# Time Series Analysis

Md Mominul Islam

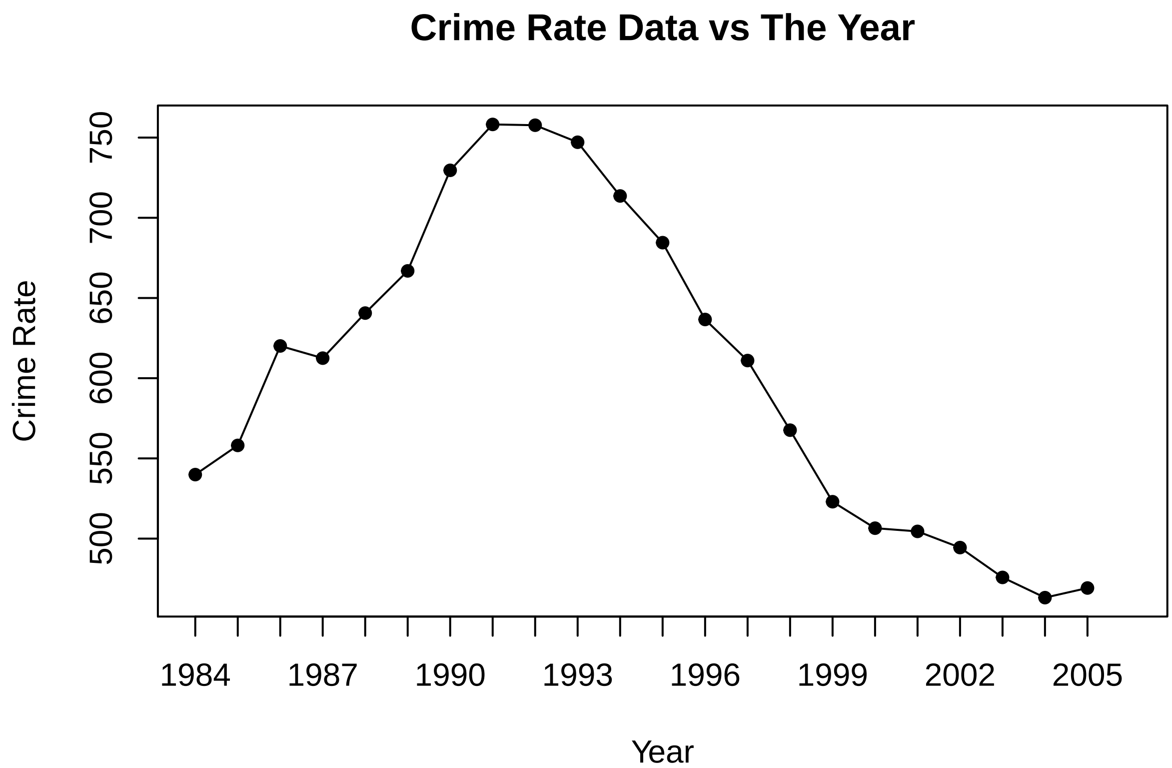
12/3/2020

**1. (5 points) Plot the crime rate data vs the year.**

**Answer**

```
## Loading required package: carData

##
## Attaching package: 'dplyr'

## The following object is masked from 'package:car':
##
##     recode

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

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

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

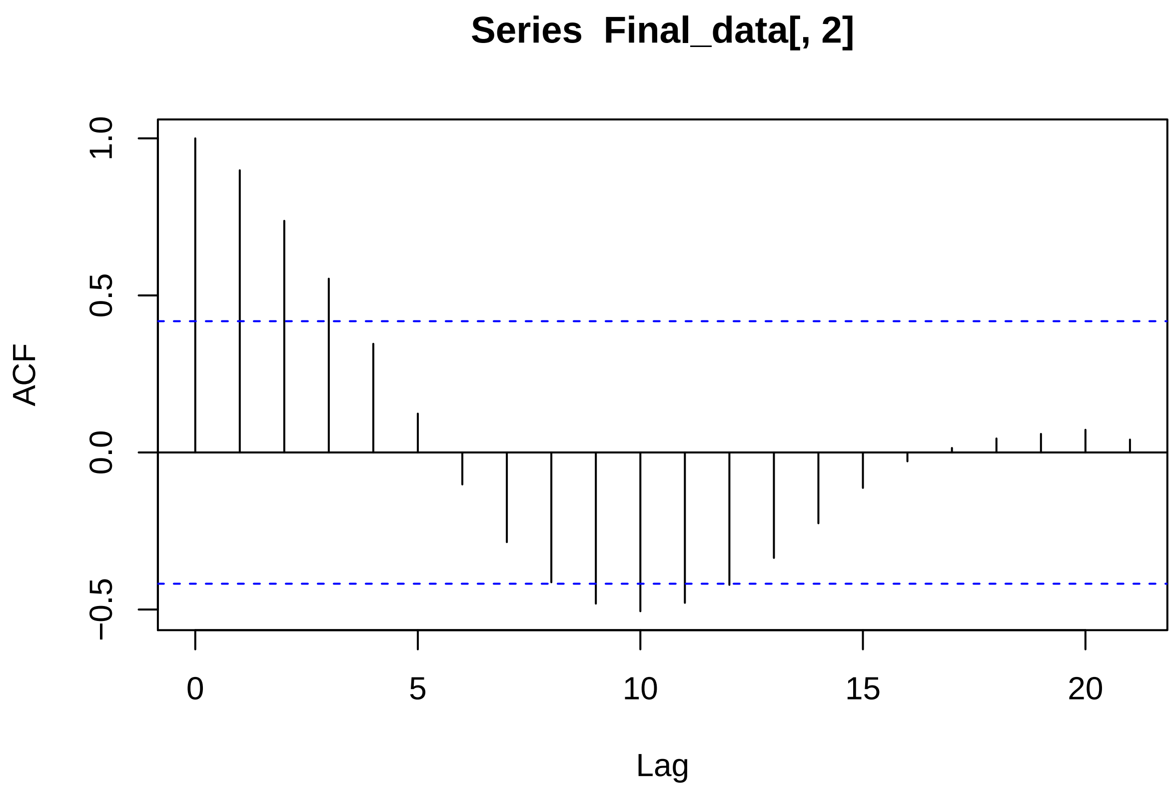
```r
Final_data <- as.data.frame(Final_data)

#Ploting Crime Rate Data vs The Year
plot(Final_data[,2],type="o",pch=19,cex=0.8,xlab='Year',
     xlim=c(1,23),xaxt= 'n', ylab='Crime Rate',
     main= "Crime Rate Data vs The Year")
axis(1, seq(1,22,1), Final_data[seq(1,22,1),1])
```



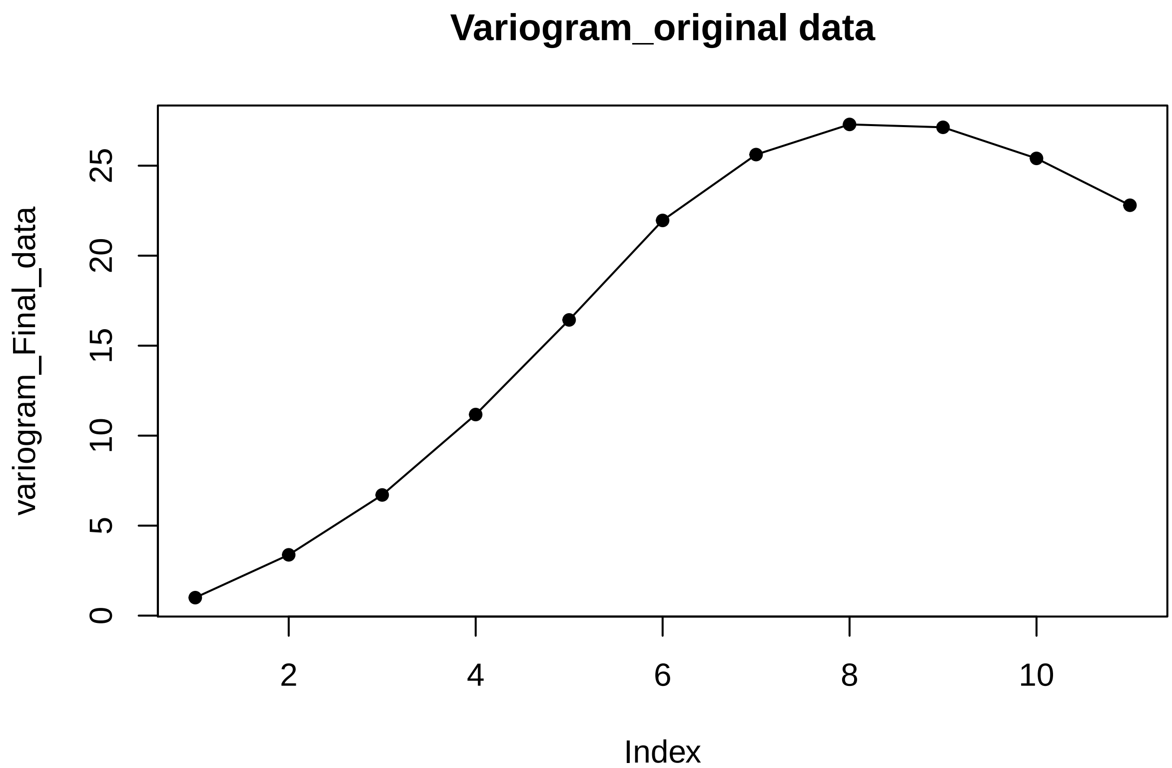

**2. (10 points) Calculate and plot the sample autocorrelation function (ACF) and variogram. List the first 10 values of ACF and variogram respectively.**

**Answer**

```
# Defining the variogram function
variogram_func <- function(x, lag) {
x <- as.matrix(x) # Make sure the x is a vector. It represents the observations of y_t.
Lag <- NULL
var_k <- NULL
vario <- NULL
for (k in 1:lag) {
Lag[k] <- k
var_k[k] <- sd(diff(x, k))^2
vario[k] <- var_k[k] / var_k[1]
}
return(as.data.frame(cbind(Lag, vario)))
}

#calculating ACF
ACF_Final_data <-acf(Final_data[,2], lag.max=25,type="correlation")
```




```r
#calculating Variogram
x <- Final_data$Rate
lag_length <- length(x) /2
lag_cheese <- 1:lag_length
z <- variogram_func(x, lag_length)
variogram_Final_data <- z$vario
#Ploting Variogram
plot(variogram_Final_data, type="o",
     main="Variogram_original data", pch=19,cex=.8, col="black" )
```




```r
#Table for ACF and Variogram
ACF <- ACF_Final_data$acf[1:10]
Variogram <- variogram_Final_data[1:10]
Variogram <- as.data.frame(Variogram)

Table <- data.frame (
  ACF,
  Variogram
)
kable(Table,
      caption = 'ACF and Variogram')
```

Table 1: ACF and Variogram

| ACF | Variogram |
|---:|---:|
| 1.0000000 | 1.000000 |
| 0.8984053 | 3.377662 |
| 0.7374591 | 6.703893 |
| 0.5532590 | 11.173629 |
| 0.3459439 | 16.429739 |
| 0.1236614 | 21.957853 |
| -0.1018317 | 25.616645 |
| -0.2854537 | 27.291365 |
| -0.4133137 | 27.131428 |

| ACF | Variogram |
|---|---|
| -0.4812716 | 25.404944 |

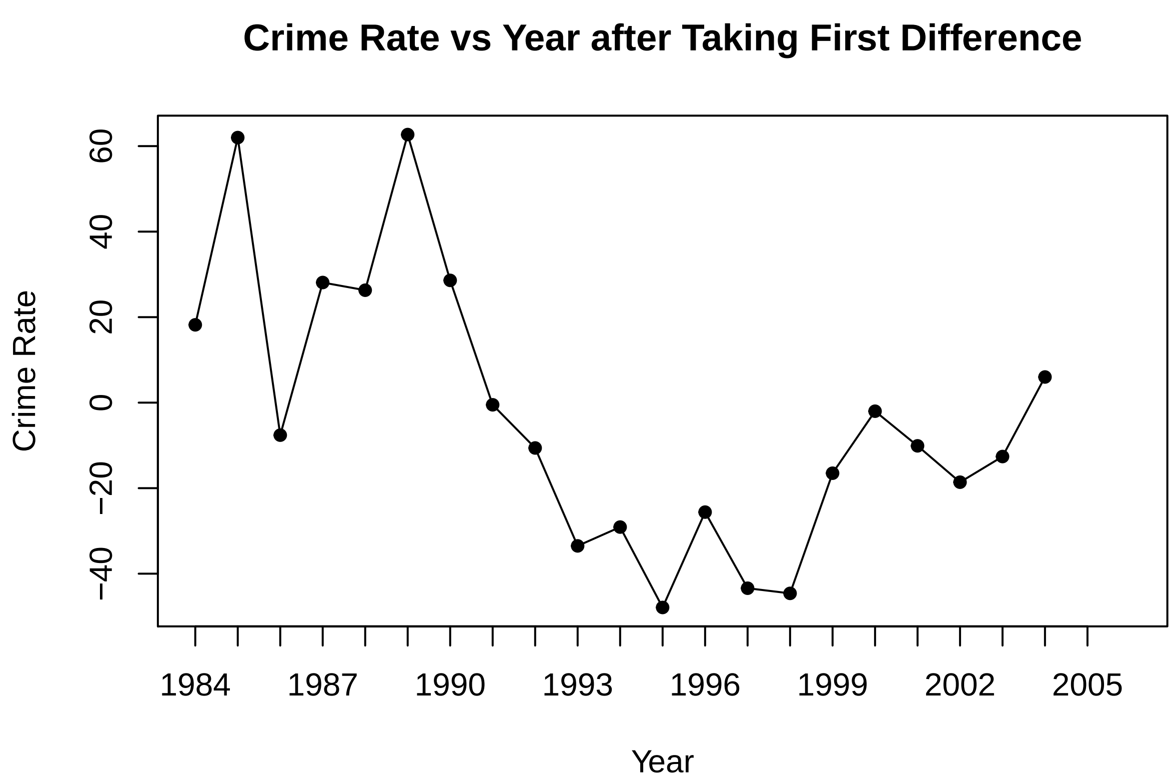
**3. (5 points) Is there an indication of nonstationary behavior in the time series? Why or why not?**

**Answer**

**4. (10 points) Calculate and plot the first difference of the timeseries. Show the first 10 differences.**

**Answer**

```
#Calculating First Difference Data
Final_data.df1 <- diff(Final_data[,2])
#Plotting First Difference Data
plot(Final_data.df1, type="o",pch=19,cex=0.8,xlab='Year',
     xlim=c(1,23), xaxt= 'n', ylab='Crime Rate',
     main= "Crime Rate vs Year after Taking First Difference")
axis(1, seq(1,22,1), Final_data[seq(1,22,1),1])
```




```
#Showing First 10 Differences
Final_data_diff <- cbind(Final_data[1:10,1],Final_data[1:10,2], Final_data.df1[1:10])
Final_data_diff <- as.data.frame(Final_data_diff)
#Replacing with suitable one
```

```r
colnames(Final_data_diff) = c("Year", "Crime Rate", "Difference Data")
kable(Final_data_diff,
      caption = 'First Differences of the Time Series')
```

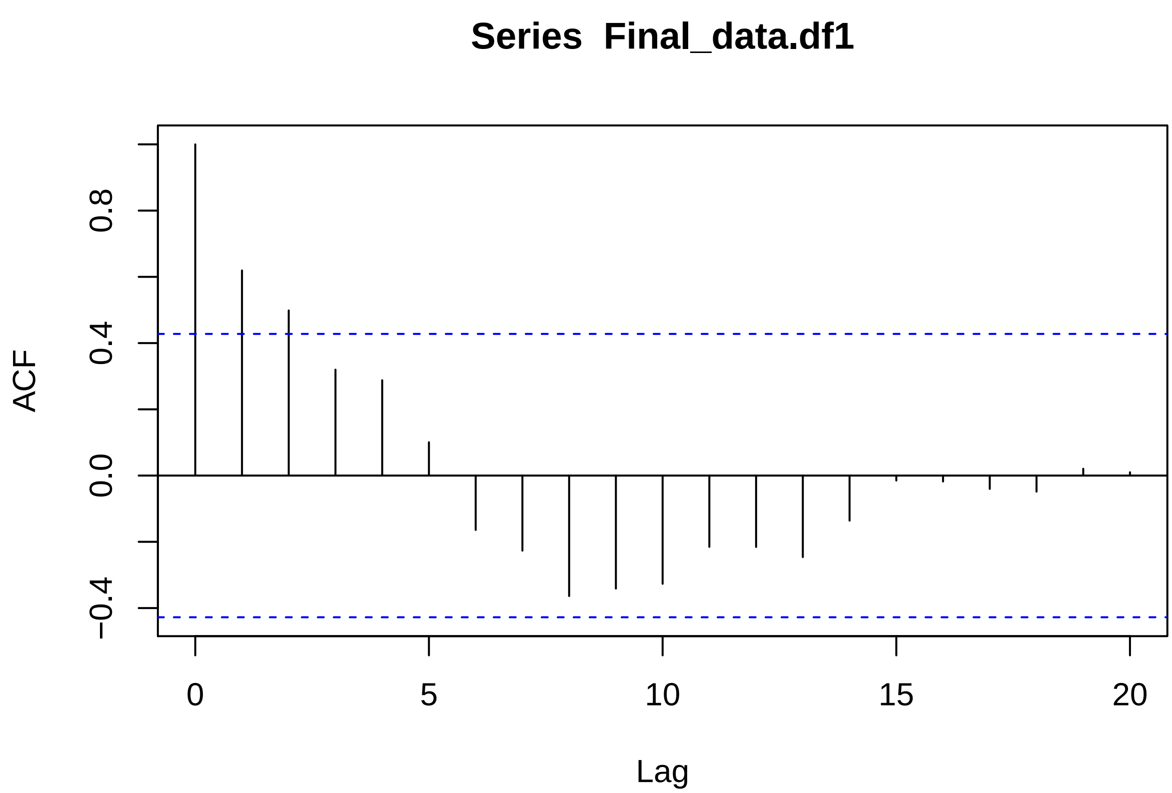
Table 2: First Differences of the Time Series

| Year | Crime Rate | Difference Data |
|---|---:|---:|
| 1984 | 539.9 | 18.2 |
| 1985 | 558.1 | 62.0 |
| 1986 | 620.1 | -7.6 |
| 1987 | 612.5 | 28.1 |
| 1988 | 640.6 | 26.3 |
| 1989 | 666.9 | 62.7 |
| 1990 | 729.6 | 28.6 |
| 1991 | 758.2 | -0.5 |
| 1992 | 757.7 | -10.6 |
| 1993 | 747.1 | -33.5 |

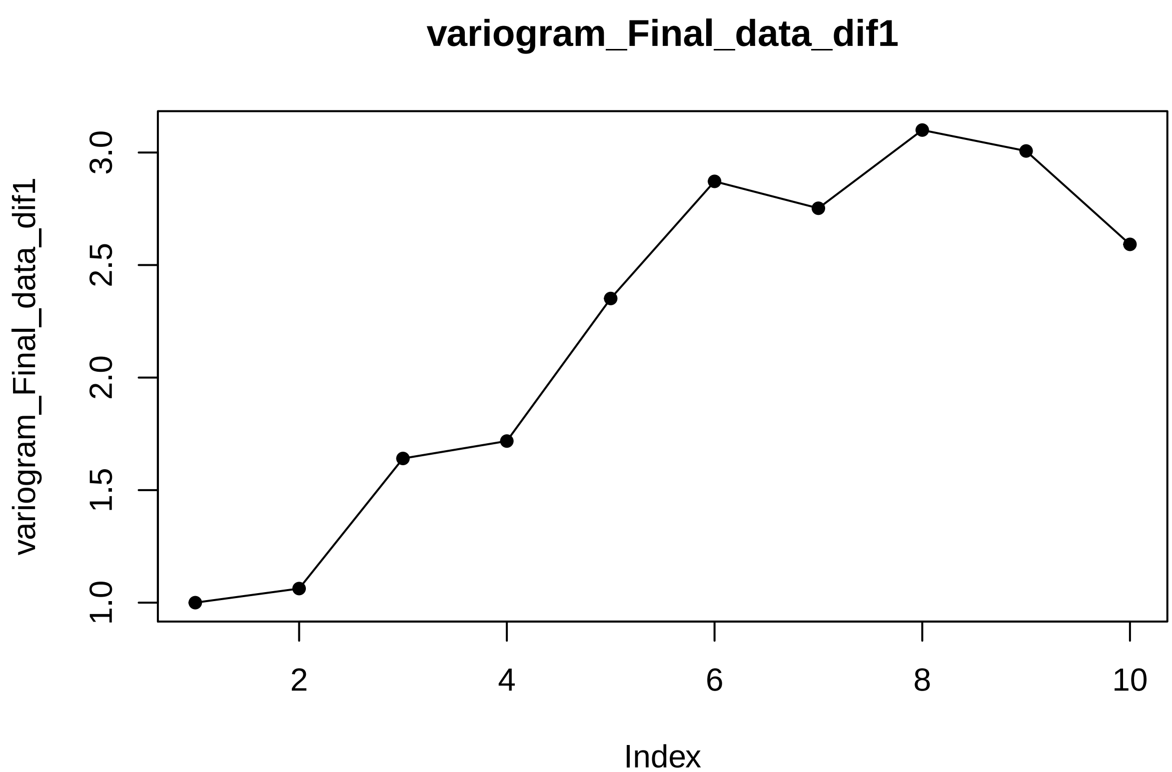
**5. (10 points) Compute the sample autocorrelation function (ACF) and variogram of the first differences.**

**Answer**

```r
#calculating ACF
ACF_Final_data_dif1 <-acf(Final_data.df1, lag.max=25,type="correlation")
```



```r
#calculating Variogram
x <- Final_data.df1
lag_length <- length(x) /2
lag_cheese <- 1:lag_length
z <- variogram_func(x, lag_length)
variogram_Final_data_dif1 <- z$vario

#Table for ACF and Variogram
ACF_dif1 <- ACF_Final_data_dif1$acf[1:10]
Variogram_dif1 <- variogram_Final_data_dif1[1:10]
Variogram_dif1 <- as.data.frame(Variogram_dif1)
plot(variogram_Final_data_dif1, type="o",
     main="variogram_Final_data_dif1", pch=19,cex=.8, col="black" )
```




```r
Table_dif1 <- data.frame (
  ACF_dif1,
  Variogram_dif1
)
kable(Table_dif1,
      caption = 'ACF with First Differences',
      col.names = c('ACF D1', 'Variogram D1'))
```

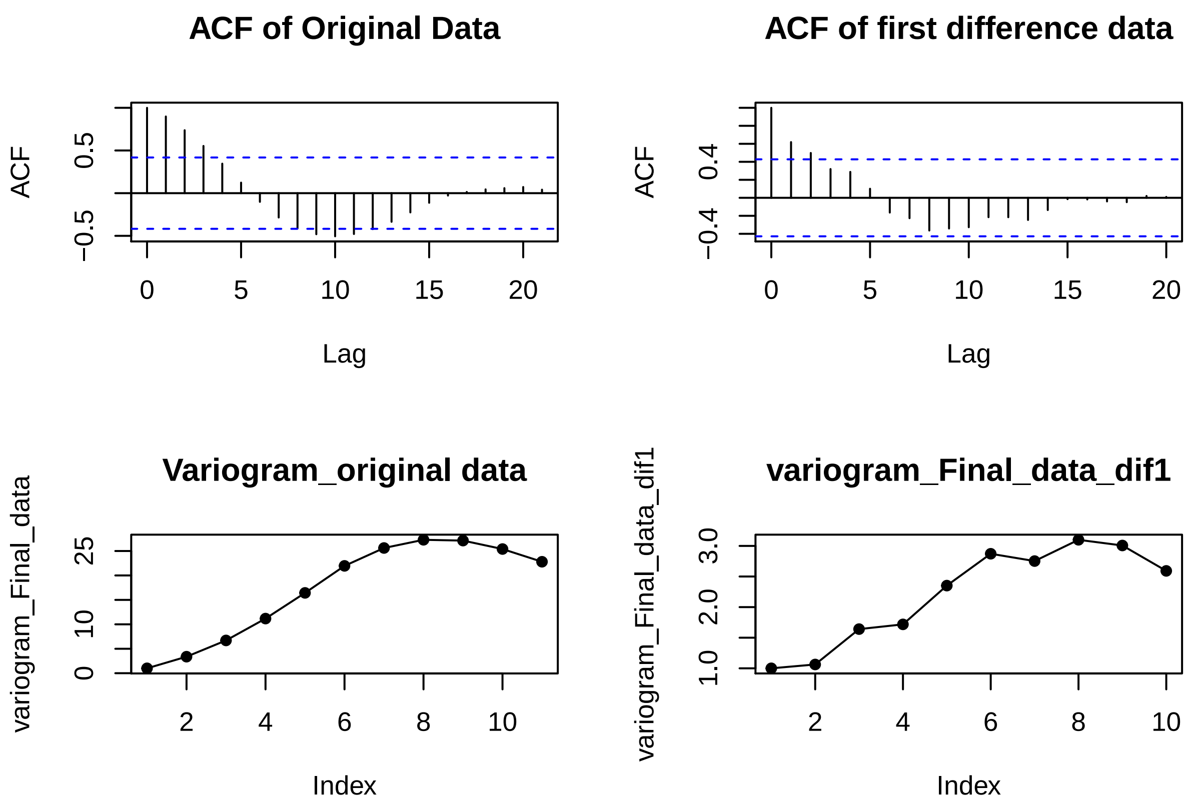
Table 3: ACF with First Differences

| ACF D1 | Variogram D1 |
|---|---|
| 1.0000000 | 1.000000 |
| 0.6193703 | 1.062543 |
| 0.4985524 | 1.640599 |
| 0.3198224 | 1.717923 |
| 0.2879014 | 2.351421 |
| 0.1007391 | 2.871869 |
| -0.1641531 | 2.752315 |
| -0.2267162 | 3.099729 |
| -0.3636319 | 3.006877 |
| -0.3412978 | 2.591916 |

**6. (5 points) What impact has differencing had on the time series?**

**Answer**

```
#Sample autocorrelation function of original data
par(mfrow=c(2, 2))
ACF_Final_data <-acf(Final_data[,2],
                     lag.max=25,type="correlation",
                     main="ACF of Original Data")
#Sample autocorrelation function of first difference data
ACF_Final_data_dif1 <-acf(Final_data.df1, lag.max=25,type="correlation",
main="ACF of first difference data")


#Variogram of Original data
plot(variogram_Final_data, type="o",
     main="Variogram_original data",
     pch=19,cex=.8, col="black" )

#Variogram of First Difference Data
plot(variogram_Final_data_dif1, type="o", main="variogram_Final_data_dif1", pch=19,cex=.8, col="black" )
```




**7. Develop an appropriate exponential smoothing forecasting procedure for the firstdifferencing data by answer the questions below.**

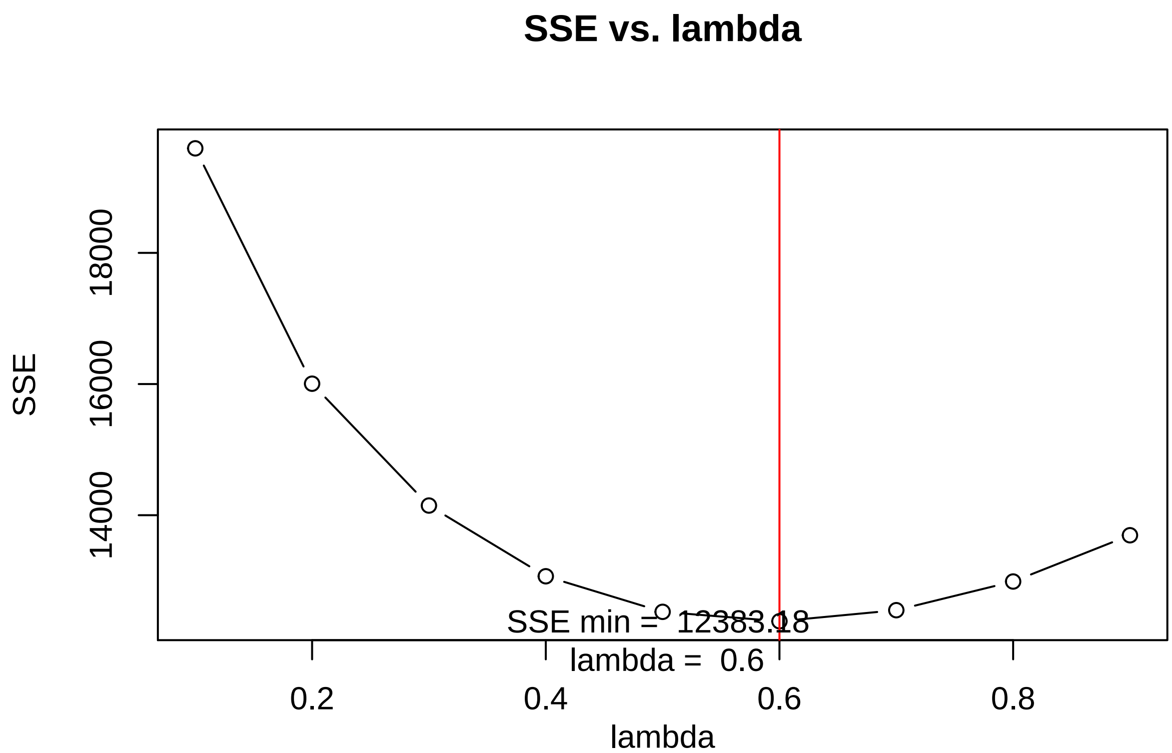
```
#Defining Smoothing Function
firstsmooth <- function(y, lambda, start=y[1]) {
           ## here the initial value is set to the first y value
          ytilde <- y
          ytilde[1] <- lambda*y[1]+(1-lambda)*start
          for (i in 2:length(y)) {
              ytilde[i] <- lambda*y[i] + (1-lambda)*ytilde[i-1]
          }
  ytilde
}
## Defining Trigg leach smooth Function
tlsmooth<-function(y,gamma,y.tilde.start=y[1],lambda.start=1){
      T<-length(y)
      #Initialize the vectors
      Qt<-vector()
      Dt<-vector()
      y.tilde<-vector()
      lambda<-vector()
      err<-vector()
      #Set the starting values for the vectors
      lambda[1]=lambda.start
      y.tilde[1]=y.tilde.start
      Qt[1]<-0
```

```
      Dt[1]<-0
      err[1]<-0
      for (i in 2:T){
          err[i]<-y[i]-y.tilde[i-1]
          Qt[i]<-gamma*err[i]+(1-gamma)*Qt[i-1]
          Dt[i]<-gamma*abs(err[i])+(1-gamma)*Dt[i-1]
          lambda[i]<-abs(Qt[i]/Dt[i])
          y.tilde[i]=lambda[i]*y[i] + (1-lambda[i])*y.tilde[i-1]
      }
return(cbind(y.tilde,lambda,err,Qt,Dt))
}

##Function for measures of accuracy
measacc.fs<- function(y,lambda){
      out<- firstsmooth(y,lambda)
      T<-length(y)
#Smoothed version of the original is the one step ahead prediction
#Hence the predictions (forecasts) are given as
      pred<-c(y[1],out[1:(T-1)])
      prederr<- (y - pred)
      SSE<-sum(prederr*prederr)
      MAPE<-100*sum(abs(prederr/y))/T
      MAD<-sum(abs(prederr))/T
      MSD<-sum(prederr*prederr)/T
      ret1<-c(SSE,MAPE,MAD,MSD)
      names(ret1)<-c("SSE","MAPE","MAD","MSD")
      return(ret1)
}
```

**a. (10 points) Assume the first-difference data is a constant process. For R user, use the HoltWinters() function to find the optimum value of Lamda to smooth the data. For JMP user, specify the Lamda given by the software.**

**Answer**

```
#Finding Out suitable Lamda Value for Low SSE
lambda.vec<-seq(0.1, 0.9, 0.1)
sse.mydata<-function(sc){measacc.fs(Final_data.df1,sc)[1]}
sse.vec<-sapply(lambda.vec, sse.mydata)
opt.lambda<-lambda.vec[sse.vec == min(sse.vec)]
plot(lambda.vec, sse.vec, type="b", main = "SSE vs. lambda\n",
xlab='lambda\n',ylab='SSE')
abline(v=opt.lambda, col = 'red')
mtext(text = paste("SSE min = ",
                   round(min(sse.vec),2), "\n lambda = ",
                   opt.lambda), side =1)
```




```r
#Fitted Data
fit1 <- HoltWinters(Final_data.df1,alpha=0.3,beta=FALSE, gamma=FALSE)
fit1
```

```
## Holt-Winters exponential smoothing without trend and without seasonal component.
##
## Call:
## HoltWinters(x = Final_data.df1, alpha = 0.3, beta = FALSE, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.3
##  beta : FALSE
##  gamma: FALSE
##
## Coefficients:
##         [,1]
## a -9.529966
```

**b. (10 points) Show the fitted values and corresponding SSE by using the lamda obtained in part a.**

```r
Final_Data_fitted<-firstsmooth(y=Final_data.df1,lambda=0.6)
data.frame(Final_Data_fitted)
```

```
##    Final_Data_fitted
```

```
## 1          18.2000000
## 2          44.4800000
## 3          13.2320000
## 4          22.1528000
## 5          24.6411200
## 6          47.4764480
## 7          36.1505792
## 8          14.1602317
## 9          -0.6959073
## 10        -20.3783629
## 11        -25.6113452
## 12        -38.9845381
## 13        -30.9538152
## 14        -38.4215261
## 15        -42.1286104
## 16        -26.7514442
## 17        -11.9005777
## 18        -10.8202311
## 19        -15.4880924
## 20        -13.7552370
## 21         -1.9020948
```

```r
Final_Data_fitted.sse<- measacc.fs(Final_Data_fitted,0.6)
kable(Final_Data_fitted.sse)
```

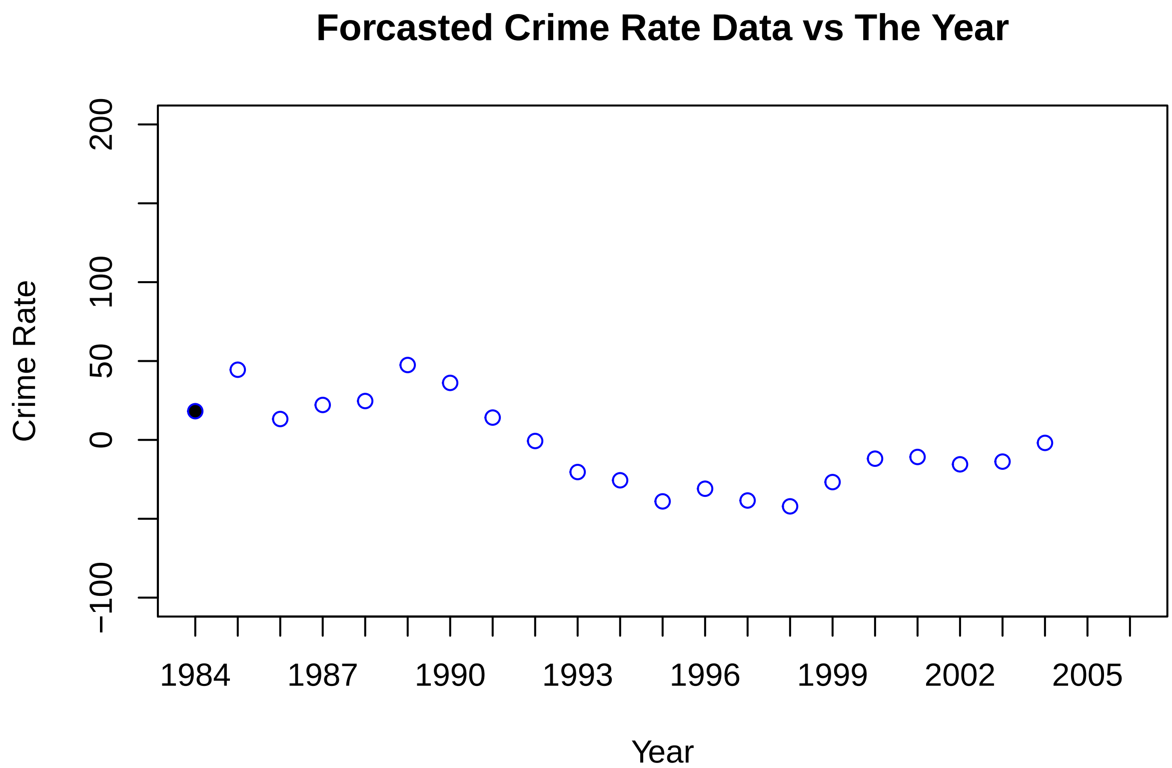
| | x |
|---|---|
| SSE | 5237.82056 |
| MAPE | 245.08845 |
| MAD | 12.36447 |
| MSD | 249.42003 |

**c. (5 points) Plot the fitted values and original values in a same plot.**

```r
#First Difference Data
Final_data_diff_all <- cbind(Final_data[,1],Final_data[,2], Final_data.df1)
```

```
## Warning in cbind(Final_data[, 1], Final_data[, 2], Final_data.df1): number of
## rows of result is not a multiple of vector length (arg 3)
```

```r
Final_data_diff_all <-as.data.frame(Final_data_diff_all)
Final_data_diff_all$V2 <- NULL


plot(Final_data_diff_all[1:T,2], type='p', pch=16, cex=1,xlim=c(1,23),xaxt='n',ylim=c(-100,200),
     xlab='Year',xaxt= 'n', ylab='Crime Rate',
     main= "Forcasted Crime Rate Data vs The Year")
axis(1, seq(1,23,1), Final_data_diff_all[seq(1,23,1),1])
points(1:21,Final_Data_fitted, lty =2, col= "blue")
```




**d. (5 points) Assume the first-difference data shows a trend. Calculate the SSE. You can get it from the HoltWinters() function. Then compare the SSE with that of obtained in part b. What can you tell from the comparison?**

```
fit1_trending <- HoltWinters(Final_data.df1,alpha=0.3,beta=0.3, gamma=FALSE)
fit1_trending$fitted
```

```
## Time Series:
## Start = 3
## End = 21
## Frequency = 1
##          xhat      level       trend
##  3 105.800000  62.000000  43.8000000
##  4 105.374000  71.780000  33.5940000
##  5 108.831140  82.191800  26.6393400
##  6 103.283335  84.071798  19.2115374
##  7 106.667372  91.108335  15.5590372
##  8  91.780134  83.247160   8.5329737
##  9  64.323856  64.096094   0.2277617
## 10  35.331314  41.846699  -6.5153853
## 11   1.971716  14.681919 -12.7102036
## 12 -22.856457  -7.349799 -15.5066580
## 13 -48.130097 -30.369520 -17.7605769
## 14 -57.103936 -41.371068 -15.7328682
## 15 -67.492269 -52.992755 -14.4995139
## 16 -73.063798 -60.624588 -12.4392097
```

```
## 17 -63.443127 -56.094659  -7.3484679
## 18 -46.828775 -45.010189  -1.8185865
## 19 -34.323139 -35.810143   1.4870032
## 20 -26.704112 -29.606198   2.9020858
## 21 -18.301422 -22.472878   4.1714558
```

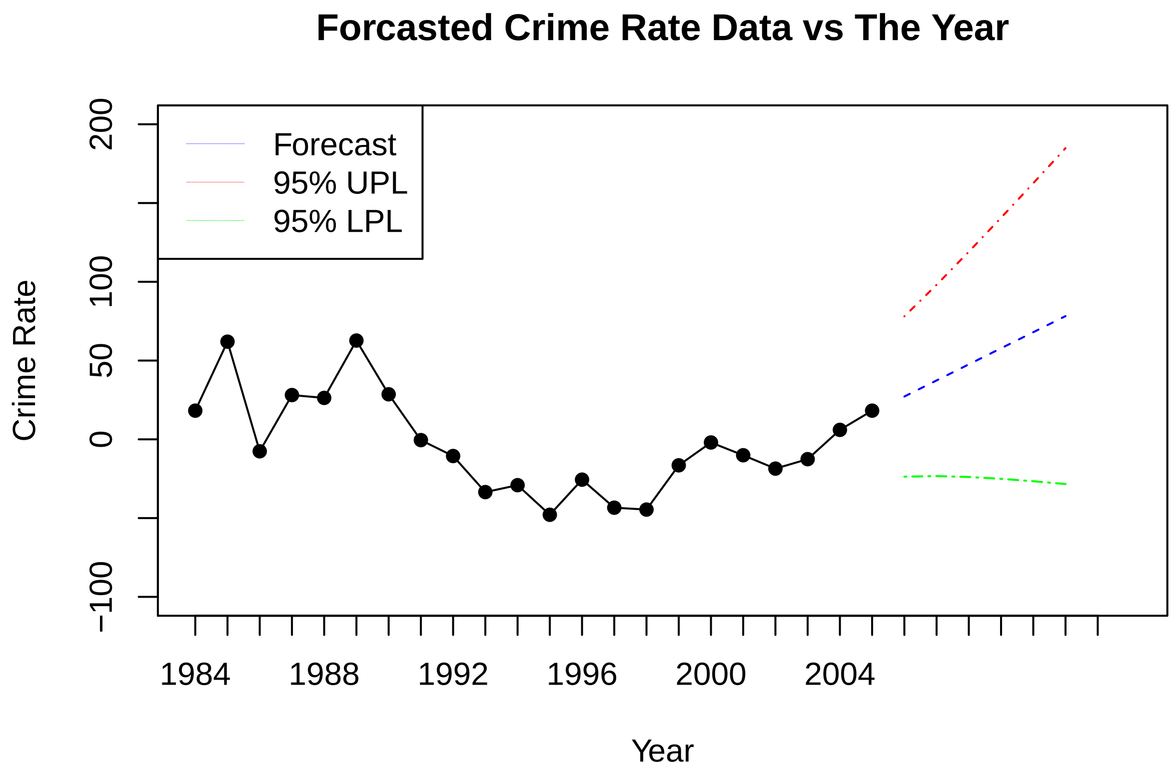
```
fit1_trending$SSE
```

```
## [1] 64423.21
```

**e. (5 points) Suppose the first-difference is a constant process. Give the forecasts of the crime rate for years from 2006 to 2010.**

**Answer**

```
# Forecast Values
lcpi<-0.6
cpi.smooth1<-firstsmooth(y=Final_data_diff_all[1:22,2],lambda=lcpi)
cpi.smooth2<-firstsmooth(y=cpi.smooth1,lambda=lcpi)
cpi.hat<-2*cpi.smooth1-cpi.smooth2
tau<-1:6
T<-length(cpi.smooth1)
cpi.forecast<-(2+tau*(lcpi/(1-lcpi)))*cpi.smooth1[T]-(1+tau*(lcpi/
(1-lcpi)))*cpi.smooth2[T]
ctau<-sqrt(1+(lcpi/((2-lcpi)^3))*(10-14*lcpi+5*(lcpi^2)+2*
                                    tau*lcpi*(4-3*lcpi)+2*(tau^2)*(lcpi^2)))
alpha.lev<-.05
sig.est<- sqrt(var(Final_data_diff_all[2:22,2]- cpi.hat[1:21]))
cl<-qnorm(1-alpha.lev/2)*(ctau/ctau[1])*sig.est


#plot forecast model with prediction intervals
plot(Final_data_diff_all[1:T,2], type='o', pch=16, cex=1,xlim=c(1,30),xaxt='n',ylim=c(-100,200),
     xlab='Year',xaxt= 'n', ylab='Crime Rate',
     main= "Forcasted Crime Rate Data vs The Year")
axis(1, seq(1,29,1), Final_data_diff_all[seq(1,29,1),1])
lines(23:28,cpi.forecast, lty =2, col= "blue")
lines(23:28,cpi.forecast+cl, lty =4, col = "red")
lines(23:28,cpi.forecast-cl, lty =6, col = "green")
legend( x="topleft",
        legend=c("Forecast","95% UPL","95% LPL"),
        col=c("blue","red","green"), lwd=.1, lty=c(2,4,6,NA),
        pch=c(NA,NA,NA,15), merge=FALSE, cex= 1)
```




**8. a. (10 points) Develop an appropriate ARIMA model and a procedure for forecasting for the crime rate data. Specify the model and estimated parameters in the model. Hint: You can use the auto.arima() and forecast() functions to answer this question.**

**Answer**

```
#Using auto.arima function
Final_Dataset_AutoArimaModel<-auto.arima(Final_data[,2])
Final_Dataset_AutoArimaModel
```

```
## Series: Final_data[, 2]
## ARIMA(1,2,0)
##
## Coefficients:
##           ar1
##       -0.4533
## s.e.   0.2091
##
## sigma^2 estimated as 623.5:  log likelihood=-92.33
## AIC=188.67   AICc=189.37   BIC=190.66
```

```
Final_Dataset_AutoArimaModel_Forecast<-as.array(forecast(Final_Dataset_AutoArimaModel))
kable(Final_Dataset_AutoArimaModel_Forecast)
```

| | Point Forecast | Lo 80 | Hi 80 | Lo 95 | Hi 95 |
|---|---|---|---|---|---|
| 23 | 466.7685 | 434.76854 | 498.7685 | 417.82878 | 515.7082 |
| 24 | 468.1591 | 409.22121 | 527.0969 | 378.02140 | 558.2967 |
| 25 | 467.8171 | 373.55509 | 562.0790 | 323.65581 | 611.9783 |
| 26 | 468.2604 | 334.67715 | 601.8437 | 263.96243 | 672.5584 |
| 27 | 468.3478 | 290.70970 | 645.9859 | 196.67377 | 740.0218 |
| 28 | 468.5965 | 243.08759 | 694.1055 | 123.71037 | 813.4827 |
| 29 | 468.7721 | 191.67339 | 745.8709 | 44.98618 | 892.5581 |
| 30 | 468.9809 | 136.90442 | 801.0573 | -38.88624 | 976.8480 |
| 31 | 469.1746 | 78.89640 | 859.4528 | -127.70440 | 1066.0536 |
| 32 | 469.3751 | 17.85539 | 920.8949 | -221.16472 | 1159.9150 |

**b. (5 points) Compare the AIC obtained from part a with that of obtained from ARIMA(0,1,0) model. Which model has a smaller AIC? What can you tell by this comparison?**

**Answer**

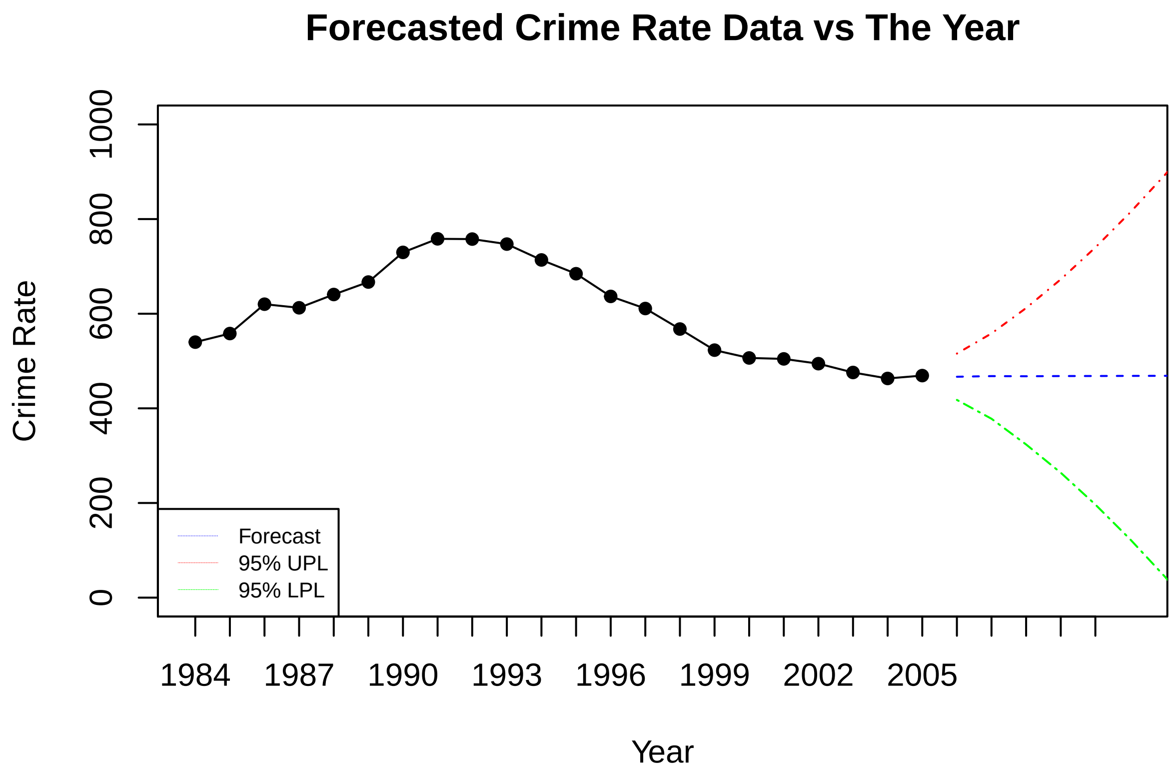
```
# AR(1) model create with code ARIMA (0,1,0)
Final_data.arima<-arima(Final_data[,2],order=c(0, 1, 0))
Final_data.arima
```

```
##
## Call:
## arima(x = Final_data[, 2], order = c(0, 1, 0))
##
##
## sigma^2 estimated as 966:  log likelihood = -101.97,  aic = 205.93
```

**c. (5 points) Show the 1- to 5- step ahead forecasts and corresponding 95% prediction intervals for the crime rate. Show only the results/outputs. Calculation process or formula are not required.**

**Answer**

```
plot(Final_data[,2],type="o",pch=19,cex=0.8,
     xlab='Year',xaxt= 'n', ylab='Crime Rate',
     xlim=c(1,28),xaxt='n', ylim=c(0,1000),
     main= "Forecasted Crime Rate Data vs The Year")
axis(1, seq(1,27,1), Final_data[seq(1,27,1),1])
lines(Final_Dataset_AutoArimaModel_Forecast$mean,col="blue", lty= 2)
lines(Final_Dataset_AutoArimaModel_Forecast$upper[,2], col="red", lty= 4)
lines(Final_Dataset_AutoArimaModel_Forecast$lower[,2], col="green",lty= 6)
legend( x="bottomleft",
        legend=c("Forecast","95% UPL","95% LPL"),
        col=c("blue","red","green"), lwd=.1, lty=c(2,4,6,NA),
        pch=c(NA,NA,NA,15), merge=FALSE, cex= .7 )
```


Forecasted Crime Rate Data vs The Year
Crime Rate
0
200
400
600
800
1000
1984
1987
1990
1993
1996
1999
2002
2005
Year
Forecast
95% UPL
95% LPL