HuYuDataInsight Zhaowei Cai

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)
```

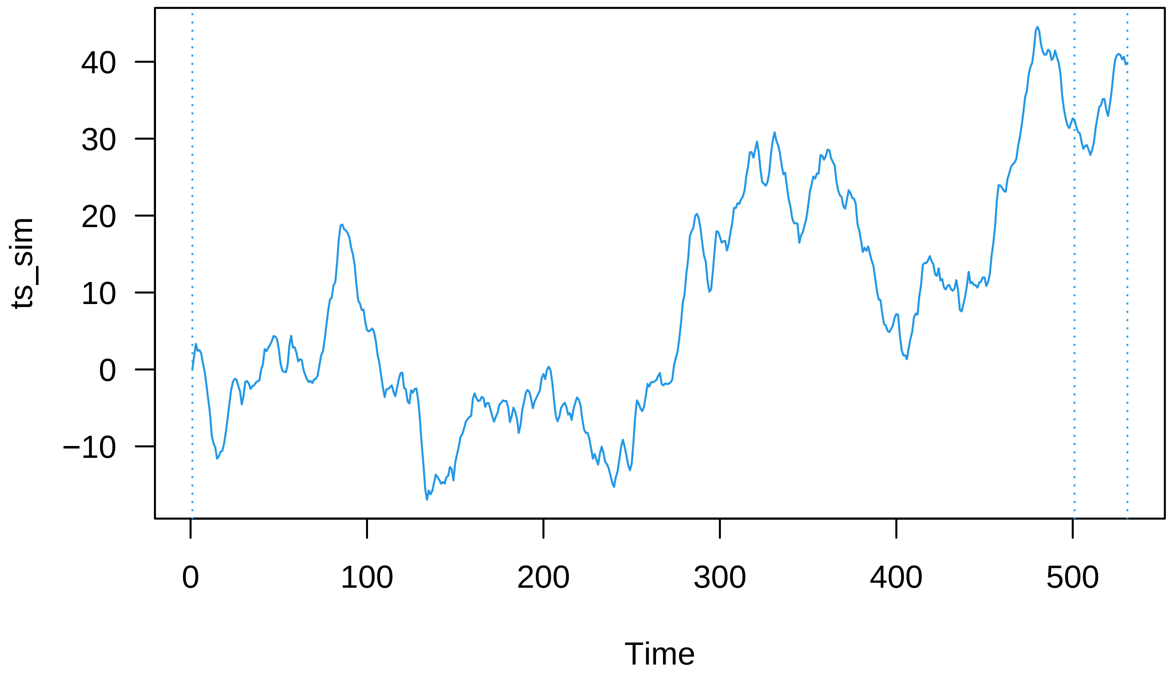
```
## Warning: package 'PerformanceAnalytics' was built under R version 4.2.3

## 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)
```

Question 1

```
set.seed(123)
#create a time series with 531 observations and first element is 0
ts_sim <- arima.sim(list(order = c(1,1,0), ar=0.65), n = 530)
```

```
plot(ts_sim, col=4, las=1)
abline(v=c(1, 501, 531), lty="dotted", col=4)
```



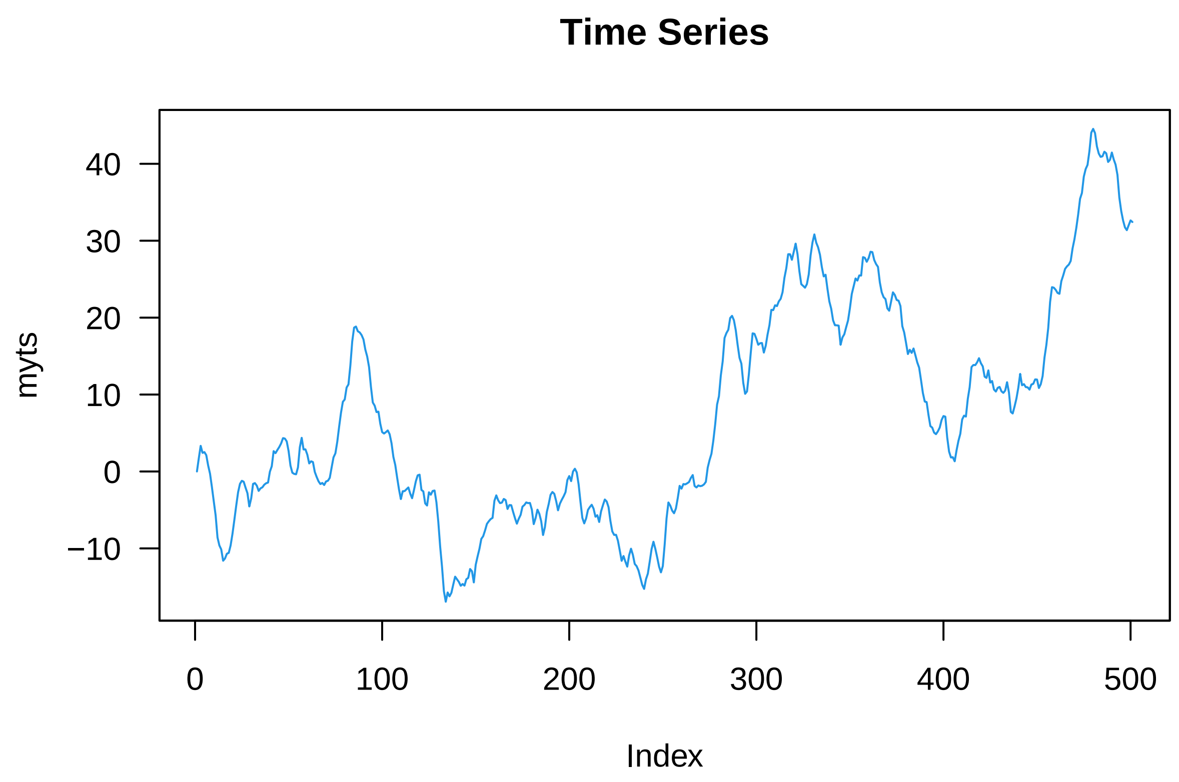
Question 2

Used 501 observations for myts because ts_sim[1] is always 0 and ts_sim has 531 observations

```
myts = subset(ts_sim, subset=rep(c(TRUE, FALSE), times=c(501, 30)))
```

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 = -2.4943, Lag order = 7, p-value = 0.3691
## 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')
```

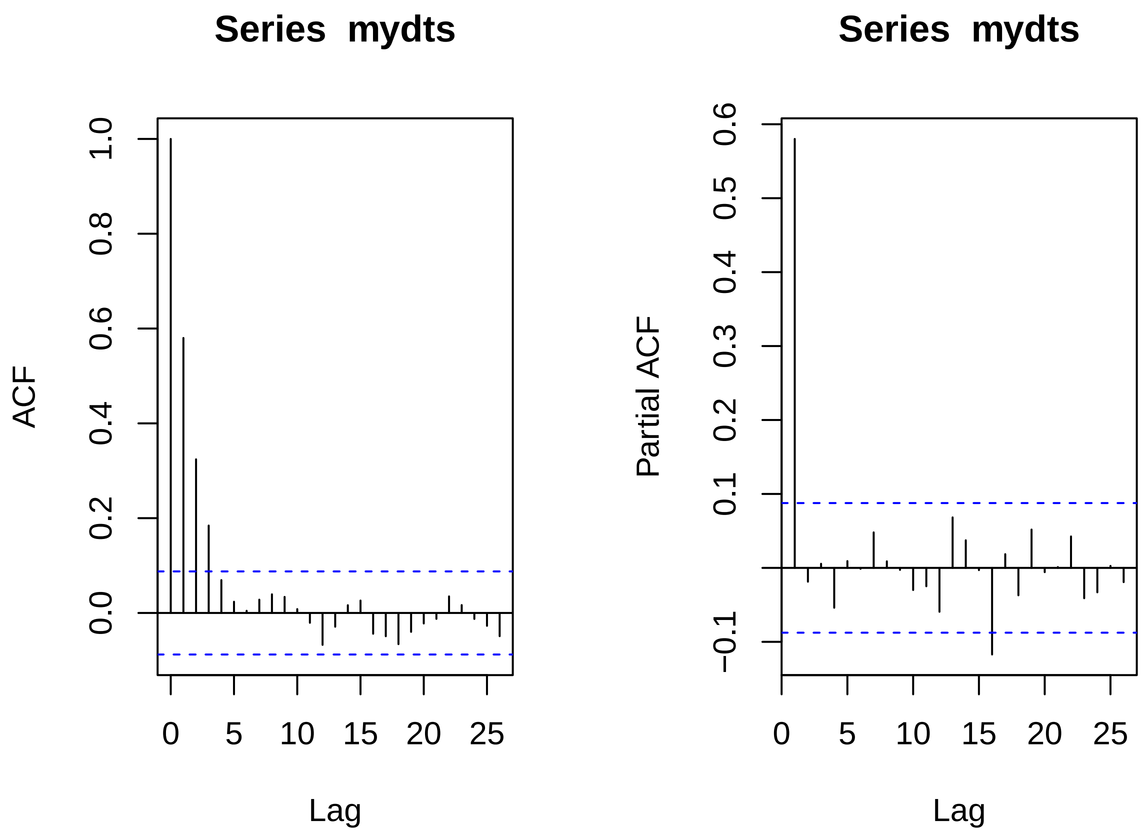
```
## Warning in adf.test(mydts, alternative = "stationary"): p-value smaller than
## printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  mydts
## Dickey-Fuller = -6.5529, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

P-value less than 0.05, not 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.  0.0363
##
## sigma^2 = 0.9408:  log likelihood = -693.93
## AIC=1391.86   AICc=1391.88   BIC=1400.29
##
## Training set error measures:
##                      ME      RMSE       MAE      MPE     MAPE      MASE
## Training set 0.02619055 0.9689955 0.7679863 129.6649 223.1909 0.8897537
##                     ACF1
## Training set 0.008341719
```

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.  0.0363
##
## sigma^2 estimated as 0.939:  log likelihood = -693.93,  aic = 1391.86
##
## Training set error measures:
##                      ME      RMSE       MAE       MPE     MAPE      MASE
## Training set 0.02613827 0.9680279 0.7664534 -51.01893 78.03226 0.8061068
##                     ACF1
## Training set 0.008265199
```

Or can directly fit the original time series

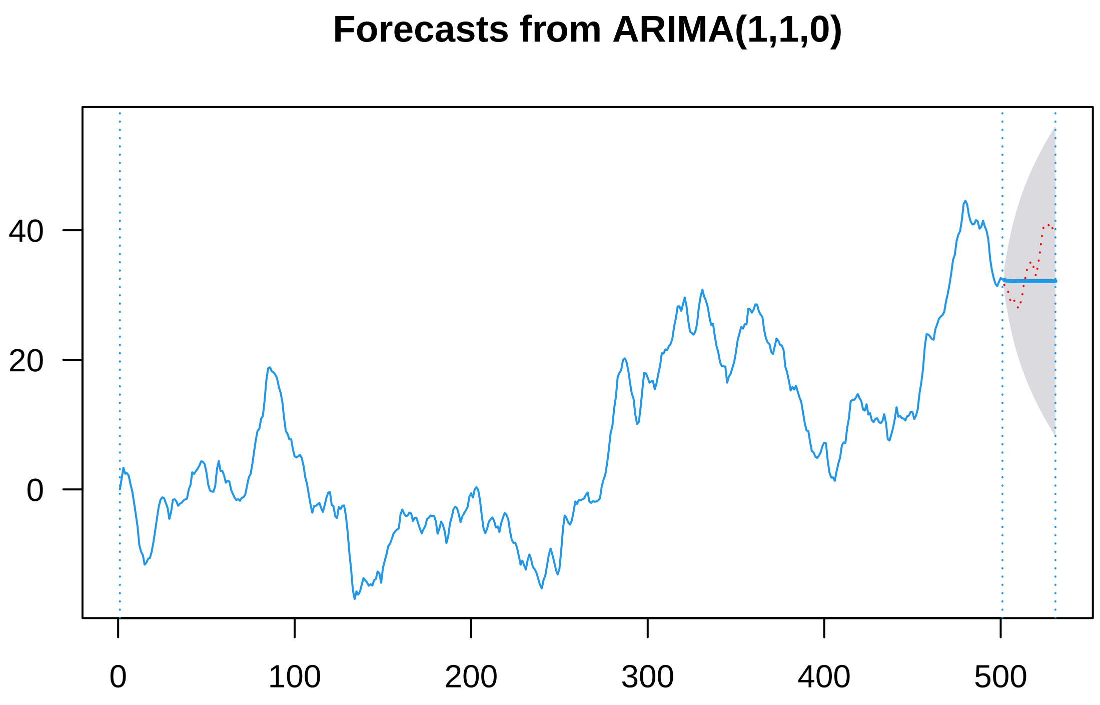
```
fit_myts2 = auto.arima(myts)
summary(fit_myts2)
```

```
## Series: myts
## ARIMA(1,1,0)
##
## Coefficients:
##          ar1
##       0.5825
## s.e.  0.0363
##
## sigma^2 = 0.9408:  log likelihood = -693.93
## AIC=1391.86   AICc=1391.88   BIC=1400.29
##
## Training set error measures:
##                      ME      RMSE       MAE       MPE     MAPE      MASE
## Training set 0.02613827 0.9680279 0.7664534 -51.01893 78.03226 0.8061068
##                     ACF1
## Training set 0.008265199
```

Question 3

Question 3(a)

```
forecast_myts = forecast(fit_myts, h=30, level=0.95)
plot(forecast_myts, col=4, las=1)
abline(v=c(1, 501, 531), lty="dotted", col=4)
lines(502:531, ts_sim[502:531], lty="dotted", col="red")
```

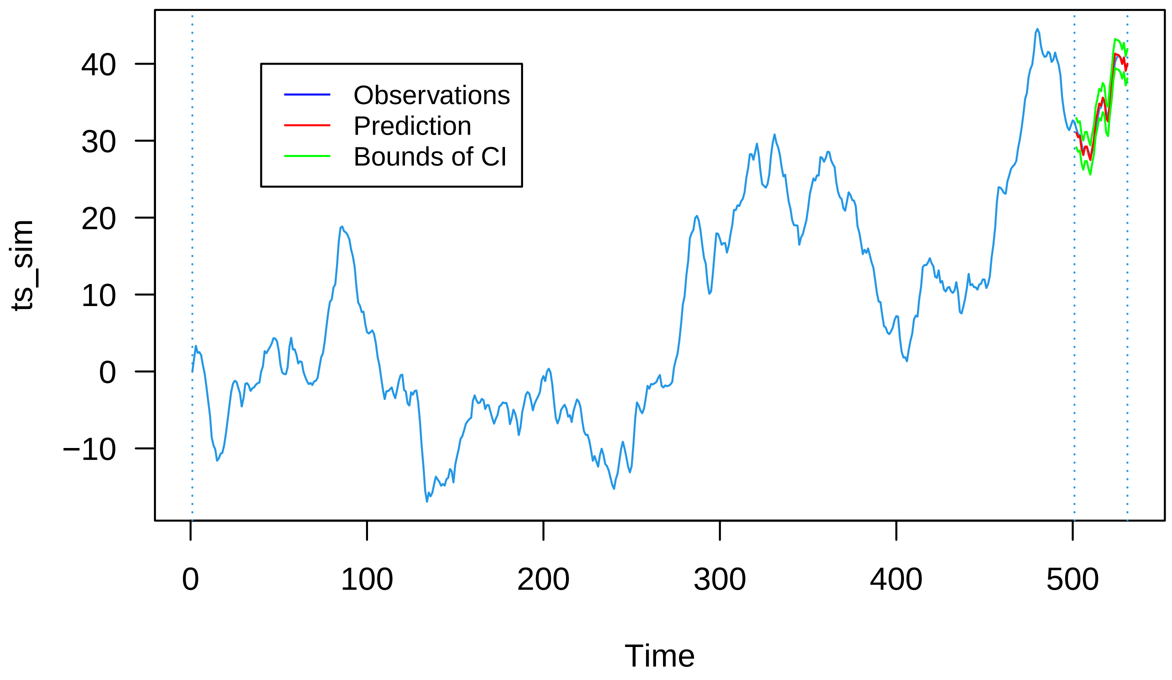


```
# red is observation and blue is prediction
```

Question 3(b)

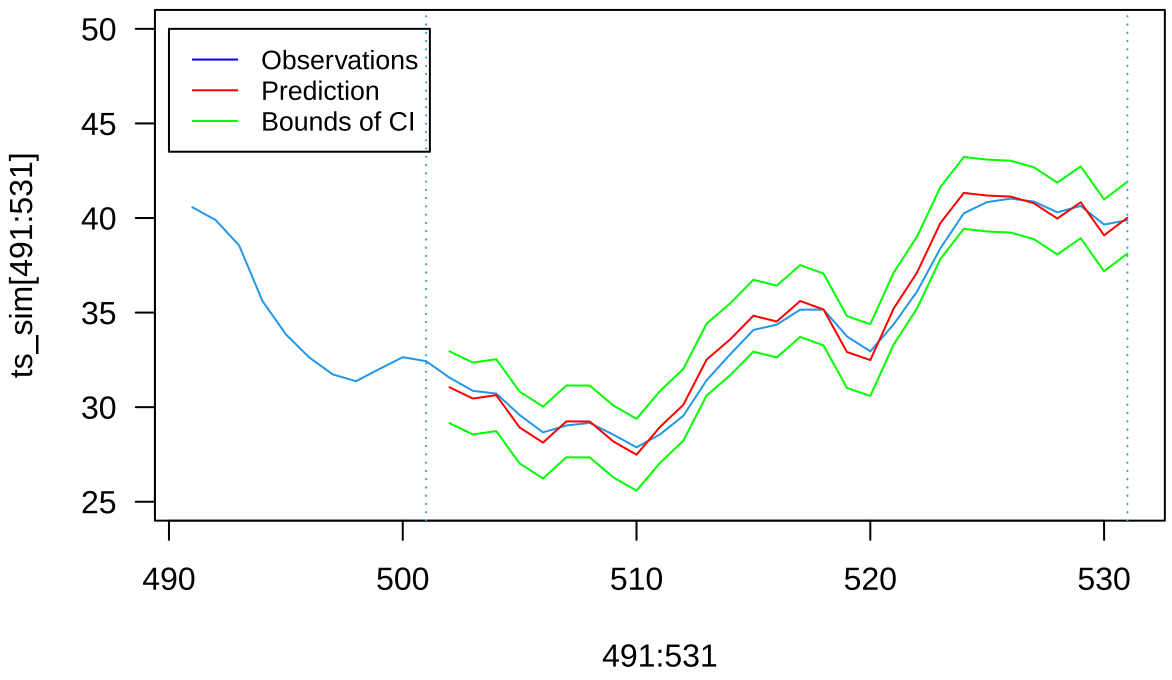
```
# since it is one step ahead prediction, so we need use for loop
pred_df <- data.frame(NULL)
for(t in 502:531){
  pred_onestep <- forecast(ts_sim[1:t], h=1, level=0.95, model = fit_myts)
  pred_df <- rbind(pred_df, data.frame(mean = pred_onestep$mean[1], lower = pred_onestep$lower[1], uppe
}
```

```
plot(ts_sim, col=4, las=1)
abline(v=c(1, 501, 531), lty="dotted", col=4)
lines(502:531, pred_df$mean, col = 'red')
lines(502:531, pred_df$lower, col = 'green')
lines(502:531, pred_df$upper, col = 'green')
legend(40, 40, legend=c("Observations", "Prediction", "Bounds of CI"),col=c("blue", "red", "green"),lty
```



make the plot more clear

```
plot(491:531, ts_sim[491:531], col=4, las=1, type = 'l', ylim = c(25,50))
abline(v=c(501, 531), lty="dotted", col=4)
lines(502:531, pred_df$mean, col = 'red')
lines(502:531, pred_df$lower, col = 'green')
lines(502:531, pred_df$upper, col = 'green')
legend(490, 50, legend=c("Observations", "Prediction", "Bounds of CI"),col=c("blue", "red", "green"),lt
```



Question 4

Generate AR(1) model data

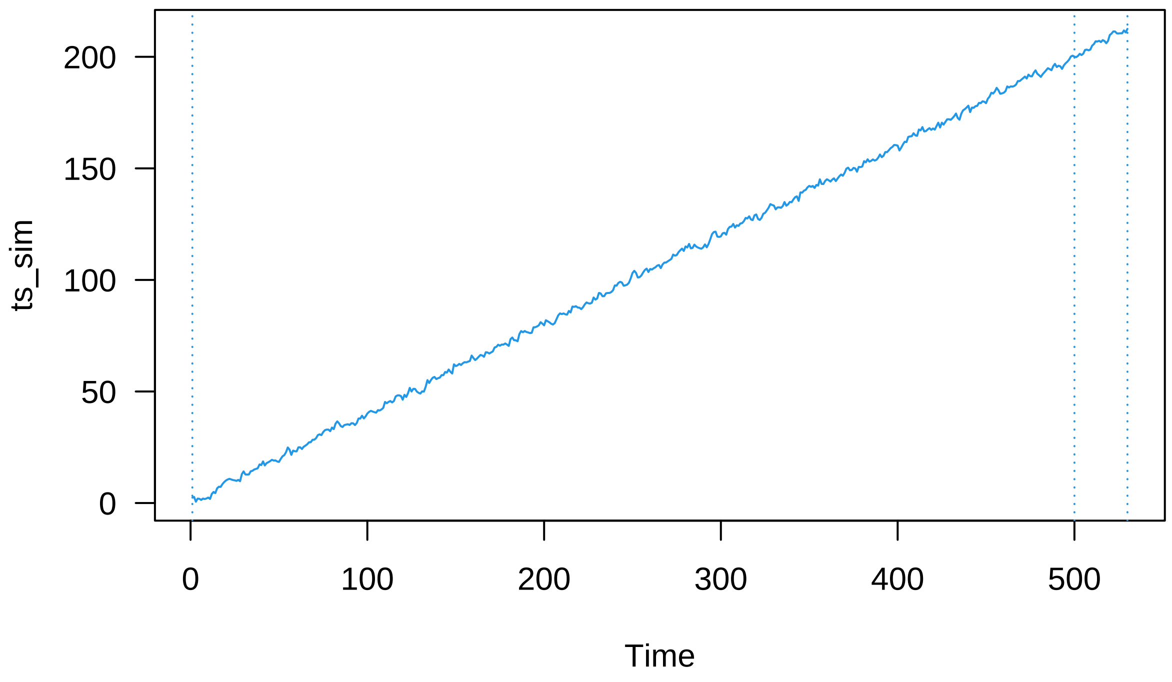
```r
set.seed(123)
ts_sim = arima.sim(list(ar=0.65),n=530)
```

add trend in the data

```r
ts_sim=ts_sim + 0.3 + 0.4*time(ts_sim)
```

Generate plots

```r
plot(ts_sim, col=4, las=1)
abline(v=c(1, 500, 530), lty="dotted", col=4)
```

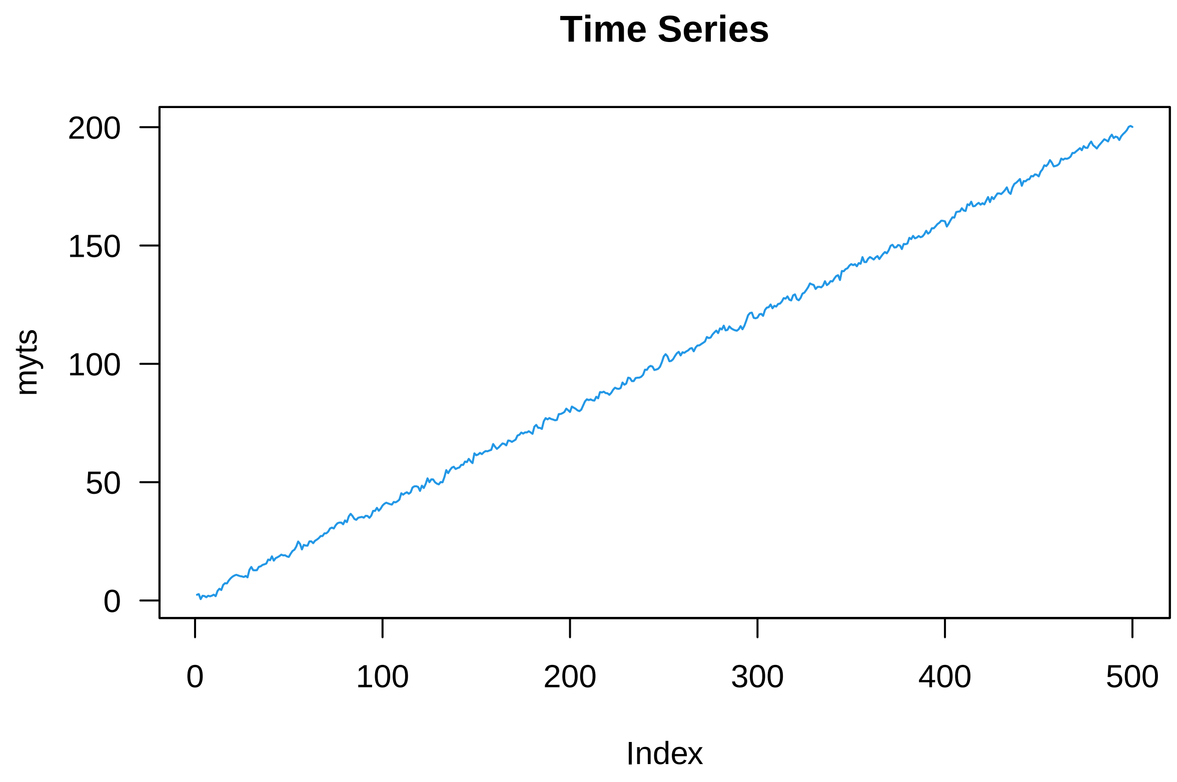


Question 5

```r
myts = subset(ts_sim, subset=rep(c(TRUE, FALSE), times=c(500, 30)))
```

Step 1: visualize myts

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

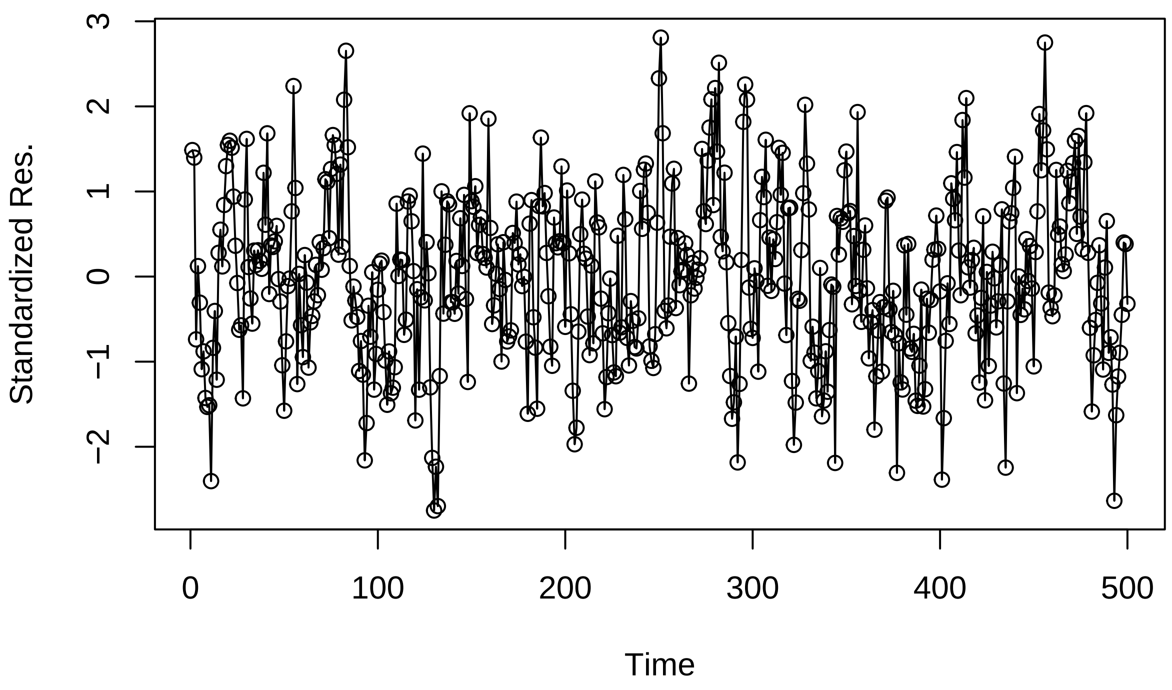


Fit trend part

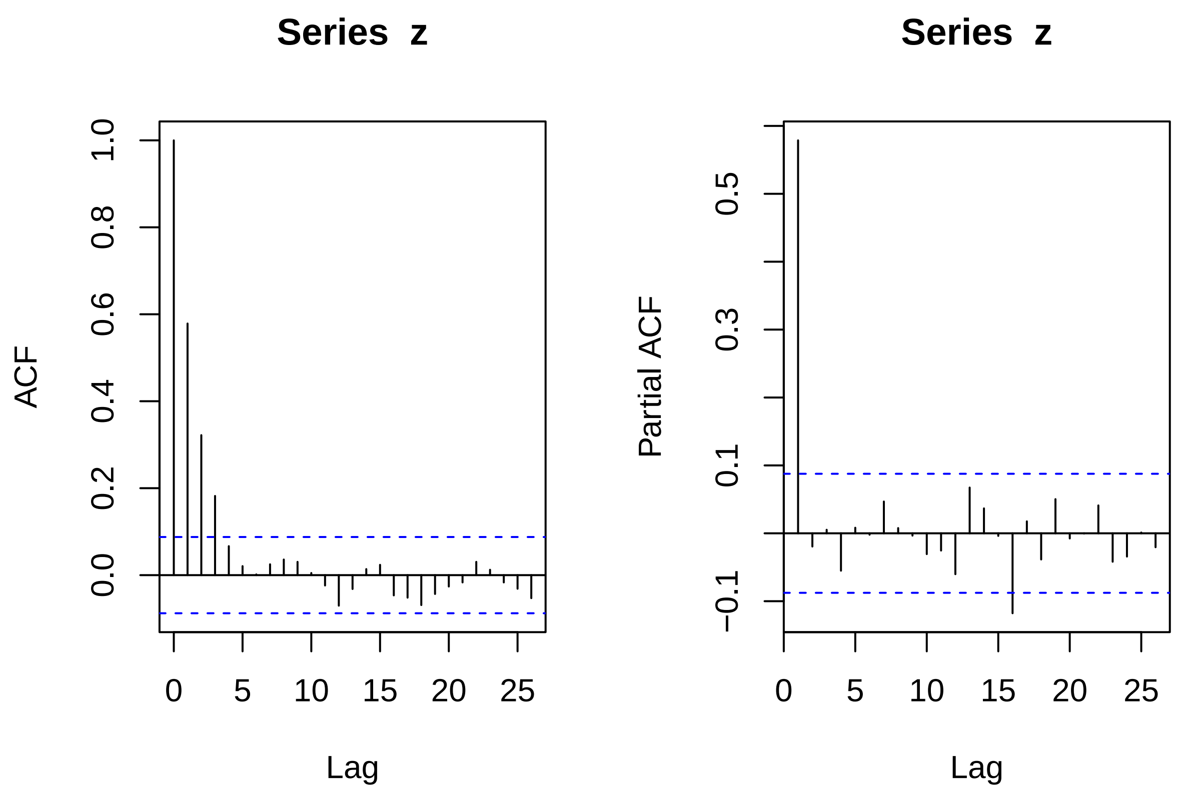
```
time = time(myts)
reg=lm(myts~time)
summary(reg)
```

```
##
## Call:
## lm(formula = myts ~ time)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2478 -0.8020 -0.0011  0.7870  3.3185
##
## Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept) 0.2579884  0.1066960    2.418    0.016 *
## time        0.4004266  0.0003691 1085.013   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.191 on 498 degrees of freedom
## Multiple R-squared:  0.9996, Adjusted R-squared:  0.9996
## F-statistic: 1.177e+06 on 1 and 498 DF,  p-value: < 2.2e-16
```

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



```
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)
```

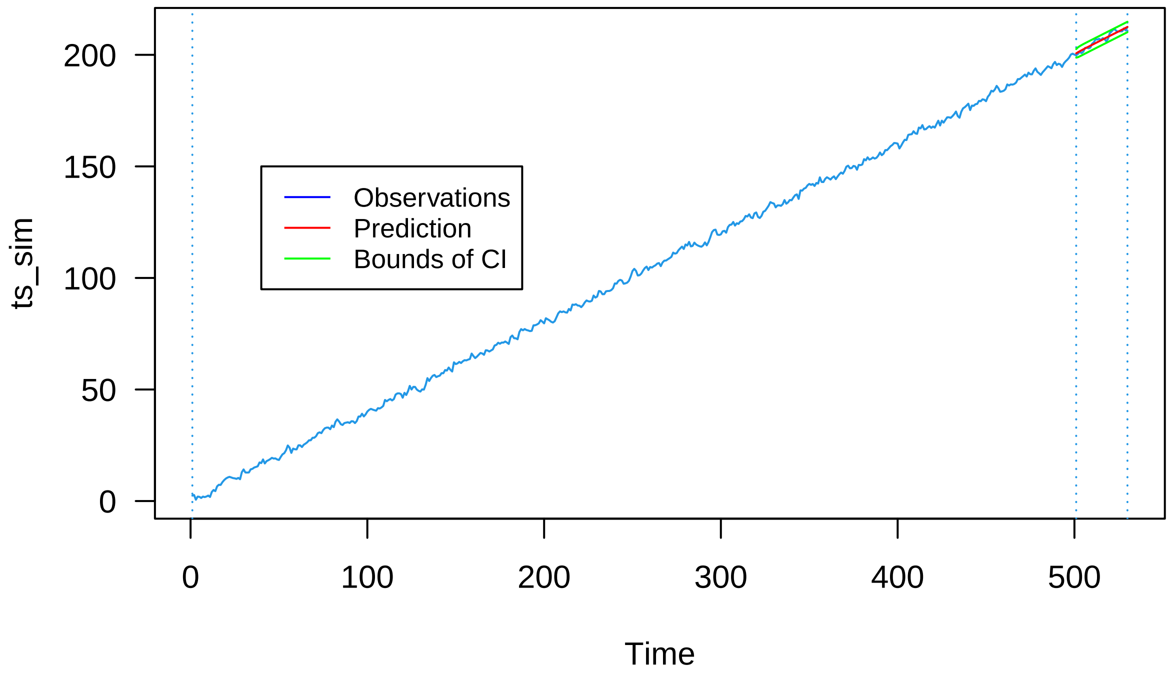
```
## Series: newts
## ARIMA(1,0,0) with zero mean
##
## Coefficients:
##          ar1
##       0.5802
## s.e.  0.0364
##
## sigma^2 = 0.9396:  log likelihood = -693.59
## AIC=1391.18   AICc=1391.2   BIC=1399.6
##
## Training set error measures:
##                        ME      RMSE       MAE      MPE     MAPE      MASE
## Training set -0.001095898 0.9683386 0.7669195 166.1226 256.9947 0.8885151
##                       ACF1
## Training set 0.009733031
```

Question 6

Question 6(a)

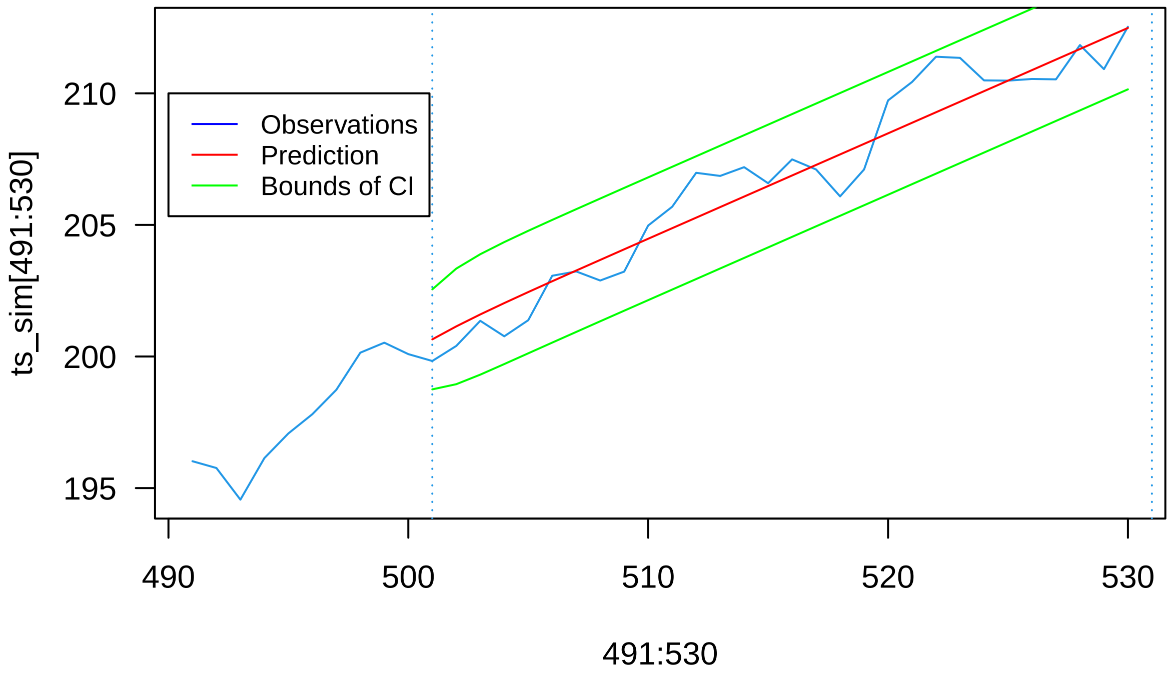
```
prediction <- forecast(fit_newts, h=30, level=0.95)
pred_df <- data.frame(time = 501:530)
pred_df$mean <- prediction$mean + predict(reg, newdata = data.frame(time = 501:530))
pred_df$lower <- prediction$lower + predict(reg, newdata = data.frame(time = 501:530))
pred_df$upper <- prediction$upper + predict(reg, newdata = data.frame(time = 501:530))
```

```
plot(ts_sim, col=4, las=1)
abline(v=c(1, 501, 530), lty="dotted", col=4)
lines(501:530, pred_df$mean, col = 'red')
lines(501:530, pred_df$lower, col = 'green')
lines(501:530, pred_df$upper, col = 'green')
legend(40, 150, legend=c("Observations", "Prediction", "Bounds of CI"),col=c("blue", "red", "green"),lt
```



```
plot(491:530, ts_sim[491:530], col=4, las=1, type = 'l')
abline(v=c(501, 531), lty="dotted", col=4)
```

```
lines(501:530, pred_df$mean, col = 'red')
lines(501:530, pred_df$lower, col = 'green')
lines(501:530, pred_df$upper, col = 'green')
legend(490, 210, legend=c("Observations", "Prediction", "Bounds of CI"),col=c("blue", "red", "green"),l
```



## Question 6(b) Predict the residuls part

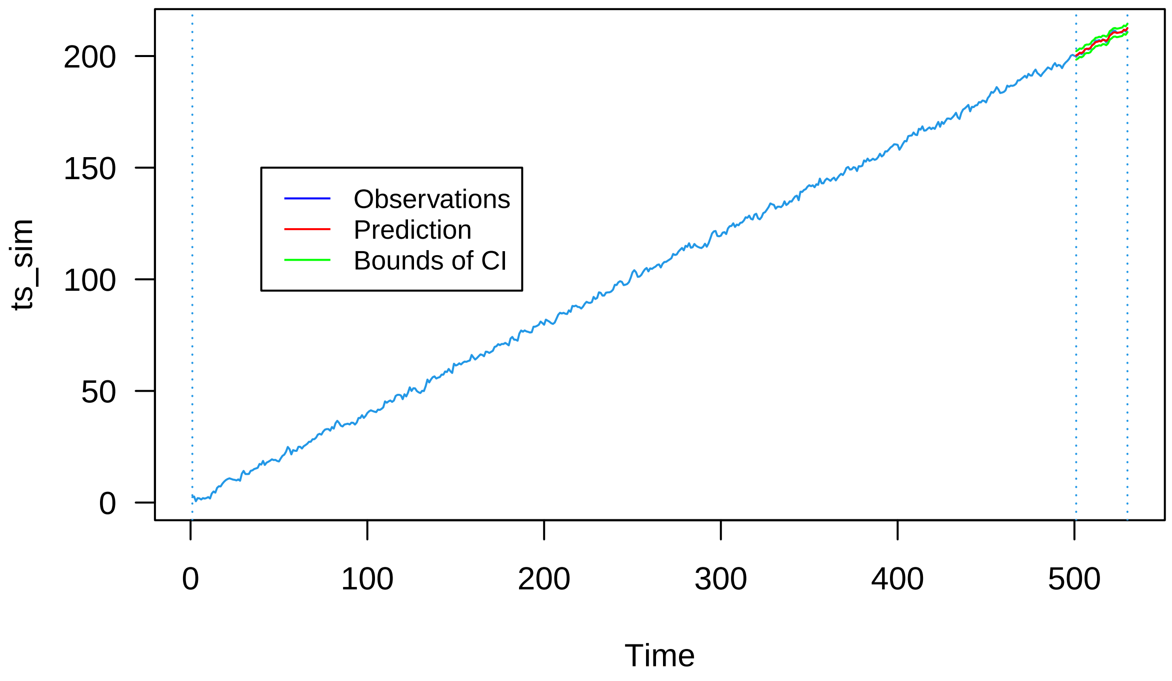
```
# since it is one step ahead prediction, we will need to use the for loop

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

pred_df <- data.frame(NULL)
for(t in 501:530){
  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

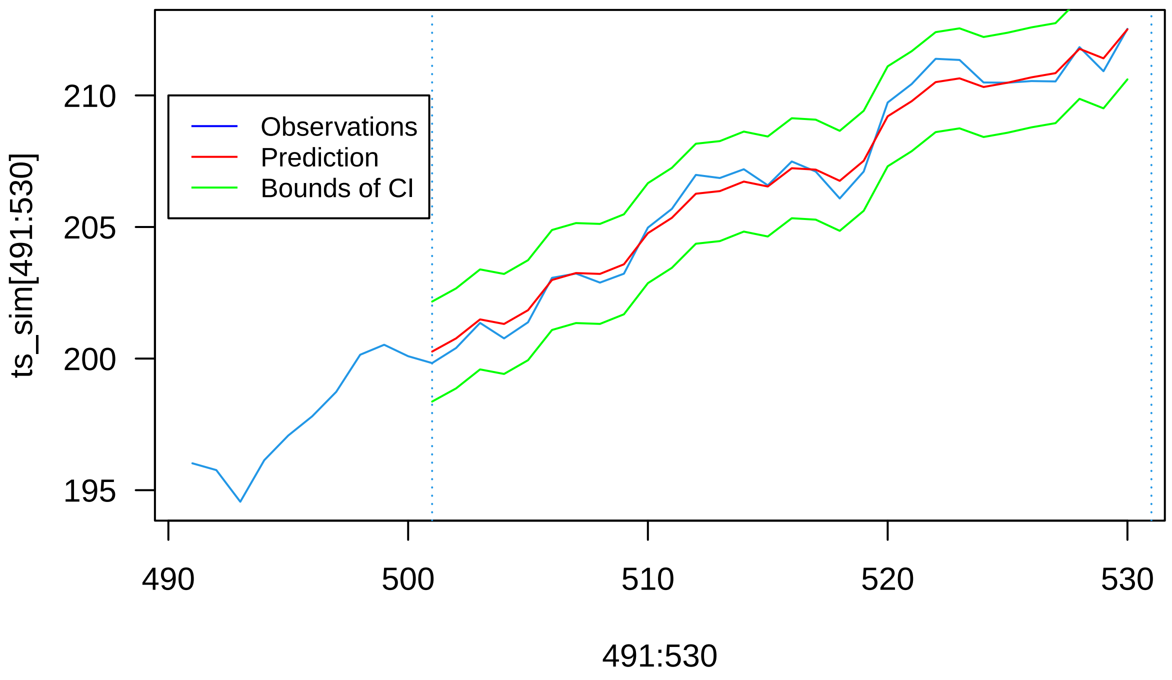
```
pred_df$mean <- pred_df$mean + predict(reg, newdata = data.frame(time = 501:530))
pred_df$lower <- pred_df$lower + predict(reg, newdata = data.frame(time = 501:530))
pred_df$upper <- pred_df$upper + predict(reg, newdata = data.frame(time = 501:530))
```

plot pred, obs and CI

```
plot(ts_sim, col=4, las=1)
abline(v=c(1, 501, 530), lty="dotted", col=4)
lines(501:530, pred_df$mean, col = 'red')
lines(501:530, pred_df$lower, col = 'green')
lines(501:530, pred_df$upper, col = 'green')
legend(40, 150, legend=c("Observations", "Prediction", "Bounds of CI"),col=c("blue", "red", "green"),lty
```



```
plot(491:530, ts_sim[491:530], col=4, las=1, type = 'l')
abline(v=c(501, 531), lty="dotted", col=4)
lines(501:530, pred_df$mean, col = 'red')
lines(501:530, pred_df$lower, col = 'green')
lines(501:530, pred_df$upper, col = 'green')
legend(490, 210, legend=c("Observations", "Prediction", "Bounds of CI"),col=c("blue", "red", "green"),l
```



210
205
200
195
490
500
510
520
530
491:530
ts_sim[491:530]
Observations
Prediction
Bounds of CI