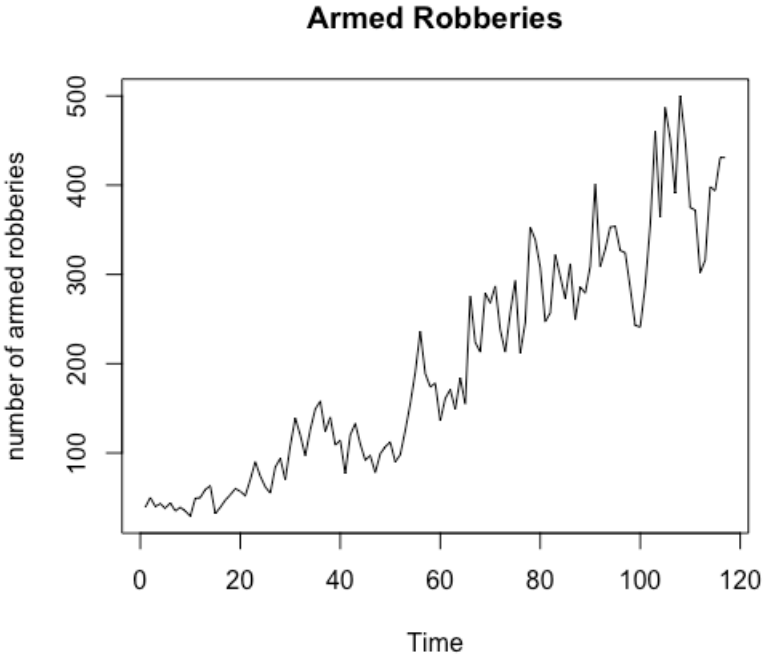
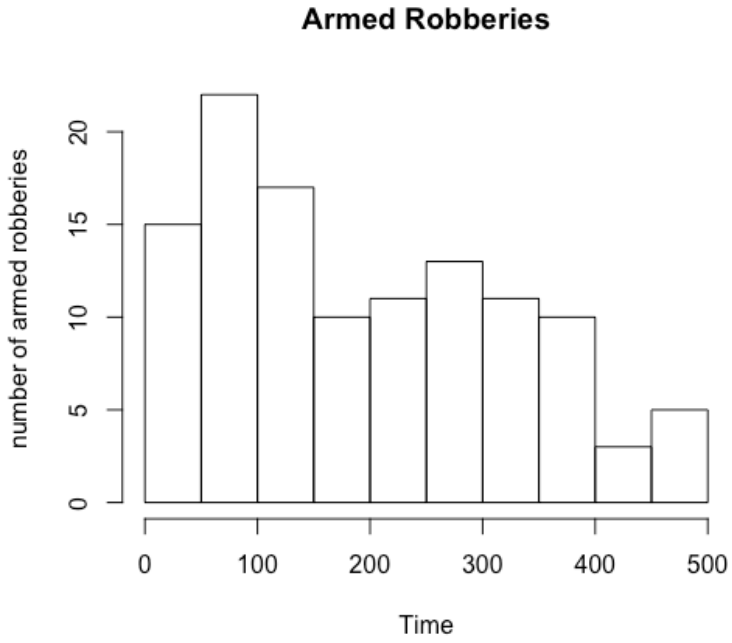
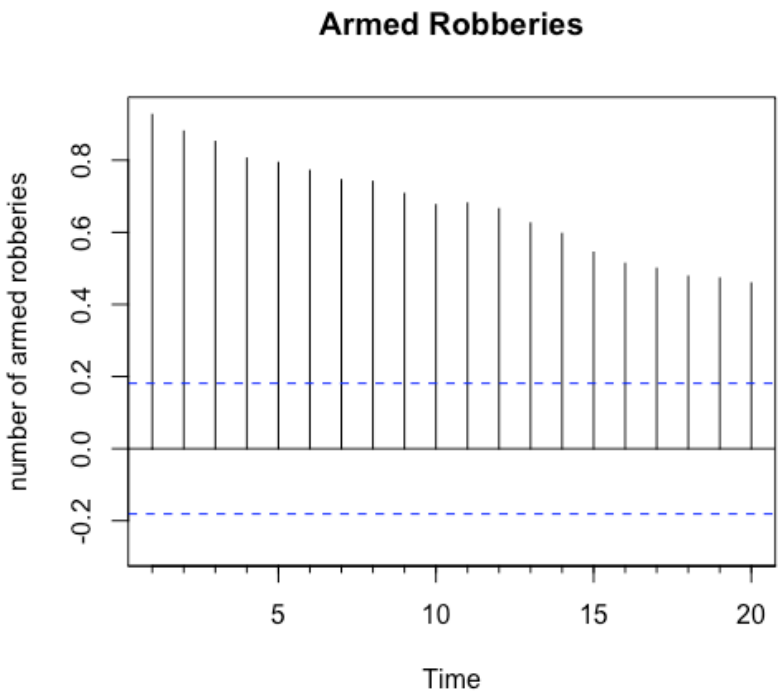
1. **Use graphical techniques to inspect the data. Describe the behavior of the data. Mention any features you think may be important for analysis and forecasting.**







From ACF plots, we show that all are significant and all are decreasing slowly. From all of the others graphs, we see that there is a definite trend, and unequal variance although there seems to be a seasonal component. Also, the histogram does not look like a Gaussian curve.

For analysis and forecasting, we need to either transform the data or difference it or both. We do this for getting no trend and we need stable variance. Also, we are trying to get the histogram to have a Gaussian curve. That is what we are trying to do in the rest of the steps.

2. **In what way(s) is the data not stationary? Use transformations and/or differencing to make the series stationary. Include a time plot of the transformed and/or differenced data. If you chose a transformation, use this transformation for the remainder of the steps.**

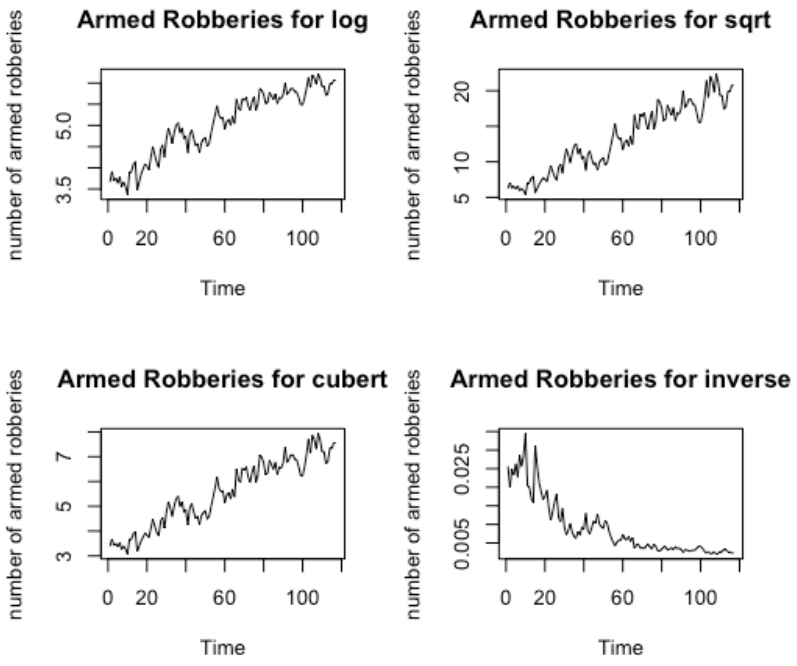
The data is not stationary because of:

- Unequal variance
- Definite trend

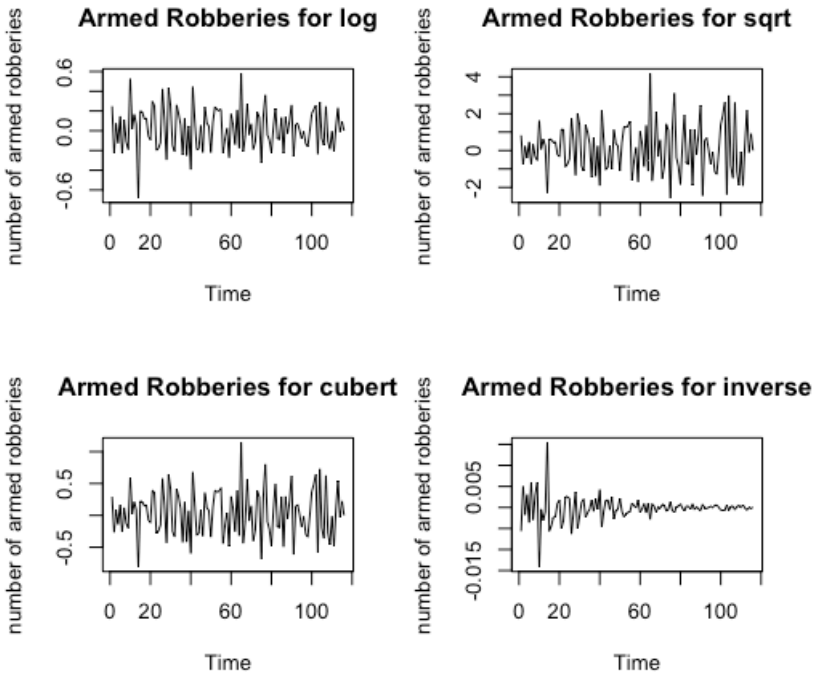
From the above ACF plots, we show that all are significant and all are decreasing slowly. Hence, shows a trend. However, there seems to be a seasonal component. Also, the histogram does not look like a Gaussian curve. **Hence, we are first trying to transform the data in the hopes of getting time series similar to white noise.**

We see that none of the transformation time plots look stationary because of unstable variance and trend.

TRANSFORMATION:



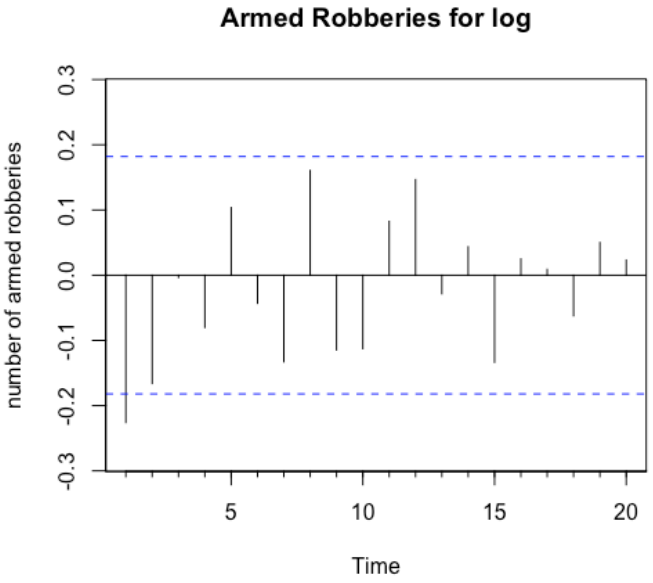
DIFFERENCING:



We see that the difference of log transformation looks the most stationary as it has almost stable variance and no trend. It looks similar to the time series plot of white noise. **So we are going to use difference of log transformation for the remainder of the steps.**

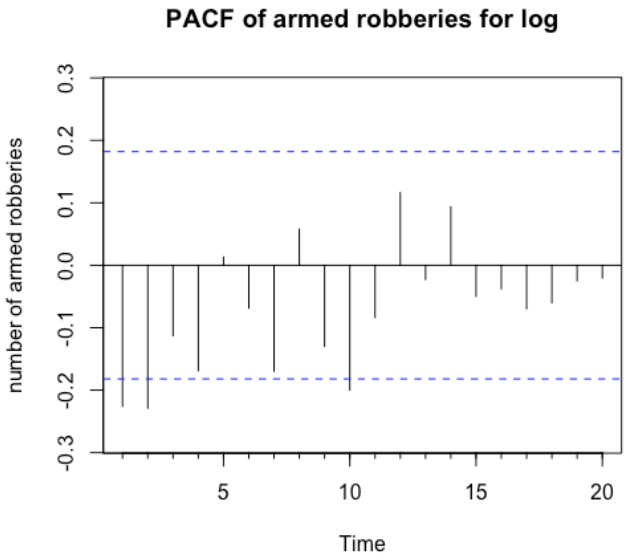
3. **Using the transformed and/or differenced data from the previous part, obtain the sample ACF and PACF plots, as well as plots of the raw periodogram, and its smoothed version. Comment on the plots. Use the plots to make a preliminary guess for an appropriate ARIMA model. Keep in mind that differencing plays a part in determining whether the model should be ARMA or ARIMA.**

ACF PLOTS:



From this ACF plot, we see that there is only one value that is statistically significant (outside the bounds). **Hence, q=1 in the ARIMA(p,d,q) model.**

PACF PLOTS:

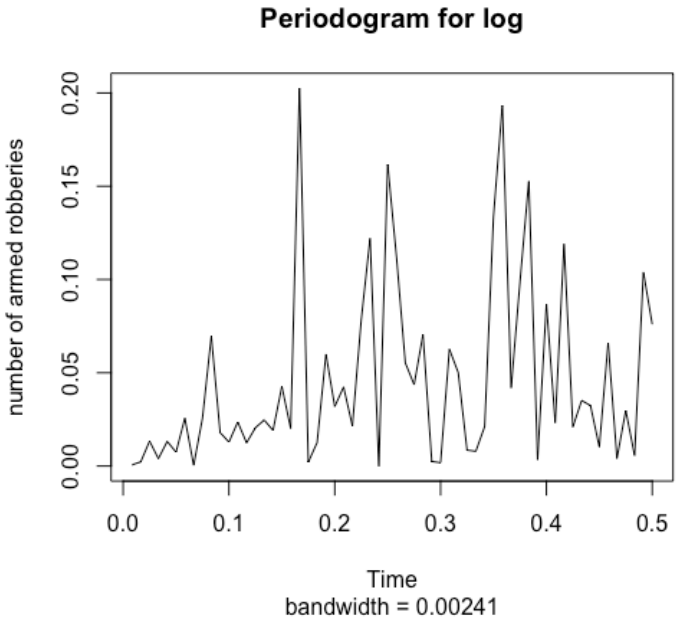


From this PACF plot, we see that there are three values that are statistically significant (outside the bounds). **Hence, p=3 in the ARIMA(p,d,q) model.**

Also by using the command ndiffs() for log transformation, we see that d=1 (meaning only one differencing is required).

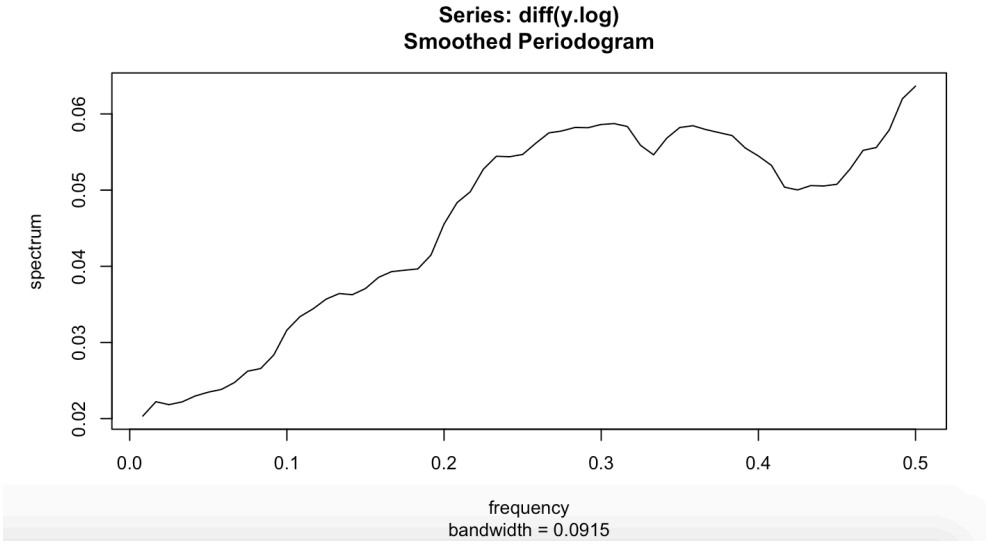
Hence, <u>the ARIMA(p,d,q) model = ARIMA(3,1,1)</u>

PERIODOGRAM:



From this periodogram, we see that there are a few peaks with dominant frequencies. Hence, it looks like there are seasonal components. To try to remove this seasonality, we try getting the smooth version of this.

SMOOTHED VERSION:



From this smooth periodogram, we see that there are no high peaks. It looks pretty smooth overall. Hence, it now looks like there are hardly any seasonal components.

4. Fit the model from the previous step. Include the model, and the parameter estimates. Plot the fitted values and the observed values on the same plot.

<u>MODEL OUTPUT:</u>

Call:
arima(x = y.log, order = c(3, 1, 1))

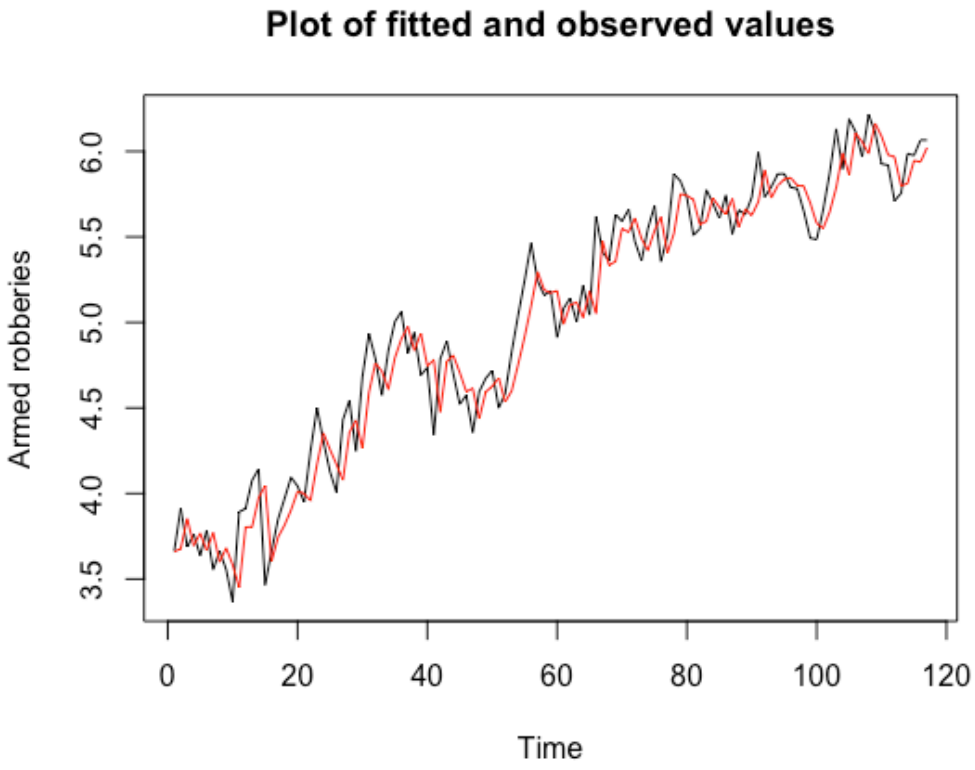
Coefficients:

	ar1	ar2	ar3	ma1
	0.3361	-0.0764	0.0245	-0.6292
s.e.	0.3307	0.1323	0.1355	0.3161

sigma^2 estimated as 0.03868: log likelihood = 23.95, aic = -37.9

PARAMETER ESTIMATES:

ar1	ar2	ar3	ma1
0.33613086	-0.07639892	0.02451413	-0.62917384

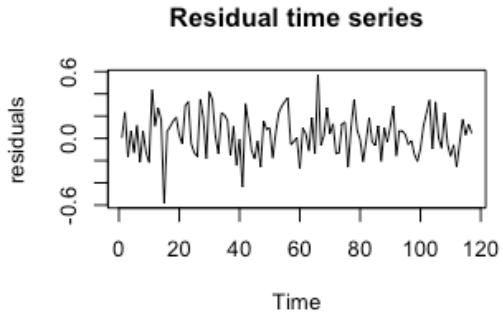


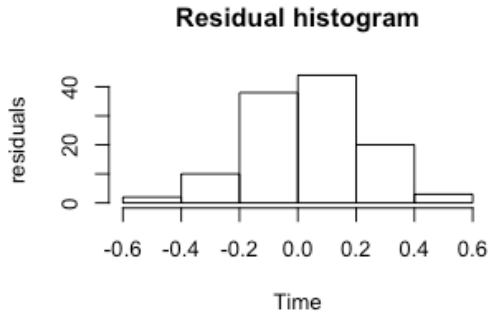
Observed values - black, fitted values – red

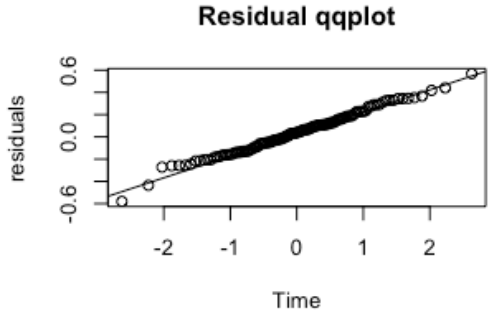
We see that both the fitted and observed values are pretty similar as seen in the above plot.

5. **Examine the residuals. Provide necessary plots and/or hypothesis test results. Do the residuals resemble Gaussian white noise?**

From the residual time series graph below, we see that it almost resembles the time series graph for Gaussian white noise. Also, the histogram looks almost like a Gaussian curve. Lastly, the residual qqplot has an approximate normal distribution as most of the residuals are on the qqline. **Hence, we can conclude that the residuals resemble Gaussian white noise.**







6. Use AICc to select an ARIMA model for the (possibly transformed) data. Keep in mind that differencing should be incorporated into the model. It is fine to use the function auto.arima() here. It is enough to consider p = 0,⋯ ,8, q = 0,⋯ ,8, and d = 0,1,2. Include the chosen model, and provide parameter estimates and their standard errors.

ARIMA MODEL USING AICc:

Series: diff(y.log)
ARIMA(2,0,2) with non-zero mean

Coefficients:

	ar1	ar2	ma1	ma2	mean
	-0.1206	0.3294	-0.2213	-0.6149	0.0213
s.e.	0.3947	0.2650	0.3749	0.3496	0.0040

sigma^2 estimated as 0.0375: log likelihood=28.03
AIC=-44.07 AICc=-43.3 BIC=-27.55

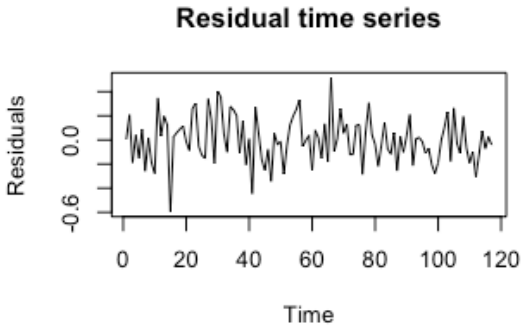
PARAMETER ESTIMATES:

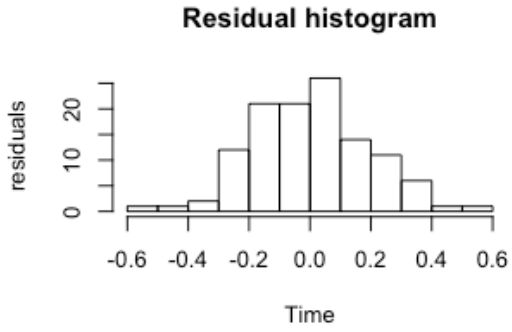
ar1	ar2	ma1	ma2	intercept
-0.12058663	0.32942322	-0.22133571	-0.61486659	0.02131093

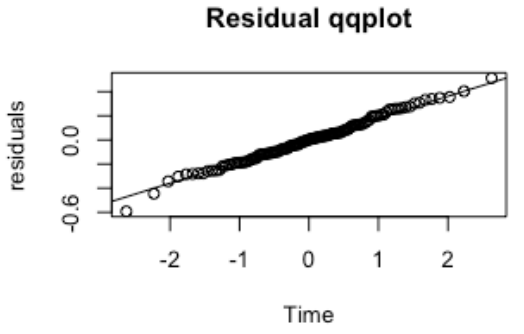
STANDARD ERRORS:

	ar1	ar2	ma1	ma2	mean
s.e.	0.3947	0.2650	0.3749	0.3496	0.0040

7. **Inspect the residuals of this model. Provide necessary plots and/or hypothesis test results. Do the residuals resemble Gaussian white noise?**



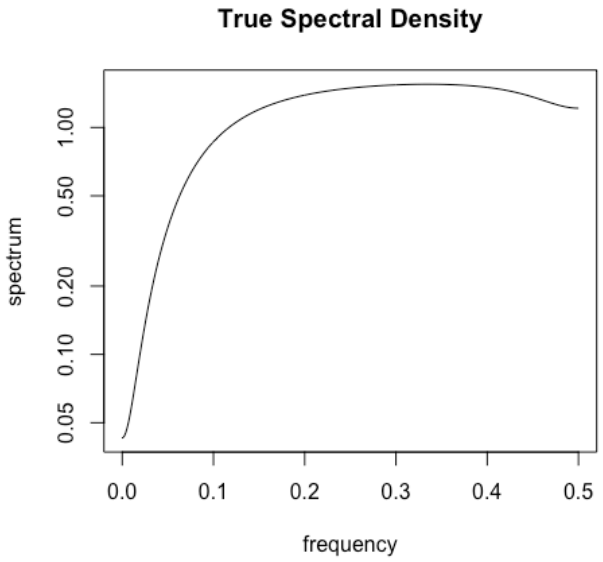


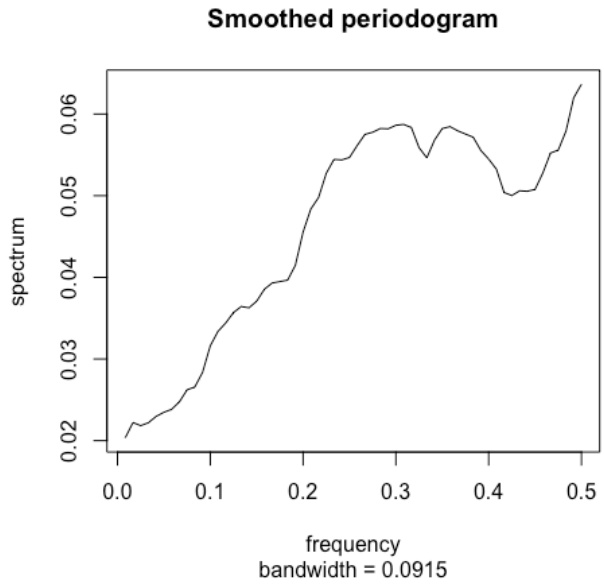


From the residual time series graph, we see that it almost resemble the time series graph for Gaussian white noise. Also, the histogram looks almost like a Gaussian curve. Lastly, the residual qqplot has an approximate normal distribution as most of the residuals are on the qqline. **Hence, we can conclude that the residuals likely resemble Gaussian white noise.**

8. **Plot the (theoretical) spectral density of the final model together with the smoothed periodogram. Comment on the plots. Describe the method you chose for smoothing the periodogram.**

From the true (theoretical) spectral density periodogram, we see that it is already smooth. There are no peaks or dominant frequencies thus indicating no seasonality. Hence, I do not really see the point of smoothing. But after smoothing, there are still no dominant frequencies or high/low peaks. Hence, there is most likely no seasonal component.



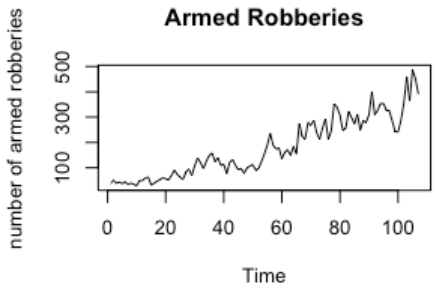


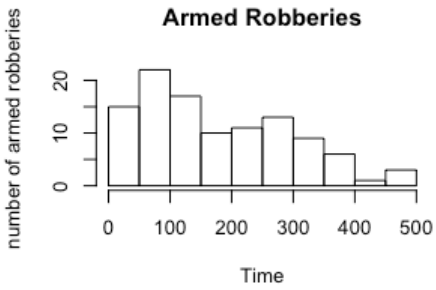
METHOD TO SMOOTH THE PERIODOGRAM:

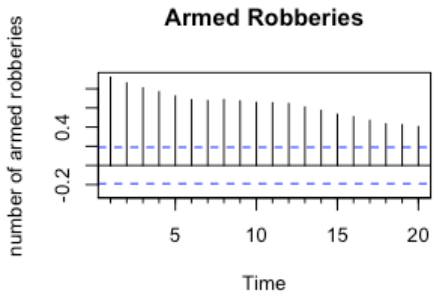
- First, we simulated data.
- Second, we got the raw periodogram values at the **Fourier frequencies**.
- Third, we chose a vector of candidate L values for smoothing.
- Fourth, we chose a vector to store criterion values for each L.
- Fifth, we went through the L values and computed Q for each.
- Lastly, we used the function spec.pgram(⋯) to smooth the periodogram with the spans= L.

9. Now remove the data for 1975 (the last 10 observations). Using only data from 1966-1975 (the first 108 observations) fit an ARIMA model using AICc. Again, it is fine to use auto.arima(). Then do the following.

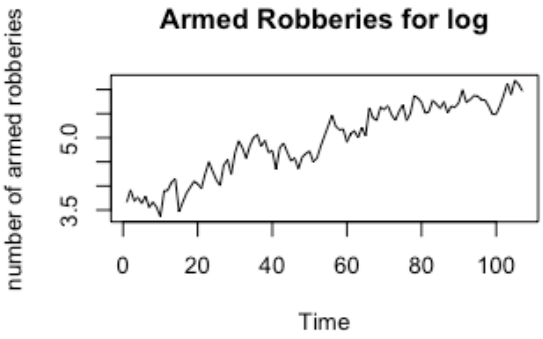
To fit an ARIMA model, we first need to see if we need to do any kind of transformation or differencing. Hence, the first step is to check if the original data (in this case everything in the data except the last 10 observations) is stationary or not**. We see from the below plots that the data is not stationary as it has unequal variance, non-Gaussian curve for histogram and from ACF plot, we see that all lags are significant.**

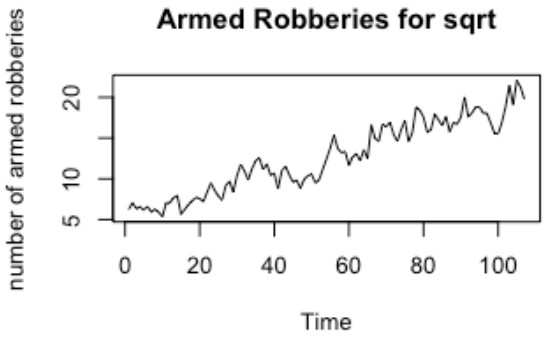


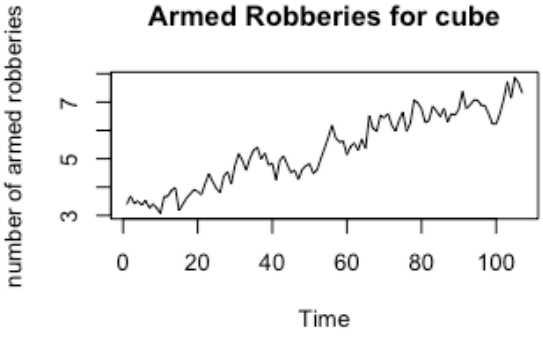


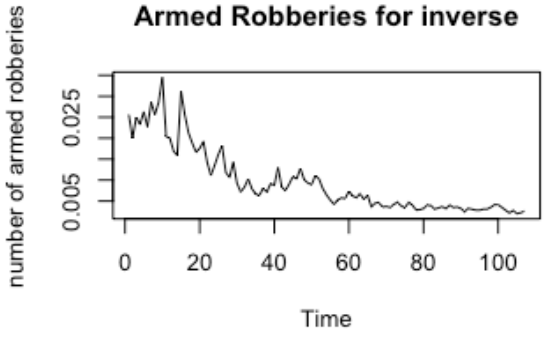


STEP 2: NOW WE HAVE TO TRANSFORM THE DATA AND CHECK FOR STATIONARITY







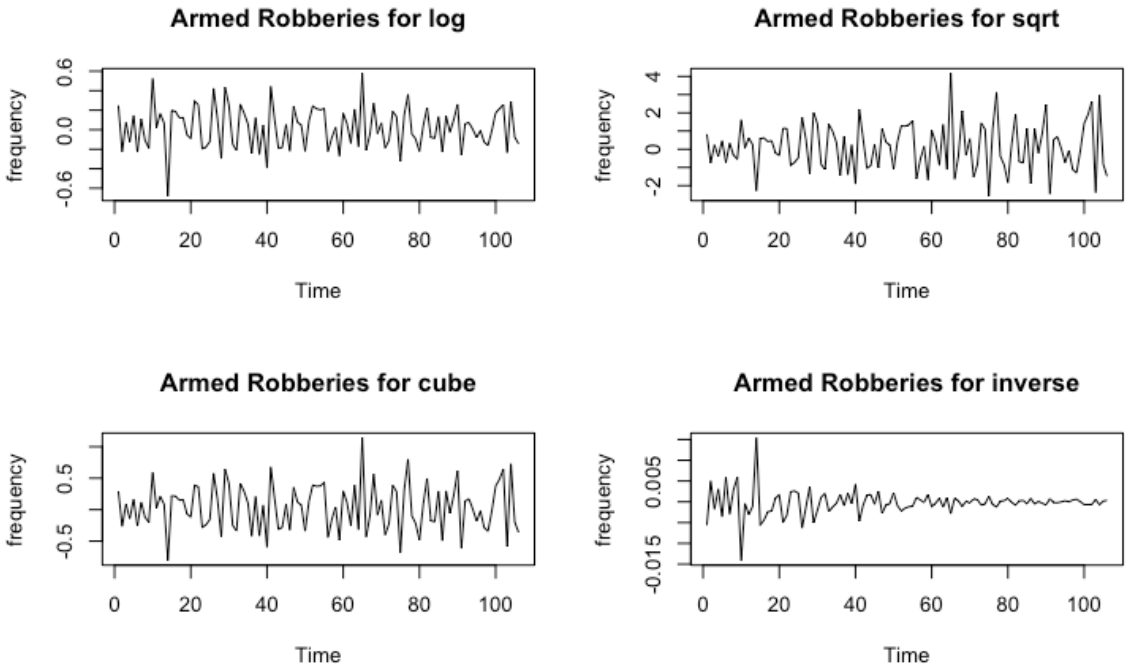


We see that none of these transformed data still look stationary because of the unequal variance and trend.

STEP 3: CHECK THE VALUE OF ndiffs() FUNCTION:

We see that for all the above transformations, the difference is one. Hence, only one differencing is required (d=1 in the ARIMA model).

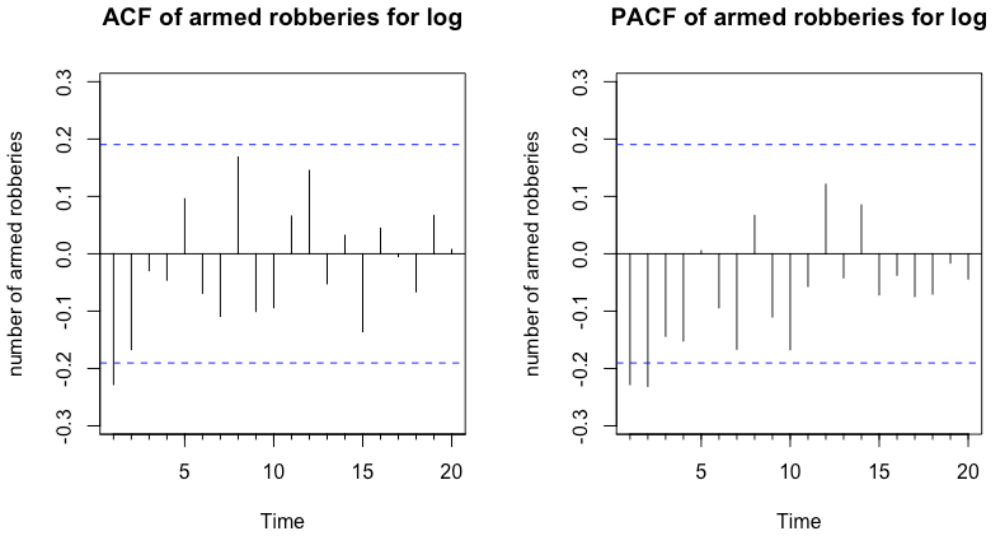
STEP 4: DIFFERENCING THE TRANSFORMED DATA



From the above time series plots, we can see that **difference of the log is the most stationary** looking one, as it closely resembles the time series plot for Gaussian white noise. **So, difference of the log transformation is what we are going to use for this problem.**

STEP 5: CHECK ACF AND PACF PLOTS:

From the below ACF and PACF plots, we can see that q=1 and p=2 for the ARIMA(p,d,q) model. Also, d=1 from the ndiffs() function used above (only one differencing). **Hence, model I have got is ARIMA(2,1,1) from ACF and PACF plots.**



PART A: Write down the chosen model. Include parameter estimates.

Series: y_new.log

ARIMA(0,1,2) with drift #We see that the model from auto.arima() function is different

Coefficients:

	ma1	ma2	drift
	-0.3755	-0.2708	0.0224
s.e.	0.0946	0.1047	0.0069

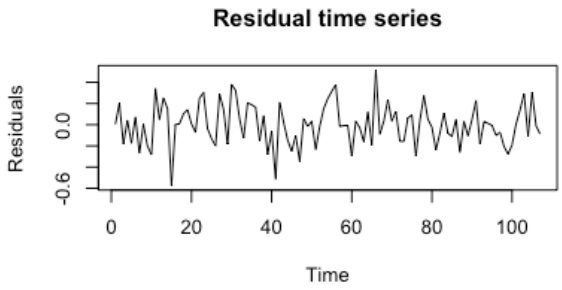
sigma^2 estimated as 0.03906: log likelihood=22.75

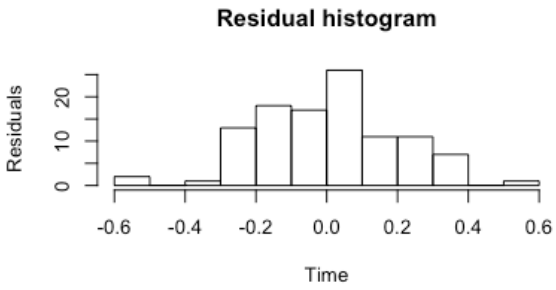
AIC=-37.49 AICc=-37.1 BIC=-26.84

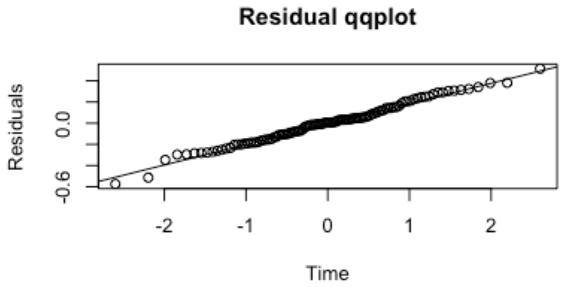
PARAMETER ESTIMATES:

ma1	ma2	drift
-0.37551128	-0.27076858	0.02238572

PART B: Inspect the residuals of this model. Do they resemble Gaussian white noise?







From the residual time series graph, we see that it almost resembles the time series graph for Gaussian white noise. Also, the histogram has an approximate looking Gaussian curve. Lastly, the residual qqplot has an approximate normal distribution as most of the residuals are on the qqline. **Hence, we can conclude that the residuals most likely resemble Gaussian white noise.**

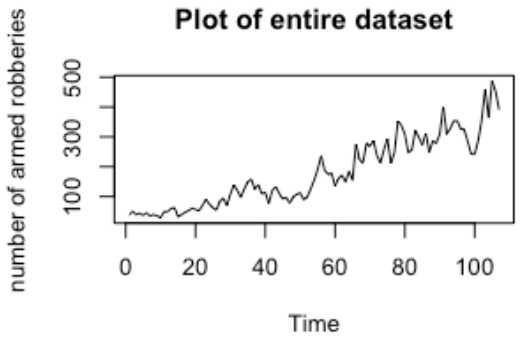
PART C: Compute point forecasts of the values for January through October 1975. If you used a transformation, then be sure to compute forecasts of the *original* data, not the *transformed* data.

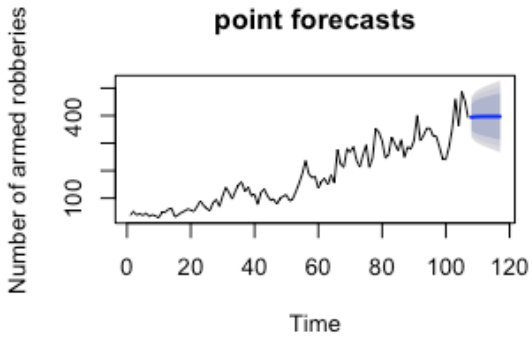
	Point Forecast	Lo 95	Hi 95
(Jan 1975) 108	394.4077	321.7554	467.0601
(Feb 1975) 109	395.8235	308.3121	483.3349
(Mar 1975) 110	396.4117	300.6570	492.1663
(Apr 1975) 111	396.6560	294.7915	498.5206
(May 1975) 112	396.7576	289.6697	503.8454
(Jun 1975) 113	396.7997	284.9393	508.6601

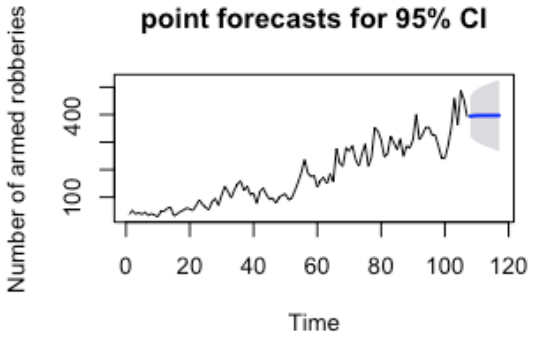
(July 1975) 114	396.8173	280.4614	513.1731
(Aug 1975) 115	396.8245	276.1731	517.4759
(Sep 1975) 116	396.8276	272.0414	521.6137
(Oct 1975) 117	396.8288	268.0458	525.6119

This table gives the point forecast values as well as the 95% confidence interval range of the low and high forecast values. For example: In Jan 1975, there is a 95% chance that the forecast value lays in between (321.7554, 467.0601).

PART D: Make a time plot of the entire dataset, the point forecasts, and 95% prediction intervals.

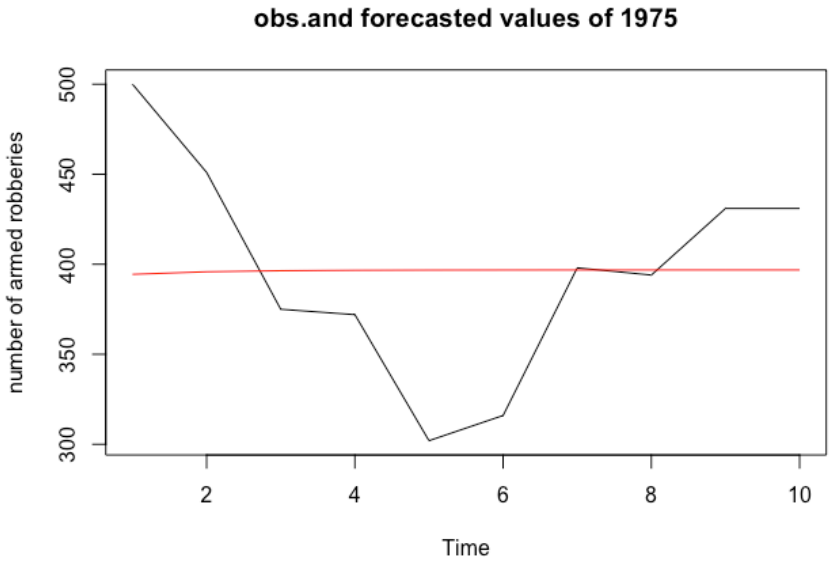






Make another plot of just the observed values from 1975 along with the point forecasts and the prediction intervals. Comment on the forecast performance.

We see from the below graph that the forecast performance is very bad because the observed values differs greatly from the point forecasts.



CODE APPENDIX:

```
par(mfrow=c(1,1))
library(forecast)
data=read.table("bostonArmedRobberies.txt",header = T)
names(data)
y = data$X41 #number of armed robberies
y
a=data$X1966.01 #years
a
time = 1:length(y)
time
ts.plot(y,xlab="Time",ylab="number of armed robberies",main="Armed
Robberies")
hist(y,xlab="Time",ylab="number of armed robberies",main="Armed
Robberies")
Acf(y,xlab="Time",ylab="number of armed robberies",main="Armed
Robberies")

#TRANSFORMATION
y.log = log(y)
ts.plot(y.log,xlab="Time",ylab="number of armed robberies",main="Armed
Robberies for log")
y.sqrt = sqrt(y)
ts.plot(y.sqrt,xlab="Time",ylab="number of armed robberies",main="Armed
Robberies for sqrt")
y.cubert = y^(1/3)
ts.plot(y.cubert,xlab="Time",ylab="number of armed
robberies",main="Armed Robberies for cubert")
#inv=ts.plot(y.inv)
y.inv = 1/y
ts.plot(y.inv,xlab="Time",ylab="number of armed robberies",main="Armed
Robberies for inverse")

#apply differencing
ts.plot(diff(y.log),xlab="Time",ylab="number of armed
robberies",main="Armed Robberies for log") #either this or cubert
```

```
ts.plot(diff(y.sqrt),xlab="Time",ylab="number of armed
robberies",main="Armed Robberies for sqrt")
ts.plot(diff(y.cubert),xlab="Time",ylab="number of armed
robberies",main="Armed Robberies for cubert")
ts.plot(diff(y.inv),xlab="Time",ylab="number of armed
robberies",main="Armed Robberies for inverse") #nope

#Checking p and q values for ARIMA model
par(mfrow=c(1,1))
Acf(diff(y.log),xlab="Time",ylab="number of armed
robberies",main="Armed Robberies for log") ##q=1
Acf(diff(y.sqrt),xlab="Time",ylab="number of armed
robberies",main="Armed Robberies for sqrt")
Acf(diff(y.cubert),xlab="Time",ylab="number of armed
robberies",main="Armed Robberies for cubert")
Acf(diff(y.inv),xlab="Time",ylab="number of armed
robberies",main="Armed Robberies for inverse")

Pacf(diff(y.log),xlab="Time",ylab="number of armed
robberies",main="PACF of armed robberies for log") ##p=3
Pacf(diff(y.sqrt),xlab="Time",ylab="number of armed
robberies",main="Armed Robberies for sqrt")
Pacf(diff(y.cubert),xlab="Time",ylab="number of armed
robberies",main="Armed Robberies for cubert")
Pacf(diff(y.inv),xlab="Time",ylab="number of armed
robberies",main="Armed Robberies for inverse")

#PERIODOGRAM
spec.pgram(diff(y.log),log="no",xlab="Time",ylab="number of armed
robberies",main="Periodogram for log") #only this one is needed.
spec.pgram(diff(y.sqrt),log="no",xlab="Time",ylab="number of armed
robberies",main="Armed Robberies for sqrt")
spec.pgram(diff(y.cubert),log="no",xlab="Time",ylab="number of armed
robberies",main="Armed Robberies for cubert")
spec.pgram(diff(y.inv),log="no",xlab="Time",ylab="number of armed
robberies",main="Armed Robberies for inverse")


#Checking d value in ARIMA model
```

```
ndiffs(y.log) #1
ndiffs(y.sqrt) #1
ndiffs(y.cubert) #1
ndiffs(y.inv) #1

#Hence, the model that I get is an ARIMA(3,1,1) model
spec.pgram(diff(y.log),log="no") ##We don't want the log version

library(astsa)
fitARIMA31$coef
arma.spec(ar=c(0.33613086,-0.07639892),ma=c(-0.62917384), main =
'True Spectral Density') ## For ARIMA(3,1,1) model

#PART 4
fitARIMA31 = arima(y.log,order=c(3,1,1))
fitARIMA31
coeff=fitARIMA31$coef
coeff
mean(diff(y.log))
log_fit = fitted(fitARIMA31)
plot(x=time[1:length(y)], y=y.log, type='l',xlab = "Time",ylab="Armed
robberies",main="Plot of fitted and observed values") #observed values
points(x=time[1:length(y)], y=log_fit, type='l', col='red') #fitted values

#PART 5
fitARIMA31 = arima(y.log,order=c(3,1,1))
res31= fitARIMA31$res
res31
par(mfrow=c(2,2))
plot(res31,xlab="Time",ylab="residuals",main="Residual time series")
hist(res31,xlab="Time",ylab="residuals",main="Residual histogram")#
Looks like a Gaussian curve
qqnorm(res31,xlab="Time",ylab="residuals",main="Residual qqplot")
#Most of the residuals are on the line
qqline(res31)

#PART 6
aiccmodel=auto.arima(diff(y.log), max.P=8, max.Q=8, max.d =2, ic="aicc" )
aiccmodel
```

```
aiccmodel$coef
aiccmodel$residuals

#PART 7
par(mfrow=c(2,2))
ARIMA_best=auto.arima(y.log)
res=ARIMA_best$residuals
res
plot(res,xlab="Time",ylab="Residuals",main="Residual time series")
hist(res,xlab="Time",ylab="residuals",main="Residual histogram")# Looks
like a Gaussian curve
qqnorm(res,xlab="Time",ylab="residuals",main="Residual qqplot") #Most
of the residuals are on the line
qqline(res)


#PART 8
par(mfrow=c(1,2))
#(theoretical) spectral density-fitted final model, auto.arima one
arma.spec(ar=c(-0.12058673,0.32942273),ma=c(-0.22133552,-
0.61486640), main = 'True Spectral Density') ##correct
#spec.pgram(diff(y.log),log="no")
# simulate data
n = length(y)
m = floor(n/2)
x =diff(y.log)

# get the raw periodogram values at the Fourier frequencies
pgrm.raw = spec.pgram(x, plot=F)$spec
pgrm.raw
# vector of candidate L values for smoothing
spans = (1:(m-1))*2+1
spans
# vector to store criterion values for each L
Q = numeric(length(spans))
Q
# go through the L values and compute Q for each
for(j in 1:length(spans)){
  L = spans[j]
```

```
  pgrm.smooth = spec.pgram(x, spans=L, plot=F)$spec
  Q[j] = sum((pgrm.smooth - pgrm.raw) ^ 2) + sum((pgrm.raw)^2)/(L-1)
}
L = spans[which.min(Q)]
L
spec.pgram(diff(y.log),log="no",spans=L,xlab="frequency",ylab="spectrum",
main="Smoothed periodogram")

#PART 9
#Now remove the data for 1975 (the last 10 observations). Using only data
from 1966-1975
#(the first 108 observations), fit an ARIMA model using AICc. Again, it is
fine to use auto.arima().
#Then do the following.
#• Write down the chosen model. Include parameter estimates.
#• Inspect the residuals of this model. Do they resemble Gaussian white
noise?
#• Compute point forecasts of the values for January through October 1975.
If you used a
#transformation, then be sure to compute forecasts of the original data, not
the trans- formed data.
#• Make a time plot of the entire data set,the point forecasts, and 95%
prediction intervals.
#Make another plot of just the observed values from 1975 along with the
point forecasts and the
#prediction intervals. Comment on the forecast performance.
changed_data=data[-108:-117,]
changed_data
y_new = changed_data$X41
y_new
par(mfrow=c(2,2))
ts.plot(y_new,xlab="Time",ylab="number of armed
robberies",main="Armed Robberies") ##UNEQUAL VARIANCE
hist(y_new,xlab="Time",ylab="number of armed robberies",main="Armed
Robberies")
Acf(y_new,xlab="Time",ylab="number of armed robberies",main="Armed
Robberies")
#TRANSFORMATION
y_new.log = log(y_new)
```

```
ts.plot(y_new.log,xlab="Time",ylab="number of armed
robberies",main="Armed Robberies for log")
y_new.sqrt = sqrt(y_new)
ts.plot(y_new.sqrt,xlab="Time",ylab="number of armed
robberies",main="Armed Robberies for sqrt")
y_new.cubert = y_new^(1/3)
ts.plot(y_new.cubert,xlab="Time",ylab="number of armed
robberies",main="Armed Robberies for cube")
#inv=ts.plot(y.inv)
y_new.inv = 1/y_new
ts.plot(y_new.inv,xlab="Time",ylab="number of armed
robberies",main="Armed Robberies for inverse")

#apply differencing
ts.plot(diff(y_new.log),xlab="Time",ylab="frequency",main="Armed
Robberies for log") #THIS IS THE ONE
ts.plot(diff(y_new.sqrt),xlab="Time",ylab="frequency",main="Armed
Robberies for sqrt")
ts.plot(diff(y_new.cubert),xlab="Time",ylab="frequency",main="Armed
Robberies for cube")
ts.plot(diff(y_new.inv),xlab="Time",ylab="frequency",main="Armed
Robberies for inverse") #worst

#Checking d value in ARIMA model
ndiffs(y_new.log) #1
ndiffs(y_new.sqrt) #1
ndiffs(y_new.cubert) #1
ndiffs(y_new.inv) #1
par(mfrow=c(1,2))
Acf(diff(y_new.log),xlab="Time",ylab="number of armed
robberies",main="ACF of armed robberies for log") ##q=0

Pacf(diff(y_new.log),xlab="Time",ylab="number of armed
robberies",main="PACF of armed robberies for log") ##p=2

model=auto.arima(y_new.log)
model ##ARIMA(0,1,2)
coeff_new=model$coef
coeff_new
```

```
#PART B
par(mfrow=c(2,2))
res_model=model$residuals
res_model
plot(res_model,xlab="Time",ylab="Residuals",main="Residual time series")
hist(res_model,xlab="Time",ylab="Residuals",main="Residual histogram")#
Looks like a Gaussian curve
qqnorm(res_model,xlab="Time",ylab="Residuals",main="Residual qqplot")
#Most of the residuals are on the line
qqline(res_model)

#PART C
original_model=auto.arima(y_new)
original_model
fitARIMA11 = arima(y_new,order=c(1,1,1))
fitARIMA11
prediction=forecast(fitARIMA11, h=10, level=c(80, 90, 95))
forecast(fitARIMA11, h=10, level=c(95))

#PART D
ts.plot(y_new,xlab="Time",ylab="number of armed robberies",main="Plot
of entire dataset")
plot(forecast(fitARIMA11, h=10, level=c(80, 90, 95)),xlab="Time",ylab =
"Number of armed robberies",main="point forecasts") #We see that most
#of the next values increases with time
plot(forecast(fitARIMA11, h=10, level=c(95)),xlab="Time",ylab = "Number
of armed robberies",main="point forecasts for 95% CI")

par(mfrow=c(1,1))
prediction=predict(fitARIMA11, n.ahead = 10)
prediction
point_forecast=prediction$pred
point_forecast
length(point_forecast)
#point_forecast=c(394.4077, 395.8235,
396.4117,396.6560,396.7576,396.7997,396.8173,396.8245,396.8276,396.8
288)
##OR USE predict(fitARIMA11, n.ahead = 10)
```

```
length(point_forecast)
plot(x=time[1:length(data[108:117,]$X41)], y=data[108:117,]$X41,
type='l',xlab="Time",ylab="number of armed robberies",main="obs.and
forecasted values of 1975") #observed values
points(x=time[1:length(point_forecast)], y=point_forecast, type='l',
col='red') #fitted values
#The forecast performance is very bad because the observed values differs
greatly from the point forecasts
#point_forecast=c(394.4077, 395.8235,
396.4117,396.6560,396.7576,396.7997,396.8173,396.8245,396.8276,396.8
288)
length(point_forecast)
```