chapter4

Erick onyango ooko scm223-0157/2019

2023-09-27

#SCATTERPLOT

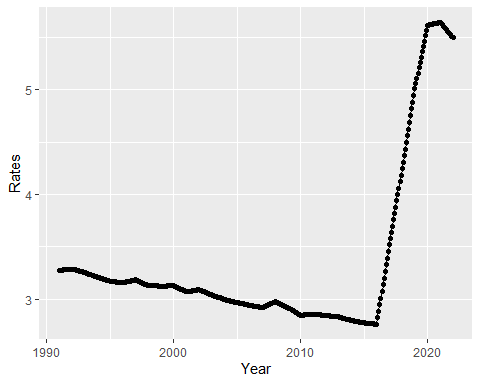
library(readxl)  
monthly\_unemp <- read\_excel("monthly\_unemp.xlsx")  
  
# loading libraries  
library(ggplot2)  
library(dplyr)

##   
## Attaching package: 'dplyr'

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

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

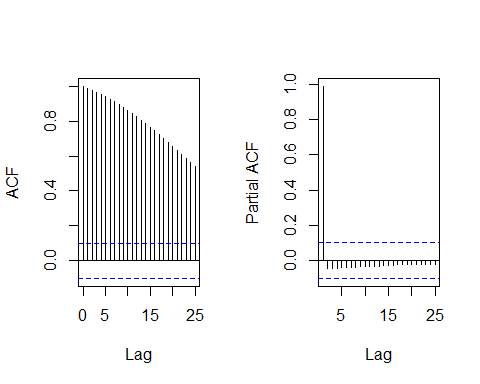
ggplot(monthly\_unemp,aes(x=Year,y=Rates))+geom\_line()+geom\_point()

 #CORELLOGRAM

library(tseries)

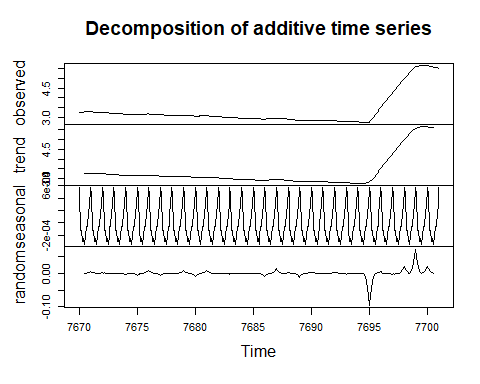
## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

par(mfrow = c(1, 2))  
acf(monthly\_unemp$Rates, main="")  
pacf(monthly\_unemp$Rates, main="")



#DECOMPOSE

library(stats)  
# Convert 'Years' column to a date format  
monthly\_unemp$Year <- as.Date(monthly\_unemp$Year)  
  
# Convert the data to a time series object  
ts\_data <- ts(monthly\_unemp$Rates, frequency = 12, start = c(min(monthly\_unemp$Year)))  
  
#decompose the ts  
decomposed\_data<-decompose(ts\_data)  
  
plot(decomposed\_data)



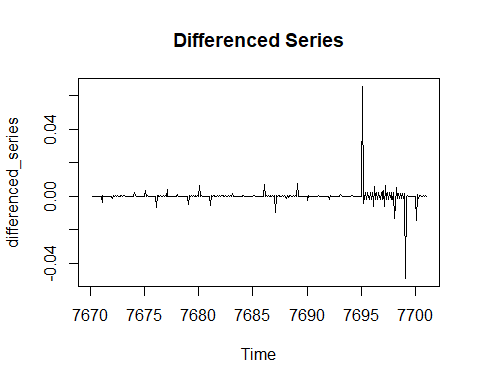
# STATIONARITY

library(tsm)  
library(tseries)  
  
#check stationarity  
adf.test(monthly\_unemp$Rates)

##   
## Augmented Dickey-Fuller Test  
##   
## data: monthly\_unemp$Rates  
## Dickey-Fuller = -2.8852, Lag order = 7, p-value = 0.2034  
## alternative hypothesis: stationary

# DIFFERENCING

# Apply differencing to make the time series stationary  
differenced\_series <- diff(ts\_data, differences = 2)  
  
# Plot the differenced series to observe any remaining trends or patterns  
plot(differenced\_series, type = "l", main = "Differenced Series")



# Perform stationarity tests on the differenced series  
adf\_test\_diff <- adf.test(differenced\_series)

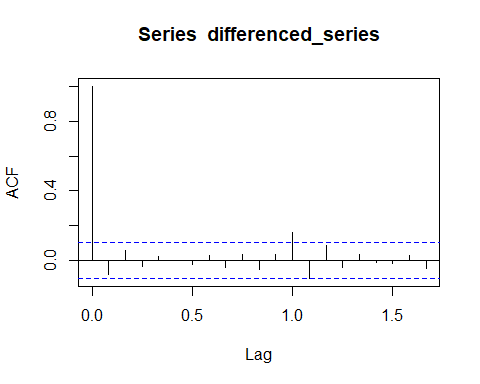
## Warning in adf.test(differenced\_series): p-value smaller than printed p-value

# Print the test results  
print(adf\_test\_diff)

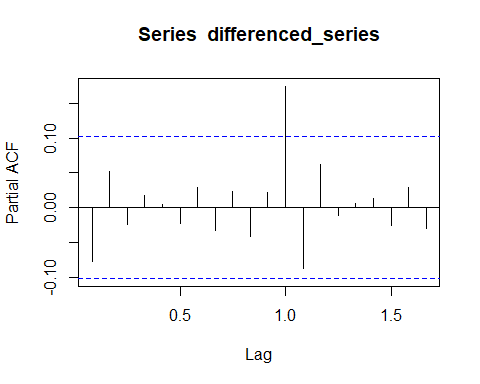
##   
## Augmented Dickey-Fuller Test  
##   
## data: differenced\_series  
## Dickey-Fuller = -6.7822, Lag order = 7, p-value = 0.01  
## alternative hypothesis: stationary

# CORELLOGRAM

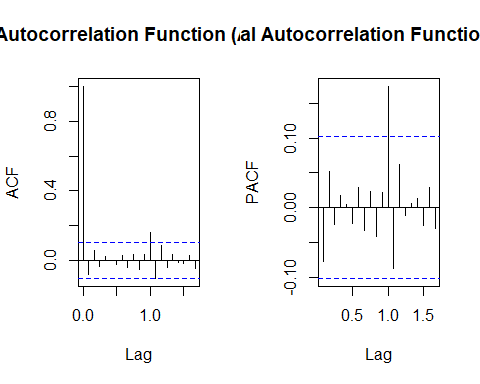
# Compute ACF and PACF  
acf\_result <- acf(differenced\_series, lag.max = 20)



pacf\_result <- pacf(differenced\_series, lag.max = 20)



par(mfrow=c(1,2))  
  
# Plot ACF  
plot(acf\_result, main = "Autocorrelation Function (ACF)", xlab = "Lag", ylab = "ACF")  
  
# Plot PACF  
plot(pacf\_result, main = "Partial Autocorrelation Function (PACF)", xlab = "Lag", ylab = "PACF")

 #ARIMA1

# Load the required library  
library(forecast)  
  
# Fit ARIMA models using stepwise search  
best\_model <- auto.arima(ts\_data, stepwise = TRUE)  
  
# Print the best model  
print(best\_model)

## Series: ts\_data   
## ARIMA(0,2,0)(0,0,1)[12]   
##   
## Coefficients:  
## sma1  
## 0.1425  
## s.e. 0.0480  
##   
## sigma^2 = 2.089e-05: log likelihood = 1473.45  
## AIC=-2942.89 AICc=-2942.86 BIC=-2935.06

#ARIMA2

# Fit ARIMA(p, 1, q) model  
model\_arima <- arima(ts\_data, order = c(0, 2, 1))  
  
# Print the fitted ARIMA model  
print(model\_arima)

##   
## Call:  
## arima(x = ts\_data, order = c(0, 2, 1))  
##   
## Coefficients:  
## ma1  
## -0.0703  
## s.e. 0.0493  
##   
## sigma^2 estimated as 2.116e-05: log likelihood = 1470.16, aic = -2936.31

# Calculate AIC and BIC  
aic <- AIC(model\_arima)  
bic <- BIC(model\_arima)  
  
# Print AIC and BIC  
cat("AIC:", aic, "\n")

## AIC: -2936.315

cat("BIC:", bic, "\n")

## BIC: -2928.482

# Get coefficient estimates, standard errors, t-values, and p-values  
coefficients <- coef(model\_arima)  
standard\_errors <- sqrt(diag(vcov(model\_arima)))  
t\_values <- coefficients / standard\_errors  
p\_values <- 2 \* (1 - pnorm(abs(t\_values)))  
  
# Create a table with coefficient estimates, standard errors, t-values, and p-values  
results\_table <- data.frame(  
 Coefficient = names(coefficients),  
 Estimate = coefficients,  
 `Std. Error` = standard\_errors,  
 `t-value` = t\_values,  
 `p-value` = p\_values  
)  
  
# Print the results table  
print(results\_table)

## Coefficient Estimate Std..Error t.value p.value  
## ma1 ma1 -0.07026031 0.04925089 -1.426579 0.1537012

# Calculate residual mean and residual variance  
residuals <- resid(model\_arima)  
residual\_mean <- mean(residuals)  
residual\_variance <- var(residuals)  
  
# Print residual mean and residual variance  
cat("Residual Mean:", residual\_mean, "\n")

## Residual Mean: -4.549252e-05

cat("Residual Variance:", residual\_variance, "\n")

## Residual Variance: 2.116146e-05

##RESIDUAL

# Obtain the residuals from the ARIMA model  
residuals <- residuals(best\_model)  
  
# Perform the Ljung-Box test  
ljung\_box\_test <- Box.test(residuals, lag = 32, type = "Ljung-Box")  
  
# Print the Ljung-Box test results  
cat("Ljung-Box Test Results:\n")

## Ljung-Box Test Results:

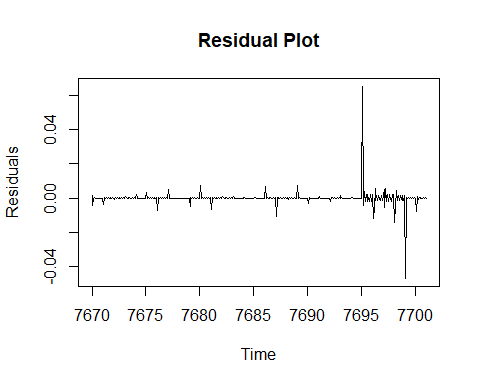
cat("Test statistic:", ljung\_box\_test$statistic, "\n")

## Test statistic: 22.53674

cat("p-value:", ljung\_box\_test$p.value, "\n")

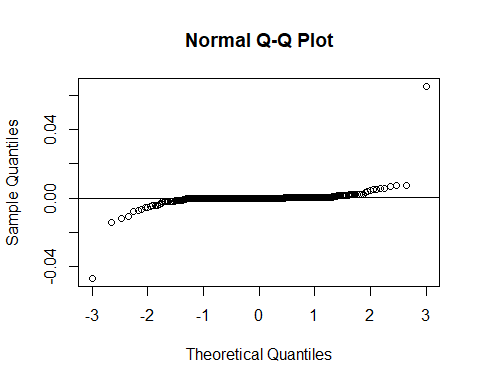
## p-value: 0.8923787

#plotting the box ljung results  
plot(residuals, type = "l", xlab = "Time", ylab = "Residuals", main = "Residual Plot")

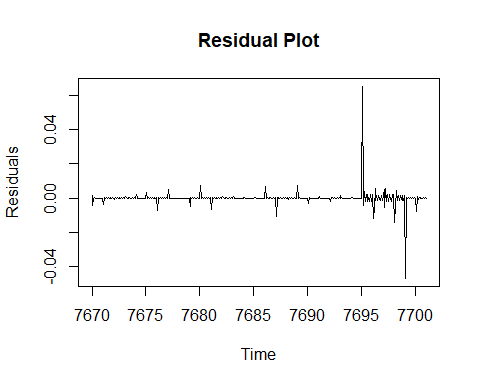
 P-value is greater than 0.05,thus suggesting that there is no significant evidence of autocorrelation in the residuals.

#Normal Q Q Plot

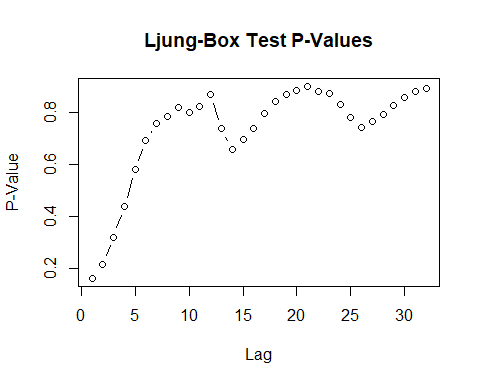
# Generate Normal Q-Q plot for the residuals  
qqnorm(residuals)  
qqline(residuals)



# Plotting the Normal Q-Q plot  
plot(residuals, type = "l", xlab = "Time", ylab = "Residuals", main = "Residual Plot")

 ##PVALUE PLOT

library(forecast)  
# Set the maximum number of lags for the Ljung-Box test  
max\_lag <- 32  
  
# Perform the Ljung-Box test for multiple lags  
p\_values <- numeric(max\_lag)  
for (lag in 1:max\_lag) {  
 ljung\_box\_test <- Box.test(residuals, lag = lag, type = "Ljung-Box")  
 p\_values[lag] <- ljung\_box\_test$p.value  
}  
  
# Plot the p-values  
plot(p\_values, type = "b", xlab = "Lag", ylab = "P-Value", main = "Ljung-Box Test P-Values")  
abline(h = 0.05, lty = 2, col = "red") # Add a horizontal line at significance level 0.05

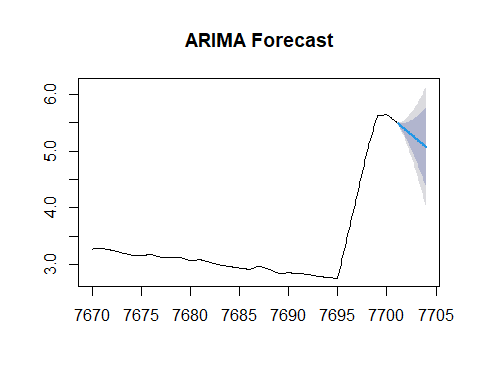


#FORECASTING

# Load the required library  
library(forecast)  
  
# Obtain the forecast using the ARIMA model  
forecast\_result <- forecast(model\_arima, h = 36) # Adjust h for desired forecast horizon  
  
# Print the forecasted values  
print(forecast\_result)

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## Feb 7701 5.489965 5.484070 5.495861 5.480949 5.498982  
## Mar 7701 5.477931 5.465117 5.490744 5.458334 5.497527  
## Apr 7701 5.465896 5.444721 5.487071 5.433511 5.498280  
## May 7701 5.453861 5.423081 5.484642 5.406786 5.500936  
## Jun 7701 5.441826 5.400336 5.483316 5.378373 5.505280  
## Jul 7701 5.429792 5.376590 5.482994 5.348426 5.511157  
## Aug 7701 5.417757 5.351919 5.483595 5.317067 5.518447  
## Sep 7701 5.405722 5.326388 5.485057 5.284390 5.527054  
## Oct 7701 5.393687 5.300046 5.487328 5.250476 5.536899  
## Nov 7701 5.381653 5.272939 5.490366 5.215389 5.547916  
## Dec 7701 5.369618 5.245102 5.494134 5.179187 5.560048  
## Jan 7702 5.357583 5.216568 5.498598 5.141919 5.573247  
## Feb 7702 5.345548 5.187365 5.503732 5.103628 5.587469  
## Mar 7702 5.333514 5.157518 5.509510 5.064351 5.602676  
## Apr 7702 5.321479 5.127049 5.515909 5.024124 5.618833  
## May 7702 5.309444 5.095980 5.522909 4.982978 5.635910  
## Jun 7702 5.297410 5.064327 5.530492 4.940941 5.653878  
## Jul 7702 5.285375 5.032108 5.538641 4.898037 5.672712  
## Aug 7702 5.273340 4.999339 5.547341 4.854292 5.692388  
## Sep 7702 5.261305 4.966034 5.556577 4.809727 5.712884  
## Oct 7702 5.249271 4.932205 5.566336 4.764361 5.734180  
## Nov 7702 5.237236 4.897866 5.576606 4.718214 5.756258  
## Dec 7702 5.225201 4.863026 5.587376 4.671303 5.779099  
## Jan 7703 5.213166 4.827699 5.598634 4.623644 5.802688  
## Feb 7703 5.201132 4.791892 5.610371 4.575254 5.827009  
## Mar 7703 5.189097 4.755616 5.622577 4.526146 5.852048  
## Apr 7703 5.177062 4.718880 5.635244 4.476333 5.877791  
## May 7703 5.165027 4.681692 5.648363 4.425830 5.904225  
## Jun 7703 5.152993 4.644060 5.661925 4.374647 5.931338  
## Jul 7703 5.140958 4.605991 5.675924 4.322797 5.959119  
## Aug 7703 5.128923 4.567494 5.690353 4.270291 5.987555  
## Sep 7703 5.116888 4.528574 5.705203 4.217139 6.016638  
## Oct 7703 5.104854 4.489238 5.720470 4.163350 6.046357  
## Nov 7703 5.092819 4.449492 5.736146 4.108936 6.076702  
## Dec 7703 5.080784 4.409343 5.752225 4.053904 6.107665  
## Jan 7704 5.068750 4.368796 5.768703 3.998263 6.139236

# Plot the forecasted values  
plot(forecast\_result, main = "ARIMA Forecast")

 #SARIMA

library(forecast)  
# Estimate the SARIMA model  
model\_sarima <- arima(ts\_data, order = c(0, 2, 1), seasonal = list(order = c(2, 0, 2), period = 12))  
  
# Print the model summary  
summary(model\_sarima)

##   
## Call:  
## arima(x = ts\_data, order = c(0, 2, 1), seasonal = list(order = c(2, 0, 2), period = 12))  
##   
## Coefficients:  
## ma1 sar1 sar2 sma1 sma2  
## -0.0596 0.0774 -0.3769 0.0418 1.0000  
## s.e. 0.0496 0.0746 0.0535 0.0475 0.0439  
##   
## sigma^2 estimated as 1.389e-05: log likelihood = 1523.05, aic = -3034.11  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE  
## Training set -4.643155e-05 0.003724131 0.0008731519 0.001349624 0.02557831  
## MASE ACF1  
## Training set 0.0852974 -0.004554328

BIC(model\_sarima)

## [1] -3010.612

# Load the forecast package  
library(forecast)  
  
# Fit SARIMA(0, 2, 1)(2, 0, 2)[12] model  
model\_sarima <- Arima(ts\_data, order = c(0, 2, 1), seasonal = list(order = c(2, 0, 2), period = 12))  
  
# Print the fitted SARIMA model  
print(model\_sarima)

## Series: ts\_data   
## ARIMA(0,2,1)(2,0,2)[12]   
##   
## Coefficients:  
## ma1 sar1 sar2 sma1 sma2  
## -0.0596 0.0774 -0.3769 0.0418 1.0000  
## s.e. 0.0496 0.0746 0.0535 0.0475 0.0439  
##   
## sigma^2 = 1.413e-05: log likelihood = 1523.05  
## AIC=-3034.11 AICc=-3033.88 BIC=-3010.61

# Calculate AIC and BIC  
aic <- AIC(model\_sarima)  
bic <- BIC(model\_sarima)  
  
# Print AIC and BIC  
cat("AIC:", aic, "\n")

## AIC: -3034.11

cat("BIC:", bic, "\n")

## BIC: -3010.612

# Get coefficient estimates, standard errors, t-values, and p-values  
coefficients <- coef(model\_sarima)  
standard\_errors <- sqrt(diag(vcov(model\_sarima)))  
t\_values <- coefficients / standard\_errors  
p\_values <- 2 \* (1 - pnorm(abs(t\_values)))  
  
# Create a table with coefficient estimates, standard errors, t-values, and p-values  
results\_table <- data.frame(  
 Coefficient = names(coefficients),  
 Estimate = coefficients,  
 `Std. Error` = standard\_errors,  
 `t-value` = t\_values,  
 `p-value` = p\_values  
)  
  
# Print the results table  
print(results\_table)

## Coefficient Estimate Std..Error t.value p.value  
## ma1 ma1 -0.05959861 0.04961669 -1.2011808 2.296811e-01  
## sar1 sar1 0.07741346 0.07464029 1.0371538 2.996642e-01  
## sar2 sar2 -0.37685926 0.05348055 -7.0466606 1.832534e-12  
## sma1 sma1 0.04182666 0.04746820 0.8811511 3.782361e-01  
## sma2 sma2 0.99999369 0.04385251 22.8035681 0.000000e+00

# Calculate residuals  
residuals <- residuals(model\_sarima)  
  
# Calculate residual mean and residual variance  
residual\_mean <- mean(residuals)  
residual\_variance <- var(residuals)  
  
# Print residual mean and residual variance  
cat("Residual Mean:", residual\_mean, "\n")

## Residual Mean: -4.643155e-05

cat("Residual Variance:", residual\_variance, "\n")

## Residual Variance: 1.390428e-05

##RESIDUAL PLOT

# Obtain the residuals from the ARIMA model  
residuals <- residuals(model\_sarima)  
  
# Perform the Ljung-Box test  
ljung\_box\_test <- Box.test(residuals, lag = 32, type = "Ljung-Box")  
  
# Print the Ljung-Box test results  
cat("Ljung-Box Test Results:\n")

## Ljung-Box Test Results:

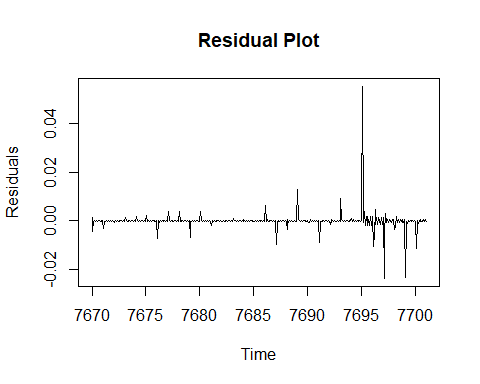
cat("Test statistic:", ljung\_box\_test$statistic, "\n")

## Test statistic: 17.7868

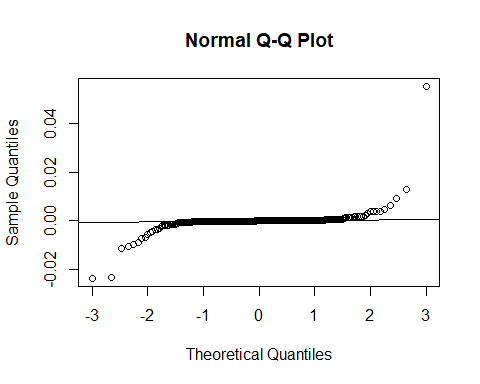
cat("p-value:", ljung\_box\_test$p.value, "\n")

## p-value: 0.9799631

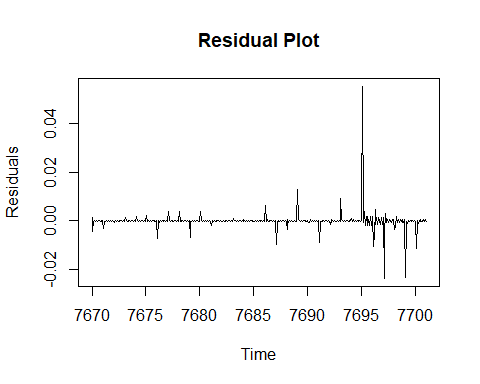
#plotting the box ljung results  
plot(residuals, type = "l", xlab = "Time", ylab = "Residuals", main = "Residual Plot")

 ##PLOT NORMALITY

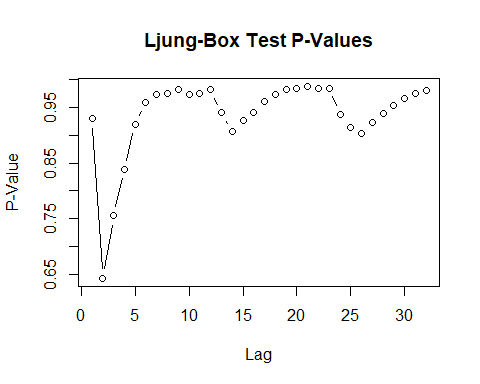
# Generate Normal Q-Q plot for the residuals  
qqnorm(residuals)  
qqline(residuals)



# Plotting the Normal Q-Q plot  
plot(residuals, type = "l", xlab = "Time", ylab = "Residuals", main = "Residual Plot")

 ##PVALUE PLOT

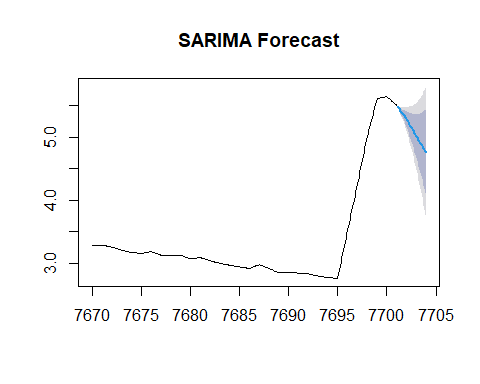
library(forecast)  
# Set the maximum number of lags for the Ljung-Box test  
max\_lag <- 32  
  
# Perform the Ljung-Box test for multiple lags  
p\_values <- numeric(max\_lag)  
for (lag in 1:max\_lag) {  
 ljung\_box\_test <- Box.test(residuals, lag = lag, type = "Ljung-Box")  
 p\_values[lag] <- ljung\_box\_test$p.value  
}  
  
# Plot the p-values  
plot(p\_values, type = "b", xlab = "Lag", ylab = "P-Value", main = "Ljung-Box Test P-Values")  
abline(h = 0.05, lty = 2, col = "red") # Add a horizontal line at significance level 0.05

 #FORECAST USING SARIMA

# Load the required library  
library(forecast)  
  
# Obtain the forecast using the ARIMA model  
forecast\_result <- forecast(model\_sarima, h = 36) # Adjust h for desired forecast horizon  
  
# Print the forecasted values  
print(forecast\_result)

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95  
## Feb 7701 5.484469 5.479510 5.489428 5.476884 5.492054  
## Mar 7701 5.468283 5.457459 5.479107 5.451730 5.484837  
## Apr 7701 5.450752 5.432833 5.468671 5.423348 5.478157  
## May 7701 5.433359 5.407286 5.459432 5.393483 5.473234  
## Jun 7701 5.415828 5.380661 5.450994 5.362045 5.469610  
## Jul 7701 5.398434 5.353321 5.443547 5.329439 5.467428  
## Aug 7701 5.380903 5.325058 5.436748 5.295495 5.466310  
## Sep 7701 5.363372 5.296063 5.430681 5.260432 5.466312  
## Oct 7701 5.345978 5.266516 5.425440 5.224452 5.467504  
## Nov 7701 5.328447 5.236181 5.420713 5.187338 5.469556  
## Dec 7701 5.311053 5.205362 5.416744 5.149413 5.472693  
## Jan 7702 5.293522 5.173814 5.413230 5.110445 5.476600  
## Feb 7702 5.270269 5.135707 5.404832 5.064474 5.476065  
## Mar 7702 5.247675 5.097504 5.397845 5.018008 5.477341  
## Apr 7702 5.224422 5.057939 5.390905 4.969808 5.479035  
## May 7702 5.201443 5.017984 5.384901 4.920867 5.482019  
## Jun 7702 5.178190 4.977125 5.379255 4.870687 5.485692  
## Jul 7702 5.155211 4.935937 5.374484 4.819861 5.490561  
## Aug 7702 5.131958 4.893898 5.370018 4.767877 5.496039  
## Sep 7702 5.108705 4.851303 5.366107 4.715043 5.502368  
## Oct 7702 5.085726 4.808444 5.363008 4.661660 5.509792  
## Nov 7702 5.062473 4.764790 5.360156 4.607206 5.517740  
## Dec 7702 5.039494 4.720905 5.358084 4.552253 5.526735  
## Jan 7703 5.016241 4.676253 5.356230 4.496274 5.536209  
## Feb 7703 4.994630 4.631773 5.357487 4.439688 5.549572  
## Mar 7703 4.972563 4.585518 5.359608 4.380629 5.564497  
## Apr 7703 4.950952 4.538481 5.363423 4.320132 5.581772  
## May 7703 4.929310 4.490247 5.368373 4.257821 5.600799  
## Jun 7703 4.907699 4.440944 5.374454 4.193859 5.621538  
## Jul 7703 4.886057 4.390570 5.381545 4.128274 5.643840  
## Aug 7703 4.864446 4.339236 5.389656 4.061206 5.667686  
## Sep 7703 4.842835 4.286959 5.398710 3.992697 5.692973  
## Oct 7703 4.821193 4.233751 5.408635 3.922778 5.719609  
## Nov 7703 4.799582 4.179709 5.419455 3.851568 5.747596  
## Dec 7703 4.777940 4.124805 5.431076 3.779056 5.776825  
## Jan 7704 4.756329 4.069131 5.443528 3.705350 5.807309

# Plot the forecasted values  
plot(forecast\_result, main = "SARIMA Forecast")



#HOLT-WINTERS

# Fit the Holt-Winters model  
fit\_multiplicative <- HoltWinters(ts\_data, seasonal = "multiplicative")  
fit\_additive <- HoltWinters(ts\_data, seasonal = "additive")  
  
# Print the model summary  
print("Multiplicative Holt-Winters Model:")

## [1] "Multiplicative Holt-Winters Model:"

print(fit\_multiplicative)

## Holt-Winters exponential smoothing with trend and multiplicative seasonal component.  
##   
## Call:  
## HoltWinters(x = ts\_data, seasonal = "multiplicative")  
##   
## Smoothing parameters:  
## alpha: 0.9591583  
## beta : 1  
## gamma: 0.08542441  
##   
## Coefficients:  
## [,1]  
## a 5.49600999  
## b -0.01487982  
## s1 1.00070984  
## s2 1.00019189  
## s3 0.99985596  
## s4 0.99961880  
## s5 0.99947927  
## s6 0.99941201  
## s7 0.99947166  
## s8 0.99961380  
## s9 0.99983539  
## s10 1.00017063  
## s11 1.00057571  
## s12 1.00109658

print("Additive Holt-Winters Model:")

## [1] "Additive Holt-Winters Model:"

print(fit\_additive)

## Holt-Winters exponential smoothing with trend and additive seasonal component.  
##   
## Call:  
## HoltWinters(x = ts\_data, seasonal = "additive")  
##   
## Smoothing parameters:  
## alpha: 0.965364  
## beta : 1  
## gamma: 0.396928  
##   
## Coefficients:  
## [,1]  
## a 5.4984656315  
## b -0.0138450667  
## s1 0.0028255973  
## s2 0.0008265825  
## s3 -0.0004761561  
## s4 -0.0015097773  
## s5 -0.0017648706  
## s6 -0.0020634664  
## s7 -0.0017852924  
## s8 -0.0013623453  
## s9 -0.0006982181  
## s10 0.0005237505  
## s11 0.0017421723  
## s12 0.0035544520