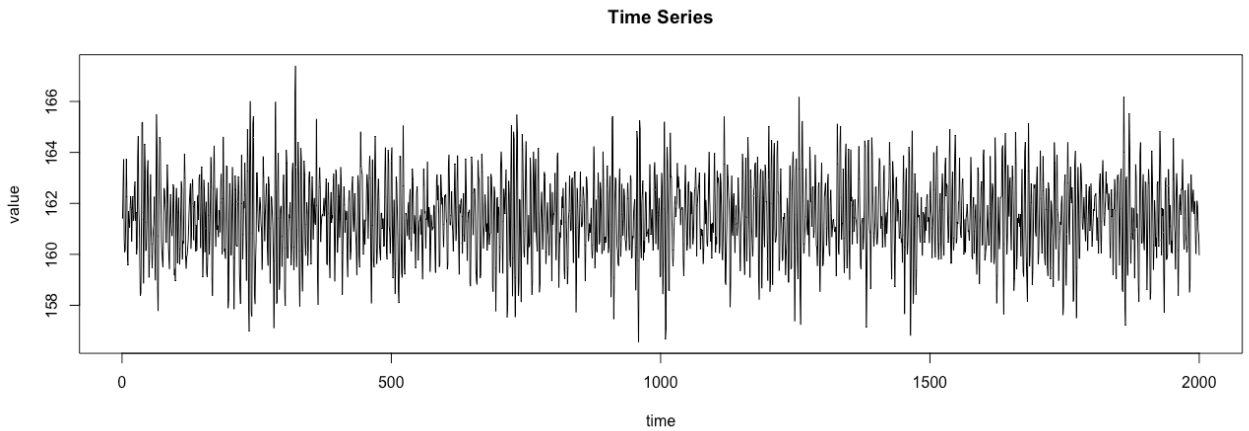
Task 1

The plot of time series are shown as follow.



Visually, we draw preliminary conclusion that it is stationary. We can also use R function `adf.test` to test it. p-value < 0.05 indicates the time series is stationary. We test at various lags and the result is shown below, p -value is 0.01, we can draw the time series is stationary. So we don't need to do data transformation here.

```
> adf.test(myts)

        Augmented Dickey-Fuller Test

data:  myts
Dickey-Fuller = -13.651, Lag order = 12, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(myts) : p-value smaller than printed p-value
> adf.test(myts,k =50)

        Augmented Dickey-Fuller Test

data:  myts
Dickey-Fuller = -6.7308, Lag order = 50, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(myts, k = 50) : p-value smaller than printed p-value
> adf.test(myts,k =100)

        Augmented Dickey-Fuller Test

data:  myts
Dickey-Fuller = -4.7308, Lag order = 100, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(myts, k = 100) : p-value smaller than printed p-value
>
```

Generally, if the time series have non-stationary feature. We can use Differencing, Transformations, Seasonal differencing to remove the non-stationary feature. There are R package to do that. For example, the `nsdiffs` and `ndiffs` from `forecast` package can help to find out how many seasonal differencing and regular differencing respectively, which are needed to make the series stationary.

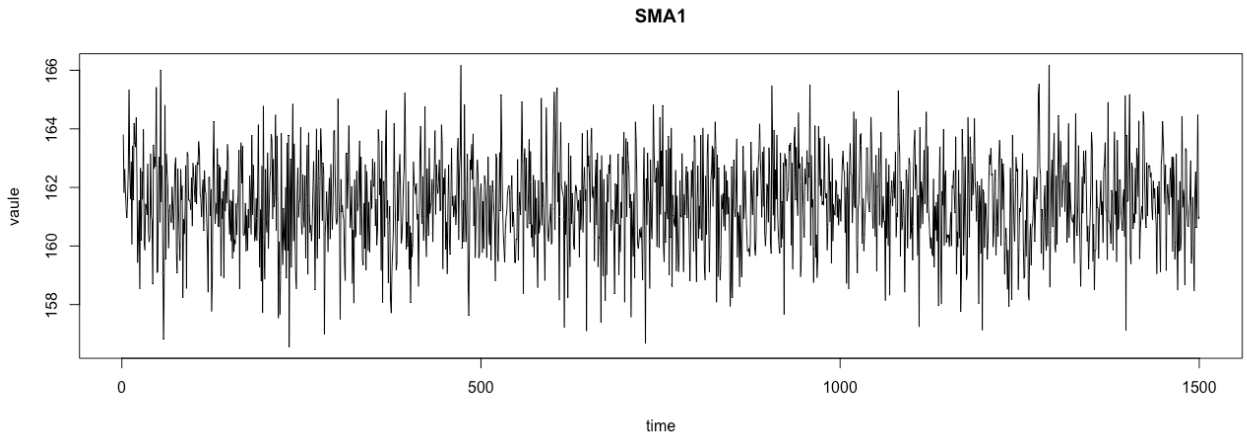
Task 2

2.1

Firstly, we partition the train data and test data. We selected the data from 1 to 1500 be the train data, and data from 1501 to 2000 be the test data. As shown below

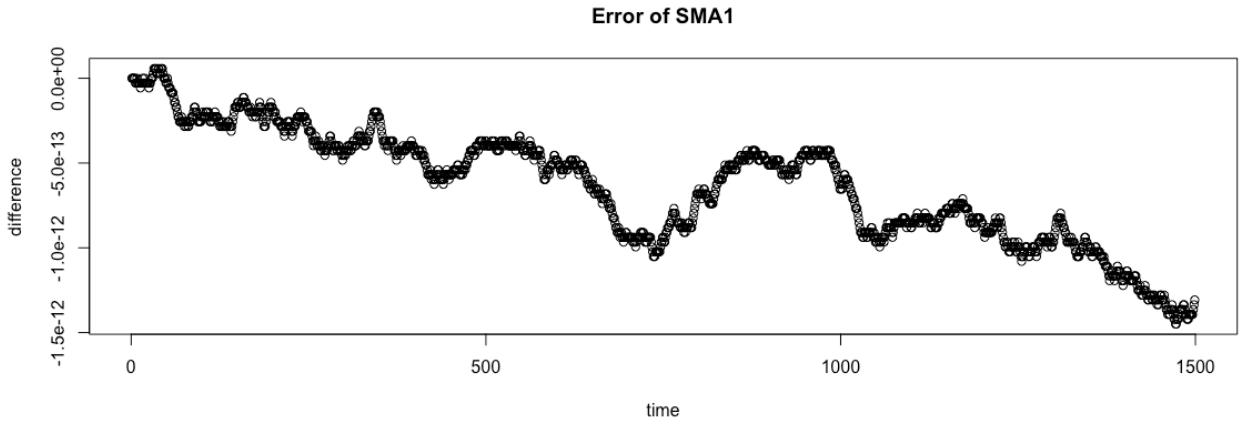
```
myts.train = subset(myts,end = length(myts) - 500)
myts.test =  subset(myts,start = length(myts) - 499)
```

Then we use `SMA` to do simple moving average model on the train data. We select k = 1 here and plot the result as follows.



2.2

Assume preditted values and actual values are stored in v1 and v2 perspectively, calculate the error by v2 -v1 to get the error series, and can plot them, as shown below.



we define a function to calculate the RMSE and put the error series into it to get the RMSE = 7.157443e-13

```
rmse <- function(error)
{
  sqrt(mean(error^2))
}
```

2.3

Here, we abstract a function doSMA to do model and return RMSE as follows

```
doSma <- function(train,k){
  library(TTR)
  sma = SMA(train,n=k)
  plot(sma,main=paste("SMA",k,sep=""),xlab ="time",ylab = "vaule")

  cActrual = c(train[(k+1):length(train)])
  cFitted =  c(sma[(k+1):length(sma)])
  error = cActrual - cFitted
  # plot(error,main=paste("Error of SMA",k,sep=""),xlab = "time",ylab =
"difference")
  return (rmse(error))
}
```

So we can call it in a loop varies k from 1 to 50 and save the RMSEs in a vector.

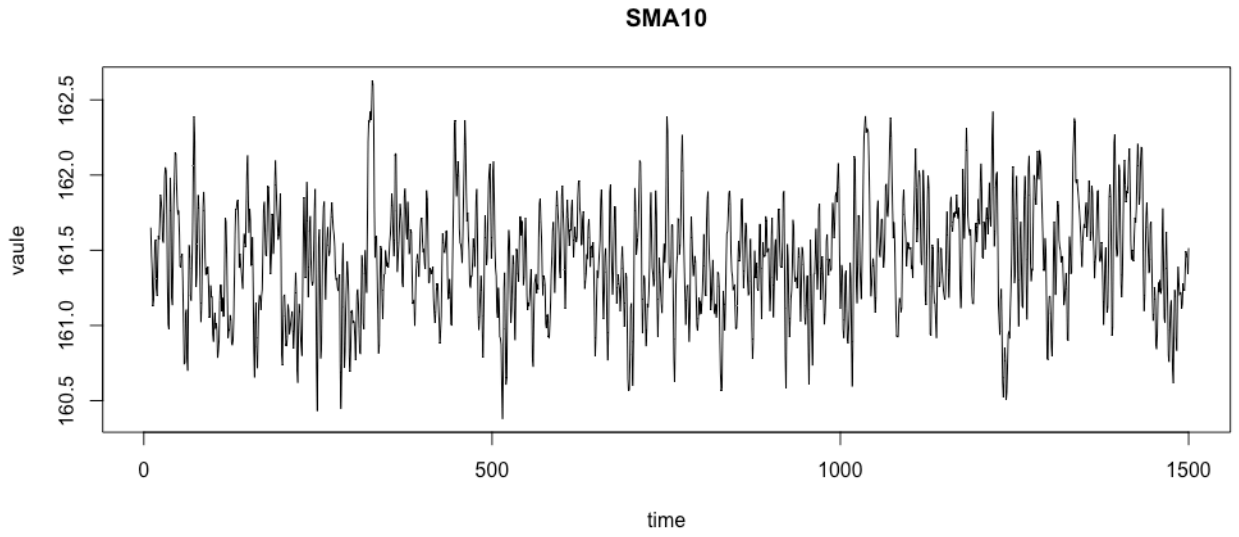
```
 rmseResults = c()
for(i in 1 : 50){
  rmseResults = c(rmseResults,doSma(myts.train,i))
}
print(rmseResults)
```

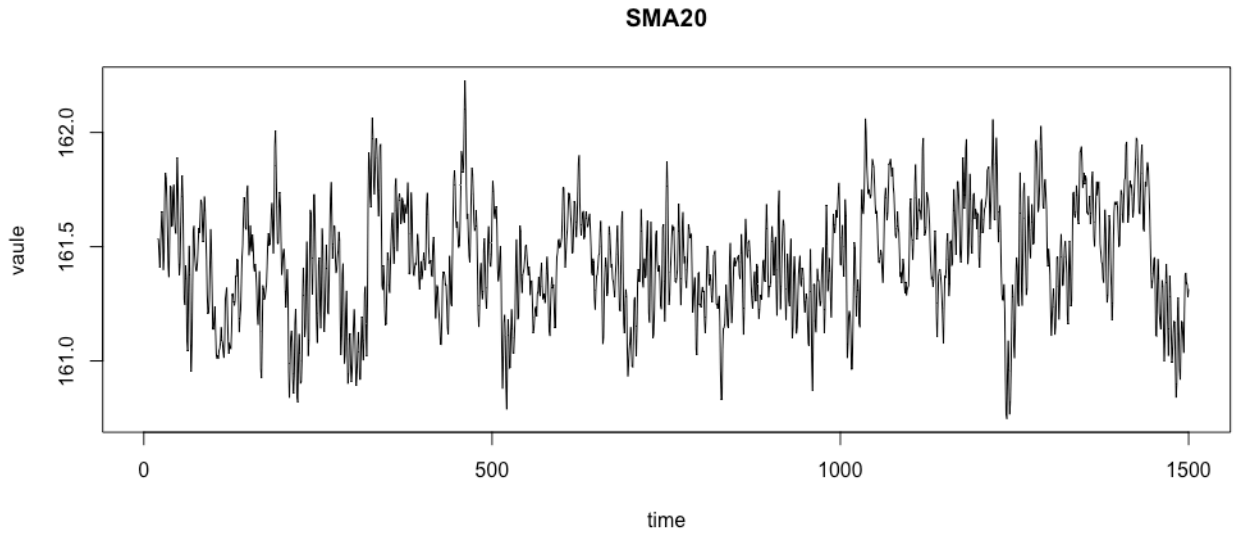
and we can get RMSEs when k varies from 1 to 50 as follows.

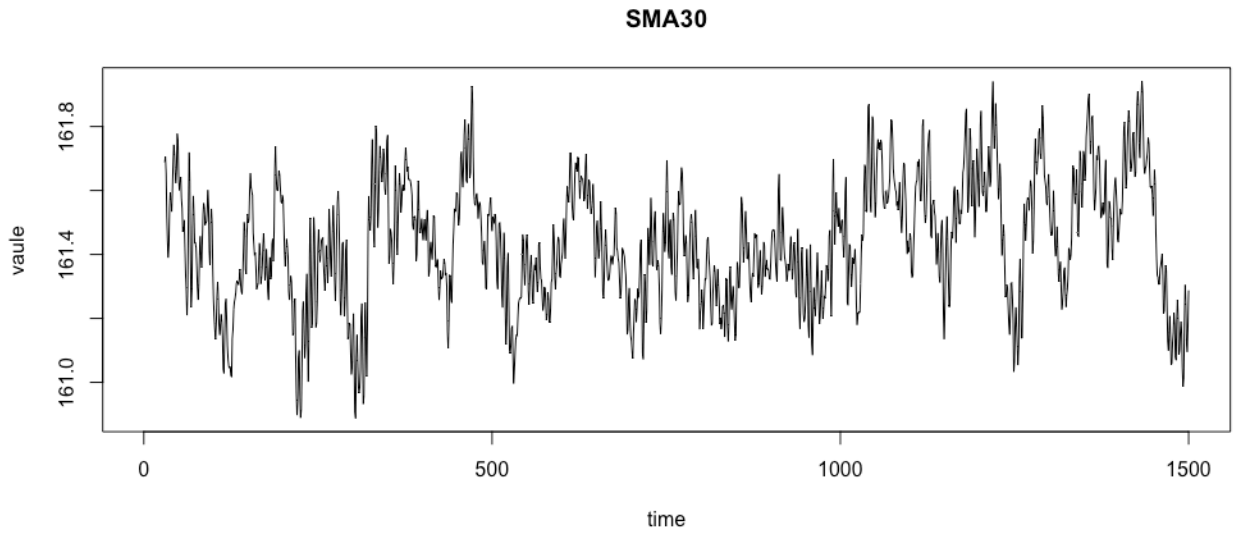
```
> source('~/Google Drive/Courses/CSC591-004/hw/ForecastingProject/forecast.R')
 [1] 4.798373e-13 8.663555e-01 1.390068e+00 1.604625e+00 1.586990e+00 1.480034e+00 1.426062e+00 1.458789e+00 1.515866e+00
[10] 1.539859e+00 1.529473e+00 1.511650e+00 1.511337e+00 1.528706e+00 1.547080e+00 1.550164e+00 1.539058e+00 1.528909e+00
[19] 1.533048e+00 1.547871e+00 1.558666e+00 1.557230e+00 1.549238e+00 1.546682e+00 1.552456e+00 1.559573e+00 1.561780e+00
[28] 1.559821e+00 1.558297e+00 1.558949e+00 1.562075e+00 1.563852e+00 1.563752e+00 1.562044e+00 1.559963e+00 1.559694e+00
[37] 1.558425e+00 1.559801e+00 1.562884e+00 1.560333e+00 1.557006e+00 1.554507e+00 1.557563e+00 1.562057e+00 1.563531e+00
[46] 1.561278e+00 1.557755e+00 1.557511e+00 1.562023e+00 1.564453e+00
Warning message:
```

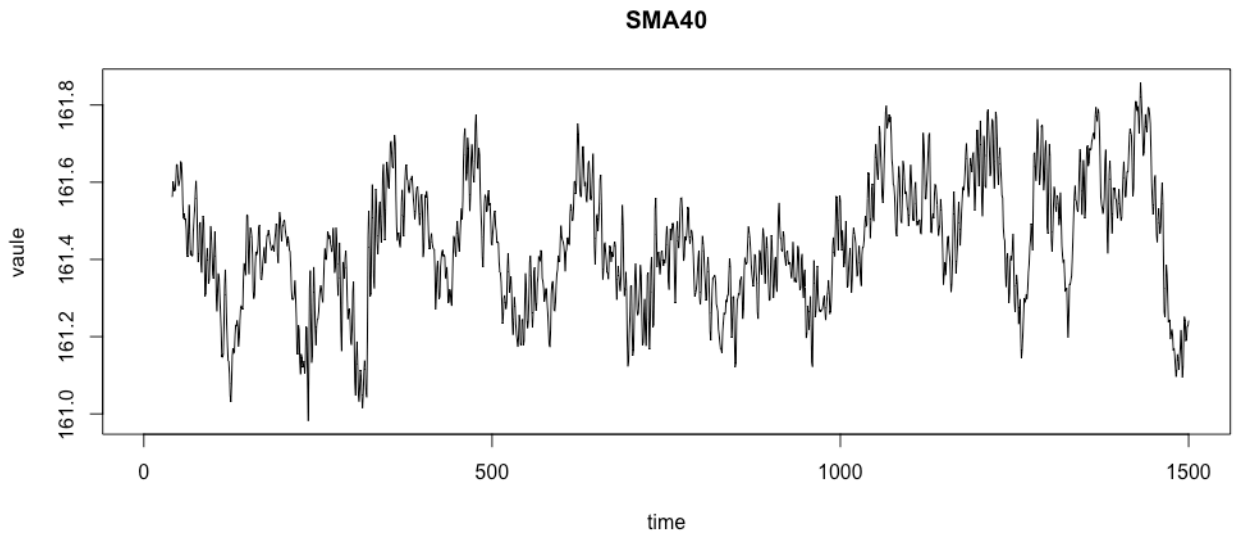
From the above results. we know that RMSE increases as k increases, particularly at the beginning.

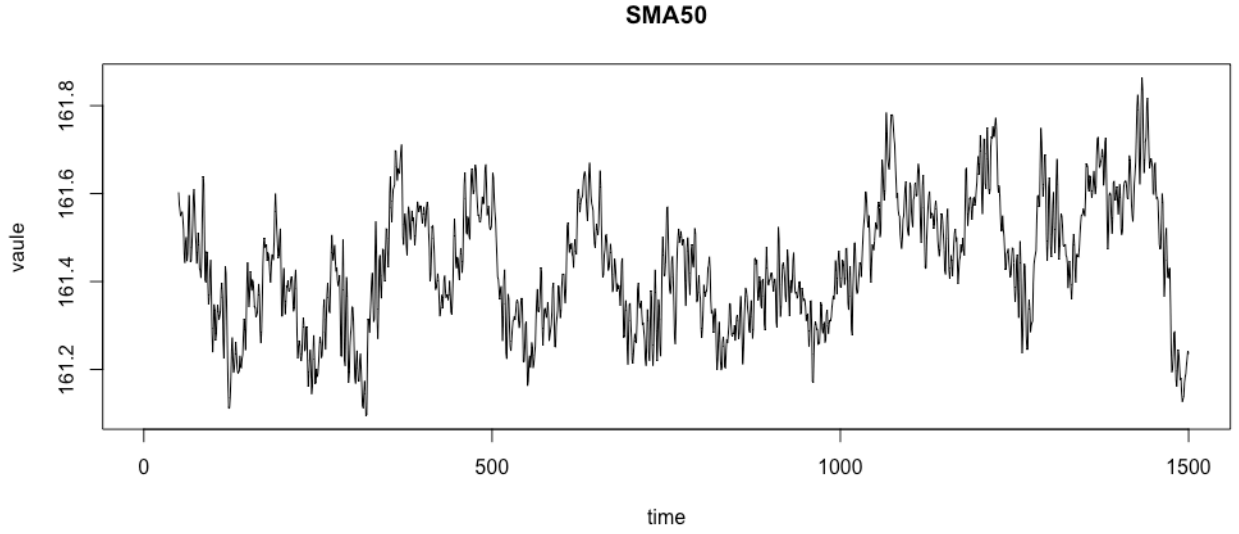
Selectedly, we print some SMA models shown below when k = 10,20,30,40,50.


SMA10
vaule
160.5
161.0
161.5
162.0
162.5
0
500
1000
1500
time


SMA20
vaule
161.0
161.5
162.0
0
500
1000
1500
time

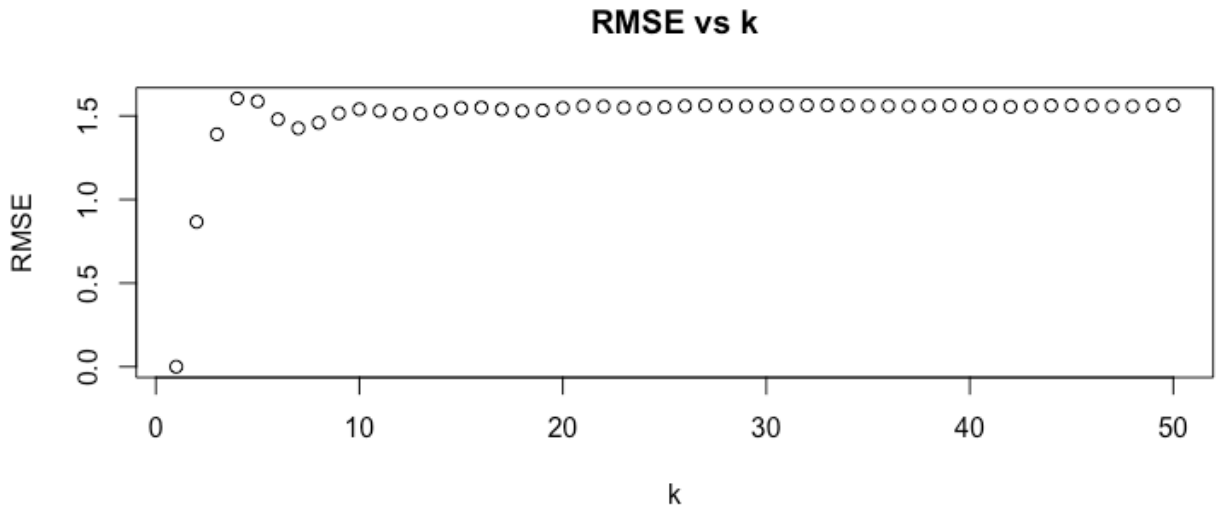

SMA30
vaule
161.0
161.4
161.8
0
500
1000
1500
time



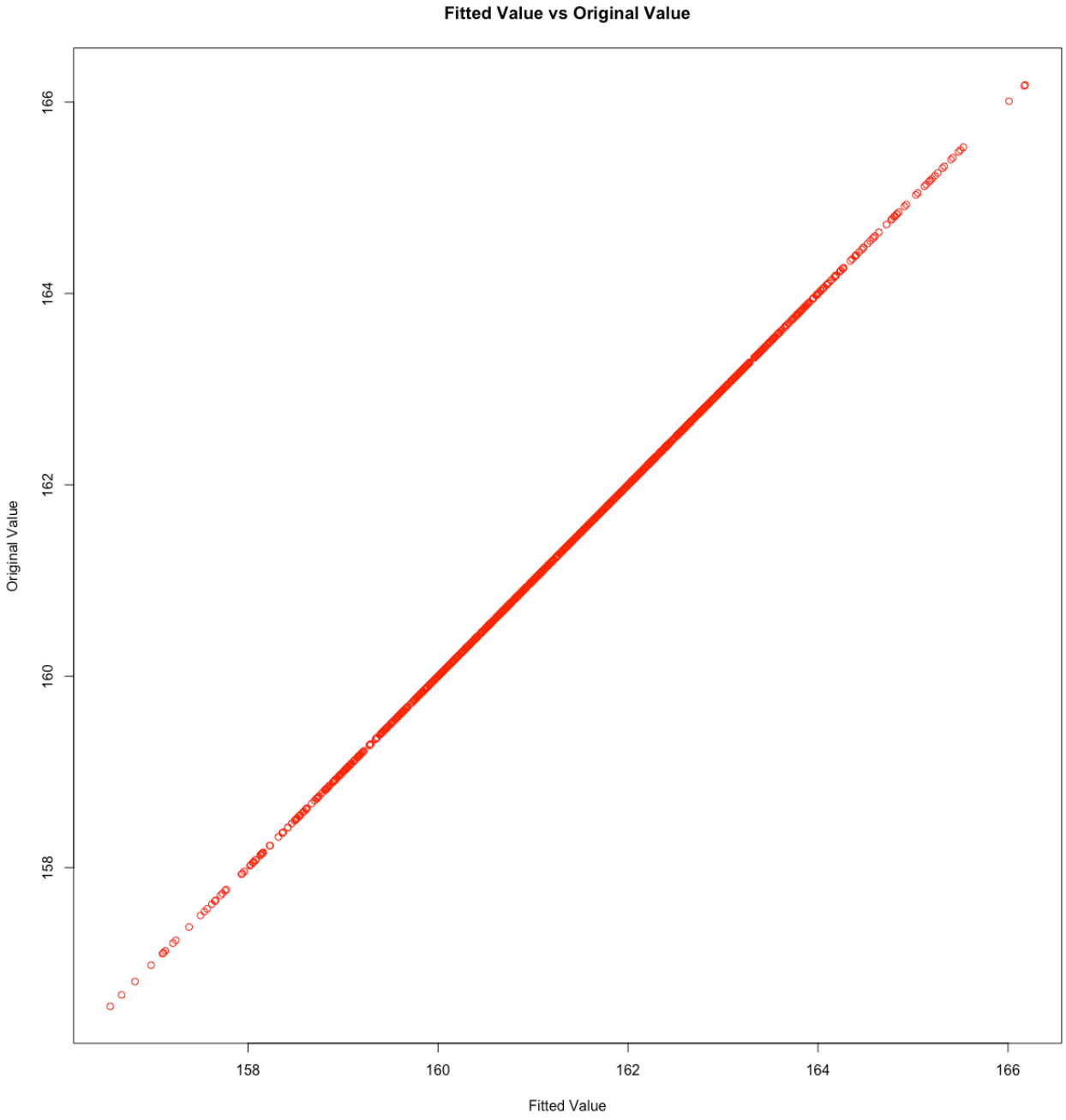


2.4

Plot RMSE vs k as follows



To plot the Fitted Value vs Original Value, we choose Fitted Value as x axis and Original Value y asix. We know RMSE = 4.798373e-13 is very small, so the the plots shows nearly like a line with slope 1, shown as follows.



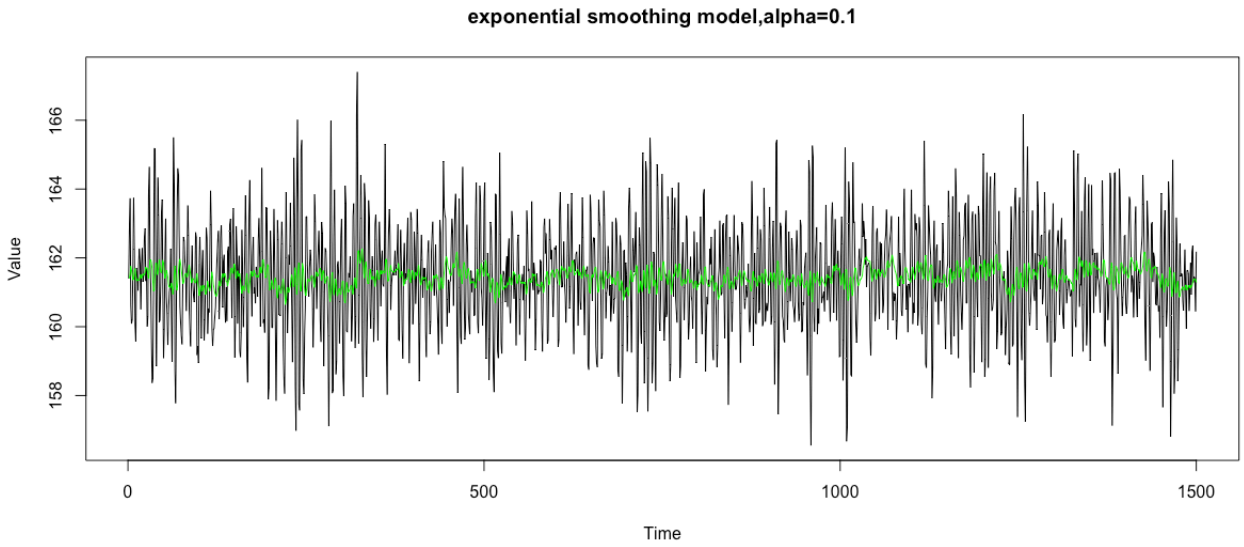
2.5

From this task, we know that modeling using simple moving average model, the k value selection affects the result of modeling. we need to select the best value for k and we use RMSE (Routed mean squared error) method for it. In this case, we conclude that, RMSE increases as k increases firstly and at some point of k, the RMSE not change much . And when k =1 ,we can get the best modeling result with RMSE 4.798373e-13

Task 3

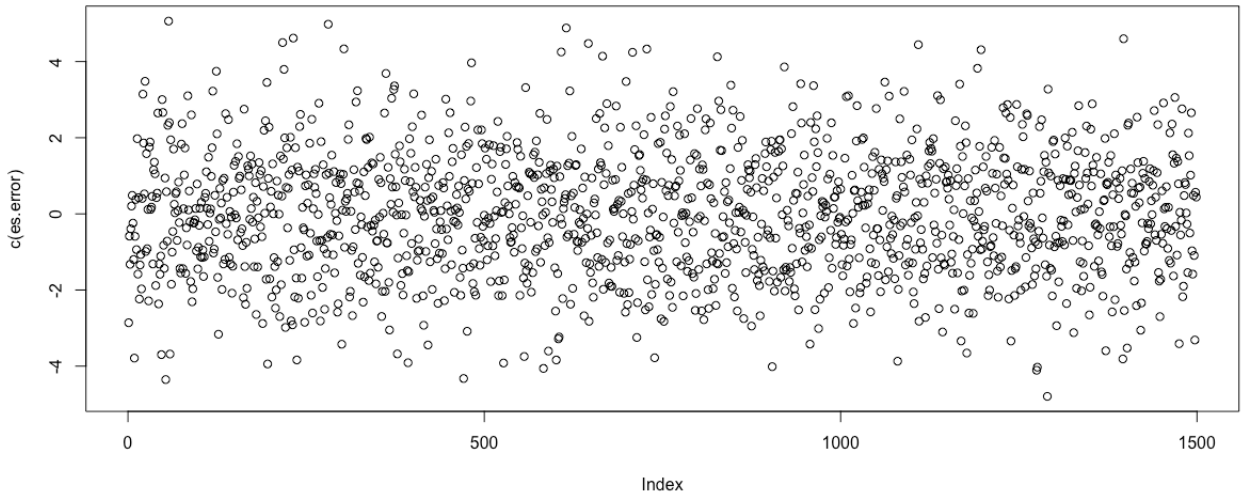
3.1

Use `HoltWinters` to do the exponential smoothing model and get the result as follows.



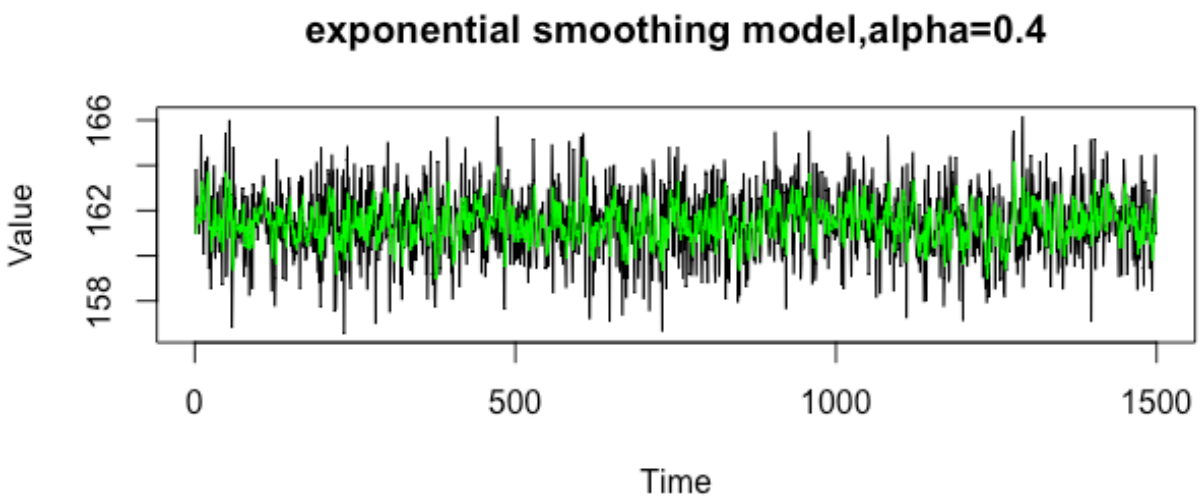
3.2

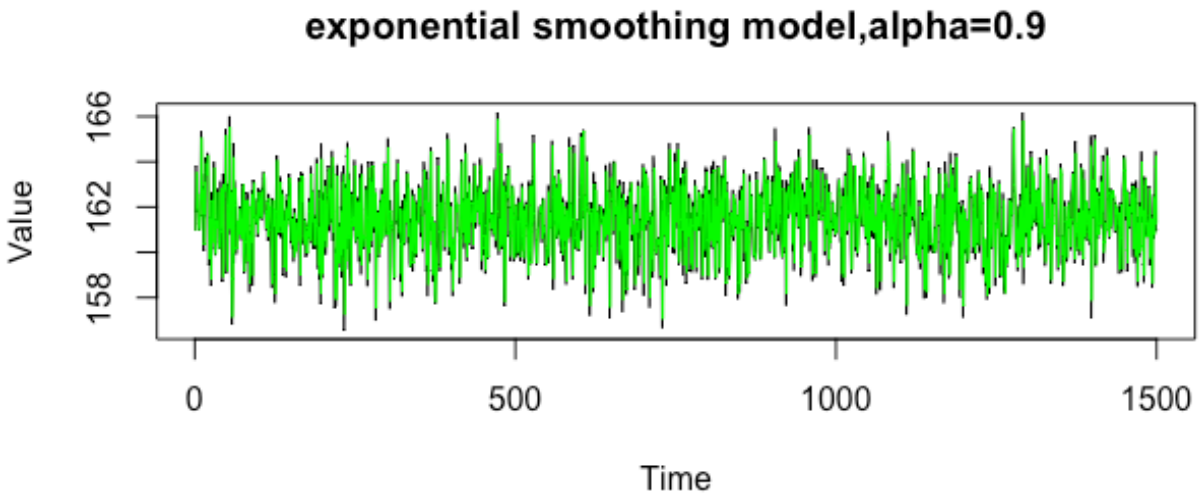
For exponential smoothing model , we just get 1499 fitted values from the 1500 observations. And the first value in fitted value vector is the predit value when t = 2. so when calculating the error series. we need to choose the train data from 2 to 1500 compared with the fitted values.We plot the error series as follows



The result of `HoltWinters` contains information of $SSE, the sum of squared errors for the in-sample forecast error. So we can use SSE divide the length of data length and then sqrt to get RMSE = 1.640987 (Same with the result we use the calculation function rmse in 2.2).

Selectedly, we plots the models when $\alpha = 0.4$ and 0.9





3.3

we abstrat function `doEsModel` to do exponential smoothing model and return RMSE . Let alpha be the param of the function.

```
doEsModel <- function(p){
  plot(myts.train,main=paste("exponential smoothing
model,alpha=",p,sep=""),ylab = "Value")
  myts.es = HoltWinters(myts.train,alpha =p ,beta = F,gamma = F)
  lines(myts.es$fitted[,1],col="green")
   return (sqrt(myts.es$SSE/length(myts.es$fitted[,1])))
}
```

then we use a loop to do the modeling varying alpha from 0.1 to 0.9 and save RMSEs in a vector.

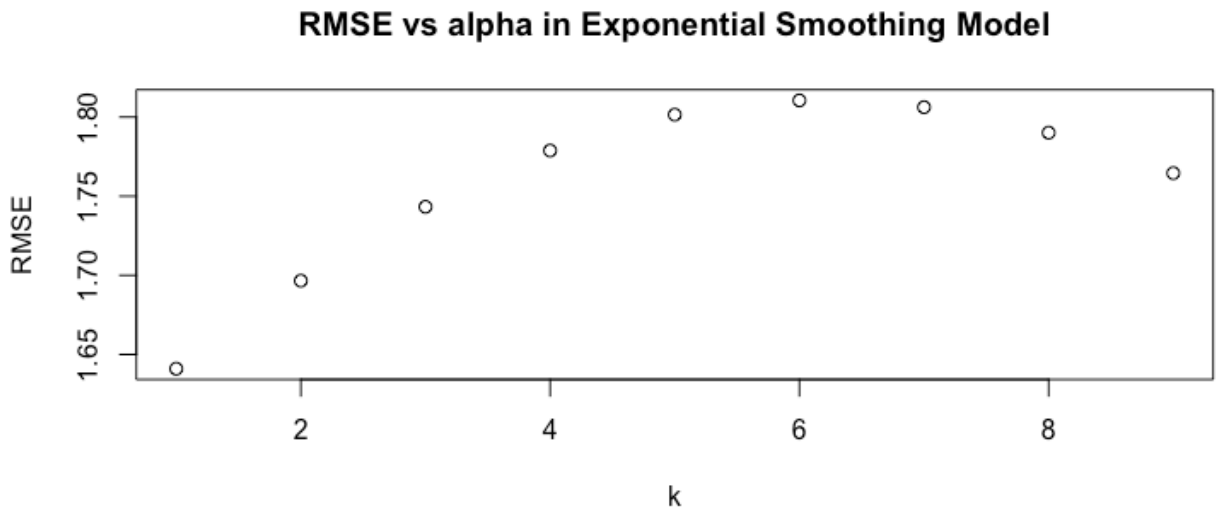
```
es.rmses = c()
for( a in 1:9){
  es.rmses = c(es.rmses,doEsModel(a/10))
}
print(es.rmses)
```

we get RMSEs as follows

```
in adf.test(myts) : p-value smaller than printed p-value
> source('~/Google Drive/Courses/CSC591-004/hw/ForecastingProject/forecast.R')
[1] 1.640987 1.696583 1.743283 1.778825 1.801498 1.810518 1.806241 1.790135 1.764565
Warning message:
```

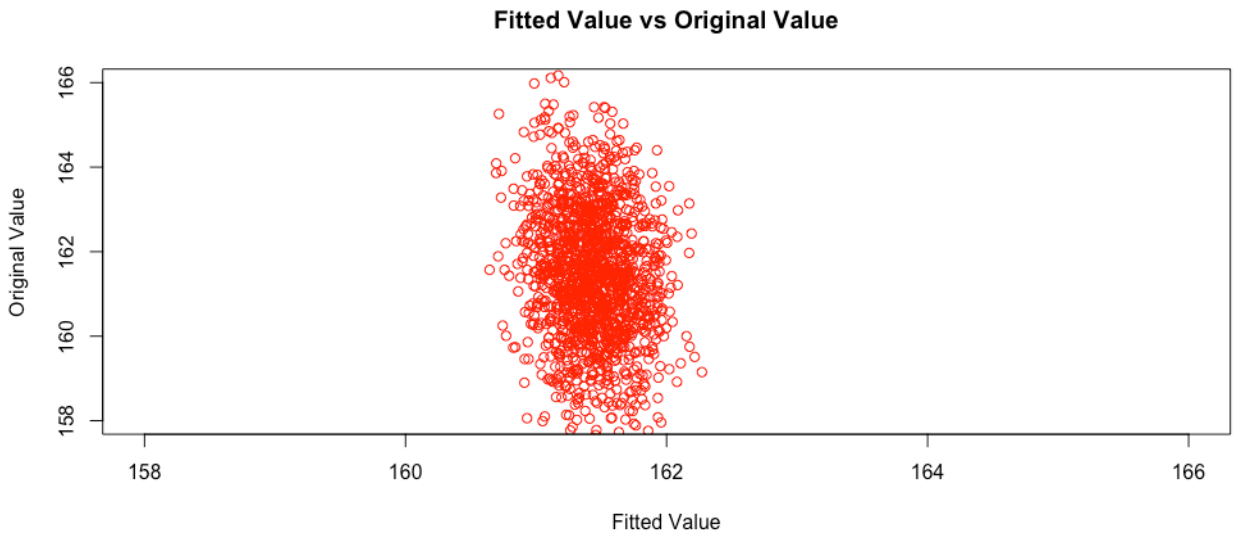
3.4

Plot RMSE vs the α as follows.We knows that when $\alpha = 0.1$ the RMSE is lowest.



3.5

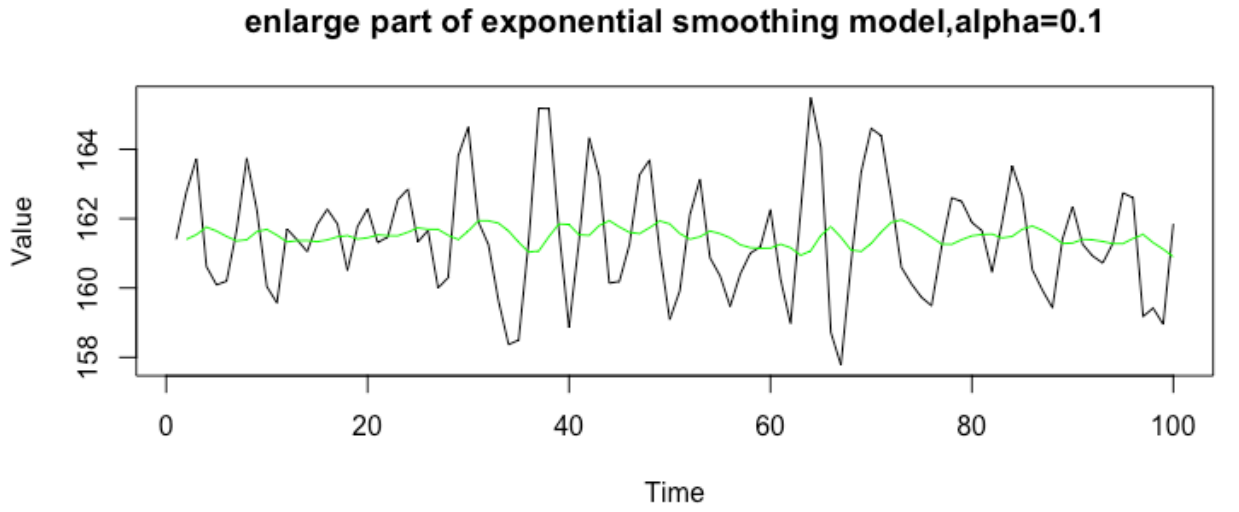
when $\alpha = 0.1$ plot the predicted values vs the original values shown as follows.

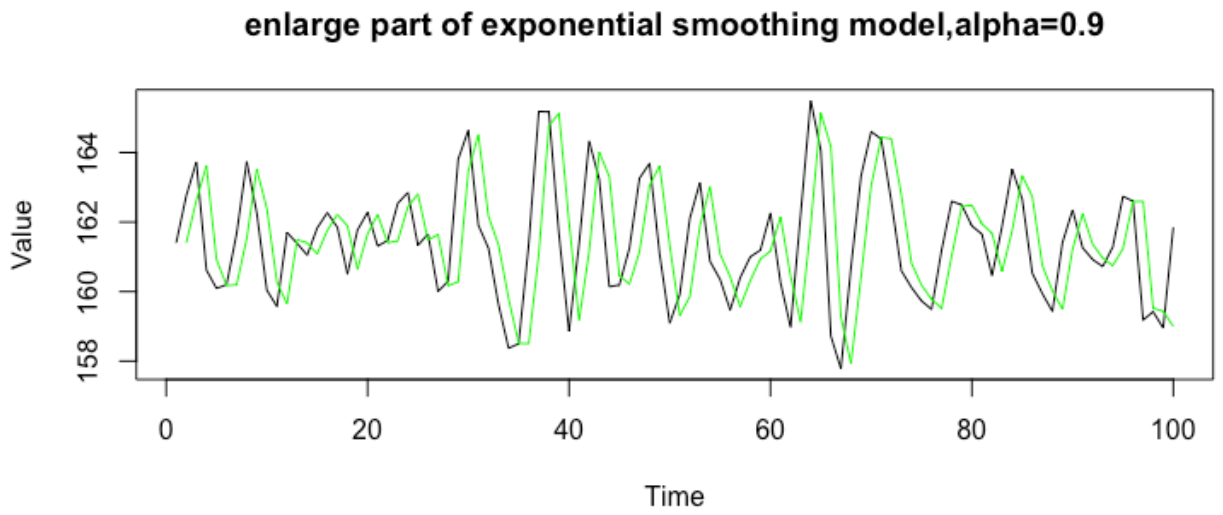


Visually, the fitted data are centered around 161-162, It is much smoother than the time series of the original data here. Also, from the RMSE above, we know that the RMSE is minimum when $\alpha = 0.1$. As α increases, the lag become more and more distinctive and the RMSE increases.

3.6

From this task, we know that modeling using exponential smoothing model, the α value selection affect the result of modeling. we need to select the best value for α and we use RMSE (Routed mean squared error) method for it. In this case, the RMSE is minimum when $\alpha = 0.1$. Maybe it is a little hard for us to visually see how fit the α affect. But we can draw and enlarge part of the plots of modeling. For example, the two images shown below enlarge the first 100 points of model when $\alpha = 0.1$ and 0.9. Now we can see when $\alpha = 0.9$. Lags are distinctive which will enlarge the error. So modeling when $\alpha = 0.9$ is not as good as modeling when $\alpha = 0.1$

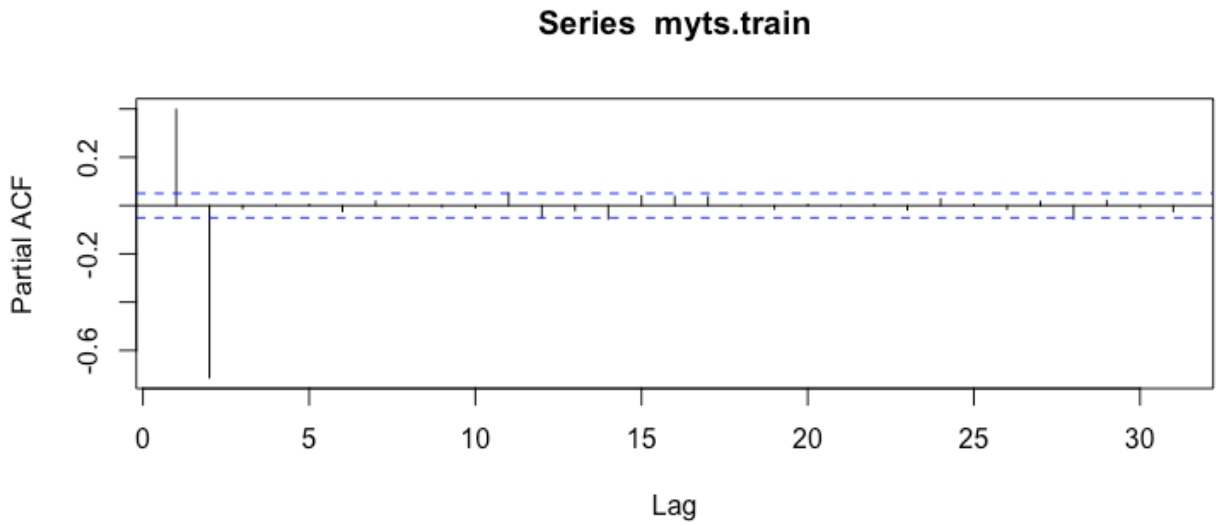




Task 4

4.1

we can use function `pacf` to plot PACF. as shown below, we should select p =2



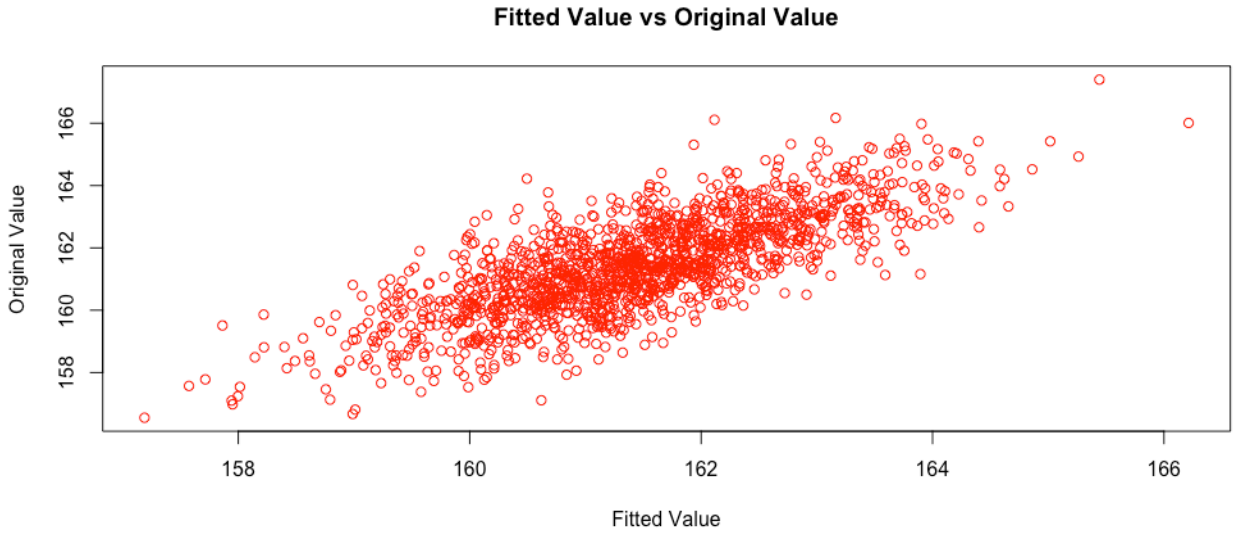
4.2

we use `ar` function to model AR(p) and get the estimate params of 0.6819 and -0.7130

```
 [ reached getOption( max.print ) -- omitted 500 entries
> myts.ar = ar(myts.train,order.max=2,aic=FALSE)
> myts.ar

Call:
ar(x = myts.train, aic = FALSE, order.max = 2)

Coefficients:
      1         2
 0.6819   -0.7130

Order selected 2  sigma^2 estimated as  1.033
>
```

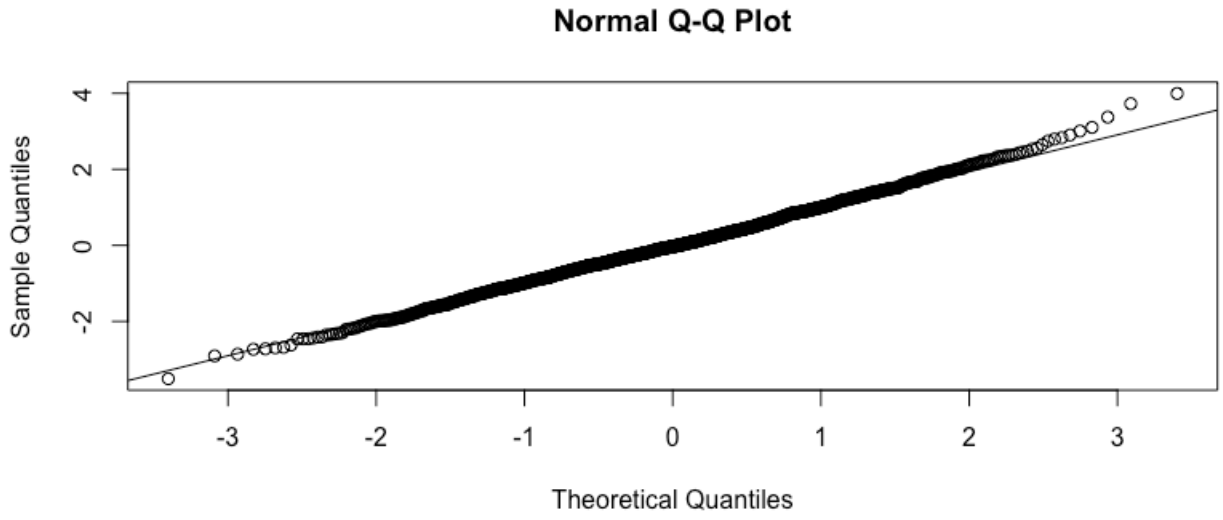
There is no fitted values in the output of `ar` but there are resid. So we can use the orginalData - resid to get the fitted value and plot them. We use the function rmse we define above to compure RMSE = 1.014927

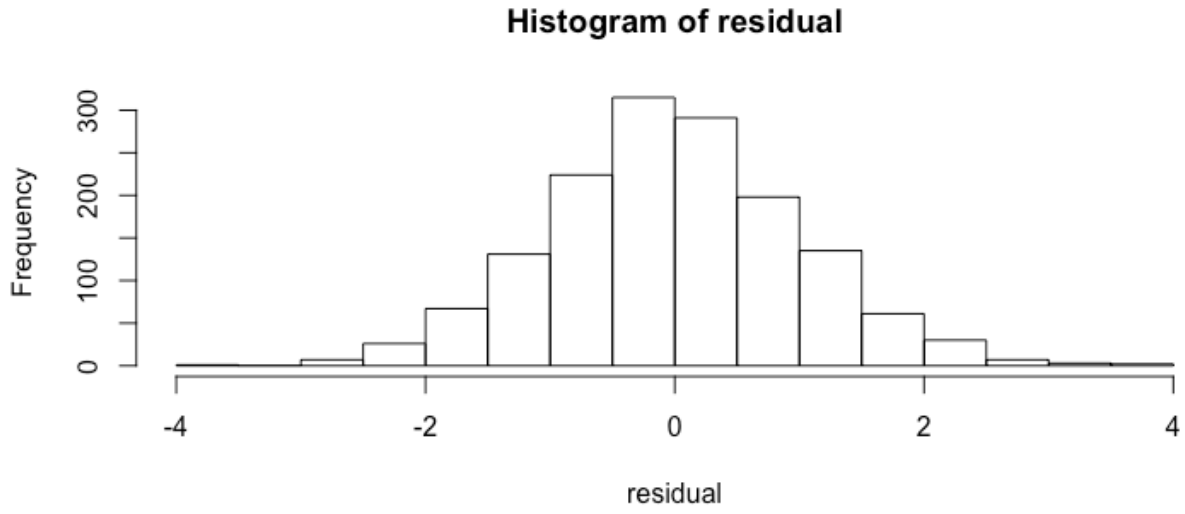
```
myts.ar = ar(myts.train,order.max=2,aic=FALSE)
myts.ar.fitted = myts.train - myts.ar$resid
plot(myts.ar.fitted,myts.train,main = "Fitted Value vs Original Value",xlab =
"Fitted Value",ylab = "Original Value",col="red")
residUsed = myts.ar$resid[3:1500]
ar.rmse =rmse(residUsed)
print(ar.rmse)
```



4.3

a.





we carried out χ2 test by the following code. From the reslut p-value < 2.2e-16, we can reject null hypothesis, means we have strong confidence that residual follows the normal distribution N(0,s^2)

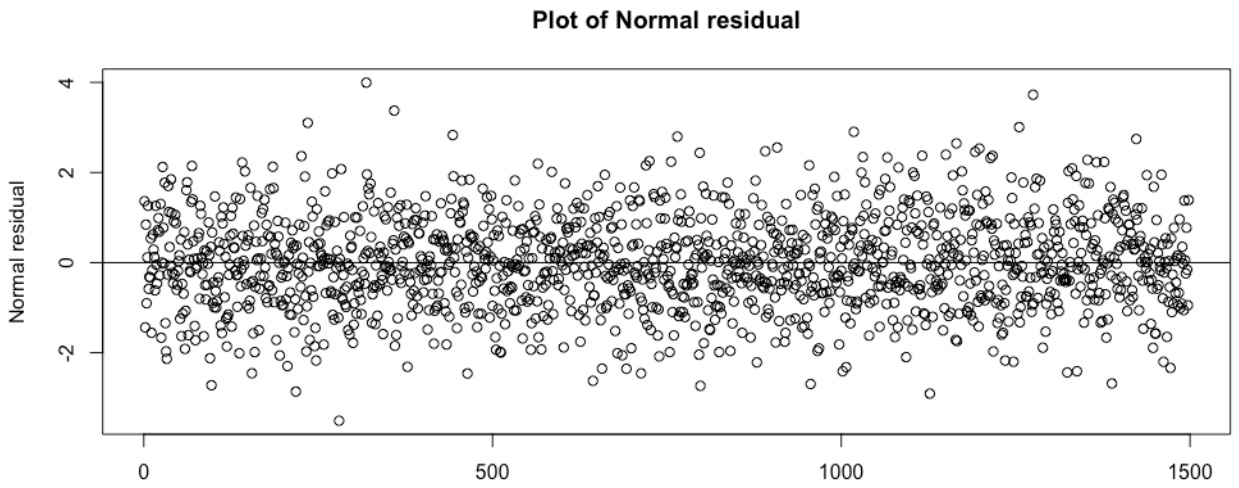
> residUsed = myts.ar$resid[3:1500]

> chisq.test(residUsed^2).

> data: residUsed^2 X-squared = 3456.7, df = 1497, p-value < 2.2e-16

b.

the residual scatter plot drawn as follows. Conclusion: residuals have no correlation trends



4.4

From this task, we know that modeling using AR(p) model. the p value selection affect the result of modeling. We use PACF to determine the lag k at which PACF cuts off to select the order p. After p selection, we can do AR modeling to estimate the params and get the resid.

From the residuals analysis, we draw histogram and carry out χ2 test and Q-Q plot, leading to the conclusion that residual follows the normal distribution N(0, s^2). From the scatter plot of residuals, we conclude residuals have no trends.

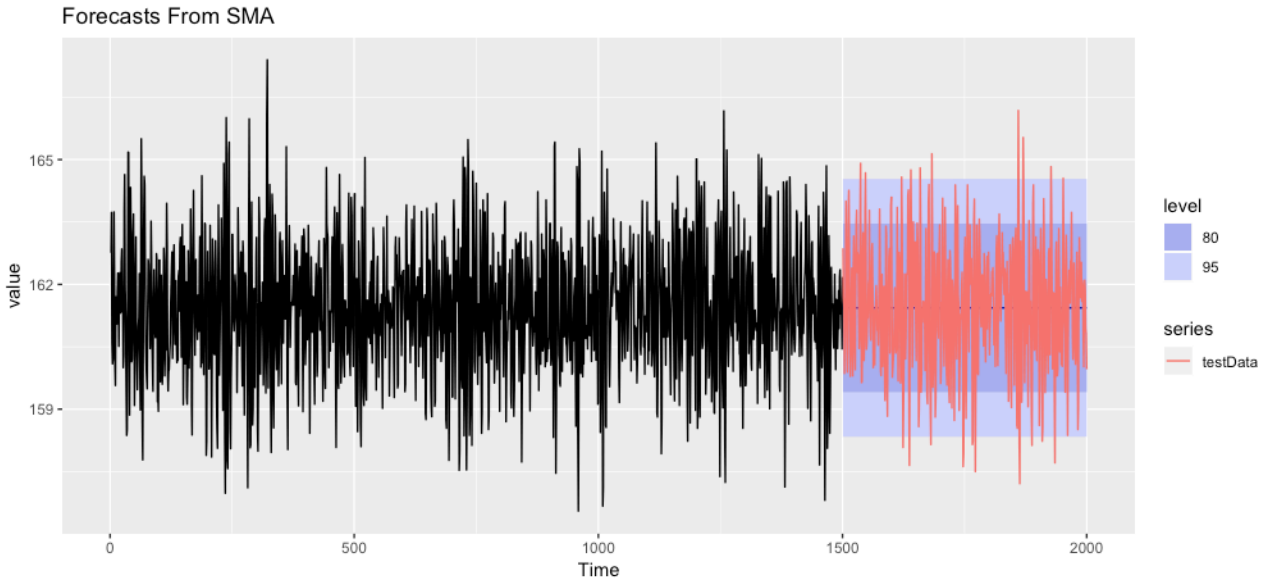
Task 5

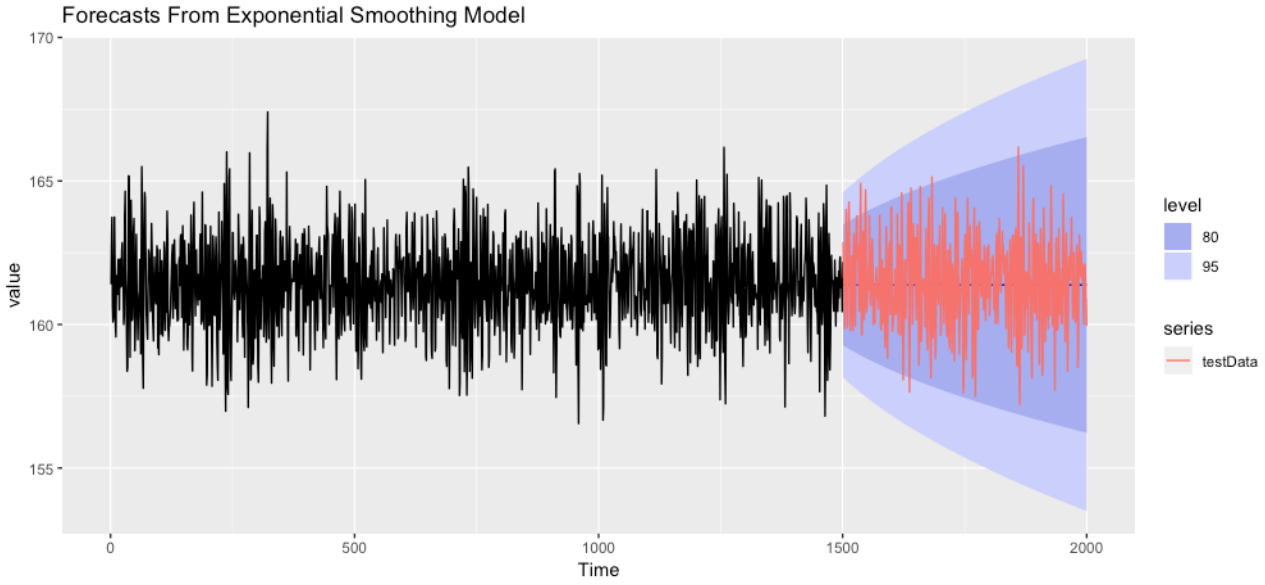
We use `forecast` function to forecast the next 500 step and then plot together with the test dataset. Then we use `accuracy(f,x)`, in which input param `f` is the forecast result and `x` is the test DataSet to calculate the accuracy.

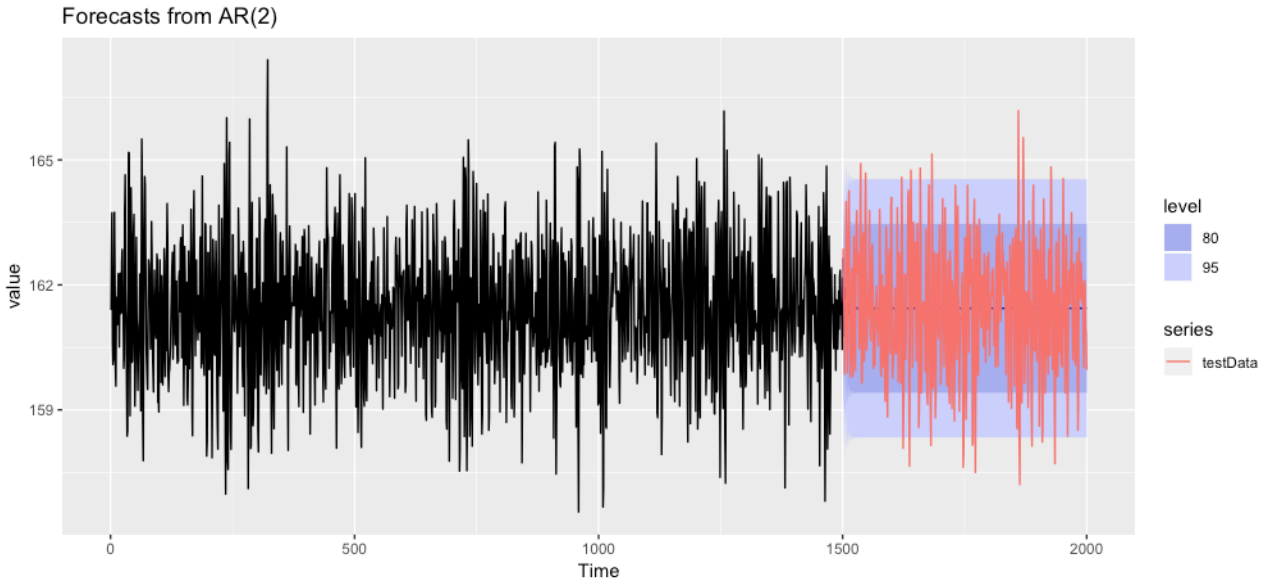
```
library(TTR)
library(forecast)
testData = myts.test
sma = SMA(myts.train,n=1)
fore.sma= sma %>%forecast(h=500)
fore.sma%>% autoplot(main="Forecasts From SMA",ylab="value") +
autolayer(testData)
accuracy(fore.sma,testData)


es = HoltWinters(myts.train,alpha =0.1 ,beta = F,gamma = F)
fore.es=es %>%forecast(h=500)
fore.es%>%autoplot(main="Forecasts From Exponential Smoothing
Model",ylab="value") + autolayer(testData)
```

```
accuracy(fore.es,testData)

ar = ar(myts.train,order.max=2,aic=FALSE)
fore.ar = ar %>%forecast(h=500)
fore.ar %>%autoplot(ylab="value") + autolayer(testData)
accuracy(fore.ar,testData)
```

plot results are shown below for SMA, Exponential Smoothing Model,AR(2) perspectively.







The accuracy between forecast models and testDate is shown below.

```
> fore.sma%>% autoplot(main="Forecasts From SMA",ylab="value") + autolayer(testData)
> accuracy(fore.sma,testData)
                     ME     RMSE      MAE          MPE      MAPE      MASE      ACF1 Theil's U
Training set 0.003271921 1.579168 1.254475 -0.007542199 0.7771332 0.9199607 0.3980748        NA
Test set     0.042039764 1.526884 1.230320  0.017094157 0.7620517 0.9022464 0.4100325 0.9210255
> es = HoltWinters(myts.train,alpha =0.1 ,beta = F,gamma = F)
> fore.es=es %>%forecast(h=500)
> fore.es%>%autoplot(main="Forecasts From Exponential Smoothing Model",ylab="value") + autolayer(testData)
> accuracy(fore.es,testData)
                       ME     RMSE      MAE         MPE      MAPE      MASE      ACF1 Theil's U
Training set -0.0001403575 1.640987 1.307375 -0.00990312 0.8098968 0.9587561 0.3862285        NA
Test set      0.0993395934 1.529535 1.233907  0.05258187 0.7640038 0.9048791 0.4100325 0.9222927
> ar = ar(myts.train,order.max=2,aic=FALSE)
> fore.ar = ar %>%forecast(h=500)
> fore.ar %>%autoplot(ylab="value") + autolayer(testData)
> accuracy(fore.ar,testData)
                       ME     RMSE      MAE          MPE      MAPE      MASE         ACF1 Theil's U
Training set -0.0004201713 1.014927 0.796098 -0.004217922 0.4931581 0.5838140 -0.008849123        NA
Test set      0.0393856789 1.522863 1.226080  0.015484875 0.7594419 0.8991393  0.409588637 0.9193969
> ?accuracy()
```

Result interpretation:

The forecasts are shown as a blue line, with the 80% prediction intervals as an darker shaded area, and the 95% prediction intervals as a lighter shaded area. Original test data are shown as the red line.mAnd we check the preditted effect by `accuracy` function in which inputs are forecast values and original test values. From the RMSE, MAE, MPE, MAPE for each one, we know that AR(2) is the best model we can use.