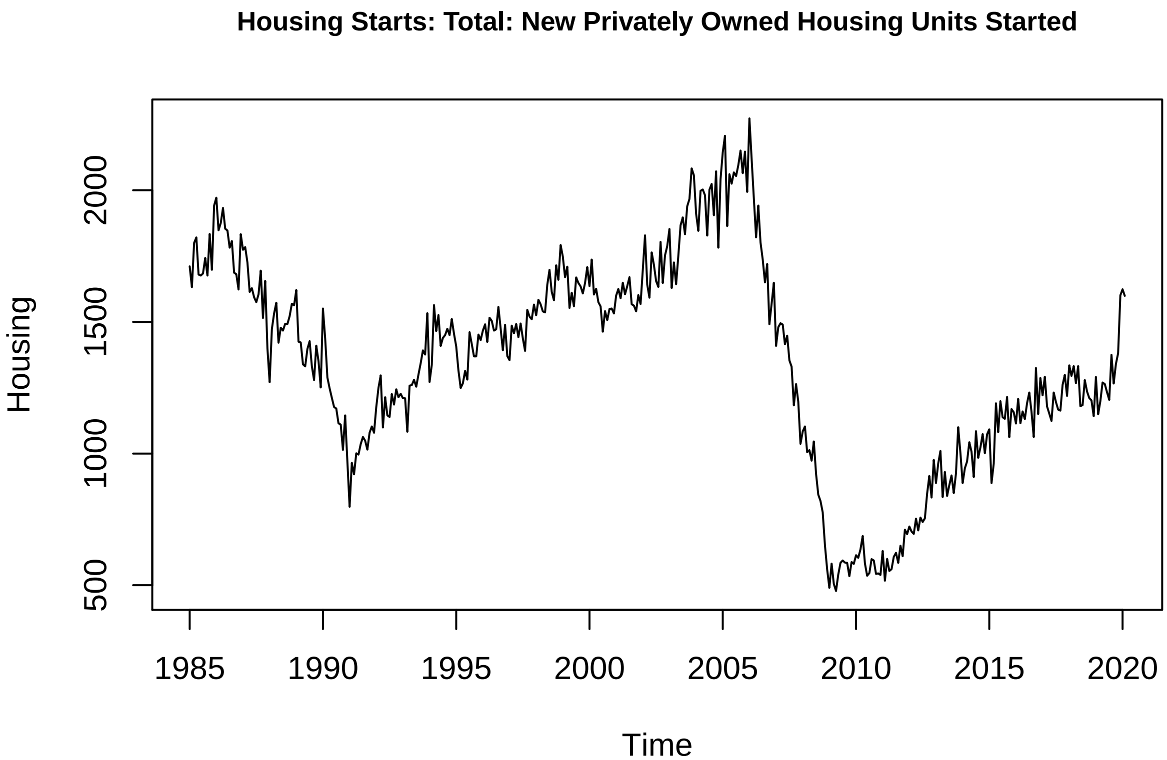
SWA Q1

b

```r
setwd("~/Desktop")
Housing_data <-read.csv(file="HOUST.csv",header=TRUE, sep=",", na = ".")
Housing <- ts(Housing_data$HOUST,start=c(1985, 1), end=c(2020, 2),frequency=12)
plot.ts(Housing,main="Housing Starts: Total: New Privately Owned Housing Units Started",cex.main=0.8)
```



c

```r
library("aTSA")
```

```
##
## Attaching package: 'aTSA'

## The following object is masked from 'package:graphics':
##
##     identify
```

```r
adf.test(Housing)
```

```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
```

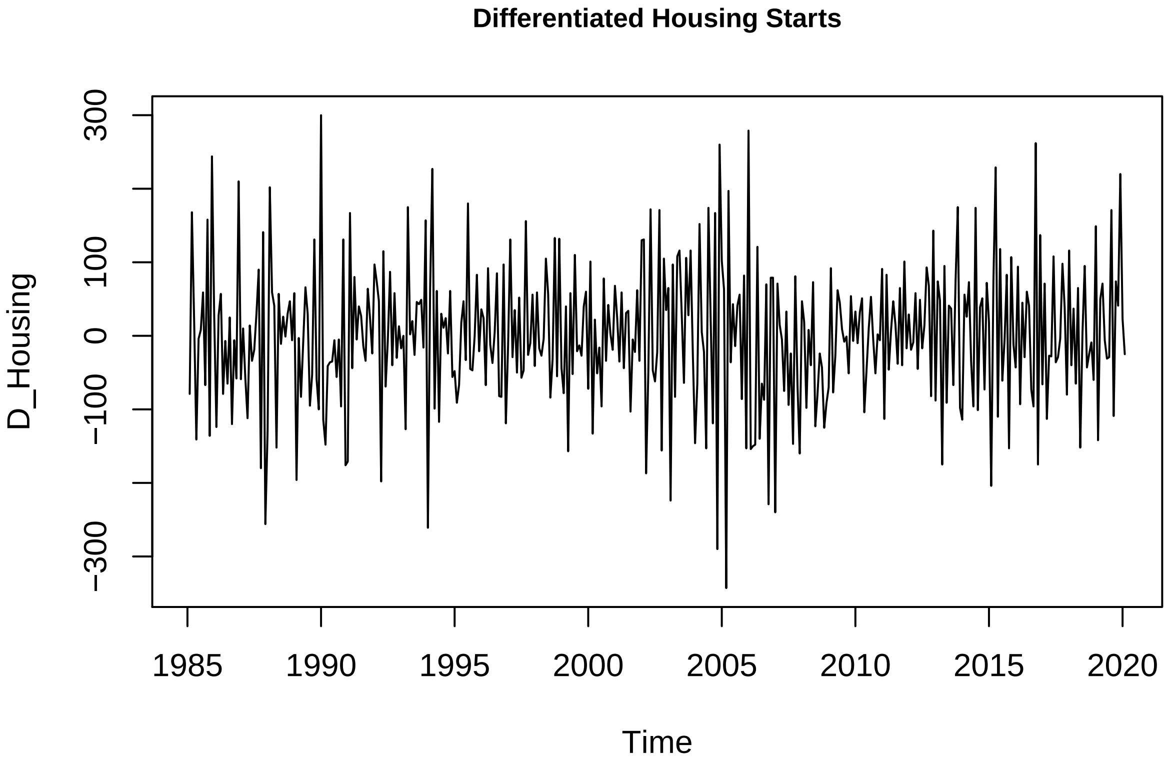
```
## [1,]   0 -0.764   0.406
## [2,]   1 -0.503   0.499
## [3,]   2 -0.535   0.487
## [4,]   3 -0.587   0.469
## [5,]   4 -0.516   0.494
## [6,]   5 -0.475   0.507
## Type 2: with drift no trend
##      lag   ADF p.value
## [1,]   0 -2.54   0.114
## [2,]   1 -1.67   0.457
## [3,]   2 -1.48   0.529
## [4,]   3 -1.46   0.537
## [5,]   4 -1.48   0.530
## [6,]   5 -1.51   0.521
## Type 3: with drift and trend
##      lag   ADF p.value
## [1,]   0 -2.53   0.351
## [2,]   1 -1.49   0.793
## [3,]   2 -1.14   0.915
## [4,]   3 -1.05   0.930
## [5,]   4 -1.14   0.915
## [6,]   5 -1.21   0.905
## ----
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

```
# library(tseries)
# adf.test(Housing)
```

Unit root tests are tests for stationarity in a time series. The null hypothesis of a presence of a unit root in AR(1) is stated as: H0: $|\phi| = 1$(non-stationary) and Ha:$|\phi| < 1$(stationary). At 5% significance level; since P-value > 0.05, ADF.TEST fail to reject H0 which equivalent to failing to reject the existence of a unit root or stochastic trend in the data series.

d

```
# difference the data
D_Housing <-diff(Housing,lag= 1,differences=1)
plot.ts(D_Housing,main="Differentiated Housing Starts",cex.main=0.8)
```




```r
adf.test(D_Housing)
```

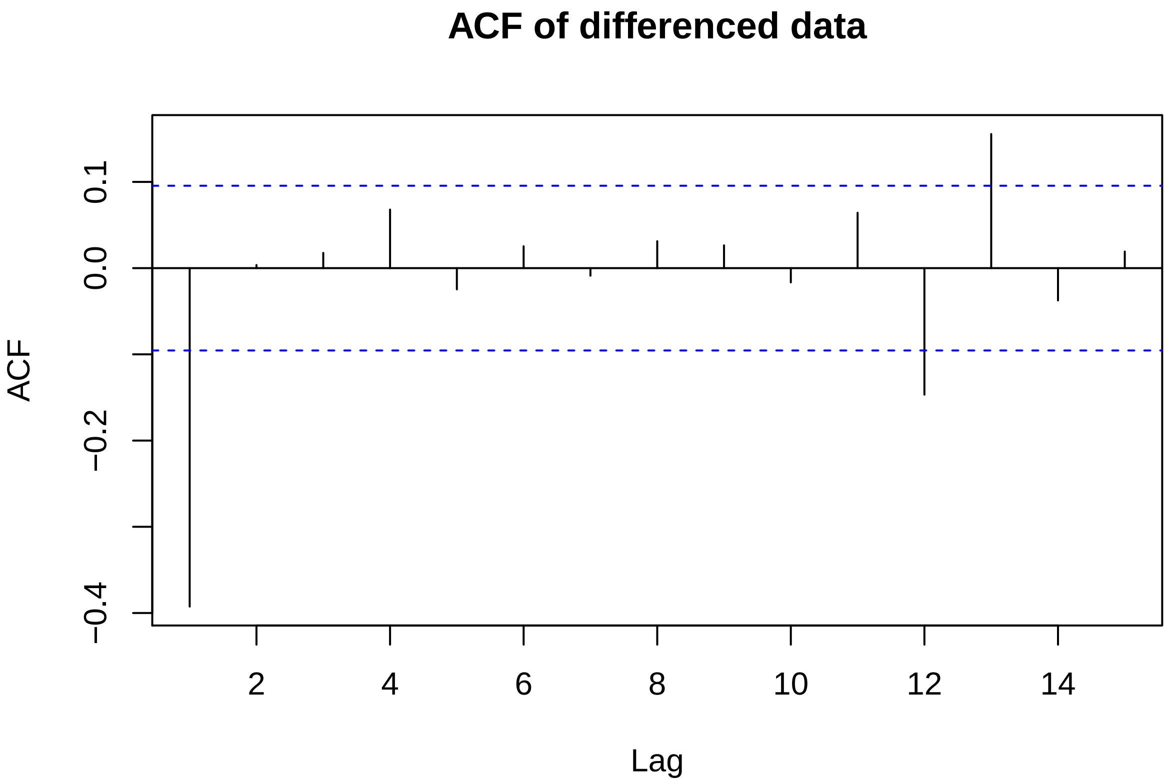
```
## Augmented Dickey-Fuller Test
## alternative: stationary
##
## Type 1: no drift no trend
##      lag    ADF p.value
## [1,]   0 -31.02    0.01
## [2,]   1 -20.40    0.01
## [3,]   2 -15.29    0.01
## [4,]   3 -11.28    0.01
## [5,]   4  -9.26    0.01
## [6,]   5  -7.86    0.01
## Type 2: with drift no trend
##      lag    ADF p.value
## [1,]   0 -30.98    0.01
## [2,]   1 -20.38    0.01
## [3,]   2 -15.28    0.01
## [4,]   3 -11.26    0.01
## [5,]   4  -9.25    0.01
## [6,]   5  -7.85    0.01
## Type 3: with drift and trend
##      lag    ADF p.value
## [1,]   0 -30.98    0.01
## [2,]   1 -20.41    0.01
## [3,]   2 -15.33    0.01
## [4,]   3 -11.31    0.01
## [5,]   4  -9.29    0.01
## [6,]   5  -7.90    0.01
## ----
```

```
## Note: in fact, p.value = 0.01 means p.value <= 0.01
```

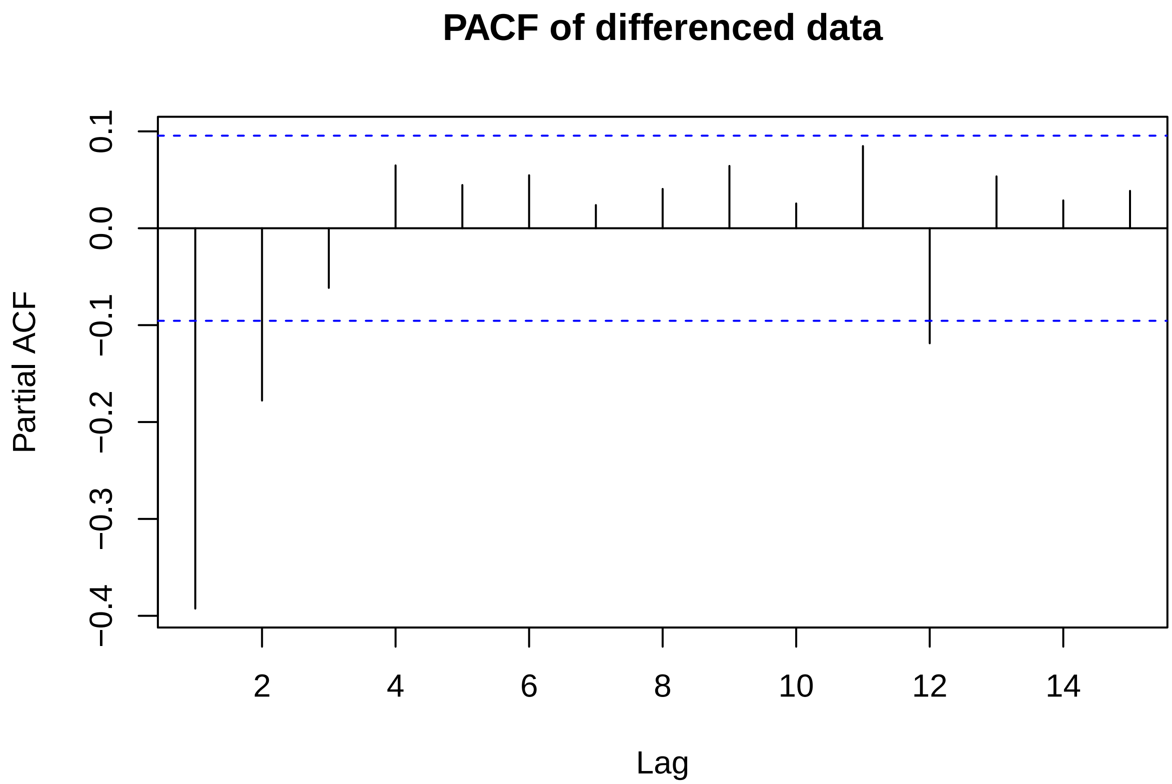
Since all P-value< 0.05, ADF.TEST will reject H0 which means we reject the existence of a unit root in AR(1).The differentiated series is "the working series".

e

```
library(stats)
ACF <- acf(D_Housing[1:length(D_Housing)],lag.max = NULL , plot = FALSE,
           demean = TRUE)
plot(ACF[1:15],main="ACF of differenced data")
```




```
PACF <- pacf(D_Housing[1:length(D_Housing)],lag.max = NULL, plot = FALSE,
             demean = TRUE,main="ACF")
plot(PACF[1:15],main="PACF of differenced data")
```



f

```
#non-seasonal
m1 <- arima(D_Housing, order = c(2,0,1), include.mean=TRUE)
m1
```

```
##
## Call:
## arima(x = D_Housing, order = c(2, 0, 1), include.mean = TRUE)
##
## Coefficients:
##           ar1      ar2      ma1  intercept
##       -0.2951  -0.1142  -0.1732    -0.2410
## s.e.   0.1838   0.0886   0.1811     2.4661
##
## sigma^2 estimated as 7417:  log likelihood = -2473.38,  aic = 4956.75
```

```
m2 <- arima(D_Housing, order = c(12,0,13), include.mean=TRUE)
```

```
## Warning in arima(D_Housing, order = c(12, 0, 13), include.mean = TRUE): possible
## convergence problem: optim gave code = 1
```

```
m2
```

```
##
## Call:
## arima(x = D_Housing, order = c(12, 0, 13), include.mean = TRUE)
##
## Coefficients:
##           ar1      ar2     ar3     ar4     ar5      ar6      ar7      ar8
##       -0.0081  -0.0121  0.1391  0.2476  0.1569  -0.1161  -0.3411  -0.1476
## s.e.   0.1227   0.1110  0.1003  0.0898  0.1160   0.0945   0.0803   0.1017
```

```
##          ar9     ar10    ar11    ar12      ma1     ma2     ma3      ma4
##      -0.1177  -0.0388  0.2058  0.5445  -0.4321  0.0160  -0.123  -0.0858
## s.e.   0.0968   0.0866  0.0864  0.0853   0.1245  0.0893   0.089   0.0923
##          ma5      ma6     ma7      ma8      ma9    ma10     ma11     ma12
##      -0.0367   0.1268  0.3730  -0.0146  0.0706  0.0382  -0.1219  -0.7358
## s.e.   0.0918   0.0744  0.0688   0.0940  0.0866  0.0875   0.0843   0.0772
##         ma13  intercept
##       0.4943     0.5161
## s.e.  0.0592     4.4430
##
## sigma^2 estimated as 6275:  log likelihood = -2444.13,  aic = 4942.25
```

```r
#seasonal
m3 <- arima(D_Housing, order = c(9,0,0), seasonal = list(order=c(0,0,1), period=4),
            include.mean=TRUE)
m3
```

```
##
## Call:
## arima(x = D_Housing, order = c(9, 0, 0), seasonal = list(order = c(0, 0, 1),
##     period = 4), include.mean = TRUE)
##
## Coefficients:
##          ar1      ar2      ar3      ar4      ar5      ar6     ar7     ar8
##      -0.4751  -0.1973  -0.0236  -0.5901  -0.2515  -0.0683  0.0323  0.1593
## s.e.   0.0487   0.0536   0.0551   0.1980   0.1087   0.0686  0.0567  0.0569
##          ar9    sma1  intercept
##       0.0533  0.6942    -0.1022
## s.e.  0.0511  0.1967     2.9693
##
## sigma^2 estimated as 7215:  log likelihood = -2467.69,  aic = 4959.38
```

g

```r
library(forecast)
```

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

##
## Attaching package: 'forecast'

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

```r
m4 <- auto.arima(D_Housing)
m4
```

```
## Series: D_Housing
## ARIMA(1,0,2)(2,0,0)[12] with zero mean
##
## Coefficients:
##          ar1      ma1     ma2     sar1     sar2
##       0.9605  -1.4136  0.4757  -0.1737  -0.2143
## s.e.  0.0237   0.0479  0.0425   0.0503   0.0504
##
```

```
## sigma^2 estimated as 6929:  log likelihood=-2457.24
## AIC=4926.47   AICc=4926.68   BIC=4950.73
```

h

```r
AIC(m1)
```

```
## [1] 4956.755
```

```r
AIC(m2)
```

```
## [1] 4942.254
```

```r
AIC(m3)
```

```
## [1] 4959.377
```

```r
AIC(m4)
```

```
## [1] 4926.474
```

After comparing m1-m4,the automatic model selection for differenced data will return the smallest AIC value(4926.474). Thus, we should choose automatic model.

i

```r
library(forecast)
m5<-nnetar(y=D_Housing,4,1,5)
m5
```

```
## Series: D_Housing
## Model:  NNAR(4,1,5)[12]
## Call:   nnetar(y = D_Housing, p = 4, P = 1, size = 5)
##
## Average of 20 networks, each of which is
## a 5-5-1 network with 36 weights
## options were - linear output units
##
## sigma^2 estimated as 4889
```

```r
m6<-nnetar(y=D_Housing,4,1,10)
m6
```

```
## Series: D_Housing
## Model:  NNAR(4,1,10)[12]
## Call:   nnetar(y = D_Housing, p = 4, P = 1, size = 10)
##
## Average of 20 networks, each of which is
## a 5-10-1 network with 71 weights
## options were - linear output units
##
## sigma^2 estimated as 3368
```

j

```r
accuracy(m1)
```

```
##                        ME     RMSE      MAE  MPE MAPE     MASE         ACF1
## Training set -0.009396967 86.12395 67.11803 -Inf  Inf 0.545042 -0.002566001
```

```r
accuracy(m2)
```

```
##                       ME     RMSE      MAE MPE MAPE      MASE         ACF1
```

```
## Training set -0.05781079 79.21442 62.8039 NaN  Inf 0.5100084 -0.007140431
```

```r
accuracy(m3)
```

```
##                       ME     RMSE      MAE  MPE MAPE      MASE         ACF1
## Training set 0.01153623 84.94262 66.40058 -Inf  Inf 0.5392159 -0.001780169
```

```r
accuracy(m4)
```

```
##                      ME     RMSE      MAE MPE MAPE      MASE         ACF1
## Training set 0.4529945 82.74605 65.17219 NaN  Inf 0.5808168 -0.001237778
```

```r
accuracy(m5)
```

```
##                        ME     RMSE      MAE  MPE MAPE     MASE        ACF1
## Training set -0.04515316 69.91966 55.30376 -Inf  Inf 0.492869 -0.01489439
```

```r
accuracy(m6)
```

```
##                        ME     RMSE      MAE  MPE MAPE      MASE        ACF1
## Training set -0.06358919 58.03391 45.81407 -Inf  Inf 0.4082966 -0.01448813
```

After comparing the m1-m6 model,the m6 model(seasonal autoregressive neural network model with 10 nodes) has the smallest RMSE and MAE. Thus we will choose m6 model.

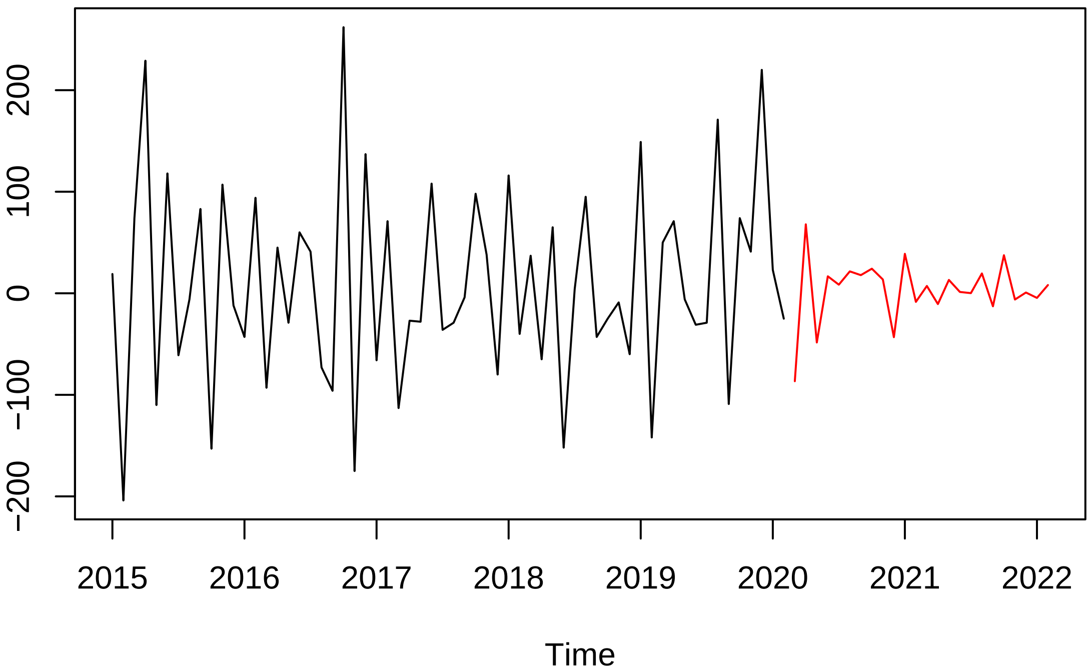
k

```r
forecast_housing <- predict(m6,n.ahead=12)
forecast_housing
```

```
##             Jan         Feb         Mar         Apr         May         Jun
## 2020                        -86.5994869  67.9108690 -48.4553420  16.7249487
## 2021  38.8845387  -8.4746990   7.2064939 -10.6364451  13.1668609   1.3053696
## 2022  -4.5539686   8.1064297
##             Jul         Aug         Sep         Oct         Nov         Dec
## 2020   8.4776229  21.5637249  17.8575249  24.2192896  13.5680886 -43.2486913
## 2021   0.1510668  19.5435473 -12.8344494  37.4869624  -6.1494667   0.7532506
## 2022
```

i

```r
D_Housing_2015<- window (D_Housing,start=c(2015,1))
ts.plot(D_Housing_2015, forecast_housing$mean,
        gpars = list(col = c("black", "red")))
```


200
100
0
−100
−200
2015
2016
2017
2018
2019
2020
2021
2022
Time

SWA Q2

a

```r
# install.packages("quantmod")
# install.packages("anytime")
# install.packages("MASS")
# install.packages("MTS")
# install.packages("tsbox")
library(xts)
```

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

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

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

```r
library(anytime)
library(quantmod)
```

```
## Loading required package: TTR
```

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

```
## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```r
library(MASS)  # functions: fitdistr
library(MTS)   # functions: EWMAvol
```

```
##
## Attaching package: 'MTS'
```

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

```r
library(tsbox) # functions: ts_ts
library(stats) # functions: acf, pacf

symbols <- "^IXIC"
indexcdat <- getSymbols(symbols, src = "yahoo", from = "1998-01-01")
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
```

```
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```r
class(IXIC)
```

```
## [1] "xts" "zoo"
```

b

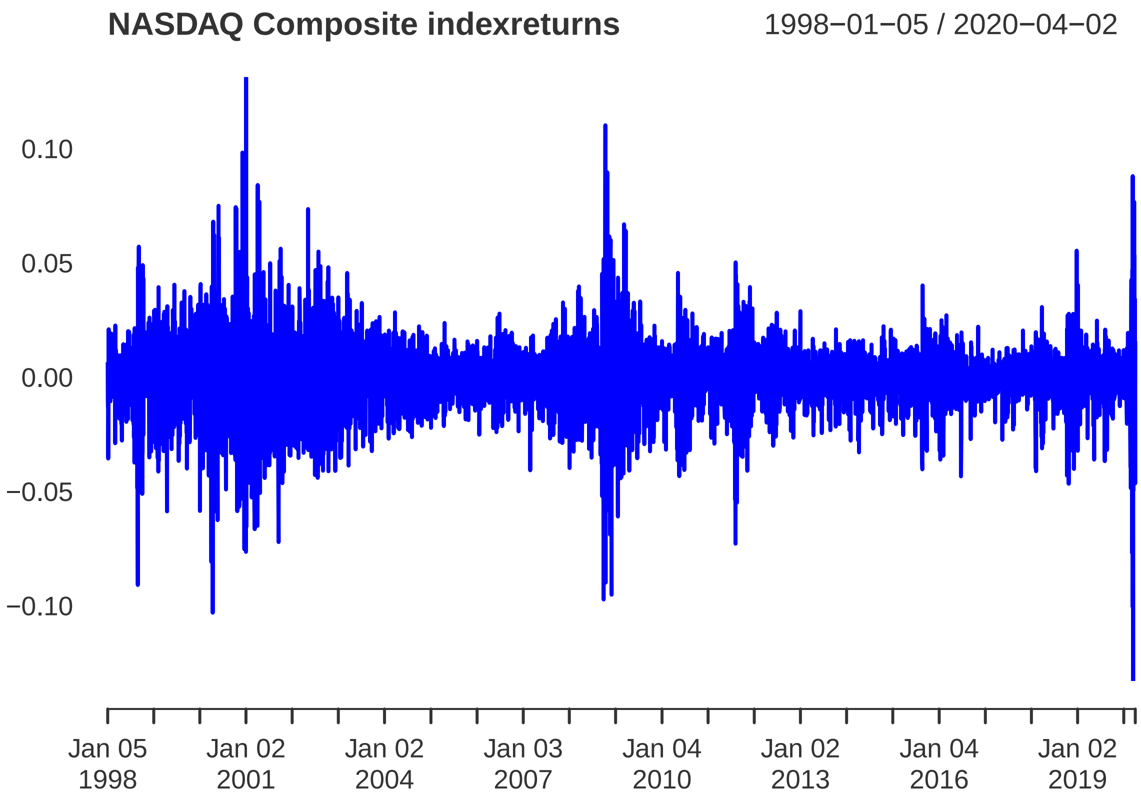
```r
NASDAQ <- IXIC$IXIC.Close
colnames(NASDAQ)[1] <- "NASDAQ_IXIC"
class(NASDAQ)
```

```
## [1] "xts" "zoo"
```

c

```r
#plot of the index
plot.xts(NASDAQ, grid.col="white",  yaxis.right=FALSE, main ="NASDAQ Composite", cex.main=0.8)
```


```r
#plot of the returns
NASDAQ_returns <- na.omit(diff(log(NASDAQ)))
plot.xts(NASDAQ_returns, grid.col="white", yaxis.right=FALSE, col="blue",
         main ="NASDAQ Composite indexreturns", cex.main=0.8)
```



d

```
symbols <- "AMZN"
indexcdat <- getSymbols(symbols, src = "yahoo", from = "1998-01-01")
class(IXIC)
```

```
## [1] "xts" "zoo"
```

e

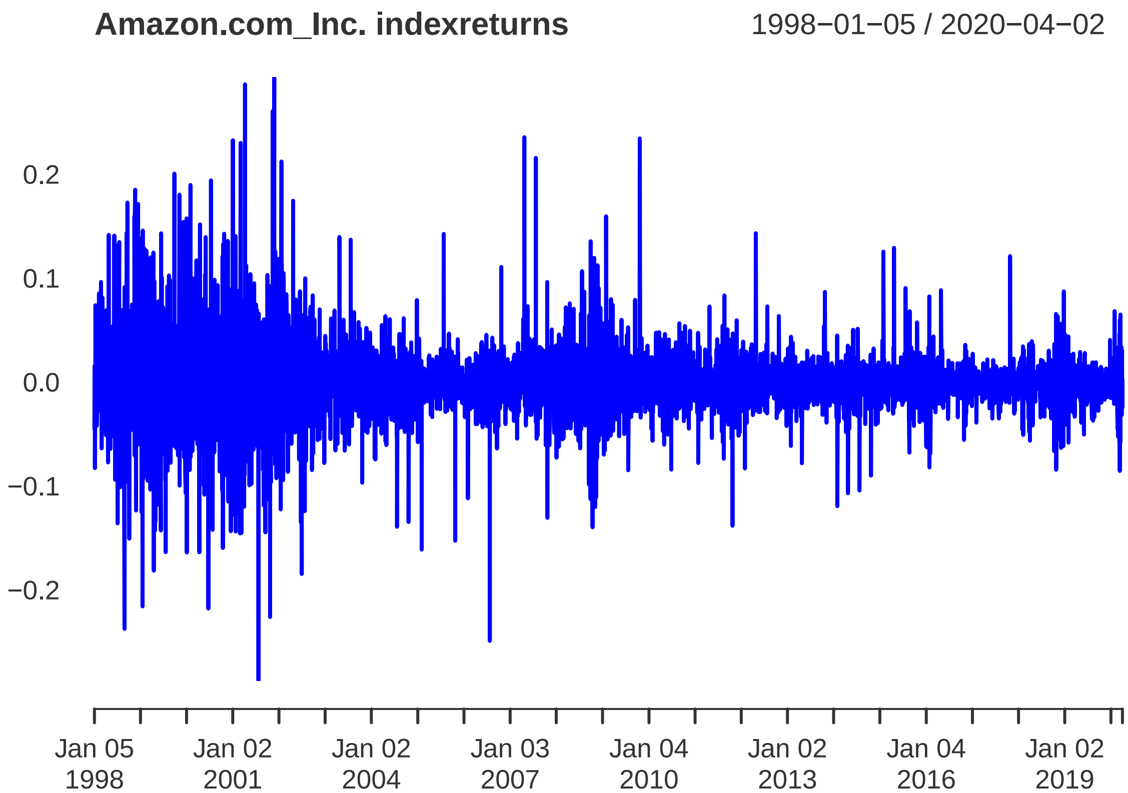
```
Amazon.com_Inc. <- AMZN$AMZN.Close
colnames(Amazon.com_Inc.)[1] <- "Close"
class(Amazon.com_Inc.)
```

```
## [1] "xts" "zoo"
```

f

```
#plot of the index
plot.xts(Amazon.com_Inc., grid.col="white",  yaxis.right=FALSE,
         main ="Amazon.com_Inc. Price series", cex.main=0.8)
```


```r
#plot of the returns
Amazon.com_Inc._returns <- na.omit(diff(log(Amazon.com_Inc.)))
plot.xts(Amazon.com_Inc._returns, grid.col="white", yaxis.right=FALSE,
         col="blue", main ="Amazon.com_Inc. indexreturns", cex.main=0.8)
```



g

```
library(quantmod) # functions: getSymbols
library(xts)      # functions: xts
library(anytime)  # functions: anytime
library(rugarch)
```

```
## Loading required package: parallel

##
## Attaching package: 'rugarch'

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

```
garch_model <- ugarchspec(variance.model = list(model="sGARCH", garchOrder=c(1, 1)),
                          mean.model = list(armaOrder=c(1,0), include.mean=TRUE),
                          distribution.model = "norm")
#garch_model for NASDAQ_returns
garch_fit1 <- ugarchfit(spec = garch_model, data = NASDAQ_returns)
garch_fit1@fit$coef
```

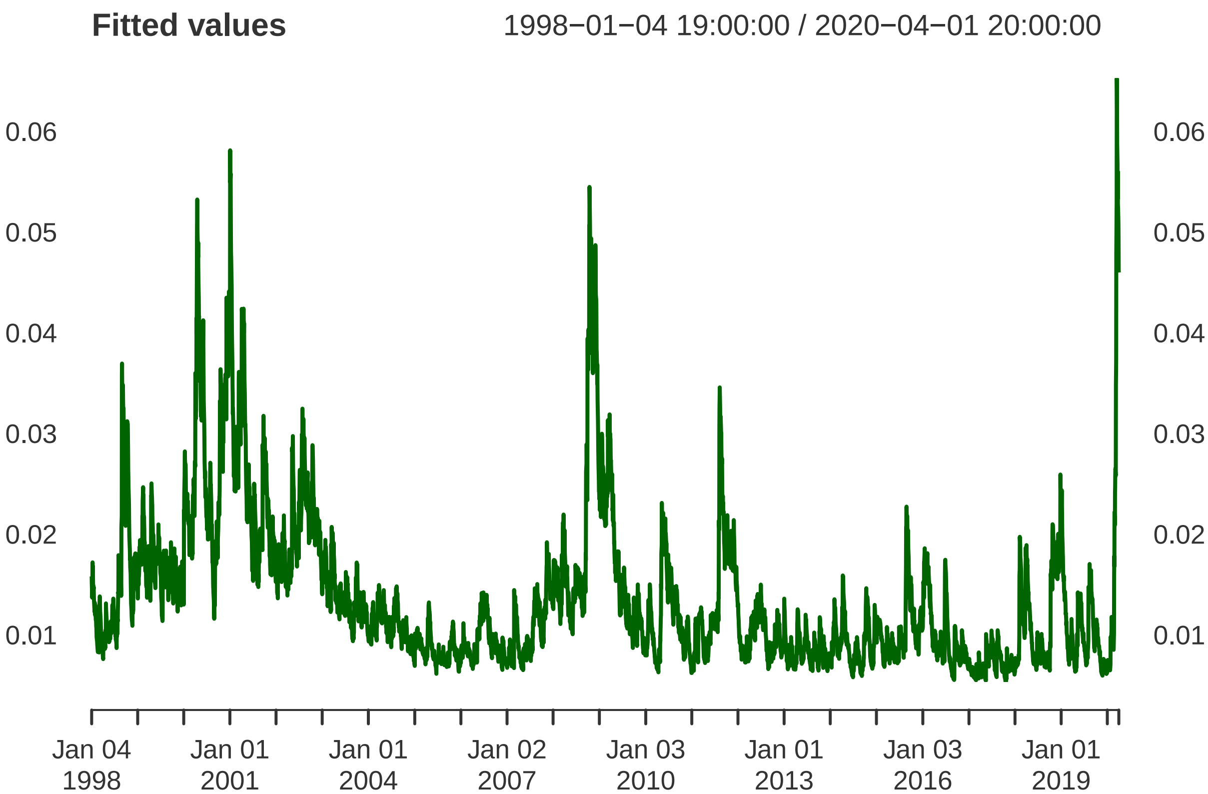
```
##            mu           ar1         omega        alpha1         beta1
##  8.047454e-04 -2.599940e-02  2.603671e-06  1.057781e-01  8.838091e-01
```

```
#garch_model for Amazon.com_Inc._returns
garch_fit2 <- ugarchfit(spec = garch_model, data = Amazon.com_Inc._returns)
garch_fit2@fit$coef
```

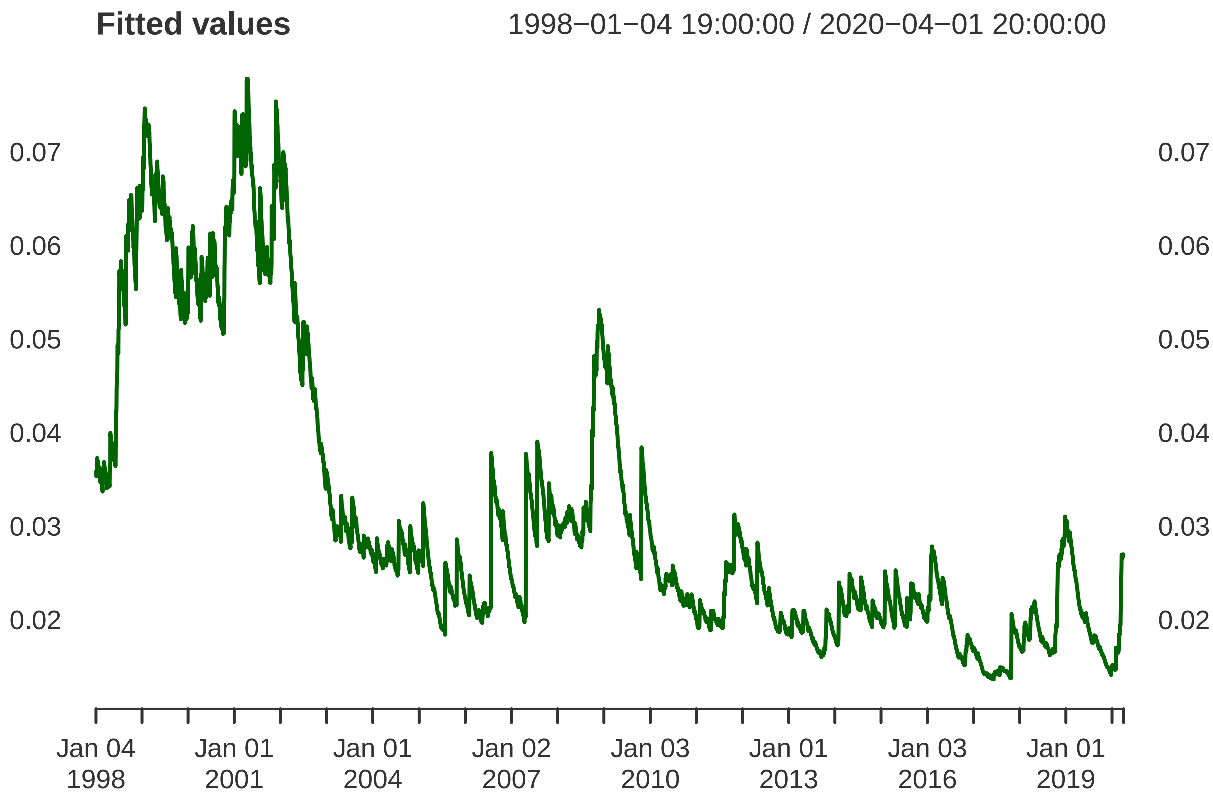
```
##           mu          ar1        omega       alpha1        beta1
## 1.251083e-03 8.647841e-03 1.856089e-06 1.599641e-02 9.820744e-01
```

h

```
# fitted volatility for NASDAQ data
garch_forecast1<- ugarchforecast(fit = garch_fit1, n.ahead = 20)
dates1 <- index(NASDAQ_returns)
t1 <- start(NASDAQ_returns)
fit_NASDAQ <- xts(garch_forecast1@model$modeldata$sigma,
                  start=t1, order.by=anytime(dates1))
plot.xts(fit_NASDAQ, grid.col="white", col="darkgreen",
         ylab="", main = "Fitted values", cex.main=0.8)
```




```
#fitted volatility for Amazon.com_Inc. data
garch_forecast2<- ugarchforecast(fit = garch_fit2, n.ahead = 20)
dates2 <- index(Amazon.com_Inc._returns)
t2 <- start(Amazon.com_Inc._returns)
fit_Amazon.com_Inc. <- xts(garch_forecast2@model$modeldata$sigma, start=t2,
                           order.by=anytime(dates2))
plot.xts(fit_Amazon.com_Inc., grid.col="white", col="darkgreen",
         ylab="", main = "Fitted values", cex.main=0.8)
```



I

```
# comparison for NASDAQ
mean(fit_NASDAQ)#average fitted value
```

```
## [1] 0.01401544
```

```
fit_NASDAQ[end(fit_NASDAQ)]#last fitted value
```

```
##                           [,1]
## 2020-04-01 20:00:00 0.04629654
```

For NASDAQ data, the average fitted value 0.01401079 smaller than last fitted value 0.04655905.

```
# comparison for Amazon
mean(fit_Amazon.com_Inc.) #average fitted value
```

```
## [1] 0.03191089
```

```
fit_Amazon.com_Inc.[end(fit_Amazon.com_Inc.)]#last fitted value
```

```
##                           [,1]
## 2020-04-01 20:00:00 0.02707155
```

For Amazon data, the average fitted value 0.03190947 larger than last fitted value 0.02717641