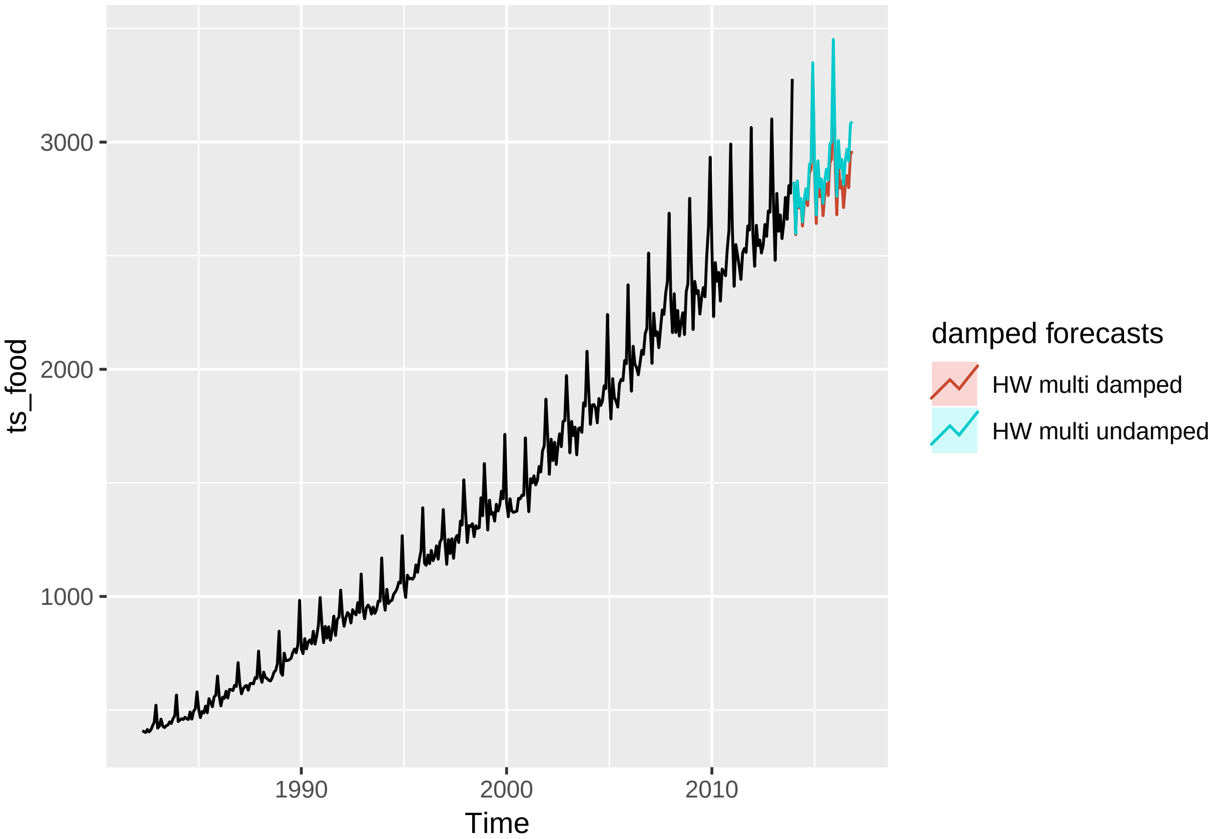
1. Problem 7.8 (i.e., Chapter 7, Problem 8) from Textbookc. Recall your retail time series data (from Exercise 3 in Section 2.10). (1)Why is multiplicative seasonality necessary for this series?

```r
setwd("/Users/Renaissance/Desktop")
retaildata <- readxl::read_excel("retail.xlsx", skip = 1)
ts_food <- ts(retaildata[,"A3349398A"],frequency=12, start=c(1982,4))

ts_food %>% decompose(type="multiplicative") %>%autoplot()
```



Yes, multiplicative seasonality is necessary. There are obvious seasonal factors. (2)Apply Holt-Winters' multiplicative method to the data. Experiment with making the trend damped.

```
# use Holt-Winters with forecast 35 steps
fit_food <- hw(ts_food, damped = TRUE, seasonal = "multiplicative", h = 35)
fit_undamp <- hw(ts_food, damped = FALSE, seasonal = "multiplicative", h = 35)
autoplot(ts_food) +
  autolayer(fit_food, series = "HW multi damped", PI = FALSE)+
  autolayer(fit_undamp, series = "HW multi undamped", PI = FALSE)+
  guides(colour=guide_legend(title="damped forecasts"))
```



(3)Compare the RMSE of the one-step forecasts from the two methods. Which do you prefer?

```
damp_onestep <- hw(ts_food, damped = TRUE, seasonal = "multiplicative", h = 1)
undamp_onestep <- hw(ts_food, damped = FALSE, seasonal = "multiplicative", h = 1)
accuracy(damp_onestep)
```

```
##                    ME     RMSE      MAE       MPE     MAPE     MASE
## Training set 4.244765 29.63087 22.26018 0.2959731 1.785469 0.292681
##                    ACF1
## Training set -0.2184672
```

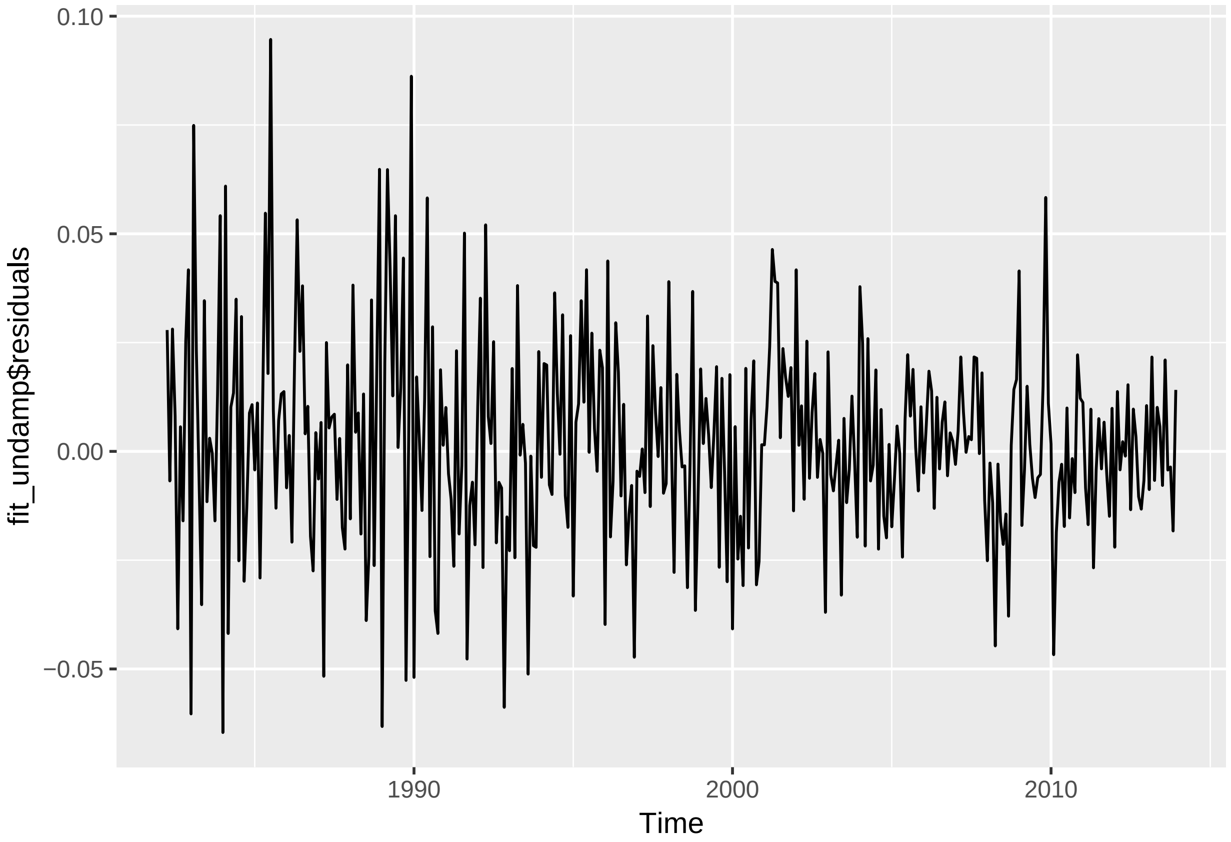
```
accuracy(undamp_onestep)
```

```
##                    ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 1.496135 29.43051 22.25676 0.1603693 1.799731 0.2926361
##                    ACF1
## Training set -0.0300701
```

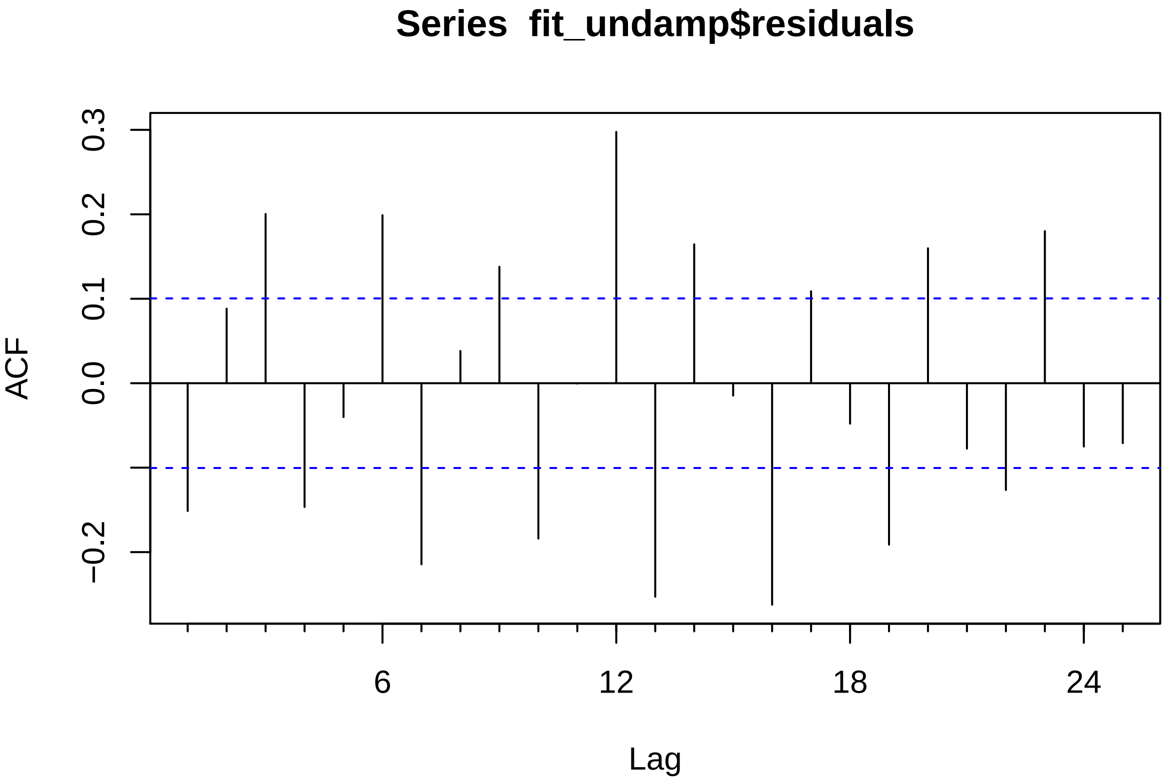
RMSE of damped is 29.63 RMSE of undamped is 29.43. Hence, from RMSE, undamped works better.

(4)Check that the residuals from the best method look like white noise.

```
autoplot(fit_undamp$residuals) #which looks like white noise
```

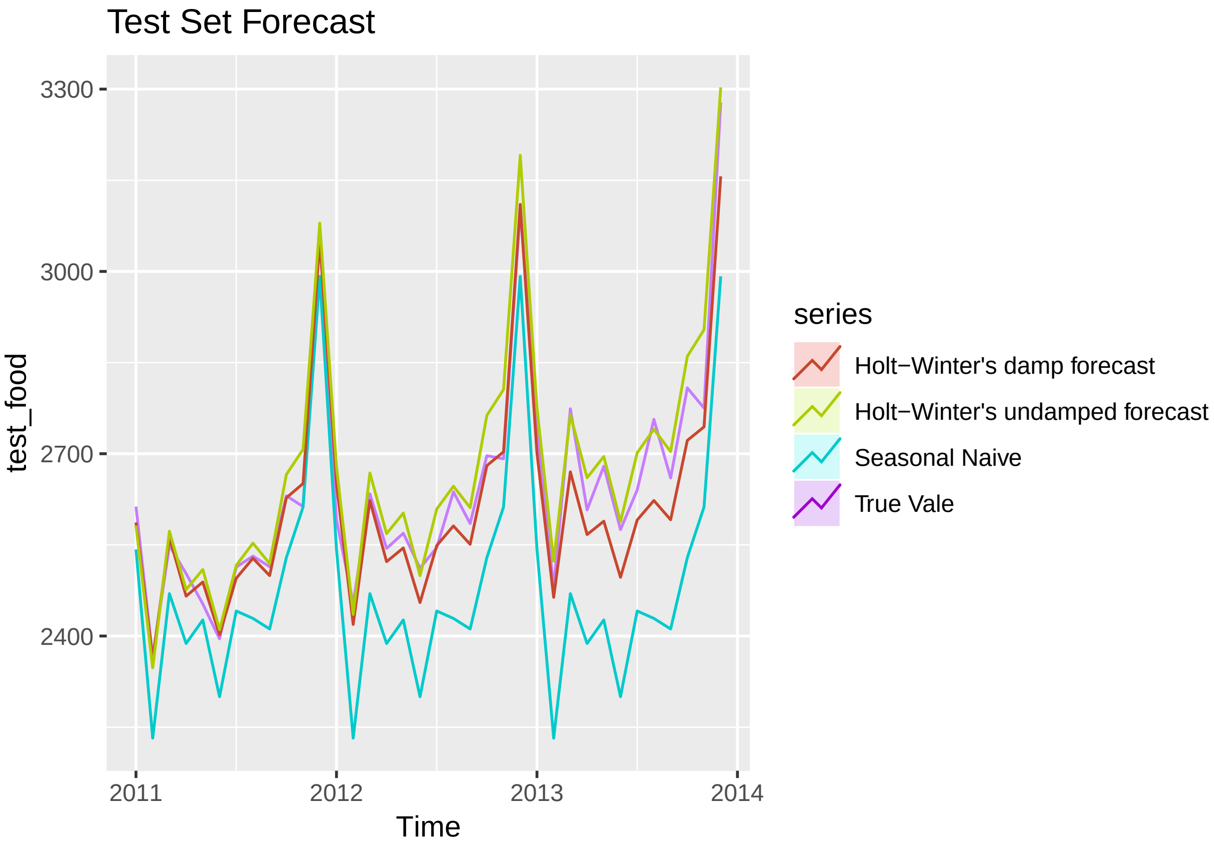


```
# check for acf
Acf(fit_undamp$residuals)
```



Residual of the best method looks like white noise, but the magnitude before 2000 seems bigger than that after 2000. After checking Acf, it is not a white noise. (5)Now find the test set RMSE, while training the model to the end of 2010. Can you beat the seasonal naïve approach from Exercise 8 in Section 3.7?

```
# set up train set and test set
train_food <- window(ts_food, end = c(2010, 12))
test_food <- window(ts_food, start = c(2011, 1))
# three forecast methods of training set
snaive_train <- snaive(train_food, h = 36)
damp_train <- hw(train_food, damped = TRUE, seasonal = "multiplicative", h = 36)
undamp_train <- hw(train_food, damped = FALSE, seasonal = "multiplicative", h = 36)

autoplot(test_food, series = "True Vale") +
  autolayer(snaive_train, series = "Seasonal Naive", PI = FALSE) +
  autolayer(damp_train, series = "Holt-Winter's damp forecast", PI = FALSE) +
  autolayer(undamp_train, series = "Holt-Winter's undamped forecast", PI = FALSE) +
  ggtitle ('Test Set Forecast')
```



```
RMSE(test_food, snaive_train$mean)
```

```
## [1] 180.1991
```

```
RMSE(test_food, damp_train$mean)
```

```
## [1] 51.78368
```

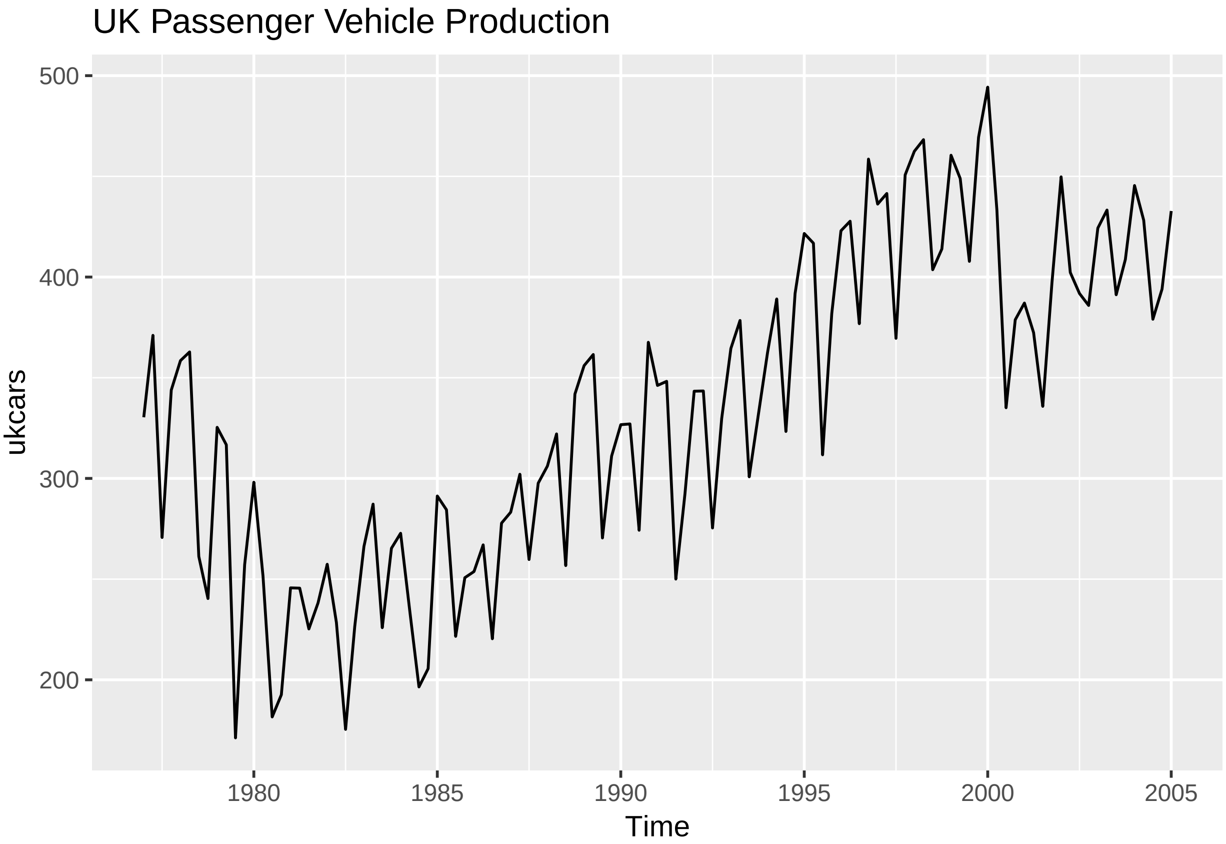
```
RMSE(test_food, undamp_train$mean)
```

```
## [1] 50.04274
```

By comparing, Undamped Holt-Winter multiplicative process gives a better approximation. It beats the seasonal naive approach.

2. Problem 7.10 (i.e., Chapter 7, Problem 10) from Textbookc. (1)For this exercise use data set ukcars, the quarterly UK passenger vehicle production data from 1977Q1–2005Q1. Plot the data and describe the main features of the series.

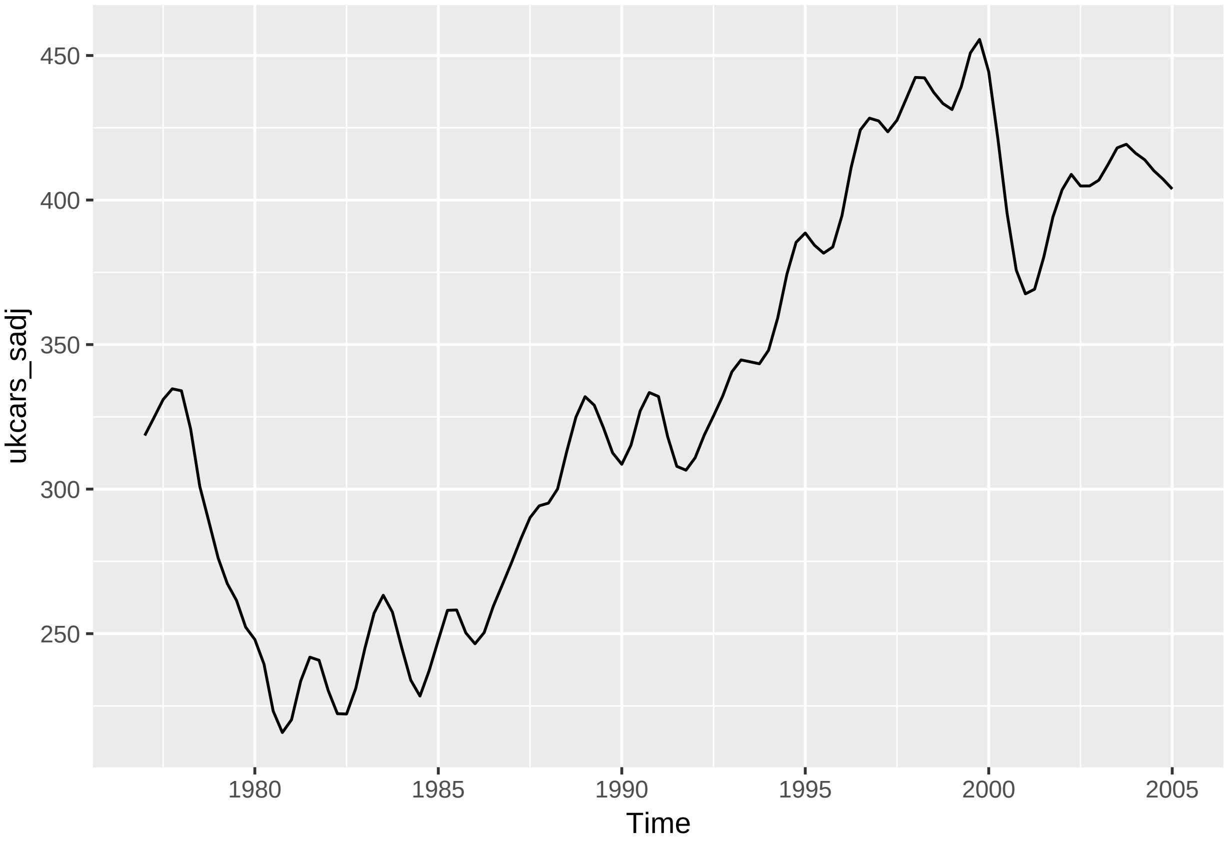
```
autoplot(ukcars) +
  ggtitle("UK Passenger Vehicle Production")
```



There is a decreasing trend before 1980, followed by a continuously increasing trend to 2000. From 2000 to 2005, some fluctuations appear. This data set has strong seasonal factors.

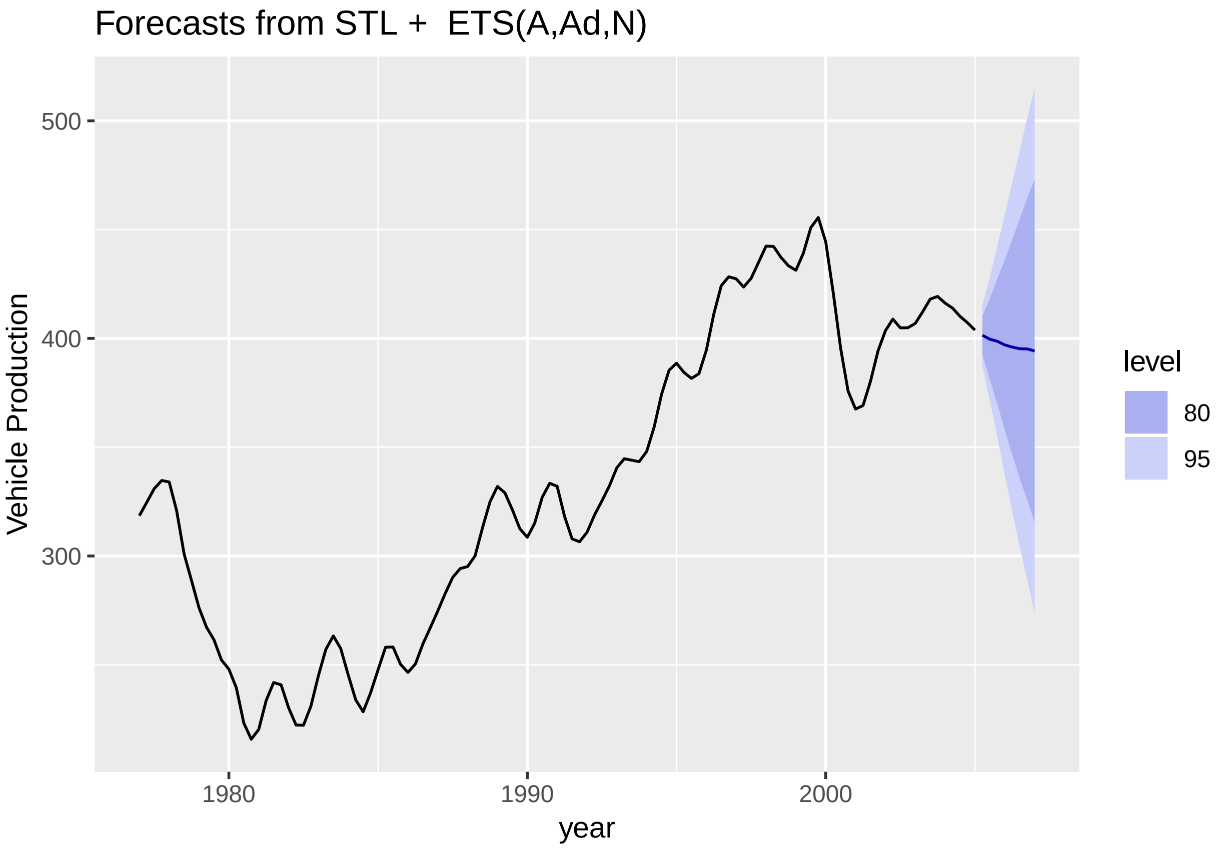
(2)Decompose the series using STL and obtain the seasonally adjusted data.

```
stl_ukcars <- stl(ukcars, s.window = "periodic")
ukcars_sadj <- stl_ukcars$time.series[,2]
autoplot(ukcars_sadj) # after seasonal adjusted
```



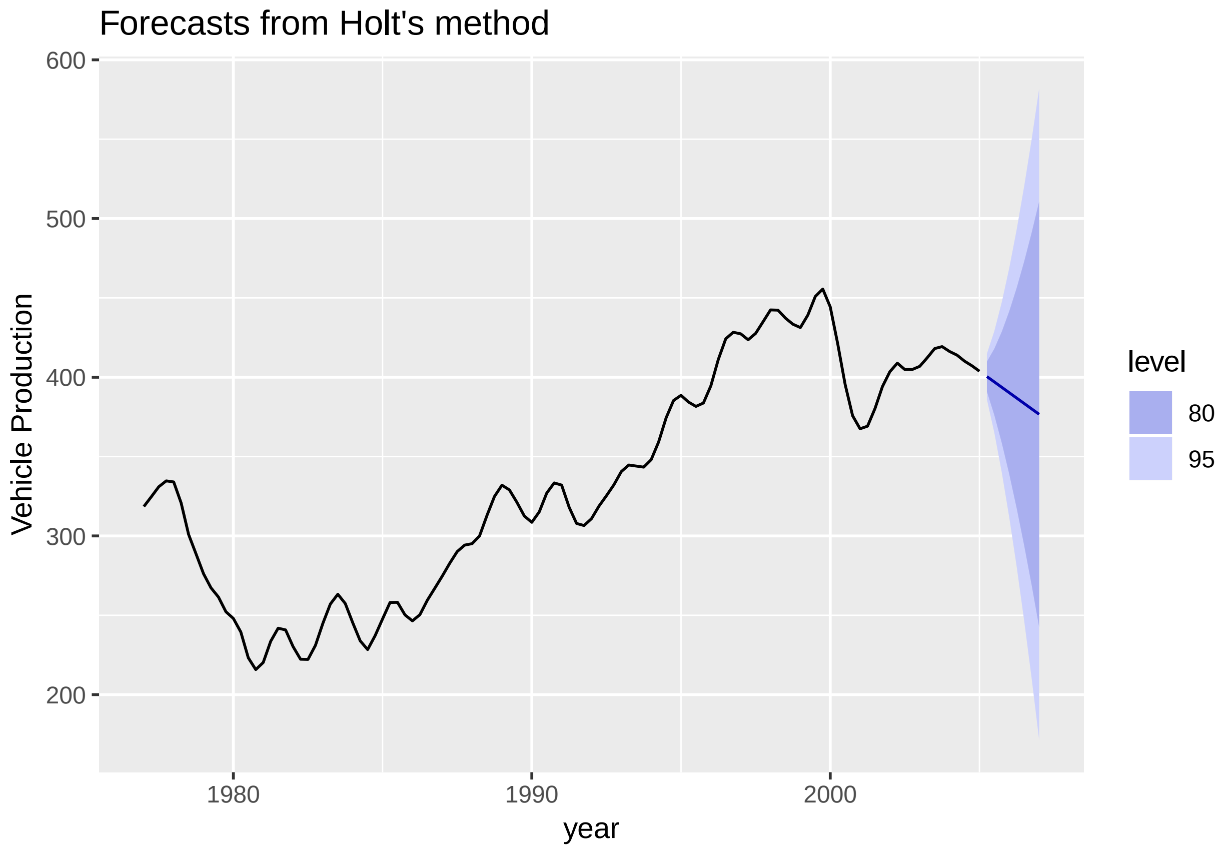
(3)Forecast the next two years of the series using an additive damped trend method applied to the seasonally adjusted data. (This can be done in one step using stlf() with arguments etsmodel="AAN", damped=TRUE.)

```
pred_sadj <- stlf(ukcars_sadj, etsmodel = "AAN", damped = TRUE, h = 8)

autoplot(pred_sadj) +
  xlab("year") +
  ylab("Vehicle Production")
```



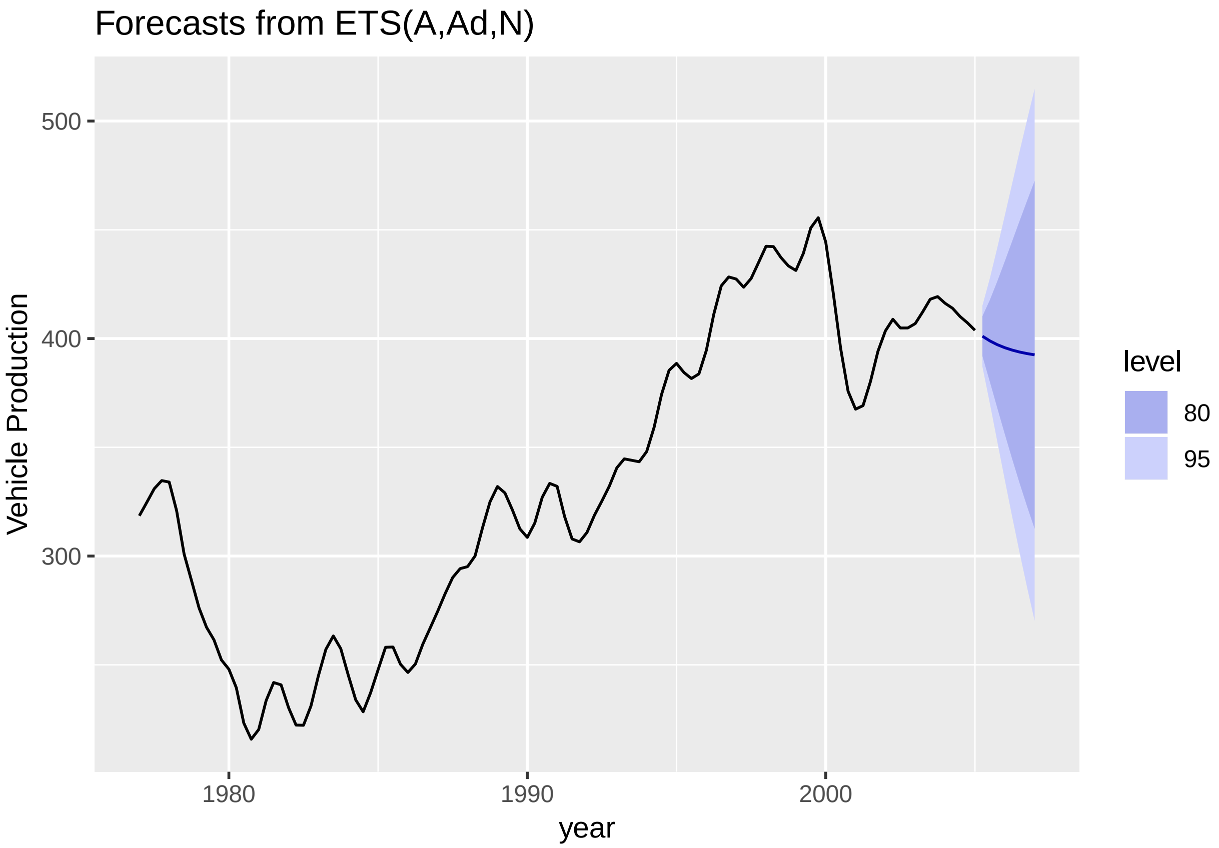
(4)Forecast the next two years of the series using Holt's linear method applied to the seasonally adjusted data (as before but with damped=FALSE).

```
hw_uk <- holt(ukcars_sadj, h = 8, damped=FALSE)
autoplot(hw_uk) +
  xlab("year") +
  ylab("Vehicle Production")
```



(5)Now use ets() to choose a seasonal model for the data.

```
ukcars_sadj %>% forecast(h = 8) %>%
  autoplot() +
  xlab("year") +
  ylab("Vehicle Production")
```



(6)Compare the RMSE of the ETS model with the RMSE of the models you obtained using STL decompositions. Which gives the better in-sample fits?

```
ets_uk <- forecast(ukcars_sadj, h = 8) # by ets

RMSE(pred_sadj$fitted, ukcars_sadj) # 6.86 by stlf
```

```
## [1] 6.860955
```

```
RMSE(ets_uk$fitted, ukcars_sadj) # 6.96 by ets
```

```
## [1] 6.96362
```

By compring RMSE, stl gives a better in-sample fits.

(7)Compare the forecasts from the three approaches? Which seems most reasonable? Check the residuals of your preferred model.

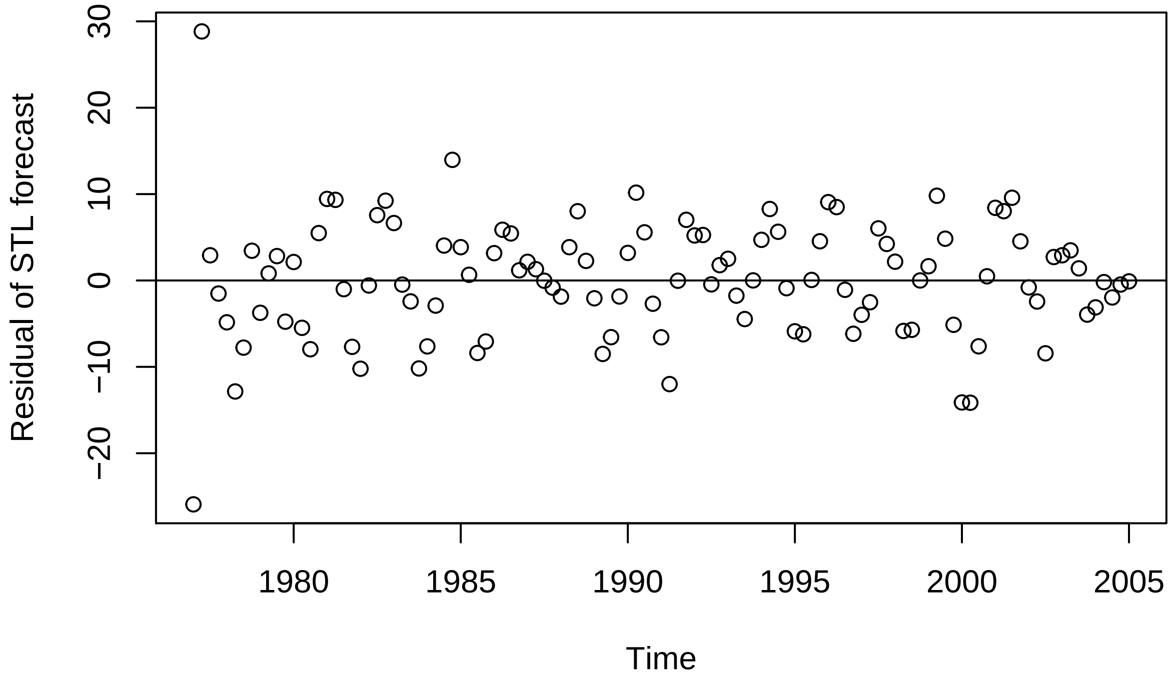
```
# check for accuracy
accuracy(pred_sadj)
```

```
##                     ME     RMSE      MAE        MPE     MAPE     MASE
## Training set 0.1087044 6.860955 5.134386 0.06791621 1.640731 0.235849
##                   ACF1
## Training set 0.2622221
```

```
accuracy(ets_uk)
```

```
##                    ME    RMSE      MAE        MPE     MAPE      MASE
## Training set 0.109292 6.96362 5.210045 0.06815573 1.665793 0.2393244
##                   ACF1
## Training set 0.2362471
```

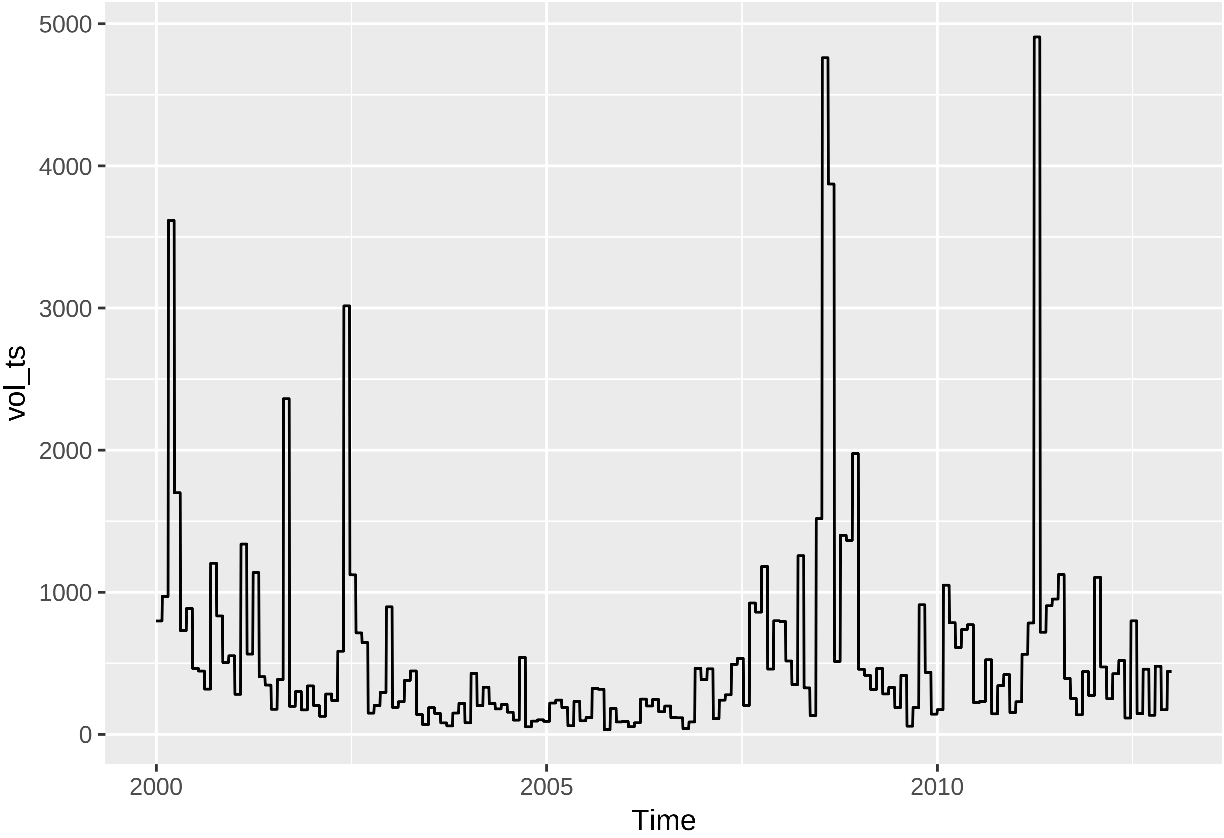
```
# check the residual of stl
plot(pred_sadj$residuals, type = "p", ylab = "Residual of STL forecast")
abline(a = 0, b = 0)
```



By observing the graphs, stl and ets gives more reasonable forecasts, since they are relatively smoother than the prediciton by Holt's linear. Holt's linear's prediction seems a stright linear line without considering trend. After comparing the RMSE, ME, MAE of stl and ets, stl is preferred. The residual plot of stl is randomly distributed and bounce between the line y = 0, with two outliers located at the top left and bottom left of the residual plot. Hence, we can say stl is an appropriate fit model for uk vehicle productions time series.

3. (a)Update the time series of the SP500 index in Section 14.1 and comment on the volatility of recent times compared to that of past times.

```
setwd("/Users/Renaissance/Desktop")
sp500 <- read_xls("hw5_14a.xls")
# I choose to use close price of sp500
# extract close and open price
sp500_close <- sp500[, c("Close")]
sp500_open <- sp500[, c("Open")]
# calculate volatility
# I set the window to 20 days
n <- dim(sp500)[1]
num_window  <- floor(n/ 20) # how many sets have 20 days
last_size <- dim(sp500)[1] %% 20 # the size of last set
vol_sp500 <- data.frame(vector(mode = "numeric", length = n))
colnames(vol_sp500) <- c("volatility")

for (i in 1: num_window){
  subset <- sp500_close[(20*(i-1)+1):(20*i),]
  mean_20 <- mean(unlist(subset))
  std_20 <- sd(unlist(subset))
  vol <- sum( (subset - mean_20)^2)/ 20
  vol_sp500[(20*(i-1)+1):(20*i), ] <- vol
}

subset <- sp500_close[(num_window*20+1):n, ]
mean_20 <- mean(unlist(subset))
std_20 <- sd(unlist(subset))
vol <- sum( (subset - mean_20)^2)/ last_size
vol_sp500[(num_window*20+1):n, ] <- vol

vol_ts <- ts(vol_sp500, 2000, 2013, frequency = 258)
autoplot(vol_ts)
```

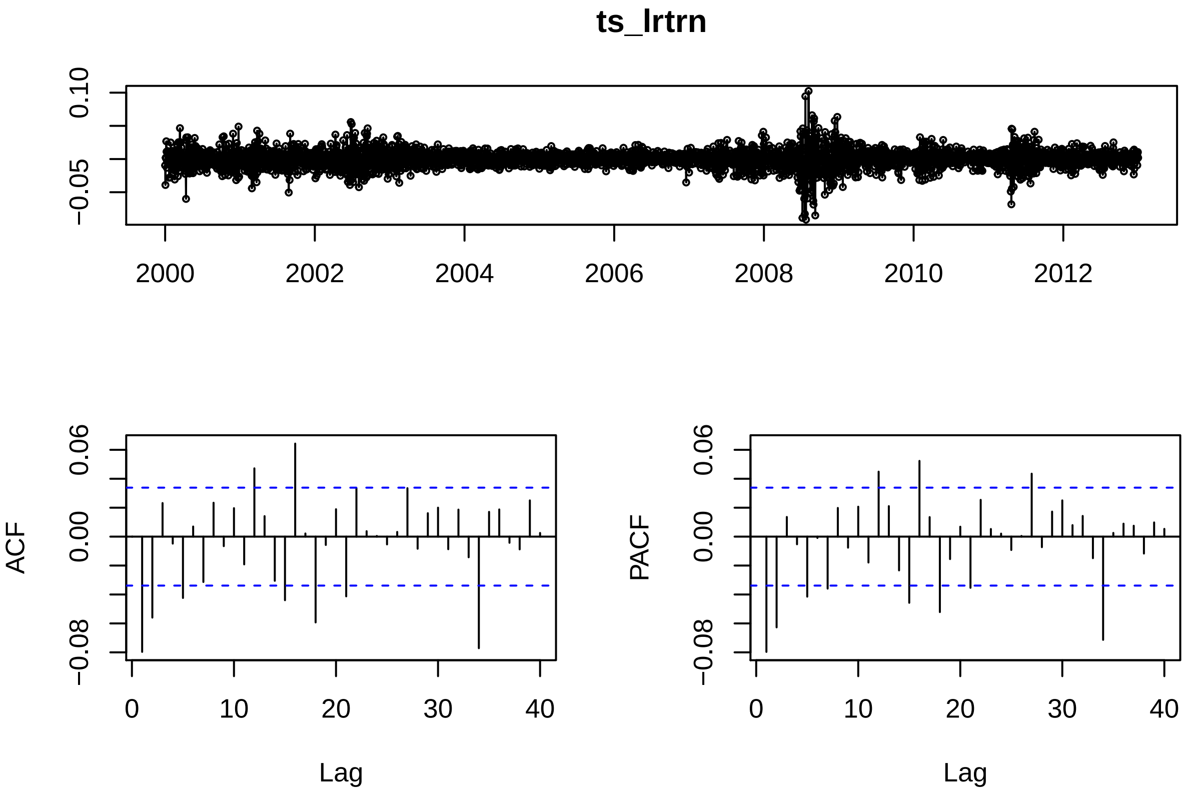


```
# Comment: By observing Volatility in recent years is more volatile than past.
```

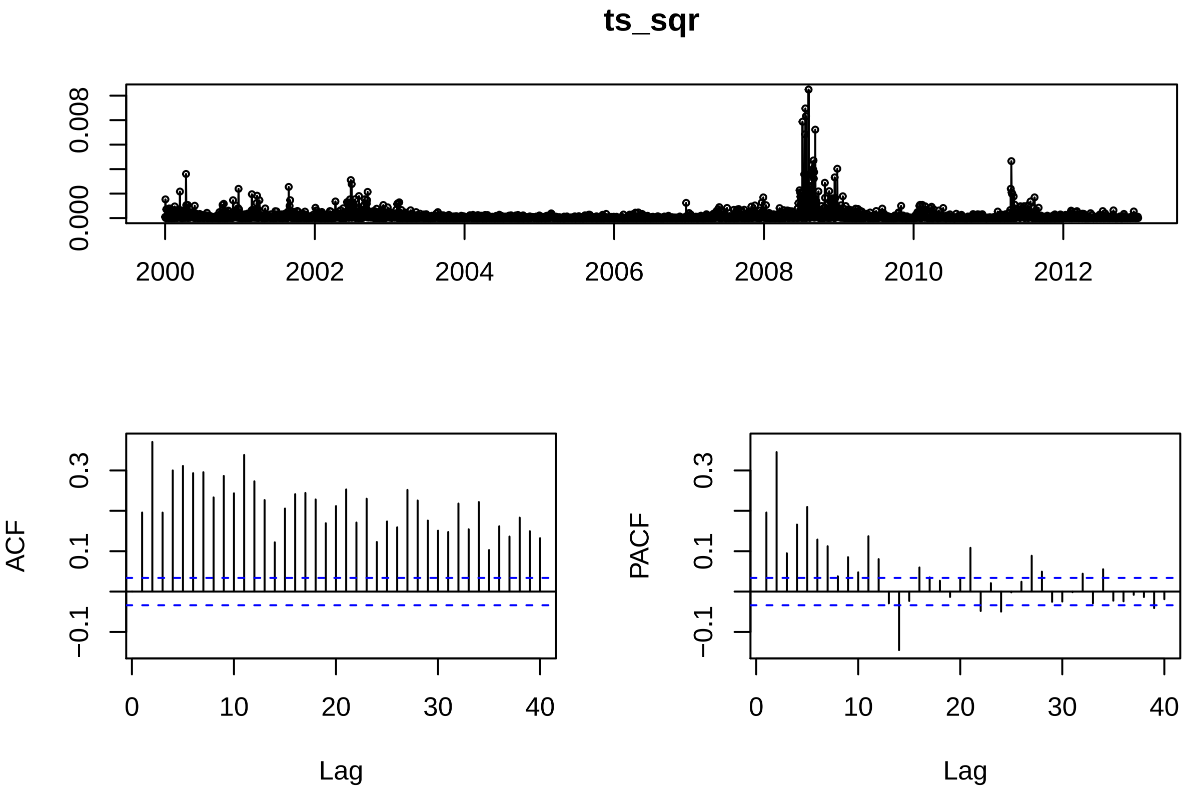
Comment: By observing Volatility graph in recent years, is more volatile than past.

3.(b)Compute the autocorrelation functions of returns and squared returns.

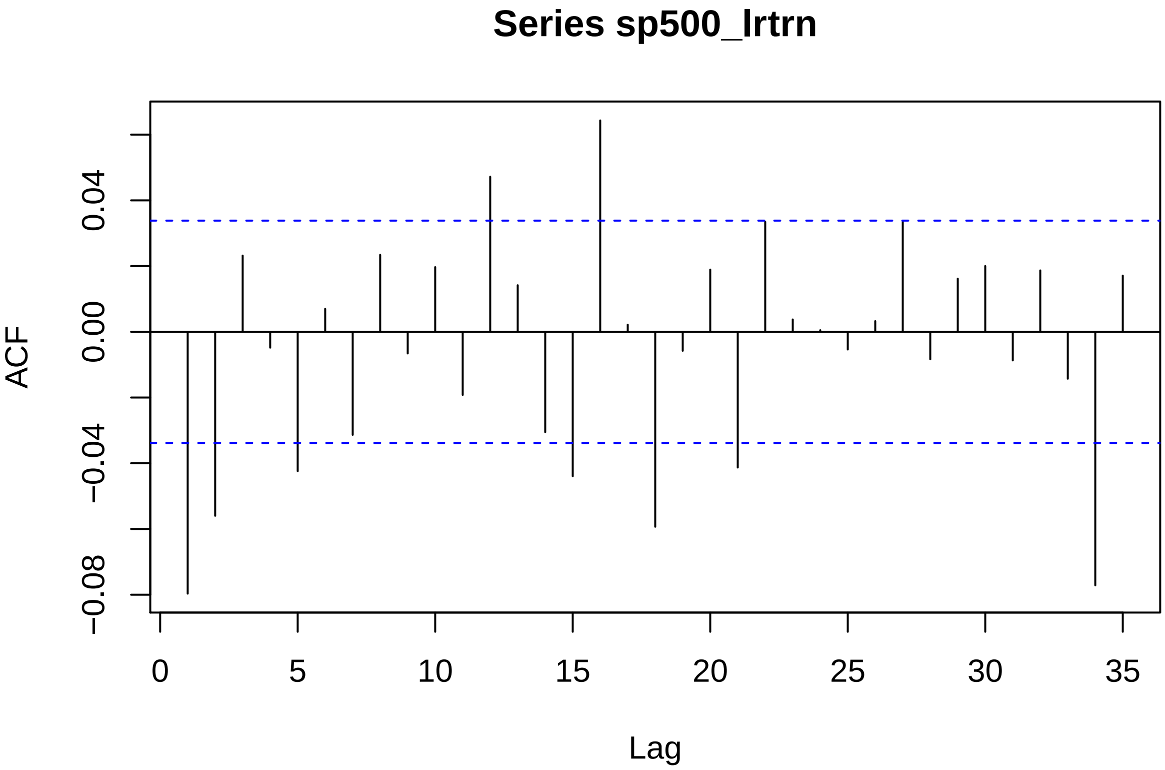
```
##calcualte log returns
sp500_lrtrn <- log(sp500_close/ sp500_open)
ts_lrtrn <- ts(sp500_lrtrn, 2000, 2013, frequency = 258)
tsdisplay(ts_lrtrn, lag.max = 40)
```



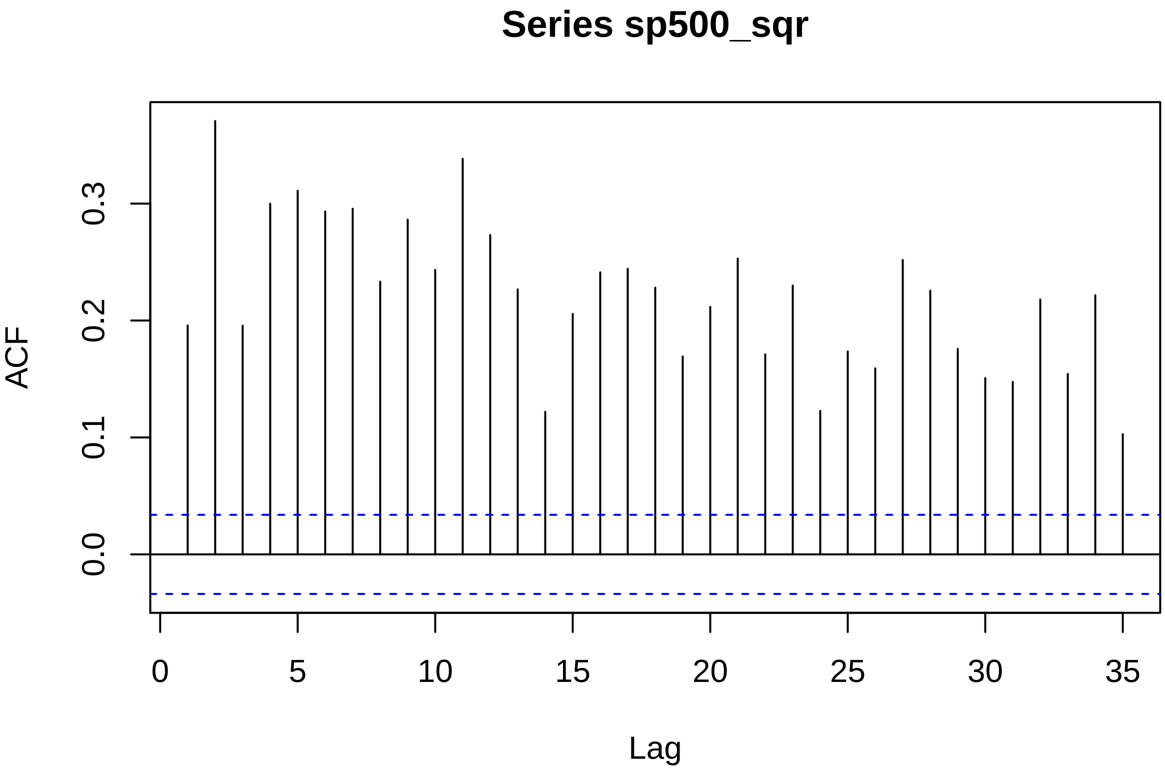
```r
##calculate square returns
sp500_sqr <- sp500_lrtrn^2
ts_sqr <- ts(sp500_sqr, 2000, 2013, frequency = 258)
tsdisplay(ts_sqr, lag.max = 40)
```



```
acf(sp500_lrtrn)
```



```
acf(sp500_sqr)
```



There are signal of seasonality in the return and squared return. All acf values are significant in tbe acf of squared return.

3.(c)Find the best ARCH process to model the volatility of the index. Could you find an equivalent more parsimonious GARCH process?

```
# conduct a box test to see whether volatility is predictable
Box.test(vol_sp500, lag = 1) # not white noise
```

```
##
##  Box-Pierce test
##
## data:  vol_sp500
## X-squared = 3155.9, df = 1, p-value < 2.2e-16
```

```
a_vol <- garch(x = vol_sp500, order = c(0, 8), trace = TRUE)
```

```
##
##  ***** ESTIMATION WITH ANALYTICAL GRADIENT *****
##
##
##      I     INITIAL X(I)        D(I)
##
##      1     3.230631e+05     1.000e+00
##      2     5.000000e-02     1.000e+00
##      3     5.000000e-02     1.000e+00
##      4     5.000000e-02     1.000e+00
##      5     5.000000e-02     1.000e+00
##      6     5.000000e-02     1.000e+00
##      7     5.000000e-02     1.000e+00
##      8     5.000000e-02     1.000e+00
##      9     5.000000e-02     1.000e+00
##
##     IT   NF      F         RELDF    PRELDF    RELDX   STPPAR   D*STEP   NPRELDF
##      0    1  2.288e+04
##      1    4  2.285e+04  1.21e-03  7.22e-03  1.8e-07  6.7e+03  1.6e-01  2.43e+01
##      2    6  2.285e+04  3.52e-04  3.35e-04  1.2e-08  7.3e+00  1.6e-02  2.96e+00
##      3    8  2.283e+04  6.19e-04  6.31e-04  2.6e-08  2.8e+00  3.1e-02  1.97e+00
##      4   10  2.283e+04  1.10e-04  1.10e-04  5.6e-09  3.0e+01  6.3e-03  1.72e+00
##      5   12  2.283e+04  2.05e-04  2.06e-04  1.1e-08  4.0e+00  1.3e-02  1.56e+00
##      6   14  2.282e+04  3.54e-04  3.60e-04  2.3e-08  2.6e+00  2.5e-02  1.50e+00
##      7   16  2.282e+04  6.45e-05  6.47e-05  4.8e-09  5.8e+01  5.0e-03  1.35e+00
##      8   18  2.282e+04  1.27e-05  1.27e-05  9.5e-10  2.6e+02  1.0e-03  1.02e+00
##      9   20  2.282e+04  2.53e-05  2.53e-05  1.9e-09  3.1e+01  2.0e-03  9.02e-01
##     10   22  2.282e+04  5.03e-06  5.03e-06  3.8e-10  6.2e+02  4.0e-04  8.47e-01
##     11   24  2.282e+04  1.01e-06  1.01e-06  7.6e-11  3.0e+03  8.0e-05  7.97e-01
##     12   26  2.281e+04  2.01e-06  2.01e-06  1.5e-10  3.7e+02  1.6e-04  7.88e-01
##     13   28  2.281e+04  4.01e-06  4.01e-06  3.0e-10  1.9e+02  3.2e-04  7.84e-01
##     14   30  2.281e+04  8.02e-07  8.02e-07  6.0e-11  3.7e+03  6.4e-05  7.76e-01
##     15   32  2.281e+04  1.60e-07  1.60e-07  1.2e-11  1.8e+04  1.3e-05  7.68e-01
##     16   34  2.281e+04  3.21e-08  3.21e-08  2.4e-12  9.2e+04  2.6e-06  7.67e-01
##     17   36  2.281e+04  6.42e-08  6.42e-08  4.8e-12  1.1e+04  5.1e-06  7.67e-01
##     18   38  2.281e+04  1.28e-08  1.28e-08  9.7e-13  2.3e+05  1.0e-06  7.66e-01
##     19   40  2.281e+04  2.57e-08  2.57e-08  1.9e-12  2.9e+04  2.1e-06  7.66e-01
##     20   42  2.281e+04  5.13e-08  5.13e-08  3.9e-12  1.4e+04  4.1e-06  7.66e-01
##     21   46  2.281e+04  1.03e-10  1.03e-10  7.7e-15  2.9e+07  8.2e-09  7.66e-01
##     22   47  2.281e+04 -4.38e+05  2.05e-10  1.5e-14  1.4e+07  1.6e-08  7.66e-01
##
```

```
##  ***** FALSE CONVERGENCE *****
##
##  FUNCTION     2.281485e+04   RELDX        1.548e-14
##  FUNC. EVALS      47         GRAD. EVALS      22
##  PRELDF       2.053e-10      NPRELDF      7.661e-01
##
##      I      FINAL X(I)        D(I)          G(I)
##
##      1    3.230631e+05     1.000e+00     2.514e-03
##      2    1.742085e-01     1.000e+00    -1.174e+02
##      3    1.124724e-01     1.000e+00    -4.630e+01
##      4    8.140559e-02     1.000e+00    -4.093e+00
##      5    6.032690e-02     1.000e+00     3.146e+01
##      6    4.354791e-02     1.000e+00     6.603e+01
##      7    2.868927e-02     1.000e+00     1.021e+02
##      8    1.444805e-02     1.000e+00     1.396e+02
##      9    9.948388e-09     1.000e+00     1.739e+02
```

```
AIC(a_vol)
```

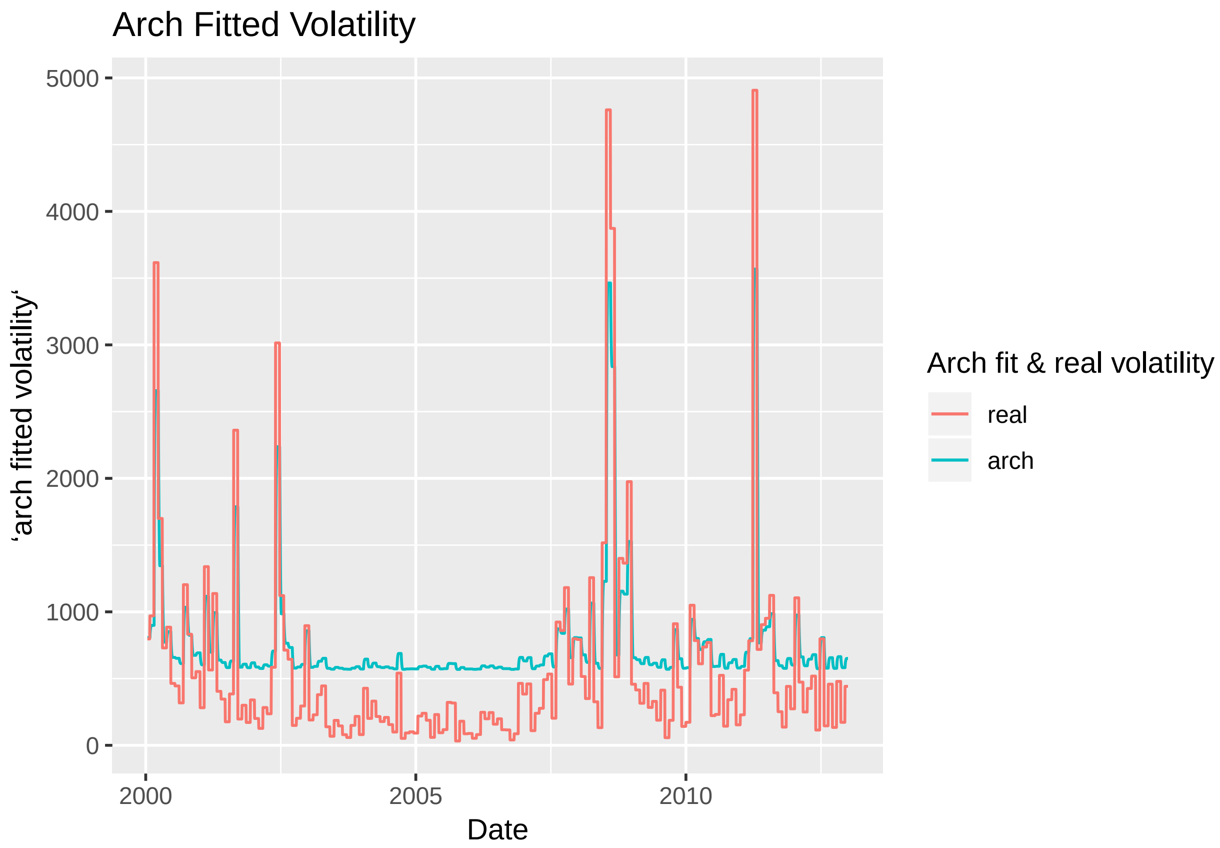
```
## [1] 51799.08
```

```
summary(a_vol)
```

```
##
## Call:
## garch(x = vol_sp500, order = c(0, 8), trace = TRUE)
##
## Model:
## GARCH(0,8)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## 0.05286 0.29692 0.51904 0.82509 6.13931
##
## Coefficient(s):
##     Estimate  Std. Error  t value Pr(>|t|)
## a0 3.231e+05   1.685e+04   19.168  < 2e-16 ***
## a1 1.742e-01   5.404e-02    3.224  0.00126 **
## a2 1.125e-01   1.315e-01    0.855  0.39248
## a3 8.141e-02   1.906e-01    0.427  0.66936
## a4 6.033e-02   1.993e-01    0.303  0.76212
## a5 4.355e-02   1.705e-01    0.255  0.79845
## a6 2.869e-02   1.375e-01    0.209  0.83468
## a7 1.445e-02   1.249e-01    0.116  0.90791
## a8 9.948e-09   8.930e-02    0.000  1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Diagnostic Tests:
##  Jarque Bera Test
##
## data:  Residuals
## X-squared = 37931, df = 2, p-value < 2.2e-16
##
```

```
##
##  Box-Ljung test
##
## data:  Squared.Residuals
## X-squared = 560.56, df = 1, p-value < 2.2e-16
```

```
x_ts <- seq(2000, 2013, length = n)
afit <- data.frame(x_ts, a_vol$fitted.values[, 1], vol_sp500)
colnames(afit) <- c("Date", "arch fitted volatility", "volatility")
afit <- na.omit(afit)

ggplot(afit, aes(Date)) +
  geom_line(aes(y = `arch fitted volatility`, color = "blue")) +
  geom_line(aes(y = afit$volatility, color = "black")) +
  ggtitle("Arch Fitted Volatility") +
  scale_color_discrete(name = "Arch fit & real volatility", labels = c("real", "arch"))
```



```
g_vol <- garch(x = vol_sp500, order = c(1, 1))
```

```
##
##  ***** ESTIMATION WITH ANALYTICAL GRADIENT *****
##
##
##      I     INITIAL X(I)        D(I)
##
##      1     4.845946e+05     1.000e+00
##      2     5.000000e-02     1.000e+00
##      3     5.000000e-02     1.000e+00
```

```
##
##     IT   NF      F         RELDF    PRELDF    RELDX   STPPAR   D*STEP   NPRELDF
##      0    1  2.374e+04
##      1    2  2.332e+04  1.75e-02  3.29e-01  1.0e-06  7.8e+03  1.0e+00  1.29e+03
##      2    5  2.332e+04  2.76e-04  2.77e-04  5.1e-09  3.3e+02  5.0e-03  3.33e+01
##      3    7  2.330e+04  5.47e-04  5.47e-04  1.0e-08  1.9e+01  1.0e-02  2.07e-02
##      4    9  2.328e+04  1.07e-03  1.07e-03  2.0e-08  1.0e+01  2.0e-02  2.02e-02
##      5   12  2.328e+04  2.12e-05  2.12e-05  4.1e-10  1.8e+03  4.0e-04  1.93e-02
##      6   14  2.328e+04  4.23e-05  4.23e-05  8.1e-10  2.3e+02  8.0e-04  1.94e-02
##      7   16  2.328e+04  8.46e-06  8.46e-06  1.6e-10  4.6e+03  1.6e-04  1.94e-02
##      8   18  2.328e+04  1.69e-05  1.69e-05  3.3e-10  5.7e+02  3.2e-04  1.93e-02
##      9   20  2.328e+04  3.38e-06  3.38e-06  6.5e-11  1.1e+04  6.4e-05  1.93e-02
##     10   23  2.327e+04  2.71e-05  2.71e-05  5.2e-10  3.6e+02  5.1e-04  1.93e-02
##     11   27  2.327e+04  5.41e-08  5.41e-08  1.0e-12  7.1e+05  1.0e-06  1.93e-02
##     12   29  2.327e+04  1.08e-07  1.08e-07  2.1e-12  8.9e+04  2.0e-06  1.93e-02
##     13   31  2.327e+04  2.16e-08  2.16e-08  4.2e-13  1.8e+06  4.1e-07  1.93e-02
##     14   33  2.327e+04  4.33e-08  4.33e-08  8.3e-13  2.2e+05  8.2e-07  1.93e-02
##     15   35  2.327e+04  8.65e-08  8.65e-08  1.7e-12  1.1e+05  1.6e-06  1.93e-02
##     16   38  2.327e+04  1.73e-09  1.73e-09  3.3e-14  2.2e+07  3.3e-08  1.93e-02
##     17   40  2.327e+04  3.46e-09  3.46e-09  6.7e-14  2.8e+06  6.6e-08  1.93e-02
##     18   42  2.327e+04  6.92e-10  6.92e-10  1.3e-14  5.6e+07  1.3e-08  1.93e-02
##     19   45  2.327e+04  5.54e-09  5.54e-09  1.1e-13  1.7e+06  1.0e-07  1.93e-02
##     20   47  2.327e+04 -4.30e+05  1.11e-09  2.1e-14  3.5e+07  2.1e-08  1.93e-02
##
##  ***** FALSE CONVERGENCE *****
##
##  FUNCTION     2.327490e+04   RELDX        2.133e-14
##  FUNC. EVALS      47          GRAD. EVALS      20
##  PRELDF       1.108e-09      NPRELDF      1.925e-02
##
##      I      FINAL X(I)        D(I)          G(I)
##
##      1    4.845946e+05     1.000e+00     1.992e-03
##      2    1.043611e+00     1.000e+00     2.081e+02
##      3    8.565609e-10     1.000e+00     1.212e+03
```

```
## Warning in sqrt(pred$e): NaNs produced
```

```
AIC(g_vol)
```

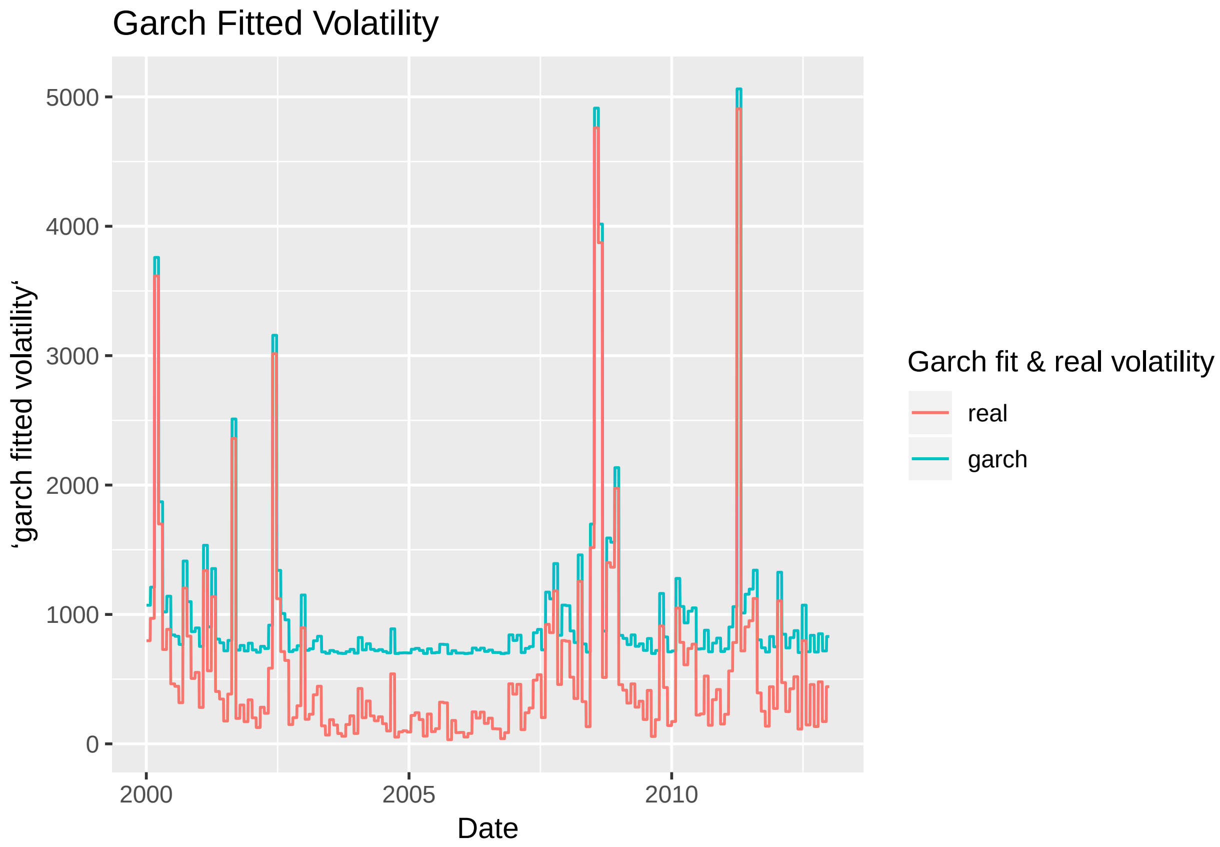
```
## [1] 52720.05
```

```
summary(g_vol) # garch(1,2) works well as arch(11)
```

```
##
## Call:
## garch(x = vol_sp500, order = c(1, 1))
##
## Model:
## GARCH(1,1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## 0.04215 0.24498 0.42266 0.63922 4.62755
##
## Coefficient(s):
```

```
##     Estimate  Std. Error  t value Pr(>|t|)
## a0 4.846e+05   5.336e+04    9.082  < 2e-16 ***
## a1 1.044e+00   1.672e-01    6.242 4.32e-10 ***
## b1 8.566e-10   9.317e-02    0.000        1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Diagnostic Tests:
##  Jarque Bera Test
##
## data:  Residuals
## X-squared = 60917, df = 2, p-value < 2.2e-16
##
##
##  Box-Ljung test
##
## data:  Squared.Residuals
## X-squared = 229.05, df = 1, p-value < 2.2e-16
```

```
gfit <- data.frame(x_ts, g_vol$fitted.values[, 1], vol_sp500)
colnames(gfit) <- c("Date", "garch fitted volatility","volatility")
gfit <- na.omit(gfit)

ggplot(gfit, aes(Date)) +
  geom_line(aes(y = `garch fitted volatility`, color = "blue")) +
  geom_line(aes(y = gfit$volatility, color = "black")) +
  ggtitle("Garch Fitted Volatility") +
  scale_color_discrete(name = "Garch fit & real volatility", labels = c("real", "garch"))
```



For volatility, garch(1,1) works similar as arch(8), which is the most accurate arch model for this volatility.

4. Based on your findings from Exercise 3, calculate the one and two-step ahead volatility forecasts. Construct a 95% interval forecast for the SP500 returns. Assume that the returns are conditionally normal distributed.

```
# one- step ahead volatility of garch
z <- rnorm(1) # random number follow normal distribution
sigma_last <- var(g_vol$fitted.values[,1], na.rm=TRUE)
sigma <- sqrt(g_vol$coef[1] + g_vol$coef[2]*(g_vol$residuals[3355])^2 + g_vol$coef[3]*sigma_last)
gf_step1 <- g_vol$fitted.values[,1][3355] + sigma*z
print("After calculating, one step forecast of garch(1,1) is ")
```

```
## [1] "After calculating, one step forecast of garch(1,1) is "
```

```
print(gf_step1)
```

```
##       a0
## 972.3217
```

```
z2 <- rnorm(1) # random number follow normal distribution
f1_data <- c(g_vol$fitted.values[,1], gf_step1)
sigma_f1 <- var(f1_data, na.rm=TRUE)
sigma_2 <- sqrt(g_vol$coef[1] + (g_vol$coef[2] + g_vol$coef[3])*sigma_f1)
gf_step2 <- gf_step1 + sigma_2*z2
print("After calculating, two step forecast of garch(1,1) is ")
```

```
## [1] "After calculating, two step forecast of garch(1,1) is "
```

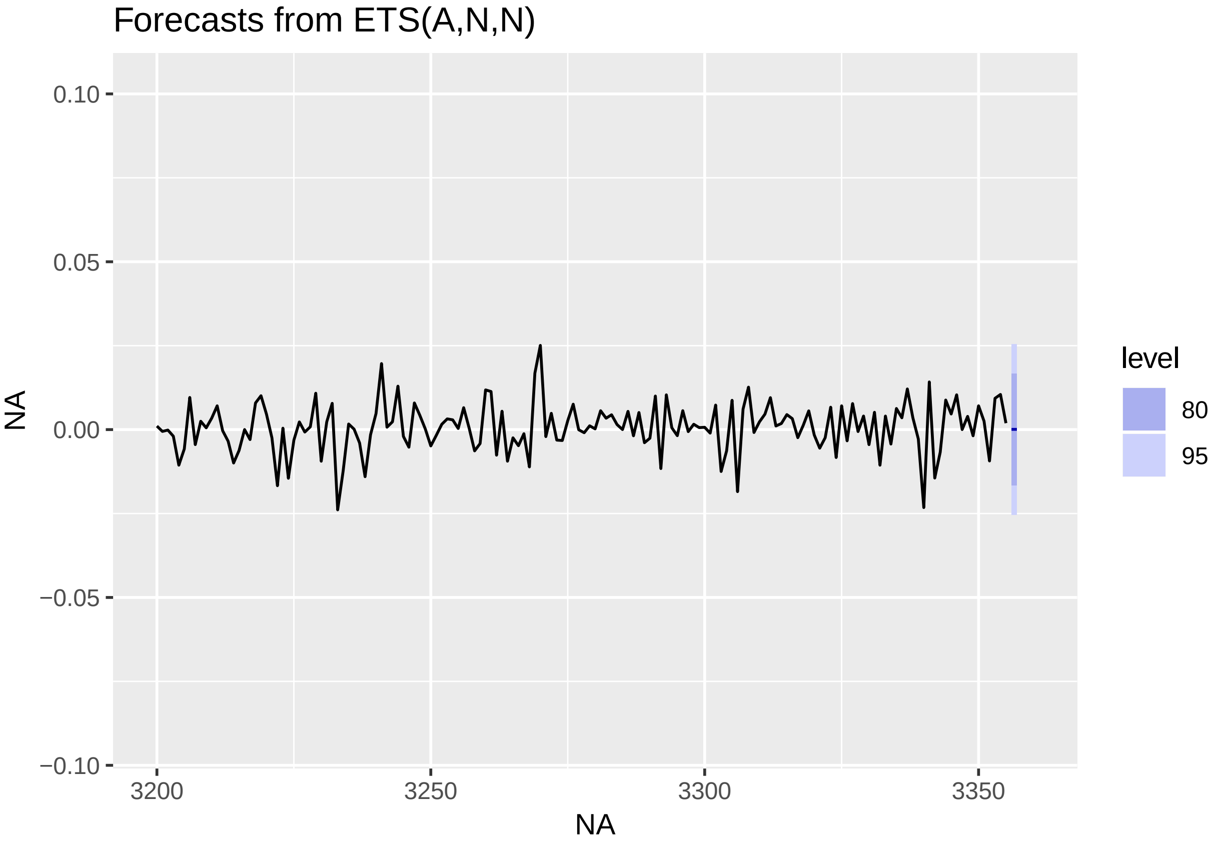
```
print(gf_step2)
```

```
##        a0
## 1454.482
```

```
# try to forecast return for 20 steps ahead
return_forecast <- forecast(unlist(sp500_lrtrn), h = 2)
autoplot(return_forecast) +
  xlim(3200, 3360) # set the window to have a better observation on the forecast
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which
## will replace the existing scale.
```

```
## Warning: Removed 3199 rows containing missing values (geom_path).
```



5. In Exercise 5 of Chapter 13, you downloaded the time series of US CPI and GDP and constructed the inflation rate and GDP growth. For each, calculate the unconditional mean, and compute the 1-step-ahead volatility forecast by implementing the best (G)ARCH model, and construct the corresponding 95% interval forecast.

```
setwd("/Users/Renaissance/Desktop")
# read cpi and gdp data
cpi <- read_xls("hw5_14a.xls", sheet = 2)
```

```
## New names:
## * `` -> ...3
```

```
gdp <- read_xls("hw5_14a.xls", sheet = 3)
```

```
## New names:
## * `` -> ...3
```

```r
cpi <- cpi[,c(1,2)]
gdp <- gdp[,c(1,2)]  #eliminate nan

cpi_ts <- ts(cpi[,2],start = c(1947,1), end = c(2013,3), frequency = 12)
gdp_ts <- ts(gdp[,2],start = c(1947,1), end = c(2013,3), frequency = 12)

# inflation
cpi_value <- unlist(cpi[,2])
inflation <- (cpi_value - Lag(cpi_value, 1))/ Lag(cpi_value, 1)
# GDP growth
gdp_value <- unlist(gdp[,2])
growth <- (gdp_value - Lag(gdp_value, 1))/ Lag(gdp_value, 1)

# calculate the unconditional mean
print("The unconditional mean of cpi is ")
```

```
## [1] "The unconditional mean of cpi is "
```

```r
print(mean(cpi[,2]))
```

```
## Warning in mean.default(cpi[, 2]): argument is not numeric or logical:
## returning NA

## [1] NA
```

```r
print("The unconditional mean of gdp is ")
```

```
## [1] "The unconditional mean of gdp is "
```

```r
print(mean(gdp[,2]))
```

```
## Warning in mean.default(gdp[, 2]): argument is not numeric or logical:
## returning NA

## [1] NA
```

```r
# fit garch
inflation_nonan <- na.omit(inflation)
g_inflation <- garch(x = inflation_nonan, order = c(1, 1))
```

```
##
##  ***** ESTIMATION WITH ANALYTICAL GRADIENT *****
##
##
##      I     INITIAL X(I)        D(I)
##
##      1     1.112159e-05     1.000e+00
##      2     5.000000e-02     1.000e+00
##      3     5.000000e-02     1.000e+00
##
##     IT   NF      F         RELDF    PRELDF    RELDX   STPPAR   D*STEP   NPRELDF
##      0    1 -3.871e+03
##      1   17 -3.885e+03  3.68e-03  3.65e-02  1.1e-04  1.2e+12  1.1e-05  2.13e+10
##      2   26 -3.948e+03  1.58e-02  1.65e-02  5.6e-01  2.0e+00  1.3e-01  2.94e+01
##      3   32 -3.949e+03  3.26e-04  5.60e-04  3.6e-06  9.9e+00  1.3e-06  6.17e+00
##      4   43 -3.977e+03  7.15e-03  7.98e-03  3.6e-01  2.0e+00  1.7e-01  4.91e+00
##      5   45 -3.993e+03  4.04e-03  3.18e-03  3.0e-01  1.6e+00  1.7e-01  6.24e-02
##      6   47 -4.028e+03  8.67e-03  7.83e-03  3.1e-01  2.0e+00  3.4e-01  3.31e+00
##      7   58 -4.030e+03  5.33e-04  1.36e-03  2.2e-07  7.9e+00  3.1e-07  6.89e-01
##      8   59 -4.031e+03  1.47e-05  1.04e-05  2.2e-07  2.0e+00  3.1e-07  3.37e-01
##      9   60 -4.031e+03  1.55e-06  1.10e-06  2.2e-07  2.0e+00  3.1e-07  3.69e-01
##     10   69 -4.031e+03 -4.05e-13  1.17e-13  9.9e-15  1.9e+00  1.4e-14 -2.03e-02
##
##  ***** FALSE CONVERGENCE *****
##
##  FUNCTION    -4.030559e+03   RELDX        9.890e-15
##  FUNC. EVALS      69         GRAD. EVALS      10
##  PRELDF       1.171e-13      NPRELDF     -2.031e-02
##
##      I      FINAL X(I)        D(I)          G(I)
##
##      1    9.491838e-07     1.000e+00    -3.405e+04
##      2    2.311585e-01     1.000e+00    -7.328e+01
##      3    7.007517e-01     1.000e+00    -8.654e+01
```

```
summary(g_inflation)
```

```
##
## Call:
## garch(x = inflation_nonan, order = c(1, 1))
##
## Model:
## GARCH(1,1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4465  0.3451  0.7437  1.1383  3.8784
##
## Coefficient(s):
##     Estimate  Std. Error  t value Pr(>|t|)
## a0 9.492e-07   2.568e-07    3.696 0.000219 ***
## a1 2.312e-01   3.293e-02    7.021 2.21e-12 ***
## b1 7.008e-01   3.385e-02   20.704  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Diagnostic Tests:
##  Jarque Bera Test
##
## data:  Residuals
## X-squared = 368.51, df = 2, p-value < 2.2e-16
##
##
##  Box-Ljung test
##
## data:  Squared.Residuals
## X-squared = 0.019688, df = 1, p-value = 0.8884
```

```
paste("In this cast, garch(1,1) gives the best fit")
```

```
## [1] "In this cast, garch(1,1) gives the best fit"
```

```
# one step ahead forecast
z_cpi <- rnorm(1) # random number follow normal distribution
```

```
sigma_last_cpi <- var(g_inflation$fitted.values[,1], na.rm=TRUE)
sigma_cpi <- sqrt(g_inflation$coef[1] + g_inflation$coef[2]*(g_inflation$residuals[794])^2 + g_inflatio
gf_step1_cpi <- g_inflation$fitted.values[,1][794] + sigma_cpi*z
print("After calculating, one step forecast of garch(1,1) is ")
```

```
## [1] "After calculating, one step forecast of garch(1,1) is "
```

```
print(as.numeric(gf_step1_cpi))
```

```
## [1] 0.04894958
```

```
print("The 95% interval of this one step ahead forecast is between :")
```

```
## [1] "The 95% interval of this one step ahead forecast is between :"
```

```
print(as.numeric(gf_step1_cpi - 1.96*sigma_cpi))
```

```
## [1] -0.3804798
```

```
print(" and ")
```

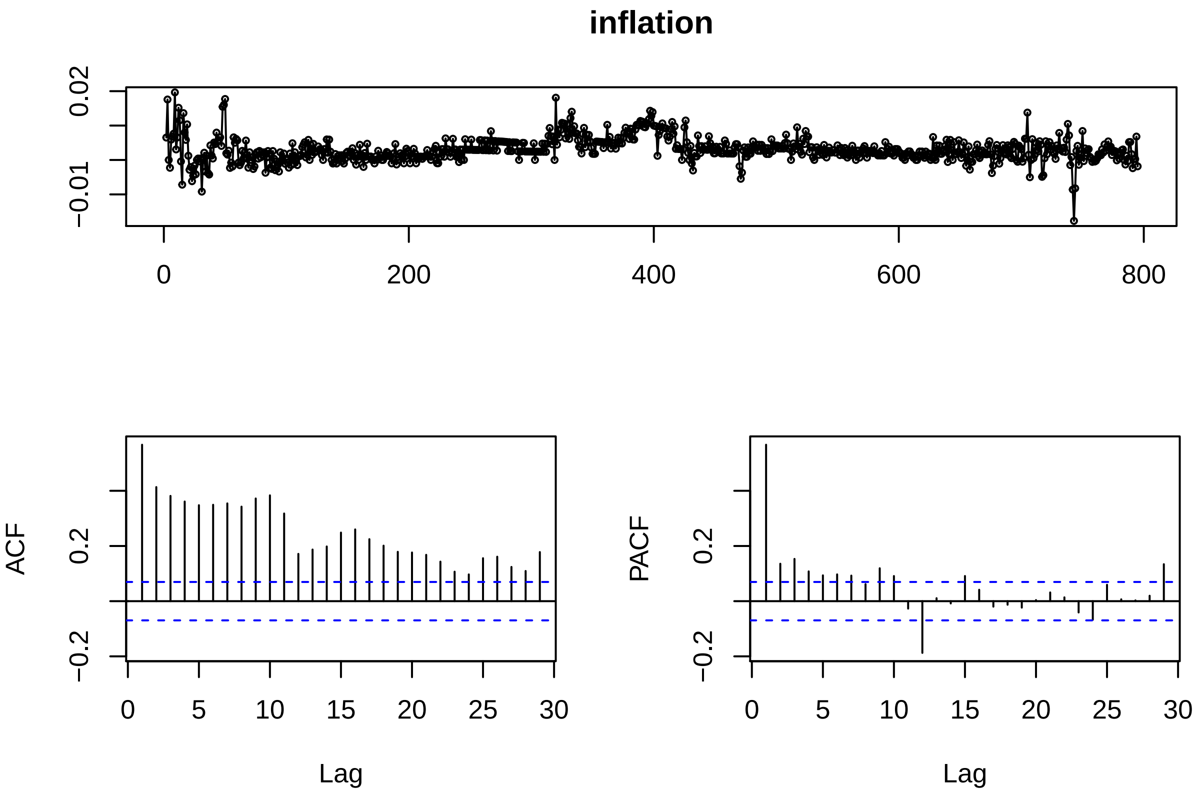
```
## [1] " and "
```

```
print(as.numeric(gf_step1_cpi +1.96*sigma_cpi))
```

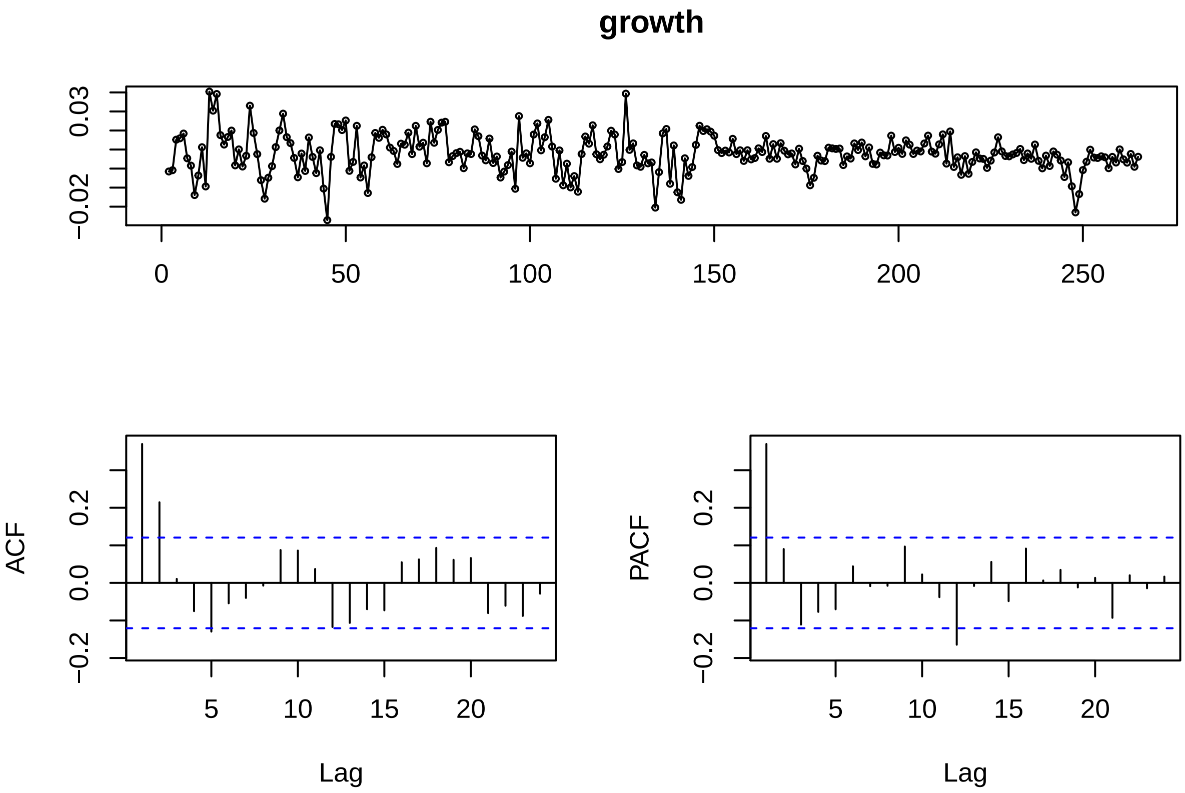
```
## [1] 0.4783789
```

6. Given this expression, retrieve the residuals and construct the 1-step-ahead volatility forecast by implementing the best (G)ARCH model. Construct the corresponding 95% interval forecast for inflation and GDP growth, and compare these intervals with those from Exercise 5.

```
tsdisplay(inflation) # view the inflation rate (from cpi)
```



```
arimafit <- auto.arima(inflation)
inflation_resi <- arimafit$residuals# retrieve the residual

tsdisplay(growth) # view the growth rate (from gdp)
```



```
arimafit_growth <- auto.arima(gdp_ts)
growth_resi <- arimafit_growth$residuals# retrieve the residual

# garch fit of inflation residual
g_ir<- garch(x = inflation_resi, order = c(1, 1))
```

```
##
##  ***** ESTIMATION WITH ANALYTICAL GRADIENT *****
##
##
##      I     INITIAL X(I)        D(I)
##
##      1     6.974545e-06     1.000e+00
##      2     5.000000e-02     1.000e+00
##      3     5.000000e-02     1.000e+00
##
##     IT   NF      F         RELDF    PRELDF    RELDX   STPPAR   D*STEP   NPRELDF
##      0    1 -4.302e+03
##      1    8 -4.306e+03  8.55e-04  1.58e-03  1.0e-05  6.8e+12  1.0e-06  5.39e+09
##      2    9 -4.306e+03  3.93e-07  1.19e-06  1.0e-05  2.0e+00  1.0e-06  2.48e+01
##      3   18 -4.325e+03  4.45e-03  7.01e-03  4.0e-01  2.0e+00  6.6e-02  2.47e+01
##      4   21 -4.340e+03  3.34e-03  3.05e-03  6.4e-01  1.9e+00  1.9e-01  4.88e-01
##      5   23 -4.385e+03  1.04e-02  6.86e-03  4.4e-01  2.0e+00  3.8e-01  6.14e+01
##      6   25 -4.398e+03  2.92e-03  2.81e-03  5.7e-02  2.0e+00  7.6e-02  1.82e+04
##      7   27 -4.399e+03  1.71e-04  5.40e-04  1.1e-02  2.0e+00  1.5e-02  2.39e+02
##      8   28 -4.402e+03  6.85e-04  8.65e-04  1.1e-02  2.0e+00  1.5e-02  2.13e+02
##      9   30 -4.404e+03  6.31e-04  1.46e-03  4.0e-02  2.0e+00  6.1e-02  1.16e+02
##     10   39 -4.405e+03  6.07e-05  2.54e-04  3.3e-08  6.0e+00  5.2e-08  2.34e-01
##     11   40 -4.405e+03  2.24e-05  1.70e-05  3.1e-08  2.0e+00  5.2e-08  7.55e-02
##     12   49 -4.405e+03  1.90e-05  2.58e-05  2.1e-03  2.0e+00  3.4e-03  6.53e-02
##     13   51 -4.405e+03  2.03e-06  2.64e-06  3.5e-04  1.9e+00  5.4e-04  9.77e-05
##     14   52 -4.405e+03  2.87e-06  5.74e-06  6.9e-04  2.0e+00  1.1e-03  3.53e-04
##     15   53 -4.405e+03  4.81e-06  7.54e-06  6.8e-04  1.8e+00  1.1e-03  6.51e-05
##     16   55 -4.405e+03  6.27e-06  8.31e-06  1.5e-03  1.3e+00  2.4e-03  3.25e-05
##     17   57 -4.405e+03  1.23e-05  1.34e-05  3.1e-03  6.5e-01  5.6e-03  1.84e-05
##     18   58 -4.405e+03  9.30e-06  1.07e-05  2.8e-03  7.5e-01  5.6e-03  1.68e-05
##     19   59 -4.405e+03  6.32e-06  7.42e-06  2.6e-03  6.6e-01  5.6e-03  1.02e-05
##     20   60 -4.405e+03  3.04e-06  3.76e-06  2.8e-03  2.9e-01  5.6e-03  3.97e-06
##     21   61 -4.405e+03  4.50e-07  4.56e-07  1.2e-03  0.0e+00  2.2e-03  4.56e-07
##     22   62 -4.405e+03  1.96e-07  2.59e-08  1.1e-04  0.0e+00  2.0e-04  2.59e-08
##     23   63 -4.405e+03  1.27e-07  5.38e-09  1.0e-04  0.0e+00  1.6e-04  5.38e-09
##     24   64 -4.405e+03  1.23e-08  8.57e-11  1.6e-05  0.0e+00  3.4e-05  8.57e-11
##     25   65 -4.405e+03  6.60e-10  1.47e-12  2.1e-06  0.0e+00  3.9e-06  1.47e-12
##     26   66 -4.405e+03 -2.65e-11  3.07e-15  3.8e-08  0.0e+00  6.3e-08  3.07e-15
##
##  ***** RELATIVE FUNCTION CONVERGENCE *****
##
##  FUNCTION    -4.405108e+03   RELDX        3.803e-08
##  FUNC. EVALS      66         GRAD. EVALS      26
##  PRELDF       3.072e-15      NPRELDF      3.072e-15
##
##      I      FINAL X(I)        D(I)          G(I)
##
##      1    3.592238e-07     1.000e+00     8.065e+01
##      2    1.948021e-01     1.000e+00     3.449e-04
##      3    7.634536e-01     1.000e+00     5.066e-04
```

```
summary(g_ir)   # garch (1, 1) gives the best fit
```

```
##
## Call:
## garch(x = inflation_resi, order = c(1, 1))
##
## Model:
## GARCH(1,1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -4.65799 -0.58537 -0.03286  0.57848  4.64900
##
## Coefficient(s):
##     Estimate  Std. Error  t value Pr(>|t|)
## a0 3.592e-07   6.447e-08    5.572 2.52e-08 ***
## a1 1.948e-01   2.056e-02    9.473  < 2e-16 ***
## b1 7.635e-01   2.058e-02   37.093  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Diagnostic Tests:
```

```
##  Jarque Bera Test
##
## data:  Residuals
## X-squared = 156.55, df = 2, p-value < 2.2e-16
##
##
##  Box-Ljung test
##
## data:  Squared.Residuals
## X-squared = 0.93926, df = 1, p-value = 0.3325
```

```
# garch fit of gdp growth residual
g_gr<- garch(x = growth_resi, order = c(0, 1))
```

```
##
##  ***** ESTIMATION WITH ANALYTICAL GRADIENT *****
##
##
##      I     INITIAL X(I)        D(I)
##
##      1     3.483404e+05     1.000e+00
##      2     5.000000e-02     1.000e+00
##
##     IT   NF      F         RELDF    PRELDF    RELDX   STPPAR   D*STEP   NPRELDF
##      0    1  5.487e+03
##      1    4  5.487e+03  3.41e-05  3.03e-05  1.4e-08  1.7e+03  1.0e-02  2.51e-02
##      2    6  5.486e+03  1.07e-04  7.69e-05  2.9e-08  2.6e+03  2.0e-02  4.06e-01
##      3    8  5.486e+03  3.40e-05  3.07e-05  5.7e-09  1.1e+06  4.0e-03  1.62e+01
##      4   10  5.486e+03  1.00e-04  7.55e-05  1.1e-08  1.6e+06  8.0e-03  2.45e+02
##      5   12  5.486e+03  2.96e-05  2.73e-05  2.3e-09  5.8e+08  1.6e-03  7.96e+03
##      6   14  5.485e+03  8.06e-05  6.46e-05  4.6e-09  8.7e+08  3.2e-03  1.12e+05
##      7   16  5.485e+03  2.17e-05  2.04e-05  9.2e-10  2.7e+11  6.4e-04  2.80e+06
##      8   18  5.485e+03  5.35e-05  4.62e-05  1.8e-09  3.9e+11  1.3e-03  3.58e+07
##      9   20  5.485e+03  1.29e-05  1.25e-05  3.7e-10  1.0e+14  2.6e-04  6.51e+08
##     10   22  5.484e+03  2.90e-05  2.68e-05  7.3e-10  1.4e+14  5.1e-04  7.52e+09
##     11   24  5.484e+03  6.42e-06  6.31e-06  1.5e-10  3.3e+16  1.0e-04  1.04e+11
##     12   26  5.484e+03  1.36e-05  1.31e-05  2.9e-10  4.3e+16  2.0e-04  1.12e+12
##     13   28  5.484e+03  2.84e-06  2.82e-06  5.9e-11  9.2e+18  4.1e-05  1.31e+13
##     14   30  5.484e+03  5.82e-06  5.73e-06  1.2e-10  1.2e+19  8.2e-05  1.46e+14
##     15   32  5.484e+03  1.19e-06  1.18e-06  2.4e-11  2.7e+21  1.6e-05  2.64e+15
##     16   34  5.484e+03  2.40e-06  2.38e-06  4.7e-11  4.1e+21  3.3e-05  1.23e+17
##     17   36  5.484e+03  4.83e-07  4.83e-07  9.4e-12  6.8e+23  6.6e-06  1.08e+19
##     18   38  5.484e+03  9.71e-07  9.68e-07  1.9e-11  3.8e+22  1.3e-05  1.07e+21
##     19   40  5.484e+03  1.95e-07  1.95e-07  3.8e-12  2.7e+22  2.6e-06  1.08e+23
##     20   42  5.484e+03  3.90e-07  3.90e-07  7.5e-12  4.1e+21  5.2e-06  1.09e+25
##     21   44  5.484e+03  7.81e-08  7.81e-08  1.5e-12  4.1e+21  1.0e-06  1.09e+27
##     22   46  5.484e+03  1.56e-07  1.56e-07  3.0e-12  6.3e+20  2.1e-06  1.09e+29
##     23   48  5.484e+03  3.13e-08  3.13e-08  6.0e-13  6.5e+20  4.2e-07  1.09e+31
##     24   50  5.484e+03  6.26e-08  6.26e-08  1.2e-12  9.9e+19  8.4e-07  1.09e+33
##     25   52  5.484e+03  1.25e-08  1.25e-08  2.4e-13  1.0e+20  1.7e-07  1.09e+35
##     26   54  5.484e+03  2.50e-08  2.50e-08  4.8e-13  1.5e+19  3.4e-07  1.09e+37
##     27   56  5.484e+03  5.01e-09  5.01e-09  9.6e-14  1.6e+19  6.7e-08  1.09e+39
##     28   58  5.484e+03  1.00e-08  1.00e-08  1.9e-13  2.4e+18  1.3e-07  1.09e+41
##     29   60  5.484e+03  2.00e-09  2.00e-09  3.9e-14  2.5e+18  2.7e-08  1.09e+43
##     30   62  5.484e+03  4.01e-09  4.01e-09  7.7e-14  3.8e+17  5.4e-08  1.09e+45
##     31   64  5.484e+03  8.02e-10  8.02e-10  1.5e-14  3.9e+17  1.1e-08  1.09e+47
##     32   66  5.484e+03  1.60e-09  1.60e-09  3.1e-14  5.9e+16  2.1e-08  1.09e+49
##     33   68  5.484e+03  3.21e-10  3.21e-10  6.2e-15  6.0e+16  4.3e-09  1.09e+51
##     34   70  5.484e+03  6.41e-10  6.41e-10  1.2e-14  9.2e+15  8.6e-09  1.09e+53
##     35   72  5.484e+03  1.28e-10  1.28e-10  2.5e-15  9.4e+15  1.7e-09  1.09e+55
##     36   74  5.484e+03  2.57e-10  2.57e-10  4.9e-15  1.4e+15  3.4e-09  1.09e+57
##     37   75  5.484e+03 -1.82e+06  5.13e-10  9.9e-15  1.5e+14  6.9e-09  1.09e+59
##
##  ***** FALSE CONVERGENCE *****
##
##  FUNCTION     5.484245e+03   RELDX        9.864e-15
##  FUNC. EVALS       75        GRAD. EVALS      37
##  PRELDF       5.131e-10      NPRELDF      1.095e+59
##
##      I      FINAL X(I)        D(I)          G(I)
##
##      1    3.483404e+05     1.000e+00    -6.073e-05
##      2    3.436141e-09     1.000e+00     4.095e+02
```

```
summary(g_gr)  # arch (0, 1) gives the best fit
```

```
##
## Call:
## garch(x = growth_resi, order = c(0, 1))
##
## Model:
## GARCH(0,1)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -20.29706   0.02525   0.07557   0.13216   0.36462
##
## Coefficient(s):
##     Estimate  Std. Error  t value Pr(>|t|)
## a0 3.483e+05   1.198e+03    290.9   <2e-16 ***
## a1 3.436e-09   3.433e-03      0.0        1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Diagnostic Tests:
##  Jarque Bera Test
##
## data:  Residuals
## X-squared = 4948700, df = 2, p-value < 2.2e-16
##
##
##  Box-Ljung test
##
## data:  Squared.Residuals
## X-squared = 0.0051985, df = 1, p-value = 0.9425
```

```
# one-step ahead forecast of garch in inflation residual
z_in <- rnorm(1) # random number follow normal distribution
sigma_last_in <- var(g_ir$fitted.values[,1], na.rm=TRUE) # last sigma
sigma_in <- sqrt(g_ir$coef[1] + g_ir$coef[2]*(g_ir$residuals[794])^2 + g_ir$coef[3]*sigma_last_in) # si
```

```
gf_step1_in <- g_ir$fitted.values[,1][794] + sigma_in*z_in # forecast 1 step
print("After calculating, one step forecast of garch(1,1) of inflation residual is ")
```

```
## [1] "After calculating, one step forecast of garch(1,1) of inflation residual is "
```

```
print(as.numeric(gf_step1_in))
```

```
## [1] -0.8881019
```

```
print("The 95% interval of this one step ahead forecast is between :")
```

```
## [1] "The 95% interval of this one step ahead forecast is between :"
```

```
print(as.numeric(gf_step1_in - 1.96*sigma_in))
```

```
## [1] -2.311577
```

```
print(" and ")
```

```
## [1] " and "
```

```
print(as.numeric(gf_step1_in +1.96*sigma_in))
```

```
## [1] 0.5353732
```

```
# one-step ahead forecast of arch in growth rate reisudal
z_gr <- rnorm(1)
meanfit_gr <- mean(g_gr$fitted.values[,1], na.rm = TRUE)
sigma_last_gr <- (g_gr$fitted.values[,1][794] - meanfit_gr)^2 # last sigma
sigma_gr <- sqrt(g_gr$coef[1] + g_gr$coef[2]*sigma_last_gr) # sigma at this time
gf_step1_gr <- g_gr$fitted.values[,1][794] + sigma_gr*z_gr  # forecast 1 step
print("After calculating, one step forecast of garch(1,1) of growthrate residual is ")
```

```
## [1] "After calculating, one step forecast of garch(1,1) of growthrate residual is "
```

```
print(as.numeric(gf_step1_gr))
```

```
## [1] 732.1386
```

```
print("The 95% interval of this one step ahead forecast is between :")
```

```
## [1] "The 95% interval of this one step ahead forecast is between :"
```

```
print(as.numeric(gf_step1_gr - 1.96*sigma_gr))
```

```
## [1] -424.6606
```

```
print(" and ")
```

```
## [1] " and "
```

```
print(as.numeric(gf_step1_gr +1.96*sigma_gr))
```

```
## [1] 1888.938
```