HuYu2_TimeSeries1

Zhaowei Cai

Mar 26,2024 - Mar 29, 2024

Loading packages

```
library(tseries)
```

```
## Warning: package 'tseries' was built under R version 4.2.3
```

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

```
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.2.3
```

```
library(stringr)
library(PerformanceAnalytics)
```

```
## Warning: package 'PerformanceAnalytics' was built under R version 4.2.3
```

```
## Loading required package: xts
```

```
## Warning: package 'xts' was built under R version 4.2.3
```

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

```
## Warning: package 'zoo' was built under R version 4.2.3
```

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

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

```
##
## Attaching package: 'PerformanceAnalytics'
```

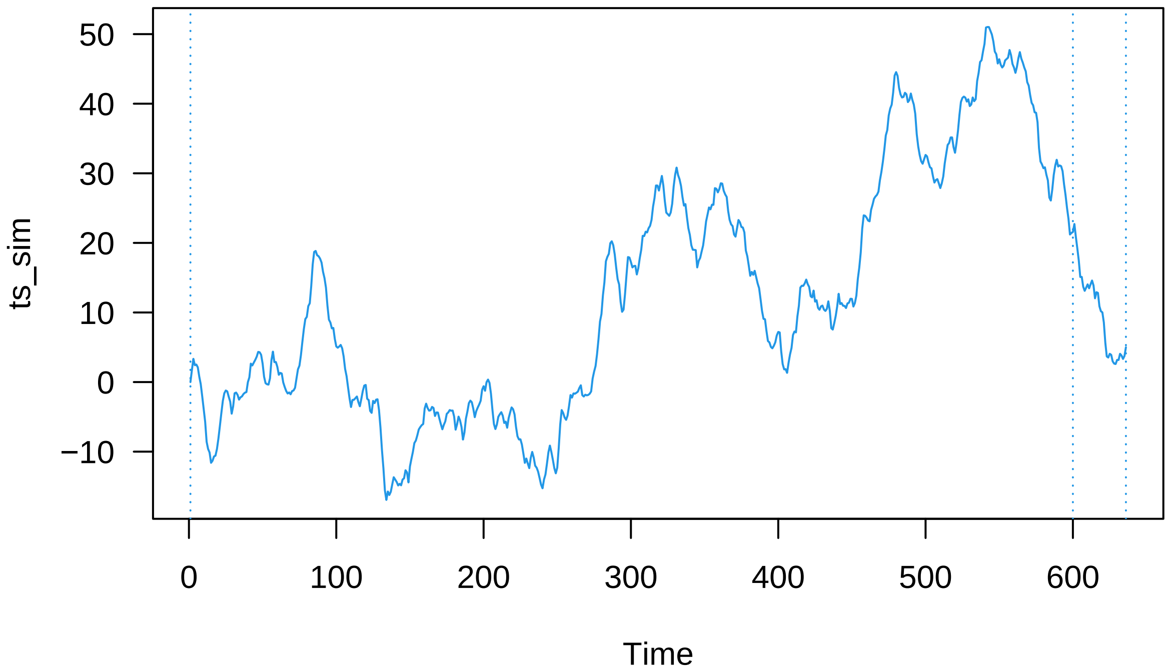
```
## The following object is masked from 'package:graphics':
##
##     legend
```

```
library(xts)
```

Part 1

```
set.seed(123)
#create a time series with right observations and first element is 0
ts_sim <- arima.sim(list(order = c(1,1,0), ar=0.65), n = 635)

left <- 600
right <- 636
it <- left:right
```

```
plot(ts_sim, col=4, las=1)
abline(v=c(1, left, right), lty="dotted", col=4)
```

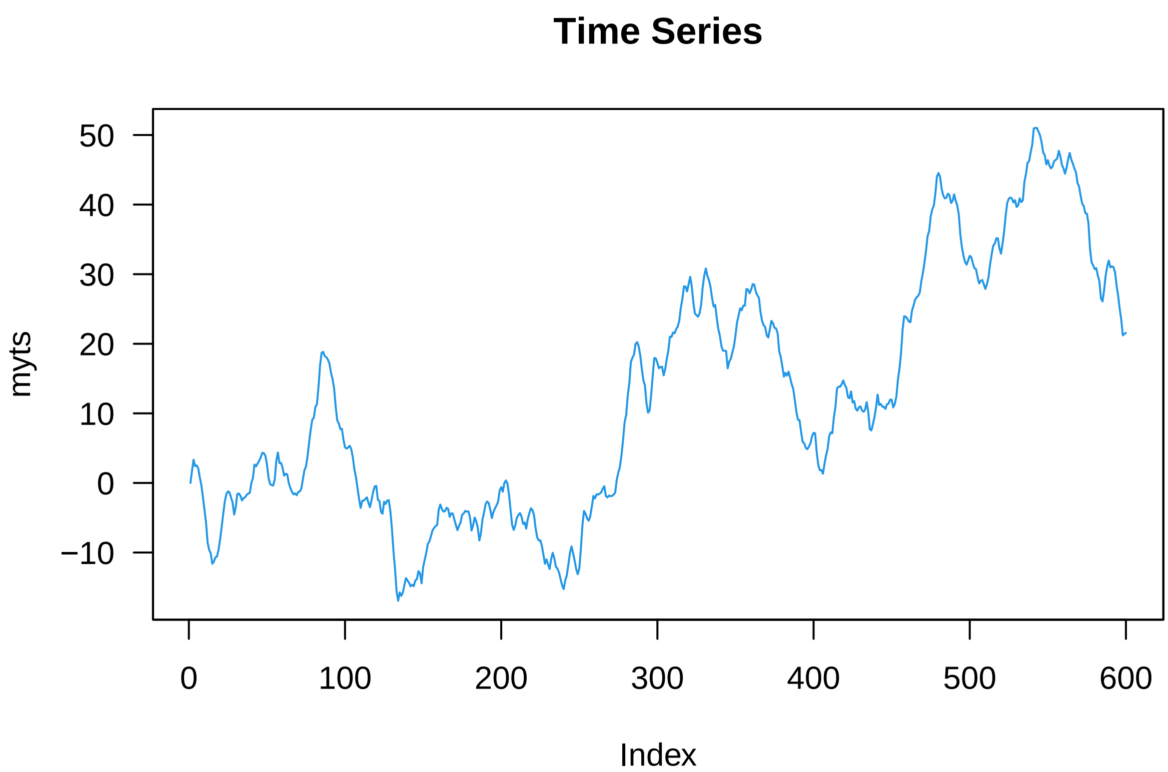


Part 2

```
myts = subset(ts_sim, subset=rep(c(TRUE, FALSE), times=c(600, 36)))
```

Step 1: visualize myts

```
plot.zoo(myts, col=4, las=1, main="Time Series")
```



Step 2: unit root test (augmented Dickey-Fuller) of myts

```
adf.test(myts, alternative = 'stationary')
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  myts
## Dickey-Fuller = -2.5821, Lag order = 8, p-value = 0.3319
## alternative hypothesis: stationary
```

P-value greater than 0.05, not reject H0, and it is not stationary.

Step 3: differentiate myts, creating mydts

```
mydts = diff(myts)
```

Step 4: unit root test (augmented Dickey-Fuller) of mydts

```
adf.test(mydts, alternative = 'stationary')
```

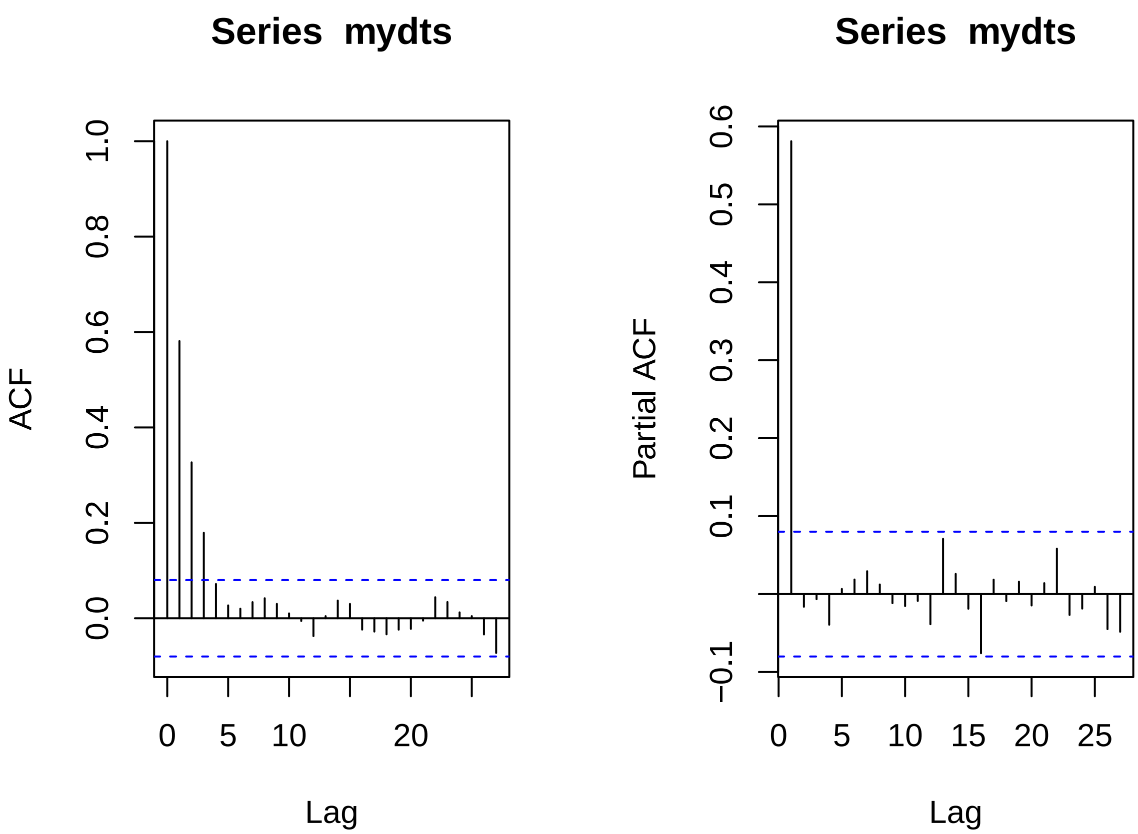
```
## Warning in adf.test(mydts, alternative = "stationary"): p-value smaller than
## printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  mydts
## Dickey-Fuller = -6.9219, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

P-value less than 0.05, reject H0, and it is stationary.

Step 5: identify lags for mydts

```
par(mfrow=c(1,2), mar=c(5,4,3,3))
acf(mydts)
pacf(mydts)
```



ACF decreases slowly, but PACF shows that it is an AR(1) (lag=1 is relevant only).

Step 6: train the model with auto.arima for mydts

```
fit_mydts = auto.arima(
  mydts,
  max.p = 3,
  max.q = 3,
  ic = "aicc",
  seasonal = FALSE,
  stationary = TRUE,
  lambda = NULL,
  stepwise = FALSE,
  approximation = FALSE
)
summary(fit_mydts)
```

```
## Series: mydts
## ARIMA(1,0,0) with zero mean
##
## Coefficients:
##          ar1
##       0.5825
## s.e.  0.0332
```

```
##
## sigma^2 = 0.9306:  log likelihood = -828.11
## AIC=1660.23   AICc=1660.25   BIC=1669.02
##
## Training set error measures:
##                      ME      RMSE       MAE      MPE     MAPE     MASE
## Training set 0.01461309 0.9638761 0.7660394 104.8523 219.6293 0.888647
##                     ACF1
## Training set 0.007771817
```

Step 7: fit the original time series, i.e. myts

```
fit_myts = arima(myts, c(1, 1, 0))
summary(fit_myts)
```

```
##
## Call:
## arima(x = myts, order = c(1, 1, 0))
##
## Coefficients:
##          ar1
##       0.5825
## s.e.  0.0332
##
## sigma^2 estimated as 0.9291:  log likelihood = -828.11,  aic = 1660.23
##
## Training set error measures:
##                      ME      RMSE       MAE       MPE     MAPE      MASE
## Training set 0.01458873 0.9630725 0.7647627 -42.61052 65.48695 0.8068719
##                     ACF1
## Training set 0.007735488
```

Or can directly fit the original time series

```
fit_myts2 = auto.arima(myts)
summary(fit_myts2)
```

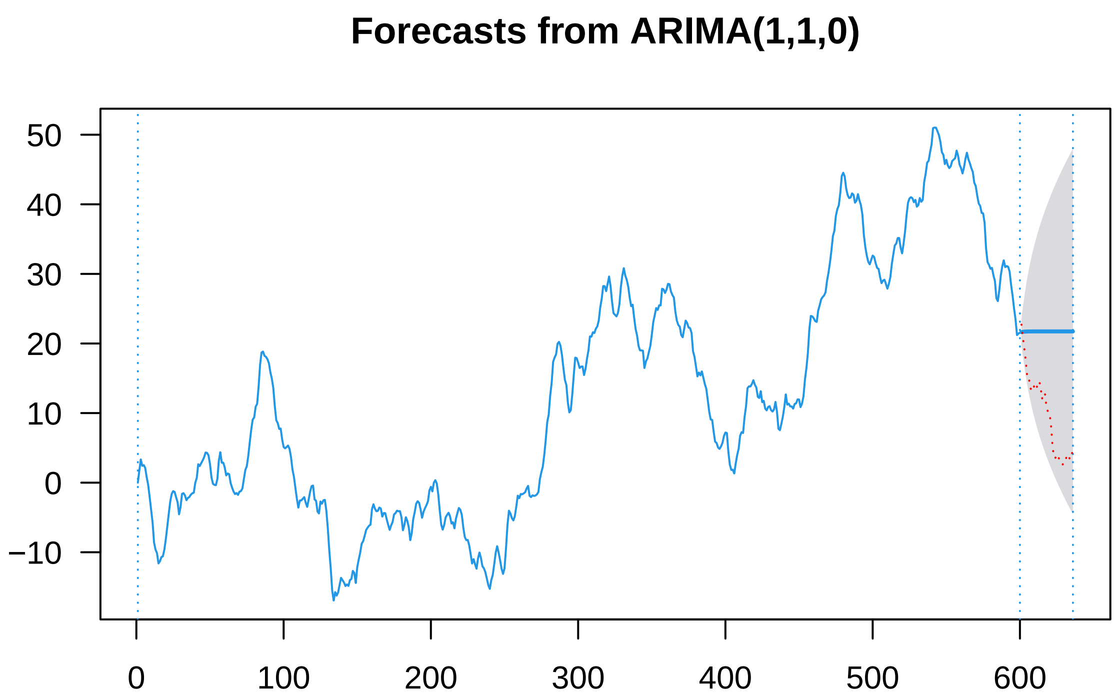
```
## Series: myts
## ARIMA(1,1,0)
##
## Coefficients:
##          ar1
##       0.5825
## s.e.  0.0332
##
## sigma^2 = 0.9306:  log likelihood = -828.11
## AIC=1660.23   AICc=1660.25   BIC=1669.02
##
## Training set error measures:
##                      ME      RMSE       MAE       MPE     MAPE      MASE
## Training set 0.01458873 0.9630725 0.7647627 -42.61052 65.48695 0.8068719
```

```
##                      ACF1
## Training set 0.007735488
```

Part 3

Part 3(a)

```r
forecast_myts = forecast(fit_myts, h=36, level=0.95)
plot(forecast_myts, col=4, las=1)
abline(v=c(1, 600, 636), lty="dotted", col=4)
lines(601:636, ts_sim[601:636], lty="dotted", col="red")
```



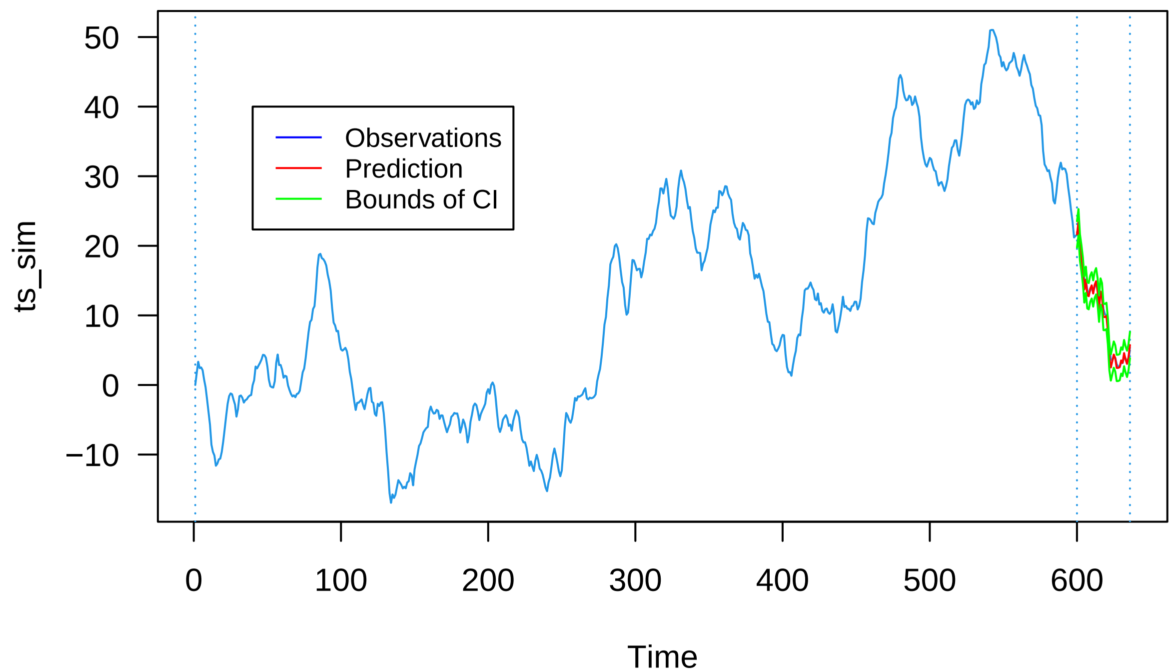

```r
# red is observation and blue is prediction
```

Part 3(b)

```r
# since it is one step ahead predictin, so we need use for loop
pred_df <- data.frame(NULL)
for(t in 600:636){
  pred_onestep <- forecast(ts_sim[1:t], h=1, level=0.95, model = fit_myts)
```

```
  pred_df <- rbind(pred_df, data.frame(mean = pred_onestep$mean[1], lower = pred_onestep$lower[1], uppe
}
```

```
plot(ts_sim, col=4, las=1)
abline(v=c(1, left, right), lty="dotted", col=4)

lines(it, pred_df$mean, col = 'red')
lines(it, pred_df$lower, col = 'green')
lines(it, pred_df$upper, col = 'green')
legend(40, 40, legend=c("Observations", "Prediction", "Bounds of CI"),col=c("blue", "red", "green"),lty
```



Part 4

Generate AR(1) model data

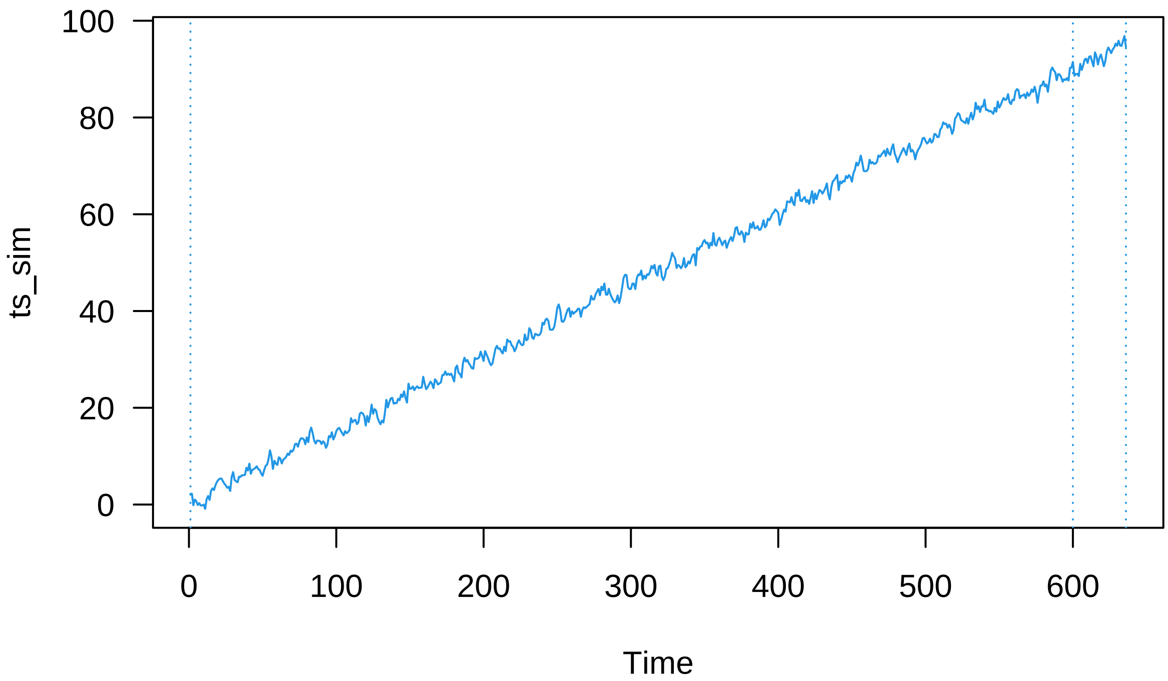
```
set.seed(123)
ts_sim = arima.sim(list(ar=0.65),n=636)
```

add trend in the data

```
ts_sim=ts_sim + 0.33 + 0.15*time(ts_sim)
```

Generate plots

```
plot(ts_sim, col=4, las=1)
abline(v=c(1, 600, 636), lty="dotted", col=4)
```

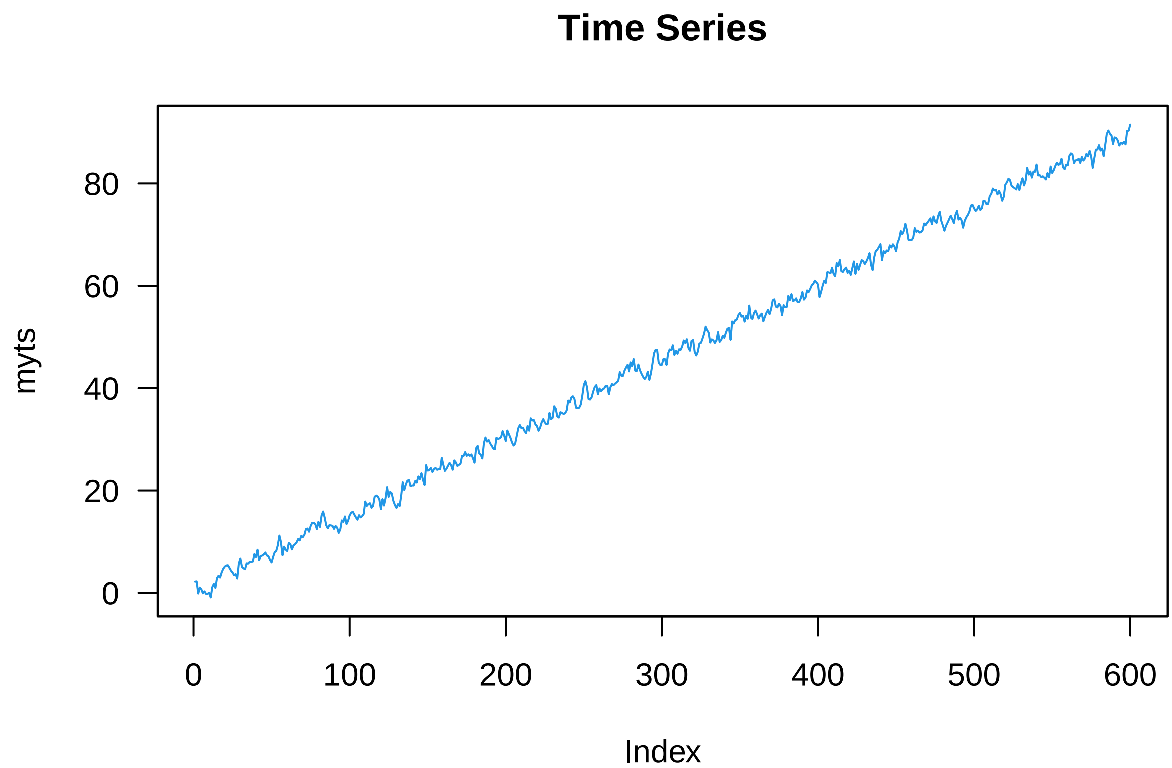


Part 5

```
myts = subset(ts_sim, subset=rep(c(TRUE, FALSE), times=c(600, 36)))
```

Step 1: visualize myts

```
plot.zoo(myts, col=4, las=1, main="Time Series")
```

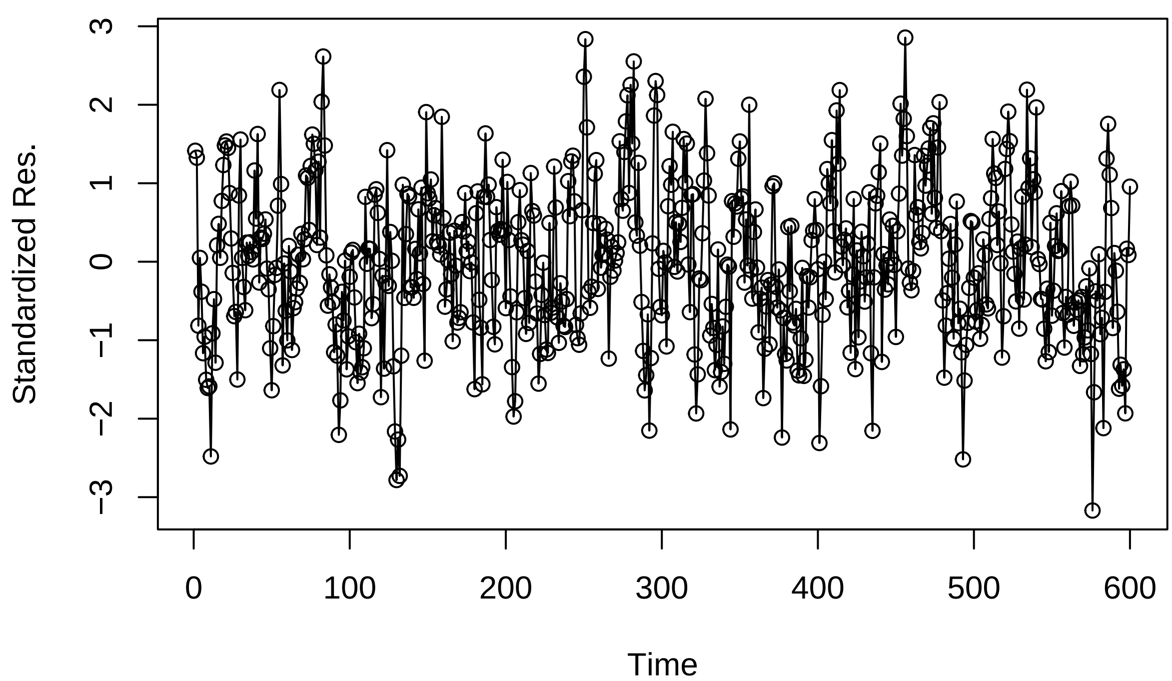


Fit trend part

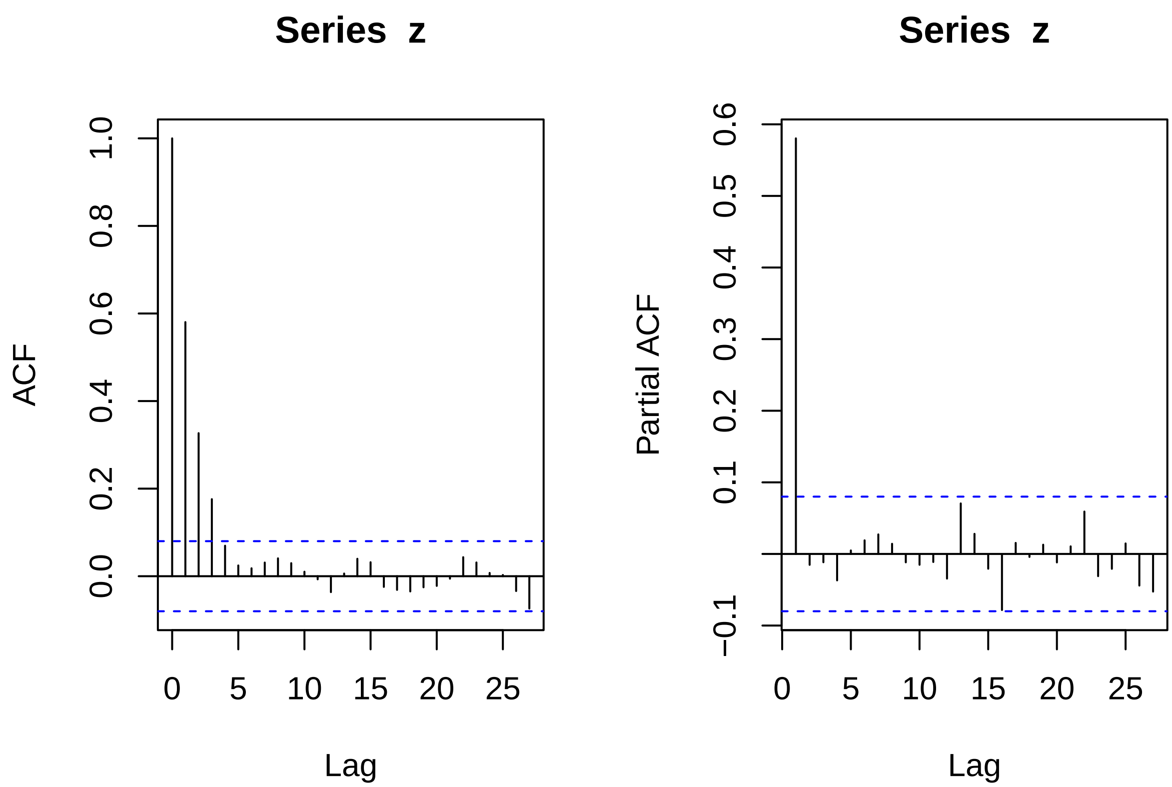
```
time = time(myts)
reg=lm(myts~time)
summary(reg)
```

```
##
## Call:
## lm(formula = myts ~ time)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7239 -0.7967 -0.0139  0.8200  3.3646
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.3782143  0.0970450   3.897 0.000108 ***
## time        0.1499655  0.0002798 535.983  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.187 on 598 degrees of freedom
## Multiple R-squared:  0.9979, Adjusted R-squared:  0.9979
## F-statistic: 2.873e+05 on 1 and 598 DF,  p-value: < 2.2e-16
```

```r
plot(as.vector(time(myts)), rstudent(reg), ylab="Standardized Res.", xlab="Time", type="o")
```




```r
z=rstandard(reg)
par(mfrow=c(1,2))
acf(z)
pacf(z)
```



Remove the trend part and fit the residuals

```
newts=ts(residuals(reg))
```

Step 6: train the model with auto.arima for newts

```
fit_newts = auto.arima(newts, max.p=3, max.q=3, ic="aicc",
                        seasonal=FALSE, stationary=TRUE, lambda=NULL,
                        stepwise=FALSE, approximation=FALSE
                        )
summary(fit_newts)
```

```
## Series: newts
## ARIMA(1,0,0) with zero mean
##
## Coefficients:
##          ar1
##       0.5822
## s.e.  0.0332
##
## sigma^2 = 0.9307:  log likelihood = -829.51
## AIC=1663.03   AICc=1663.05   BIC=1671.82
##
## Training set error measures:
##                        ME     RMSE       MAE      MPE     MAPE      MASE
## Training set 0.0005768971 0.963908 0.7668777 124.2796 219.0426 0.8893309
```

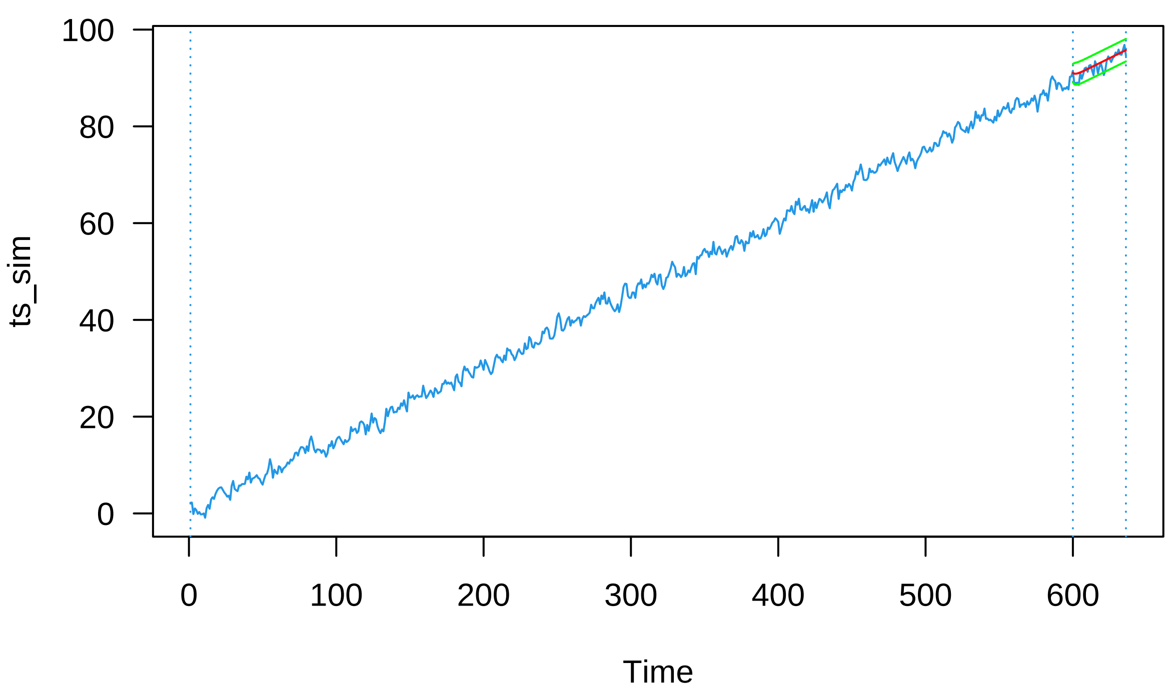
```
##                      ACF1
## Training set 0.008009318
```

Part 6

Part 6(a)

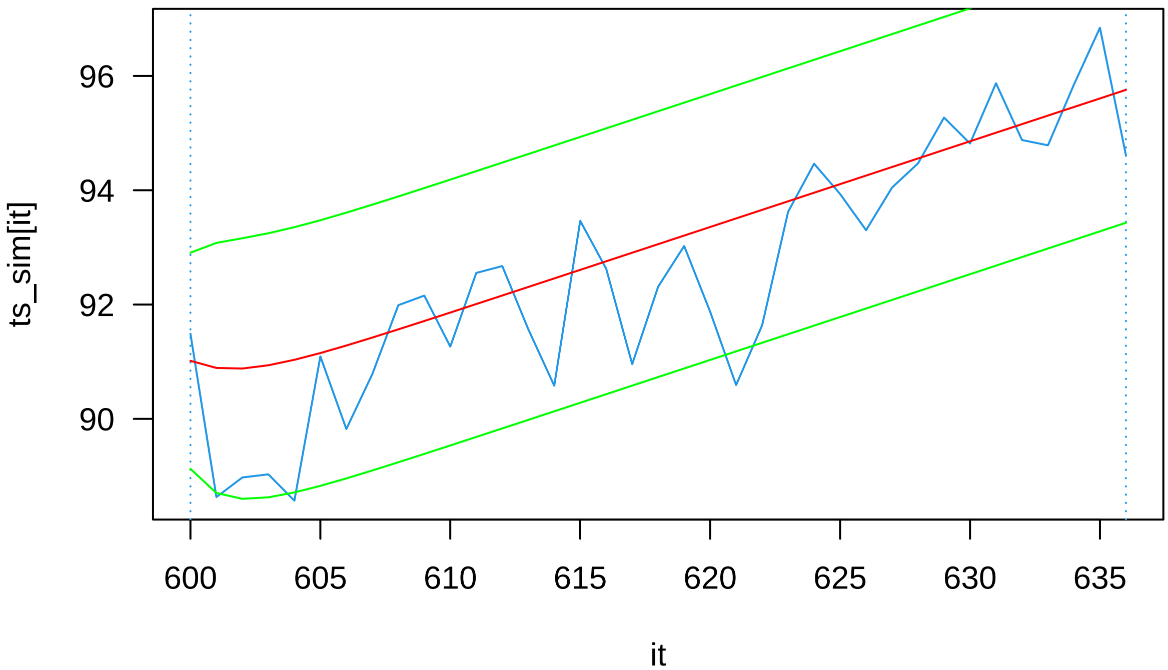
```r
prediction <- forecast(fit_newts, h=length(it), level=0.95)
pred_df <- data.frame(time = it)
pred_df$mean <- prediction$mean + predict(reg, newdata = data.frame(time = it))
pred_df$lower <- prediction$lower + predict(reg, newdata = data.frame(time = it))
pred_df$upper <- prediction$upper + predict(reg, newdata = data.frame(time = it))
```

```r
plot(ts_sim, col=4, las=1)
abline(v=c(1, left, right), lty="dotted", col=4)
lines(it, pred_df$mean, col = 'red')
lines(it, pred_df$lower, col = 'green')
lines(it, pred_df$upper, col = 'green')
legend(40, 150, legend=c("Observations", "Prediction", "Bounds of CI"),col=c("blue", "red", "green"),lty
```




```r
plot(it, ts_sim[it], col=4, las=1, type = 'l')
abline(v=c(left, right), lty="dotted", col=4)
```

```
lines(it, pred_df$mean, col = 'red')
lines(it, pred_df$lower, col = 'green')
lines(it, pred_df$upper, col = 'green')
legend(left, 128, legend=c("Observations", "Prediction", "Bounds of CI"),col=c("blue", "red", "green"),l
```



Part 6(b) Predict the residuls part

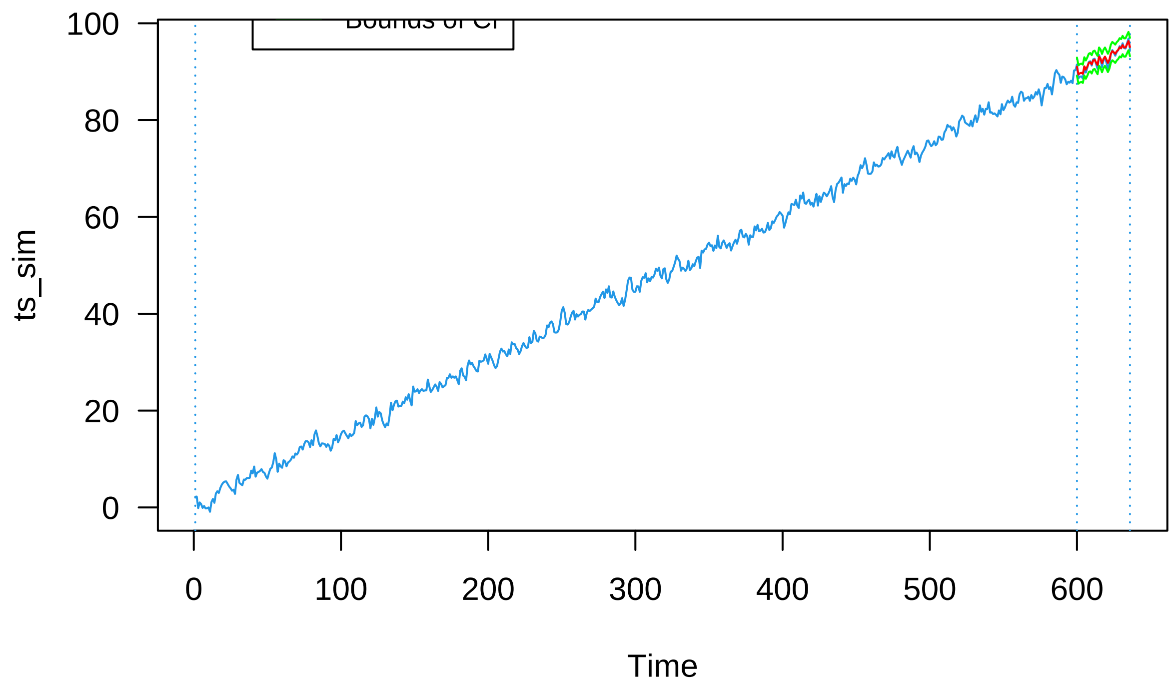
```
# since it is one step ahead predictin, so we need use for loop

# generate the residuals
rests <- ts_sim - predict(reg, newdata = data.frame(time = time(ts_sim)))

pred_df <- data.frame(NULL)
for(t in it){
  pred_onestep <- forecast(rests[1:t], h=1, level=0.95, model = fit_newts)
  pred_df <- rbind(pred_df, data.frame(mean = pred_onestep$mean[1], lower = pred_onestep$lower[1], uppe
}
```

add predicted trend back

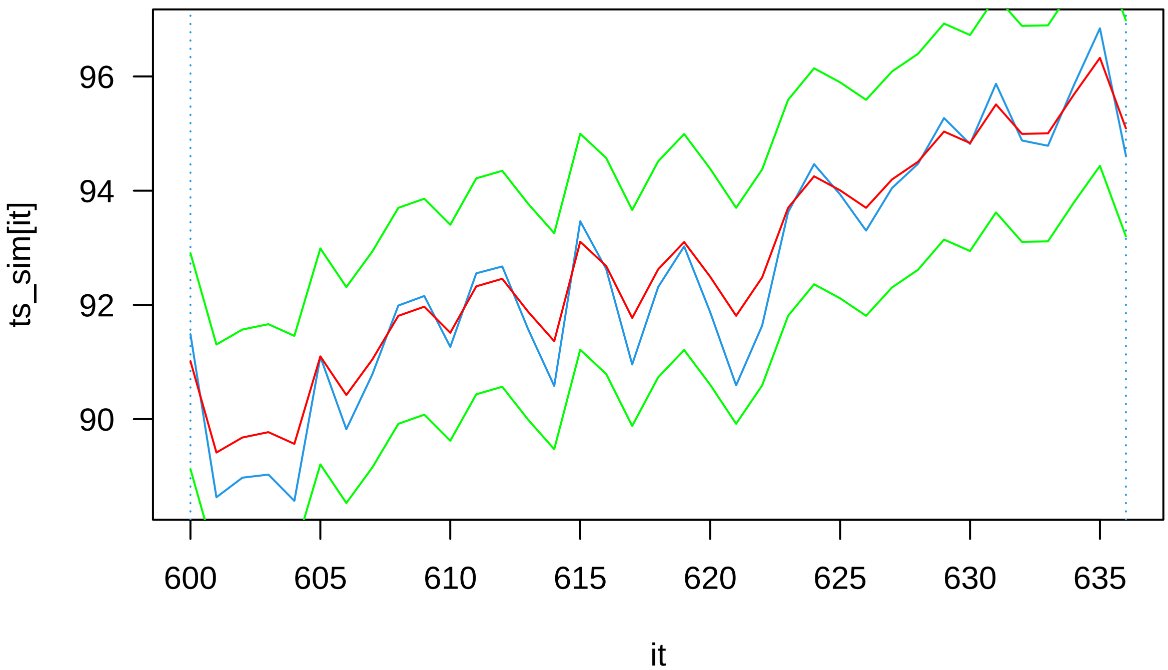
```
pred_df$mean <- pred_df$mean + predict(reg, newdata = data.frame(time = it))
pred_df$lower <- pred_df$lower + predict(reg, newdata = data.frame(time = it))
pred_df$upper <- pred_df$upper + predict(reg, newdata = data.frame(time = it))
```

plot pred, obs and CI

```
plot(ts_sim, col=4, las=1)
abline(v=c(1, 600, 636), lty="dotted", col=4)
lines(it, pred_df$mean, col = 'red')
lines(it, pred_df$lower, col = 'green')
lines(it, pred_df$upper, col = 'green')
legend(40, 120, legend=c("Observations", "Prediction", "Bounds of CI"),col=c("blue", "red", "green"),lty
```

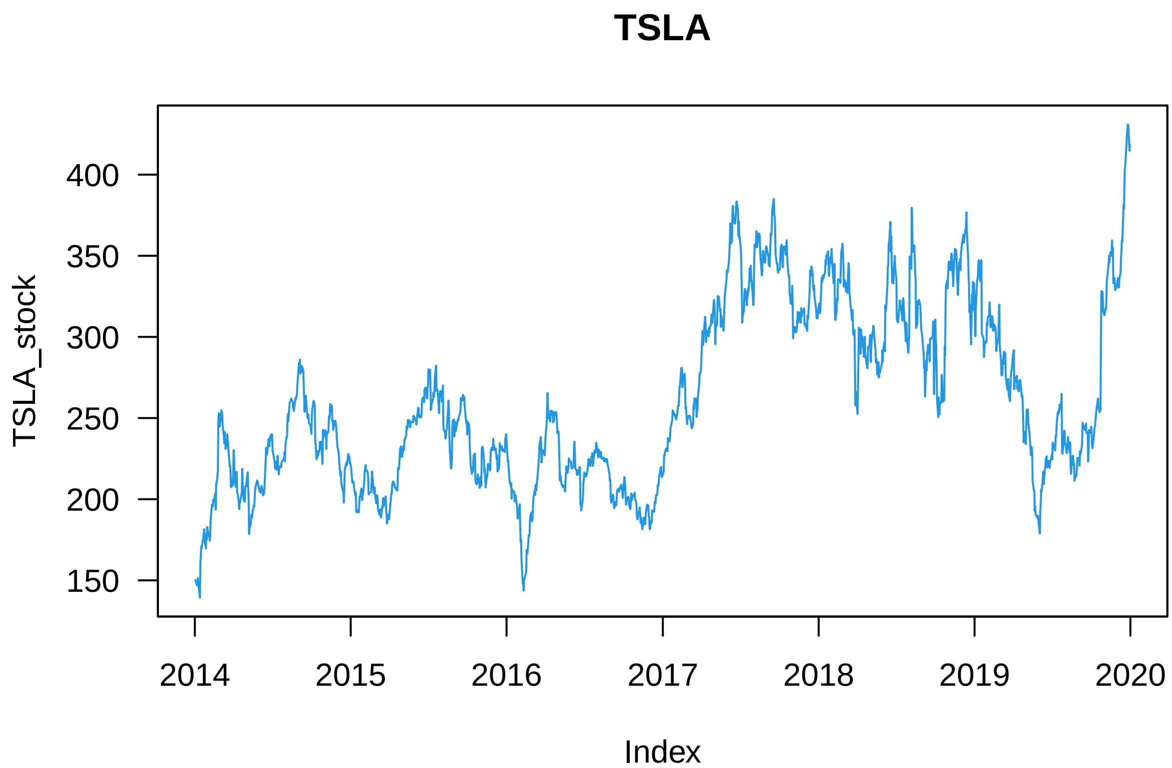



```
plot(it, ts_sim[it], col=4, las=1, type = 'l')
abline(v=c(600, 636), lty="dotted", col=4)
lines(it, pred_df$mean, col = 'red')
lines(it, pred_df$lower, col = 'green')
lines(it, pred_df$upper, col = 'green')
legend(left, 130, legend=c("Observations", "Prediction", "Bounds of CI"),col=c("blue", "red", "green"),l
```

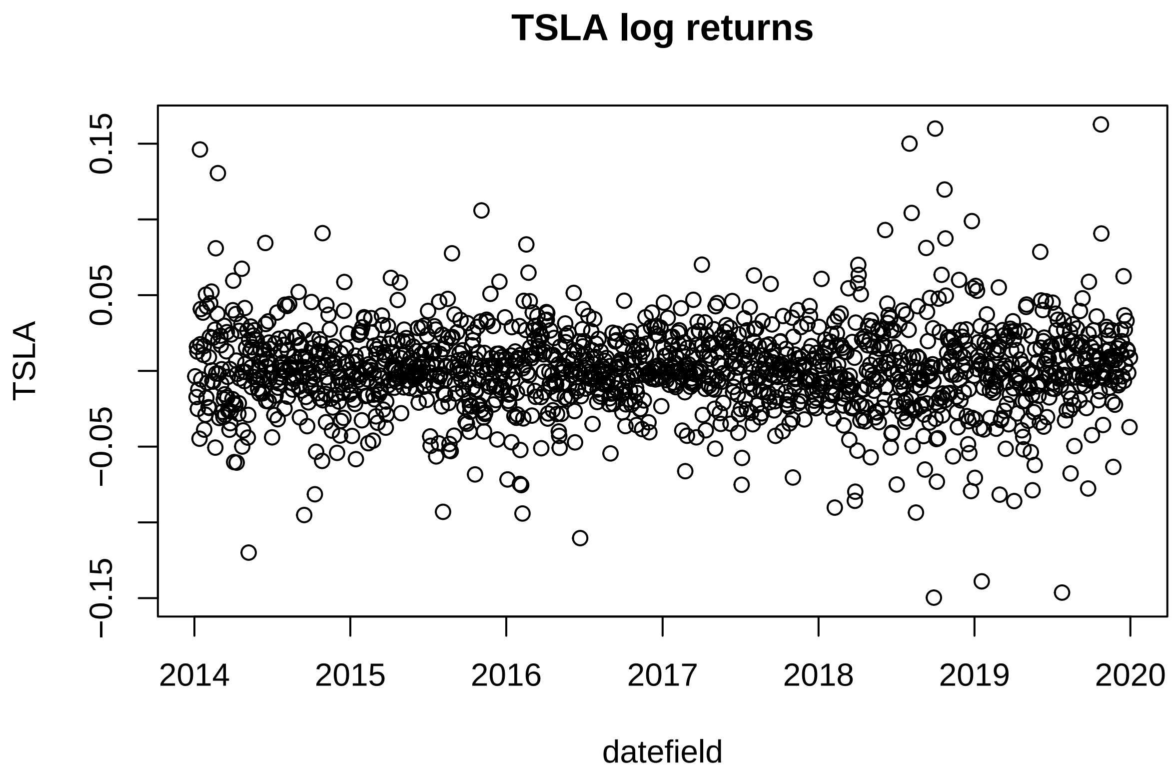


Part 7

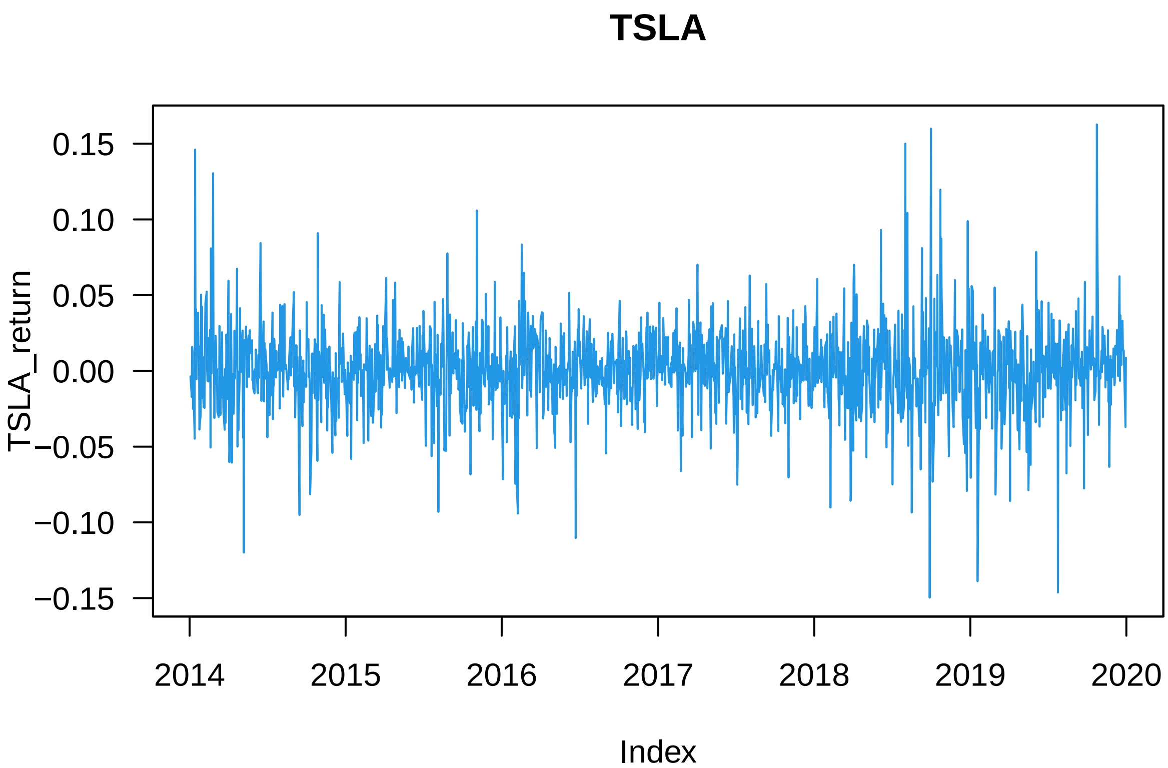
```
#(a)
data = read.csv('TSLA1.csv')

library(forecast)
library(zoo)
library(tseries)
TSLA = data$Close
time = as.Date(data$Date, format = '%m/%d/%y')
df = data.frame(datefield = time, TSLA = TSLA)
TSLA_stock = with(df, zoo(TSLA, order.by = time))
plot.zoo(TSLA_stock, col=4, las=1, main="TSLA")
```




```r
# Use the closing price to get log return
log_return = na.omit(diff(log(data$Close))) # log return
time = as.Date(data$Date, format = '%m/%d/%y')[-1]
df = data.frame(datefield = time, TSLA = log_return)
TSLA_return = with(df, zoo(TSLA, order.by = time))
plot(df, main = "TSLA log returns")
```




```
plot.zoo(TSLA_return, col=4, las=1, main="TSLA")
```




```r
# It seems there is no drift or a trend from the plot.
# We can use Augmented Dickey-Fuller Test Unit Root Test for more details:

library(urca)
```

```
## Warning: package 'urca' was built under R version 4.2.3
```

```r
summary(ur.df(log_return, type='trend', lags=20, selectlags="BIC"))
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.150974 -0.014073 -0.000256  0.015433  0.161771
##
## Coefficients:
```

```
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.372e-04  1.501e-03   0.225    0.822
## z.lag.1     -9.729e-01  3.661e-02 -26.572   <2e-16 ***
## tt           2.942e-07  1.712e-06   0.172    0.864
## z.diff.lag  -2.265e-02  2.596e-02  -0.873    0.383
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02836 on 1484 degrees of freedom
## Multiple R-squared:  0.4981, Adjusted R-squared:  0.4971
## F-statistic: 490.9 on 3 and 1484 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -26.5723 235.3642 353.0462
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.96 -3.41 -3.12
## phi2  6.09  4.68  4.03
## phi3  8.27  6.25  5.34
```

```
# no time trend, no drift

#(b)
tseries::adf.test(TSLA_return)
```

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

```
##
##  Augmented Dickey-Fuller Test
##
## data:  TSLA_return
## Dickey-Fuller = -11.651, Lag order = 11, p-value = 0.01
## alternative hypothesis: stationary
```

```
model <- auto.arima(TSLA_return, ic = 'bic',stationary = T, trace = T)
```

```
##
##  Fitting models using approximations to speed things up...
##
##  ARIMA(2,0,2) with non-zero mean : Inf
##  ARIMA(0,0,0) with non-zero mean : -6434.86
##  ARIMA(1,0,0) with non-zero mean : -6481.537
##  ARIMA(0,0,1) with non-zero mean : Inf
##  ARIMA(0,0,0) with zero mean     : -6441.325
##  ARIMA(2,0,0) with non-zero mean : -6511.177
##  ARIMA(3,0,0) with non-zero mean : -6420.078
##  ARIMA(2,0,1) with non-zero mean : Inf
##  ARIMA(1,0,1) with non-zero mean : Inf
##  ARIMA(3,0,1) with non-zero mean : Inf
##  ARIMA(2,0,0) with zero mean     : -6517.457
##  ARIMA(1,0,0) with zero mean     : -6488.731
```

```
##  ARIMA(3,0,0) with zero mean     : -6419.552
##  ARIMA(2,0,1) with zero mean     : Inf
##  ARIMA(1,0,1) with zero mean     : Inf
##  ARIMA(3,0,1) with zero mean     : Inf
##
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(2,0,0) with zero mean     : -6427.534
##
##  Best model: ARIMA(2,0,0) with zero mean
```

```
model # BIC = -6427.534 ARIMA(2,0,0)
```

```
## Series: TSLA_return
## ARIMA(2,0,0) with zero mean
##
## Coefficients:
##           ar1     ar2
##       -0.0110  0.0279
## s.e.   0.0287  0.0340
##
## sigma^2 = 0.0008161:  log likelihood = 3224.75
## AIC=-6443.49   AICc=-6443.48   BIC=-6426.42
```

```
# (c)
predict(model, newdata = data.frame(time = '2020-01-02'))
```

```
## $pred
## Time Series:
## Start = 18262
## End = 18262
## Frequency = 1
## [1] -0.001130221
##
## $se
## Time Series:
## Start = 18262
## End = 18262
## Frequency = 1
## [1] 0.02856823
```

```
# The closing price of Jan 2, 2020:
exp(log(TSLA[1510])+TSLA_return[1509])
```

```
## 2019-12-31
##   421.9917
```