TIME SERIES ANALYSIS MODEL

Enock Bereka

2024-11-21

#Time Series Analysis  
  
# Install and load necessary packages  
# install.packages(c("forecast", "tseries", "ggplot2"))  
library(forecast)

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

library(tseries)  
library(ggplot2)  
  
# Load the AirPassengers dataset  
data("AirPassengers")  
?AirPassengers

## starting httpd help server ...

## done

# 1. Data Exploration and Preprocessing  
str(AirPassengers)

## Time-Series [1:144] from 1949 to 1961: 112 118 132 129 121 135 148 148 136 119 ...

class(AirPassengers)

## [1] "ts"

start(AirPassengers)

## [1] 1949 1

end(AirPassengers)

## [1] 1960 12

frequency(AirPassengers)

## [1] 12

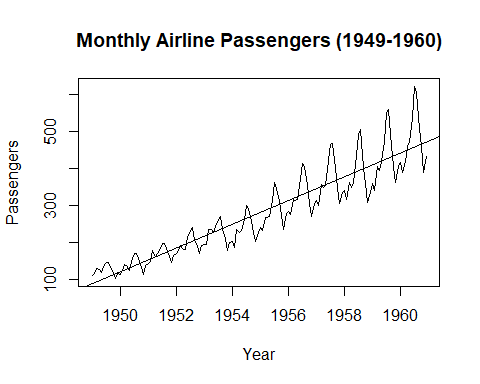
attributes(AirPassengers)

## $tsp  
## [1] 1949.000 1960.917 12.000  
##   
## $class  
## [1] "ts"

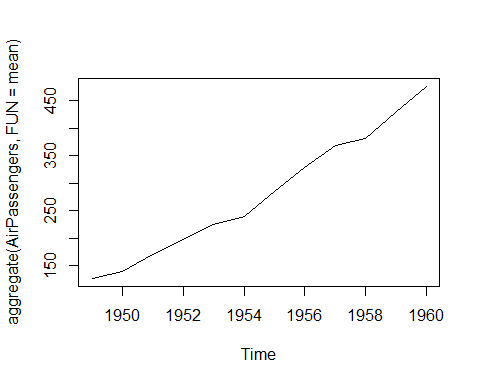
time(AirPassengers)

## Jan Feb Mar Apr May Jun Jul Aug  
## 1949 1949.000 1949.083 1949.167 1949.250 1949.333 1949.417 1949.500 1949.583  
## 1950 1950.000 1950.083 1950.167 1950.250 1950.333 1950.417 1950.500 1950.583  
## 1951 1951.000 1951.083 1951.167 1951.250 1951.333 1951.417 1951.500 1951.583  
## 1952 1952.000 1952.083 1952.167 1952.250 1952.333 1952.417 1952.500 1952.583  
## 1953 1953.000 1953.083 1953.167 1953.250 1953.333 1953.417 1953.500 1953.583  
## 1954 1954.000 1954.083 1954.167 1954.250 1954.333 1954.417 1954.500 1954.583  
## 1955 1955.000 1955.083 1955.167 1955.250 1955.333 1955.417 1955.500 1955.583  
## 1956 1956.000 1956.083 1956.167 1956.250 1956.333 1956.417 1956.500 1956.583  
## 1957 1957.000 1957.083 1957.167 1957.250 1957.333 1957.417 1957.500 1957.583  
## 1958 1958.000 1958.083 1958.167 1958.250 1958.333 1958.417 1958.500 1958.583  
## 1959 1959.000 1959.083 1959.167 1959.250 1959.333 1959.417 1959.500 1959.583  
## 1960 1960.000 1960.083 1960.167 1960.250 1960.333 1960.417 1960.500 1960.583  
## Sep Oct Nov Dec  
## 1949 1949.667 1949.750 1949.833 1949.917  
## 1950 1950.667 1950.750 1950.833 1950.917  
## 1951 1951.667 1951.750 1951.833 1951.917  
## 1952 1952.667 1952.750 1952.833 1952.917  
## 1953 1953.667 1953.750 1953.833 1953.917  
## 1954 1954.667 1954.750 1954.833 1954.917  
## 1955 1955.667 1955.750 1955.833 1955.917  
## 1956 1956.667 1956.750 1956.833 1956.917  
## 1957 1957.667 1957.750 1957.833 1957.917  
## 1958 1958.667 1958.750 1958.833 1958.917  
## 1959 1959.667 1959.750 1959.833 1959.917  
## 1960 1960.667 1960.750 1960.833 1960.917

# Plot the time series data  
plot(AirPassengers,   
 main = "Monthly Airline Passengers (1949-1960)",  
 xlab = "Year", ylab = "Passengers")  
#Fitting a smooth line  
abline(reg = lm(AirPassengers ~ time(AirPassengers)))

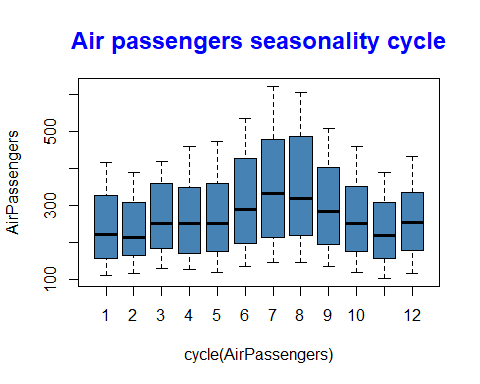


#General trend  
plot(aggregate(AirPassengers, FUN = mean))

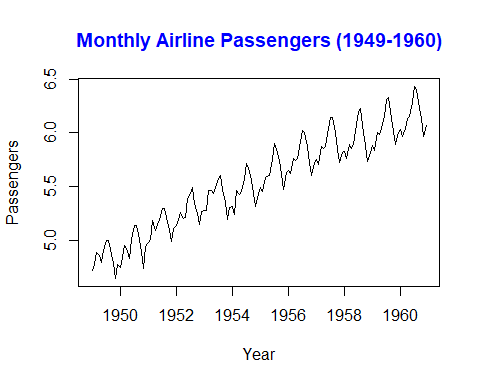


#Our graph indicates a general increase in the number  
#of air passengers over time.

boxplot(AirPassengers ~ cycle(AirPassengers),   
 main = "Air passengers seasonality cycle",   
 cex.main = 1.5, col = "steelblue", col.main = "blue")



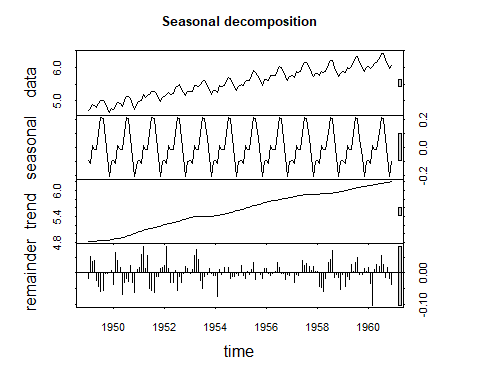
#Making the data to be stationary  
  
AirPassengers <- log(AirPassengers)  
plot(AirPassengers,   
 main = "Monthly Airline Passengers (1949-1960)",  
 xlab = "Year", ylab = "Passengers", col.main = "blue")



#The plot shows an upward trend and clear seasonality in  
#the air passenger data.

# Convert to a time series object  
passengers\_ts <- ts(AirPassengers, start = c(1949, 1),  
 frequency = 12)

# 2. Decompose the Time Series  
# Perform seasonal decomposition of time series (STL decomposition)  
passengers\_decomposed <- stl(passengers\_ts,   
 s.window = "periodic")  
plot(passengers\_decomposed, main = "Seasonal decomposition")



#Stl decomposition breaks down the time series into trend,  
#seasonal, and remainder components, helping to visualize   
#each part separately.

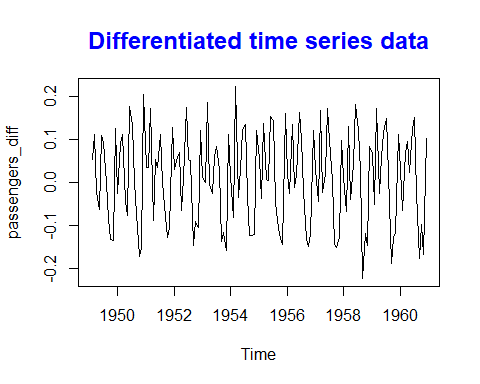
# Perform Augmented Dickey-Fuller Test for stationarity  
adf.test(passengers\_ts)

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

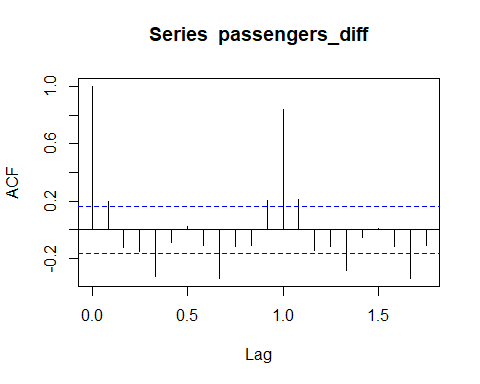
##   
## Augmented Dickey-Fuller Test  
##   
## data: passengers\_ts  
## Dickey-Fuller = -6.4215, Lag order = 5, p-value = 0.01  
## alternative hypothesis: stationary

#The Augmented Dickey-Fuller test checks if the time   
#series is stationary (mean and variance constant over   
#time). If not, differencing might be needed.

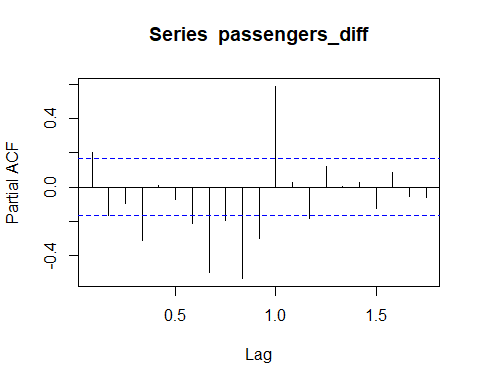
# 4. Differencing (if necessary)  
# If not stationary, apply differencing to make it stationary  
passengers\_diff <- diff(passengers\_ts)  
plot(passengers\_diff, main = "Differentiated time series data",  
 col.main = "blue", cex.main = 1.5)



acf(passengers\_diff)



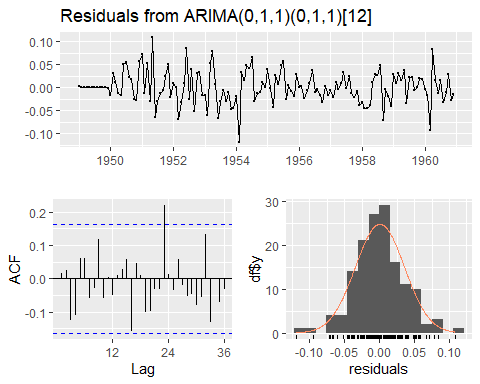
pacf(passengers\_diff)



# 5. Model Building (ARIMA)  
# Find the best ARIMA model using auto.arima()  
arima\_model <- auto.arima(passengers\_ts)  
summary(arima\_model)

## Series: passengers\_ts   
## ARIMA(0,1,1)(0,1,1)[12]   
##   
## Coefficients:  
## ma1 sma1  
## -0.4018 -0.5569  
## s.e. 0.0896 0.0731  
##   
## sigma^2 = 0.001371: log likelihood = 244.7  
## AIC=-483.4 AICc=-483.21 BIC=-474.77  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 0.0005730622 0.03504883 0.02626034 0.01098898 0.4752815 0.2169522  
## ACF1  
## Training set 0.01443892

#Predicting the model  
pred <- predict(arima\_model, n.ahead = 10\*12)  
checkresiduals(arima\_model)

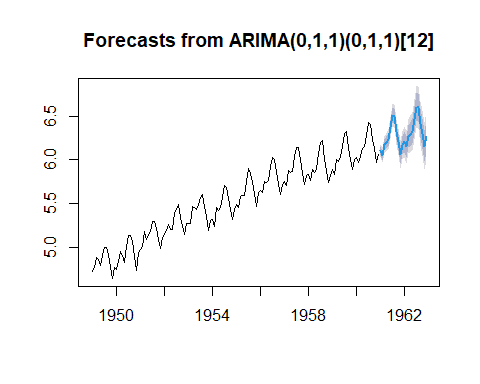


##   
## Ljung-Box test  
##   
## data: Residuals from ARIMA(0,1,1)(0,1,1)[12]  
## Q\* = 26.446, df = 22, p-value = 0.233  
##   
## Model df: 2. Total lags used: 24

#Checking residuals helps ensure the model assumptions  
#are met. Look for randomness and no patterns in the   
#residual plots.

# 7. Forecasting  
# Forecast future values using the ARIMA model  
forecast\_values <- forecast(arima\_model, h = 24) # Forecast for next 2 years (24 months)  
#The `forecast()` function uses the fitted ARIMA model   
#to predict future values.

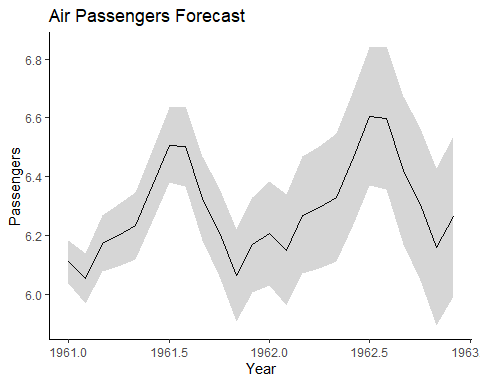
# 8. Visualization and Interpretation  
# Plot the original data, fitted values, and forecast  
plot(forecast\_values)



# Extract forecast values and confidence intervals  
forecast\_df <- data.frame(  
 Date = time(forecast\_values$mean),  
 Forecast = as.numeric(forecast\_values$mean),  
 Lower = as.numeric(forecast\_values$lower[, 2]),  
 Upper = as.numeric(forecast\_values$upper[, 2]))

# Plot the forecast using ggplot2  
ggplot(forecast\_df, aes(x = Date, y = Forecast)) +  
 geom\_line() +  
 theme\_classic()+  
 geom\_ribbon(aes(ymin = Lower, ymax = Upper), alpha = 0.2) +  
 labs(title = "Air Passengers Forecast", x = "Year", y = "Passengers")

## Don't know how to automatically pick scale for object of type <ts>. Defaulting  
## to continuous.



#The final plot displays the original data, the fitted   
#values from the ARIMA model, and the forecasted   
#values with confidence intervals.