Air_Passengers_TimeSeries_Forecast

Mouleeswaran B

2025-04-17

Introduction

- Forecasting Air Passengers Dataset using methods such as SMA,EMA,Holt and HoltWinters methods and identifying the best models to predict the forecast values for next one Year
- Log Transformation was used to stabilize the variance of the time series

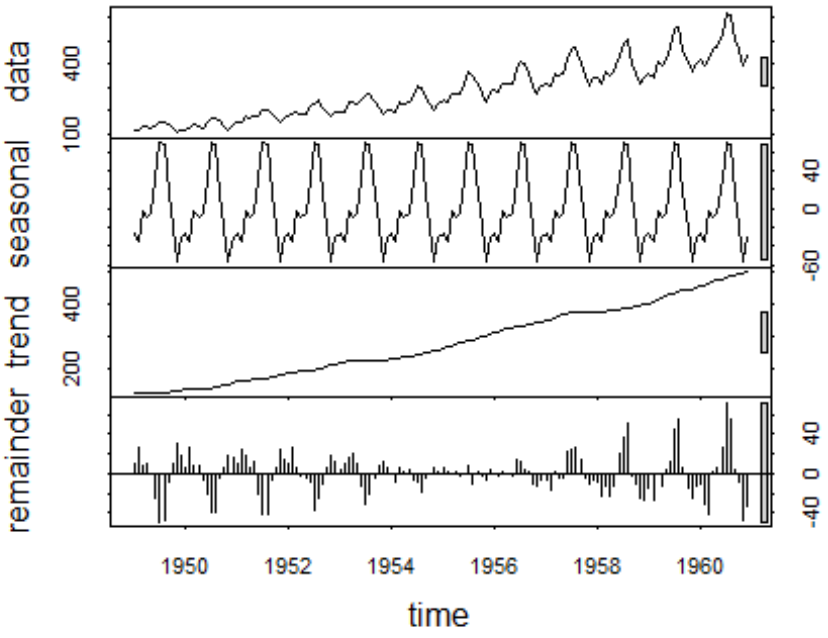
Reading The Data

```
data("AirPassengers")
AirPassengers
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1949 112 118 132 129 121 135 148 148 136 119 104 118
## 1950 115 126 141 135 125 149 170 170 158 133 114 140
## 1951 145 150 178 163 172 178 199 199 184 162 146 166
## 1952 171 180 193 181 183 218 230 242 209 191 172 194
## 1953 196 196 236 235 229 243 264 272 237 211 180 201
## 1954 204 188 235 227 234 264 302 293 259 229 203 229
## 1955 242 233 267 269 270 315 364 347 312 274 237 278
## 1956 284 277 317 313 318 374 413 405 355 306 271 306
## 1957 315 301 356 348 355 422 465 467 404 347 305 336
## 1958 340 318 362 348 363 435 491 505 404 359 310 337
## 1959 360 342 406 396 420 472 548 559 463 407 362 405
## 1960 417 391 419 461 472 535 622 606 508 461 390 432
```

Decomposing the data to identify Trend, Seasonality, Cycle and Remainder aspect of the time series

```
stl_airpassengers<-stl(AirPassengers,s.window='periodic')
plot(stl_airpassengers)
```



Installing Necessary Libraries to do the forecast

```
library(tseries)
```

```
## Warning: package 'tseries' was built under R version 4.4.3
```

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

```
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.4.3
```

```
library(TTR)
```

```
## Warning: package 'TTR' was built under R version 4.4.3
```

Checking For Stationarity in the time series using ADF test

```
adf.test(AirPassengers)
```

```
## Warning in adf.test(AirPassengers): p-value smaller than printed p-value
```

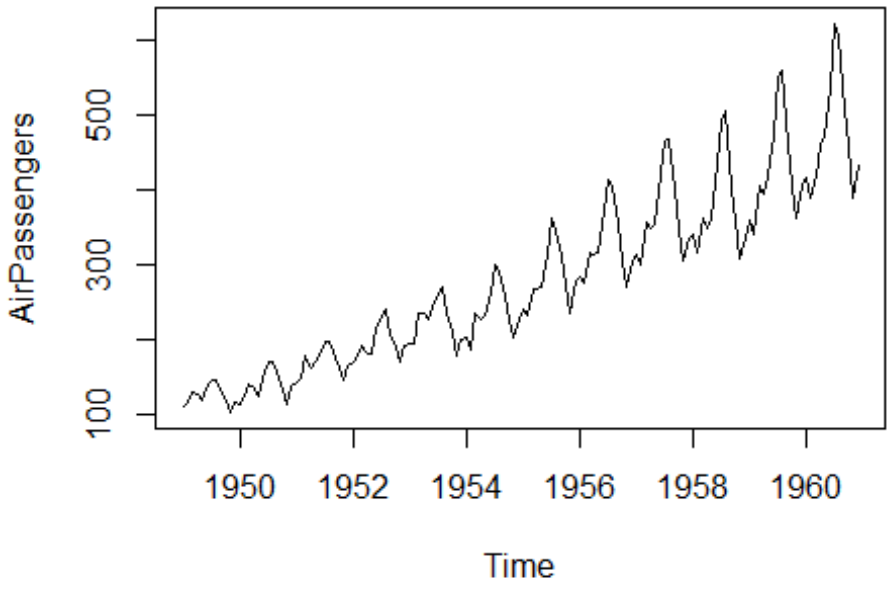
```
##
##  Augmented Dickey-Fuller Test
##
## data:  AirPassengers
```

```
## Dickey-Fuller = -7.3186, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

Augumented Dicky Fuller Test concluded that given time series is indeed Stationary so we can further proceed with our forecasting

Trying Out Simple Moving Average Technique to smoothen the time series

```
plot(AirPassengers,type='l')
```




```
AirPassengers_ts=ts(AirPassengers)
AirPassengers_ts

## Time Series:
## Start = 1
## End = 144
## Frequency = 1
##   [1] 112 118 132 129 121 135 148 148 136 119 104 118 115 126 141 135 125
149
##  [19] 170 170 158 133 114 140 145 150 178 163 172 178 199 199 184 162 146
166
##  [37] 171 180 193 181 183 218 230 242 209 191 172 194 196 196 236 235 229
243
##  [55] 264 272 237 211 180 201 204 188 235 227 234 264 302 293 259 229 203
229
```

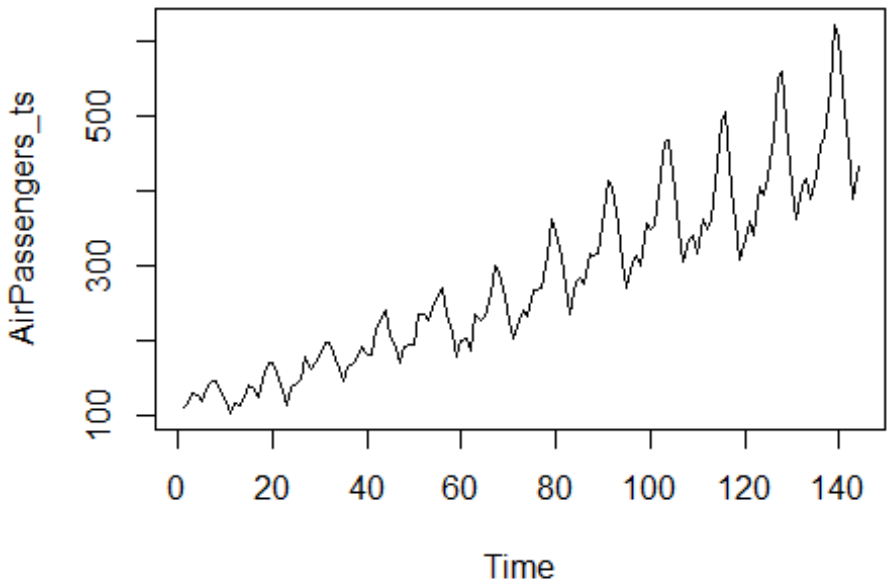
```
##  [73] 242 233 267 269 270 315 364 347 312 274 237 278 284 277 317 313 318
374
##  [91] 413 405 355 306 271 306 315 301 356 348 355 422 465 467 404 347 305
336
## [109] 340 318 362 348 363 435 491 505 404 359 310 337 360 342 406 396 420
472
## [127] 548 559 463 407 362 405 417 391 419 461 472 535 622 606 508 461 390
432
```

```
AirPassengers_ts_sma3<-SMA(AirPassengers_ts,n=3)
AirPassengers_ts_sma3
```

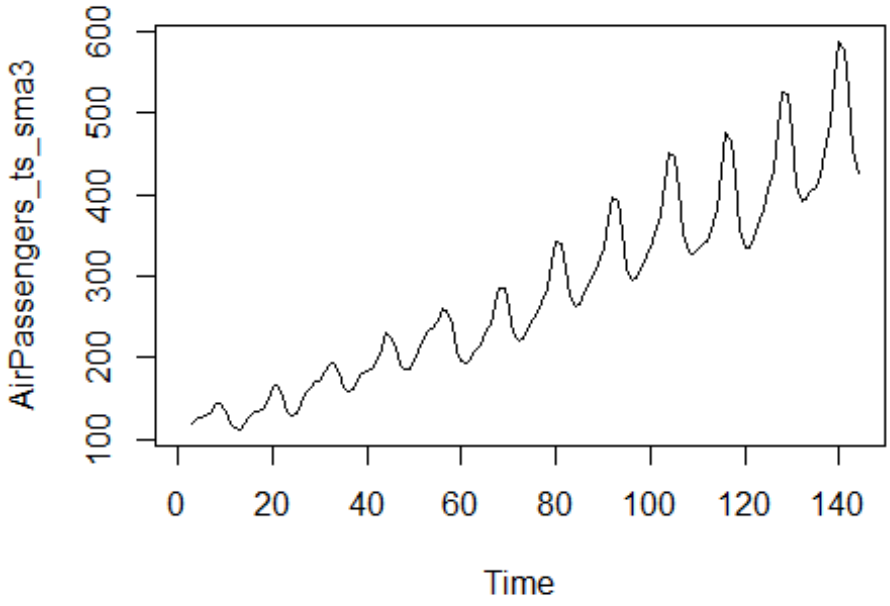
```
## Time Series:
## Start = 1
## End = 144
## Frequency = 1
##   [1]       NA       NA 120.6667 126.3333 127.3333 128.3333 134.6667 143.6
667
##   [9] 144.0000 134.3333 119.6667 113.6667 112.3333 119.6667 127.3333 134.0
000
##  [17] 133.6667 136.3333 148.0000 163.0000 166.0000 153.6667 135.0000 129.0
000
##  [25] 133.0000 145.0000 157.6667 163.6667 171.0000 171.0000 183.0000 192.0
000
##  [33] 194.0000 181.6667 164.0000 158.0000 161.0000 172.3333 181.3333 184.6
667
##  [41] 185.6667 194.0000 210.3333 230.0000 227.0000 214.0000 190.6667 185.6
667
##  [49] 187.3333 195.3333 209.3333 222.3333 233.3333 235.6667 245.3333 259.6
667
##  [57] 257.6667 240.0000 209.3333 197.3333 195.0000 197.6667 209.0000 216.6
667
##  [65] 232.0000 241.6667 266.6667 286.3333 284.6667 260.3333 230.3333 220.3
333
##  [73] 224.6667 234.6667 247.3333 256.3333 268.6667 284.6667 316.3333 342.0
000
##  [81] 341.0000 311.0000 274.3333 263.0000 266.3333 279.6667 292.6667 302.3
333
##  [89] 316.0000 335.0000 368.3333 397.3333 391.0000 355.3333 310.6667 294.3
333
##  [97] 297.3333 307.3333 324.0000 335.0000 353.0000 375.0000 414.0000 451.3
333
## [105] 445.3333 406.0000 352.0000 329.3333 327.0000 331.3333 340.0000 342.6
667
## [113] 357.6667 382.0000 429.6667 477.0000 466.6667 422.6667 357.6667 335.3
333
## [121] 335.6667 346.3333 369.3333 381.3333 407.3333 429.3333 480.0000 526.3
333
## [129] 523.3333 476.3333 410.6667 391.3333 394.6667 404.3333 409.0000 423.6
667
```

```
## [137] 450.6667 489.3333 543.0000 587.6667 578.6667 525.0000 453.0000 427.6
667
```

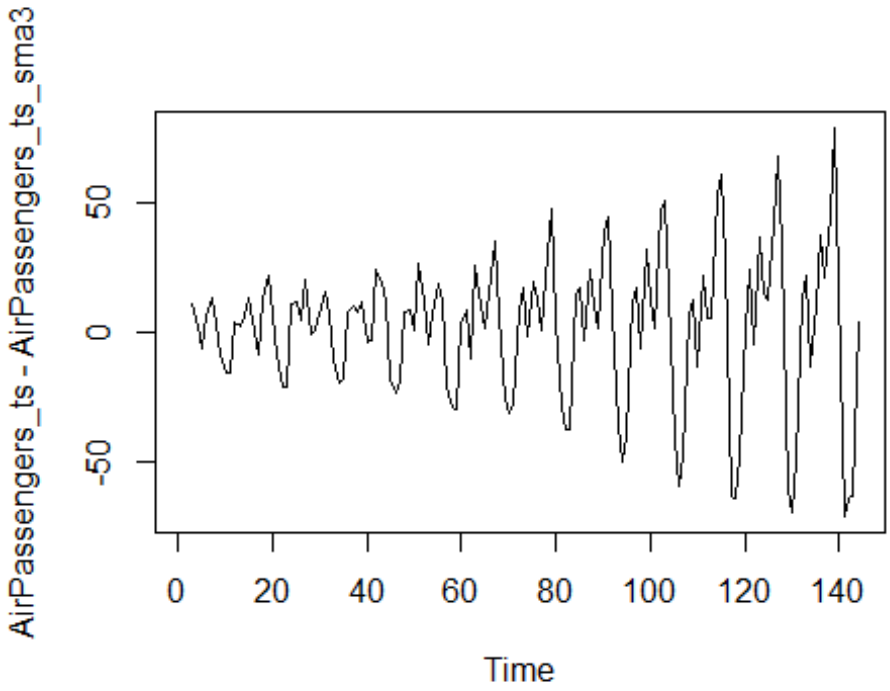
```
plot(AirPassengers_ts)
```




```
plot(AirPassengers_ts_sma3)
```



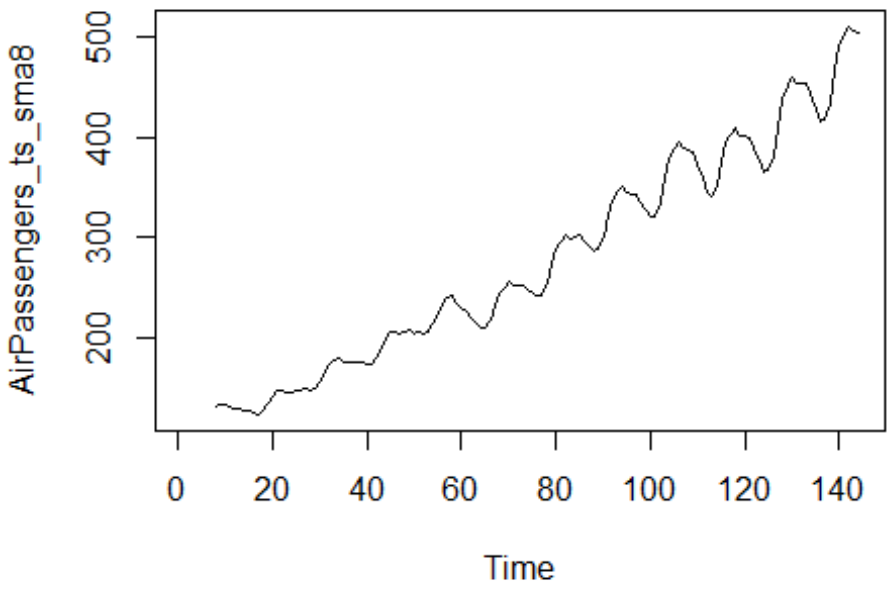

```
plot(AirPassengers_ts - AirPassengers_ts_sma3)
```



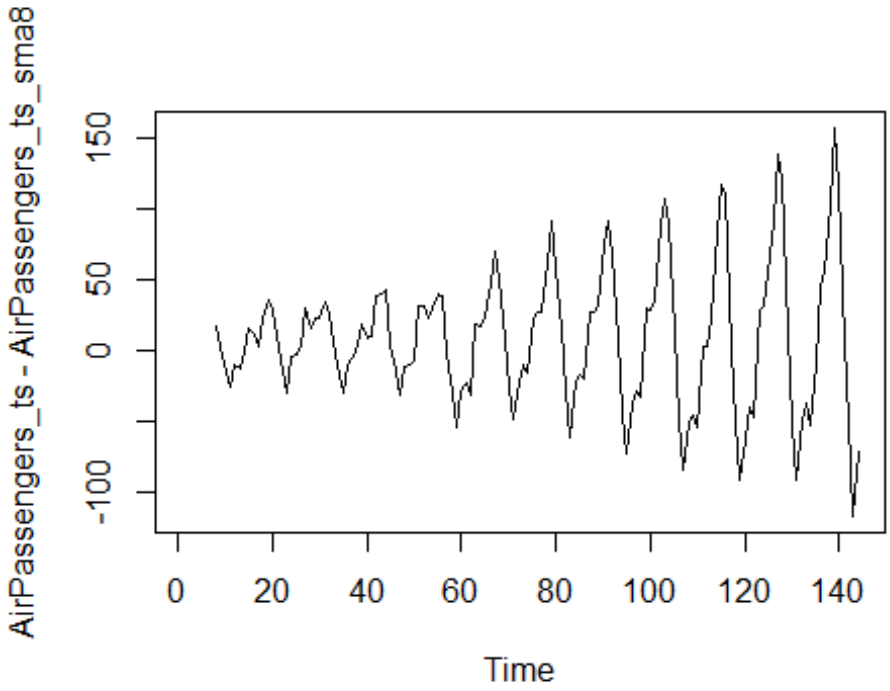
```
AirPassengers_ts_sma8<-SMA(AirPassengers_ts,n=8)
AirPassengers_ts_sma8
```

```
## Time Series:
## Start = 1
## End = 144
## Frequency = 1
##   [1]      NA      NA      NA      NA      NA      NA      NA 130.375 133.
375
##  [10] 133.500 130.000 128.625 127.875 126.750 125.875 124.250 122.875 126.
625
##  [19] 134.875 141.375 146.750 147.625 144.250 144.875 147.375 147.500 148.
500
##  [28] 147.625 149.375 155.000 165.625 173.000 177.875 179.375 175.375 175.
750
##  [37] 175.625 175.875 175.125 172.875 172.750 179.750 190.250 199.750 204.
500
##  [46] 205.875 203.250 204.875 206.500 203.750 204.500 203.625 206.125 212.
625
##  [55] 224.125 233.875 239.000 240.875 233.875 229.625 226.500 219.625 216.
000
##  [64] 210.375 210.000 216.625 231.875 243.375 250.250 255.375 251.375 251.
625
##  [73] 252.625 248.750 244.375 241.375 242.750 253.500 273.625 288.375 297.
125
##  [82] 302.250 298.500 299.625 301.375 296.625 290.750 286.500 287.250 299.
750
##  [91] 321.750 337.625 346.500 350.125 344.375 343.500 343.125 334.000 326.
875
## [100] 319.750 319.750 334.250 358.500 378.625 389.750 395.500 389.125 387.
625
## [109] 385.750 372.750 359.875 345.000 339.875 350.875 374.125 395.250 403.
250
## [118] 408.375 401.875 400.500 400.125 388.500 377.875 364.250 366.250 380.
375
## [127] 410.125 437.875 450.750 458.875 453.375 454.500 454.125 444.000 427.
875
## [136] 415.625 416.750 432.750 465.250 490.375 501.750 510.500 506.875 503.
250
```

```
plot(AirPassengers_ts_sma8)
```

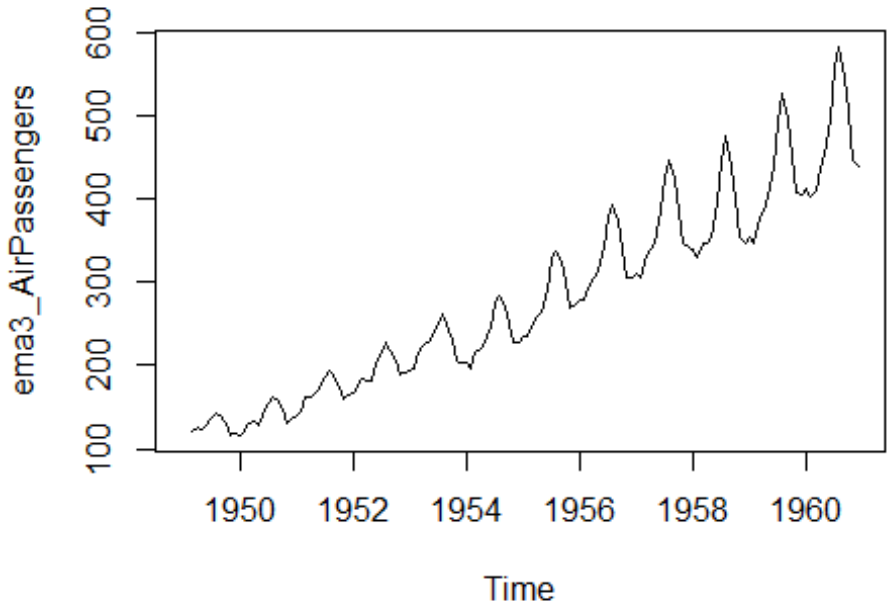



```
plot(AirPassengers_ts - AirPassengers_ts_sma8)
```

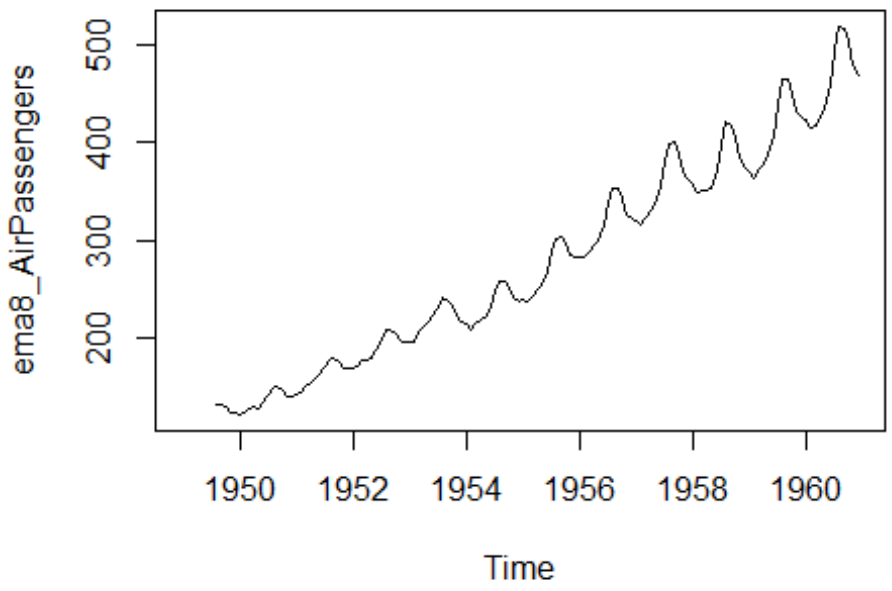


Since Simple Moving Average technique doesn't seems to make the time Series variance constant ## Trying Exponential Moving Average Technique to make variance constant

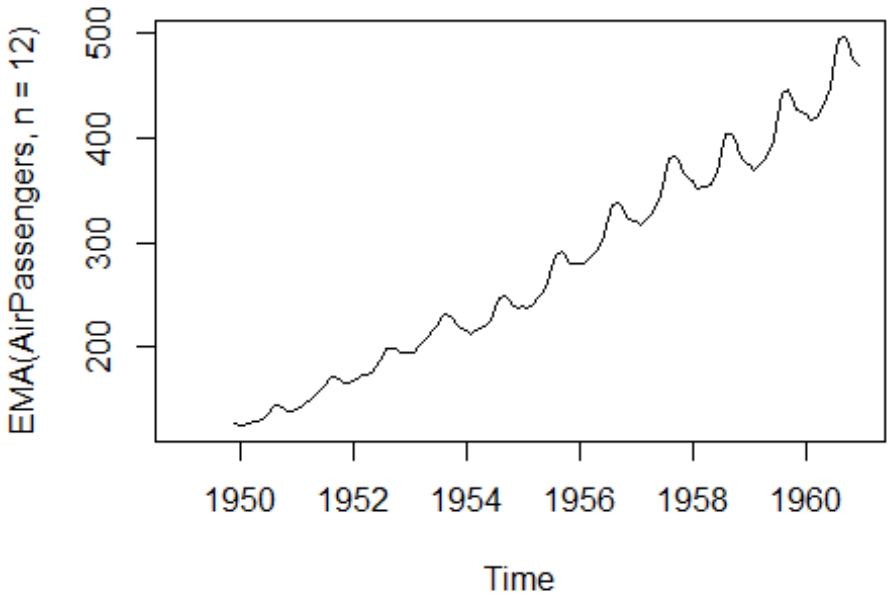
```
ema3_AirPassengers<-EMA(AirPassengers,n=3)
ema8_AirPassengers<-EMA(AirPassengers,n=8)
plot.ts(ema3_AirPassengers)
```



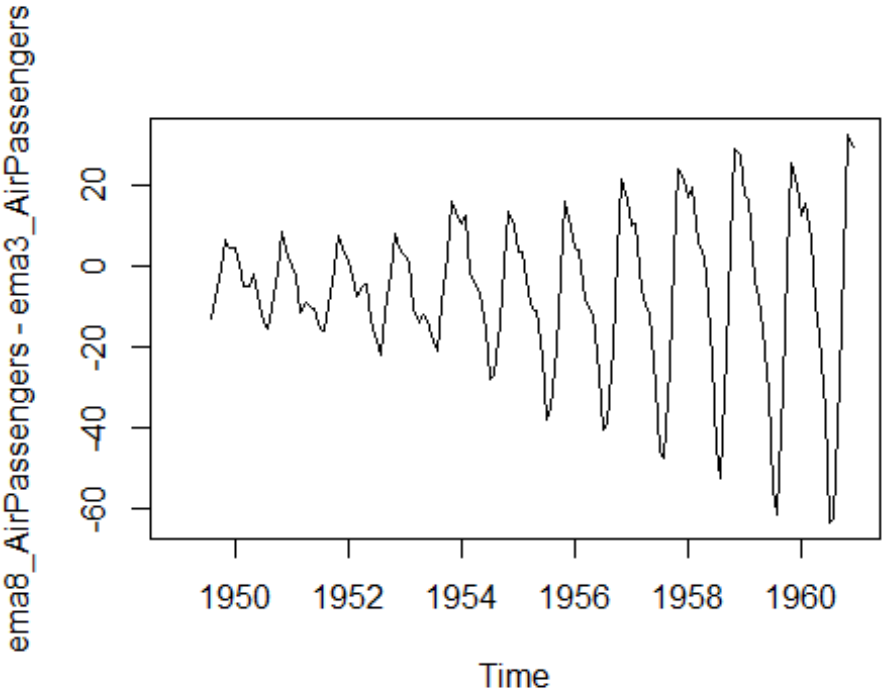

```
plot.ts(ema8_AirPassengers)
```




```
plot.ts(EMA(AirPassengers,n=12))
```




```
plot(ema8_AirPassengers-ema3_AirPassengers)
```



Exponential Moving Average Technique doesn't make the time series variance constant

Using HoltWinters Algorithm for the time Series to accomodate Seasonality

```
hw_airpassengers<-HoltWinters(log(AirPassengers))
hw_airpassengers
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal compon
ent.
##
## Call:
## HoltWinters(x = log(AirPassengers))
##
## Smoothing parameters:
##  alpha: 0.3266015
##  beta : 0.005744138
##  gamma: 0.8206654
##
## Coefficients:
##             [,1]
## a    6.172308435
## b    0.008981893
## s1  -0.073201087
## s2  -0.140973564
## s3  -0.036703294
## s4   0.014522733
```

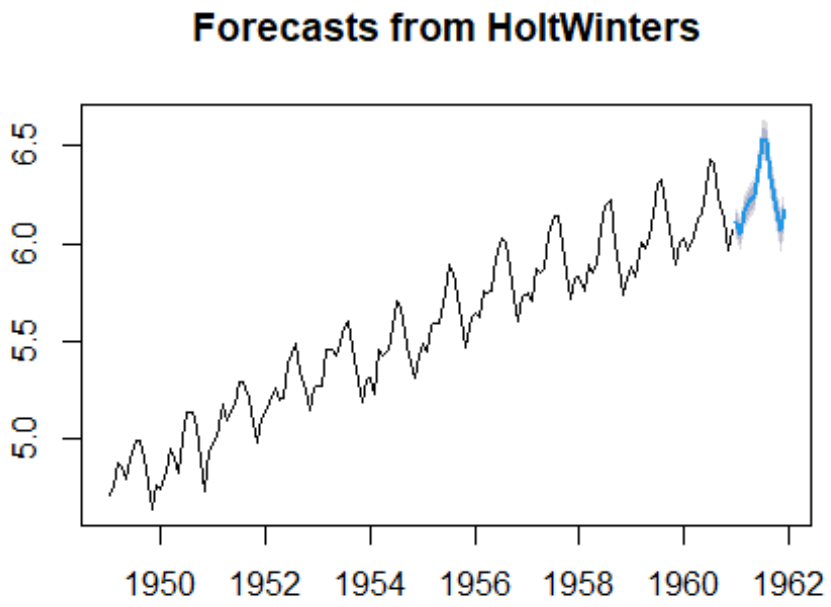
```
## s5   0.032554237
## s6   0.154873570
## s7   0.294317062
## s8   0.276063997
## s9   0.088237657
## s10 -0.032657089
## s11 -0.198012716
## s12 -0.102863837

hw_airpassengers$SSE

## [1] 0.2030765
```

Forecast 12 period(months) ahead

```
AirPassengersF12 <- forecast(hw_airpassengers,h=12)
plot(AirPassengersF12)
```



Forecasted Values mean

```
forecast_values<-exp(AirPassengersF12$mean)
forecast_values

##           Jan      Feb      Mar      Apr      May      Jun      Jul      A
ug
## 1961 449.4790 423.8157 474.6372 504.0920 517.8948 590.5613 685.0547 678.73
28
```

```
##           Sep      Oct      Nov      Dec
## 1961 567.5808 507.4866 434.0226 481.6546
```

95 percent Confidence Intervals

```
lower_95 <- exp(AirPassengersF12$lower[,"95%"])
upper_95 <- exp(AirPassengersF12$upper[,"95%"])
lower_95
```

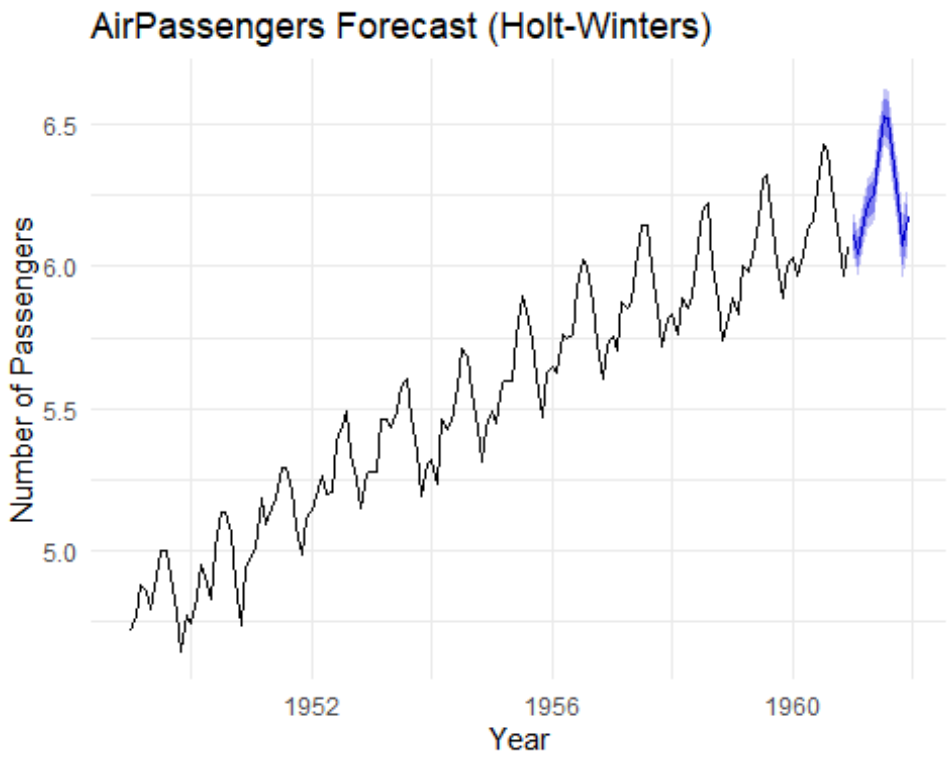
```
##           Jan      Feb      Mar      Apr      May      Jun      Jul      A
ug
## 1961 416.2477 390.9003 436.0759 461.3999 472.3087 536.6726 620.3955 612.59
83
##           Sep      Oct      Nov      Dec
## 1961 510.5867 455.0505 387.9417 429.1732
```

```
upper_95
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul      A
ug
## 1961 485.3634 459.5027 516.6085 550.7342 567.8808 649.8610 756.4530 752.00
71
##           Sep      Oct      Nov      Dec
## 1961 630.9369 565.9649 485.5771 540.5538
```

plotting the forecasted Time Series

```
library(ggplot2)
autoplot(AirPassengersF12) +
  ggtitle('AirPassengers Forecast (Holt-Winters)') +
  ylab('Number of Passengers') +
  xlab('Year') +
  theme_minimal()
```




```
ggsave("AirPassengers_Forecast_Plot.png",width=8,height=5)
```

Key Insights

1) Trend : Passenger growth is projected to increase steadily through next year
2) Peak Season : Highest number of Passengers expected in July and August
3) Precautions : Resource allocation needs to be increased by 85% to accomodate with the traffic during peak seasons