APPLY ANALYTICS FOR FORECASTING AIR PASSENGERS

May 13, 2023

0.1 Load Data

[7]:
```
library (dplyr)
library(tseries)
library(forecast)
data(AirPassengers)
```

```
Warning message:
"package 'forecast' was built under R version 4.2.3"
```

[8]:
```
AirPassengers
```

A Time Series: 12 × 12

	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

[9]:
```
class(AirPassengers)
```

'ts'

[10]:
```
end(AirPassengers)
```

1. 1960 2. 12

[11]:
```
sum(is.na(AirPassengers))
```

0

[12]:
```
frequency(AirPassengers)
```

12

[13]:
```
summary(AirPassengers)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  104.0   180.0   265.5   280.3   360.5   622.0 
```

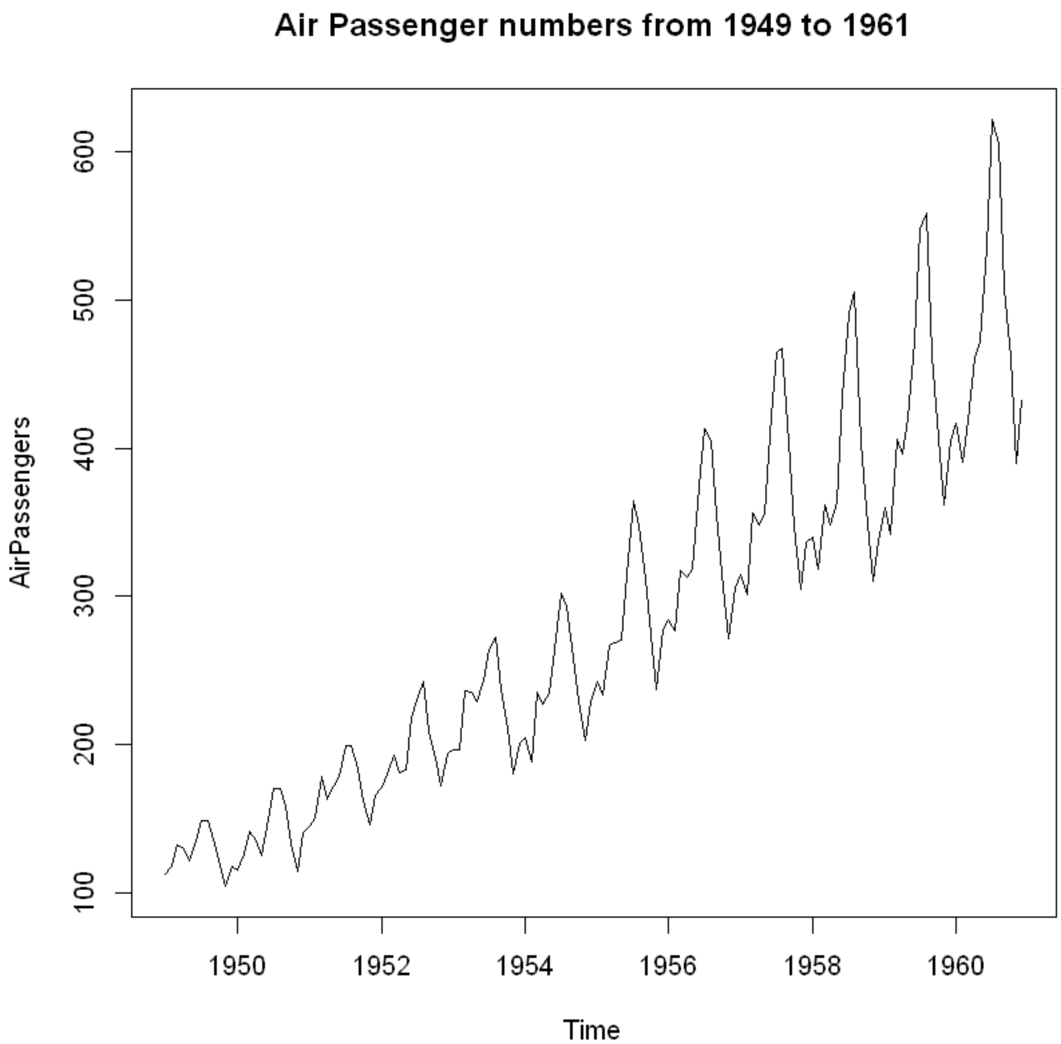
0.2 TEST THE STATIONARITY OF TIME SERIES

[14]:
```
adf.test(AirPassengers, alternative ="stationary", k=12)
```

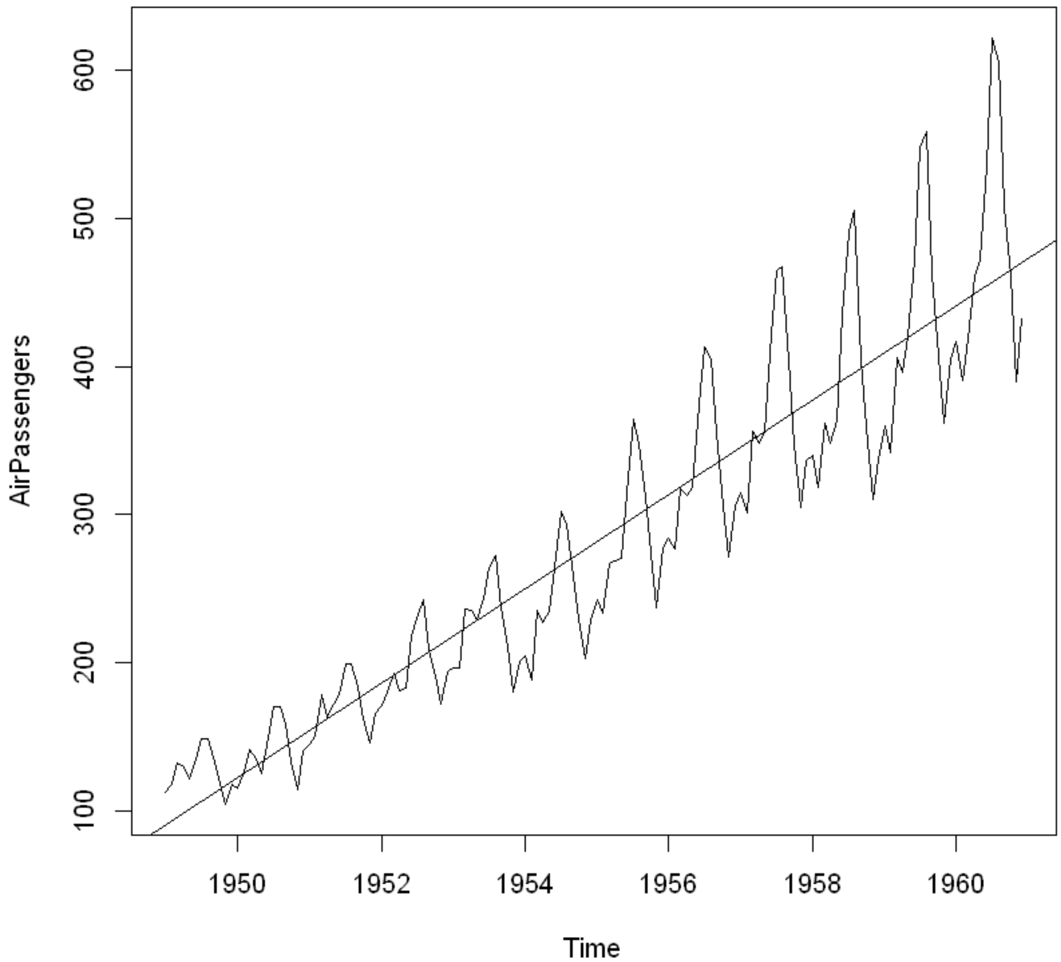
```
	Augmented Dickey-Fuller Test

data:  AirPassengers
Dickey-Fuller = -1.5094, Lag order = 12, p-value = 0.7807
alternative hypothesis: stationary
```

[15]:
```
plot(AirPassengers,main="Air Passenger numbers from 1949 to 1961") #show
 ↪sparkine
```

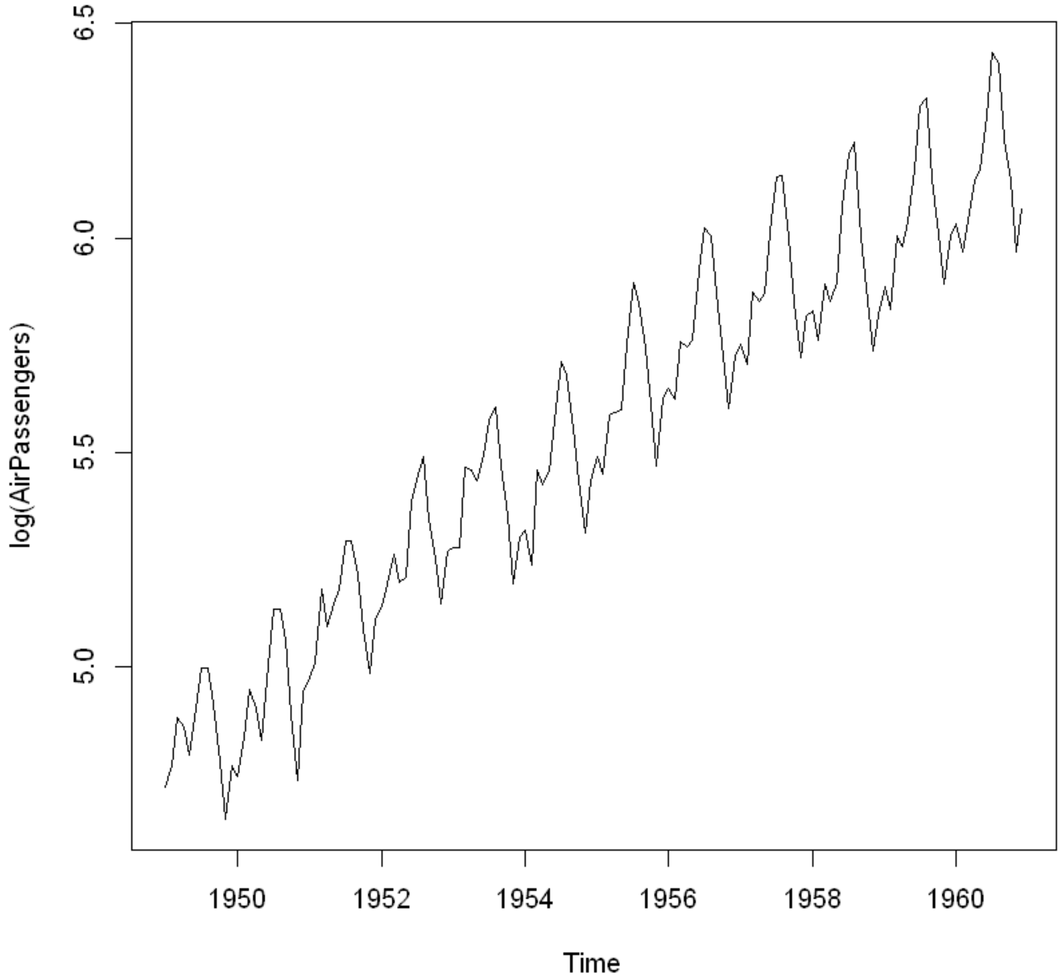


[16]:
```
plot(AirPassengers)+abline(reg=lm(AirPassengers~time(AirPassengers)))
```

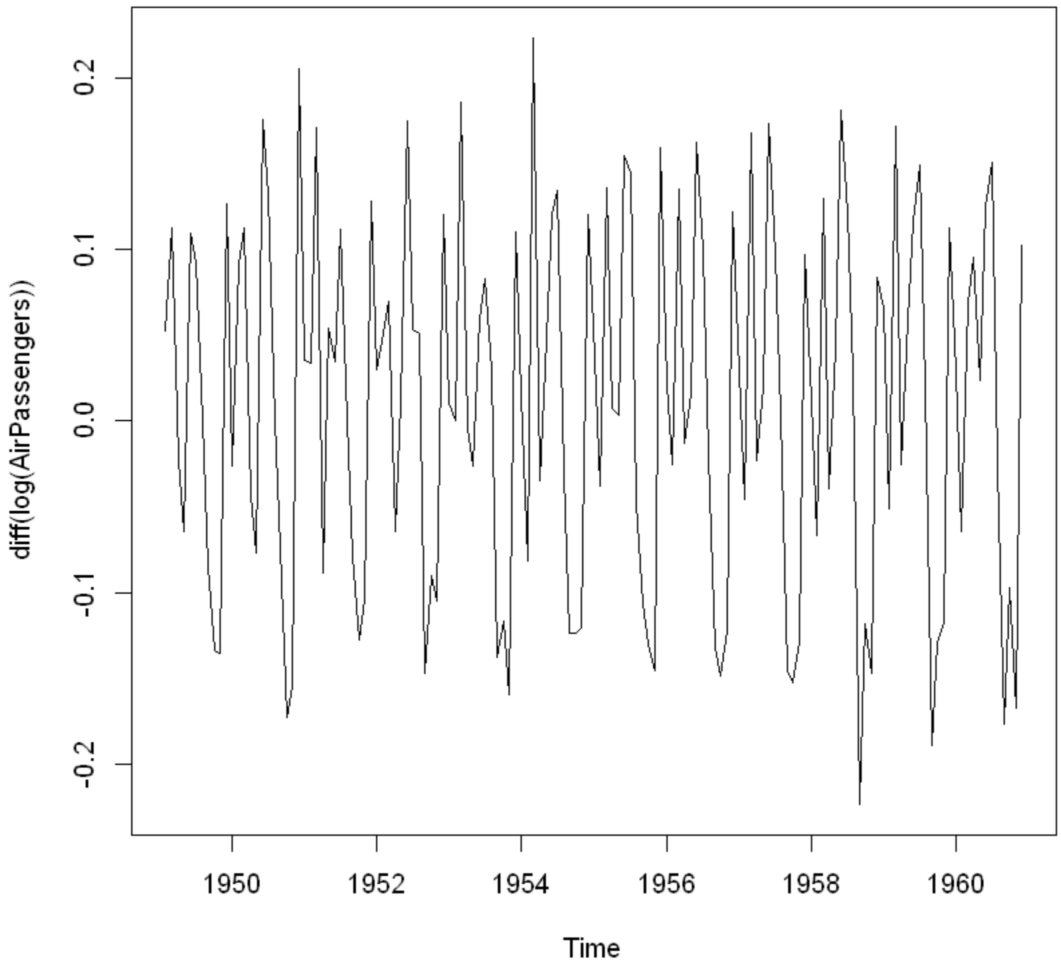


0.3 MAKE IT STATIONARY

[17]:
```
plot(log(AirPassengers))
```



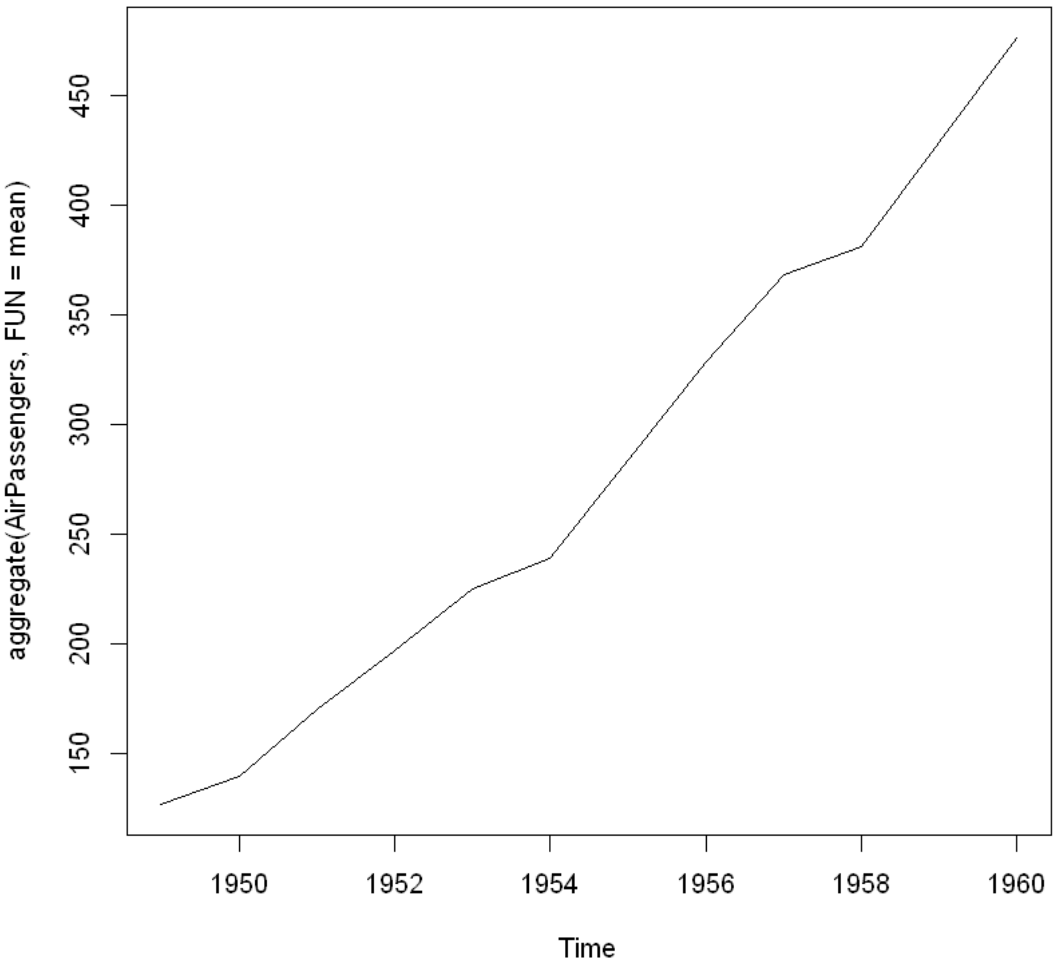
[18]:
```
plot(diff(log(AirPassengers)))
```



Now you can see mean and variance both are stationary

0.3.1 Check General Trend

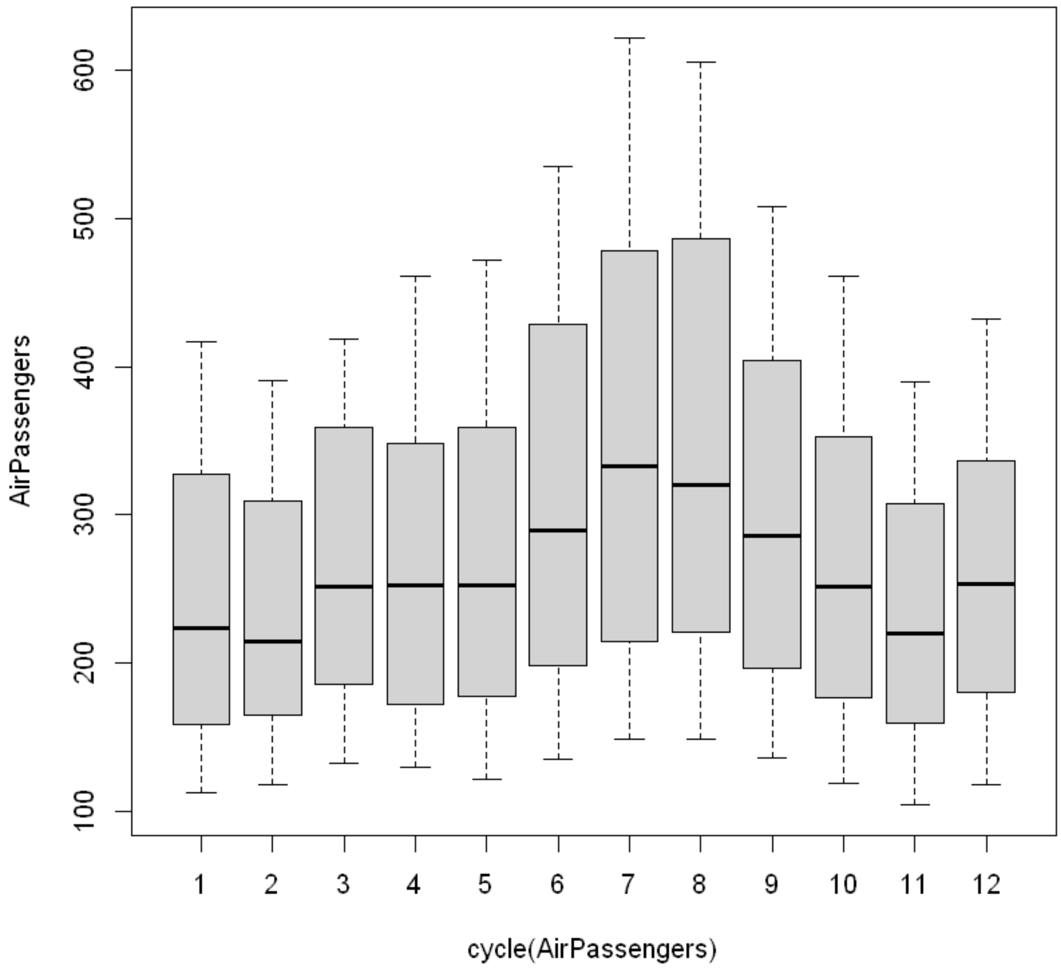
[19]:
```
plot(aggregate(AirPassengers,FUN = mean))
```



It is upword trend Use the boxplot function to see any seasonal effects.

0.3.2 show seasonality

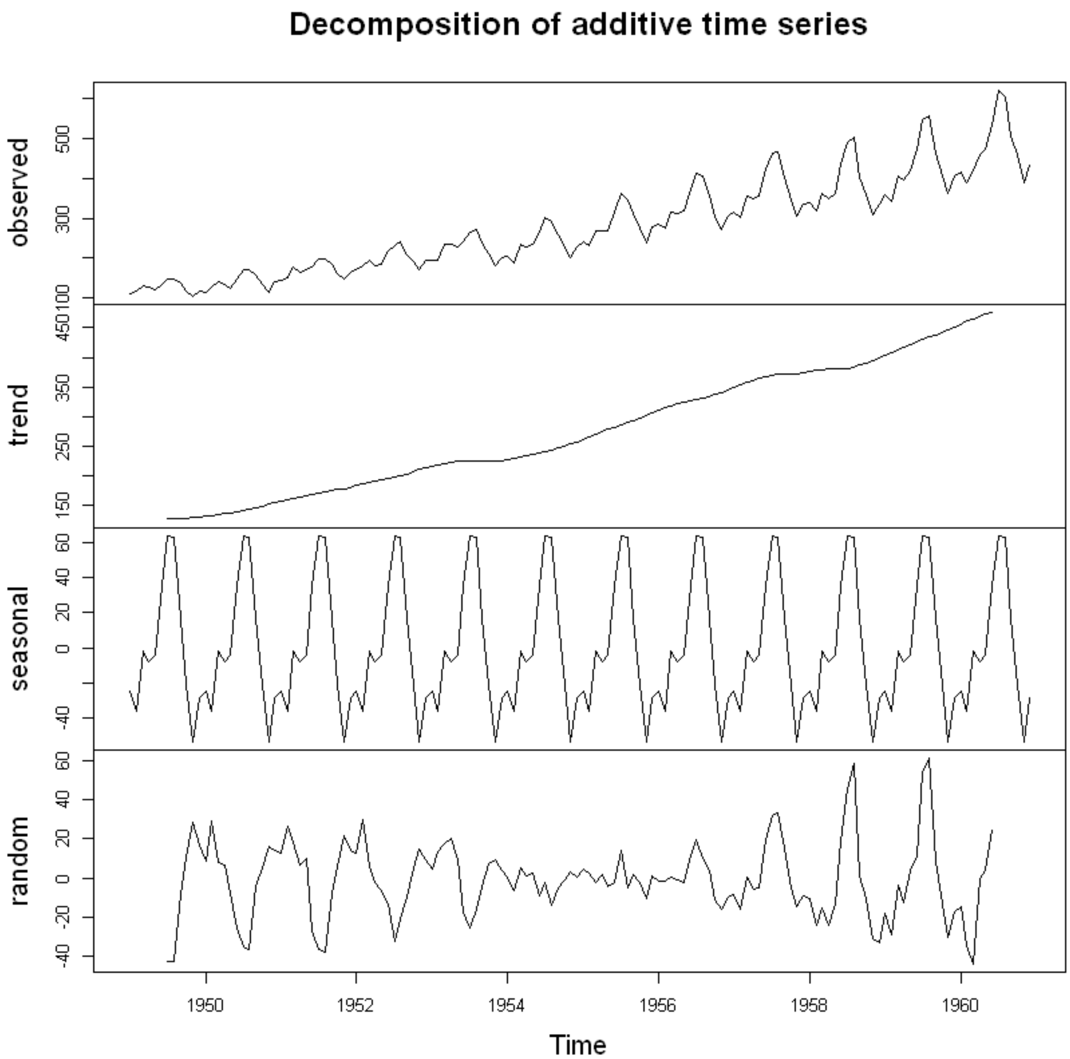
[20]:
```
boxplot(AirPassengers~cycle(AirPassengers))
```



In the boxplot you can see in month july and aug no of passangers traveling are more. The rationale for this could be more people taking holidays and fly over the summer months

0.4 TIME SERIES DECOMPOSITION

[21]:
```
plot(decompose(AirPassengers)) # time series decomposition
```



The above figure shows the time series decomposition into trend, seasonal and random (noise) . It is clear that the time series is non-stationary (has random walks) because of seasonal effects and a trend (linear trend).

0.5 MODEL IDENTIFICATION AND ESTIMATION

[22]:
```
tsdata <- ts(log(AirPassengers),frequency = 12)
```

. Calculate p d q value

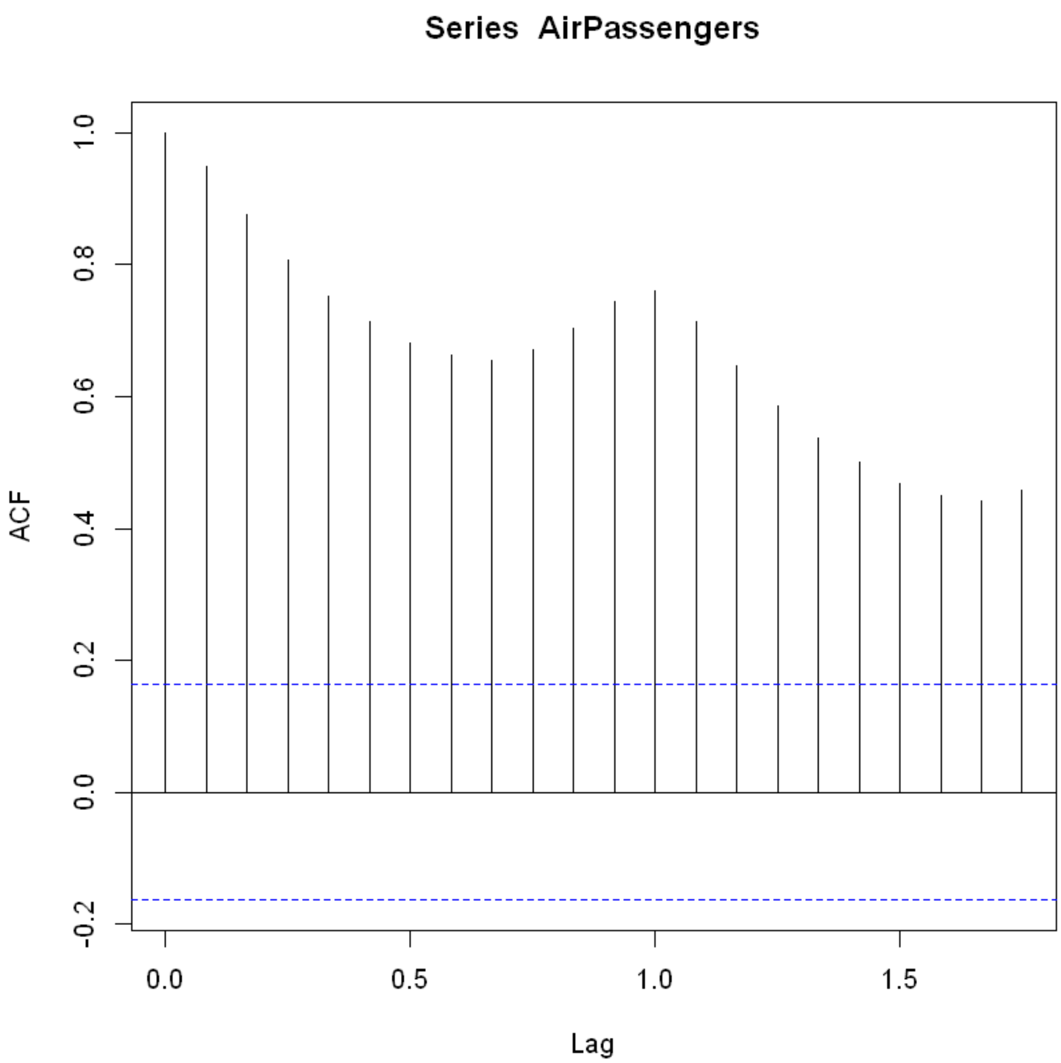
. AR(Autoregreesive) :- by seeing the past value, predict own value Integration

. MA(Moving Average) :- you take diff intervals and calculate the average Autocorrelation function and partial autocorrelation function to determine value of p and q

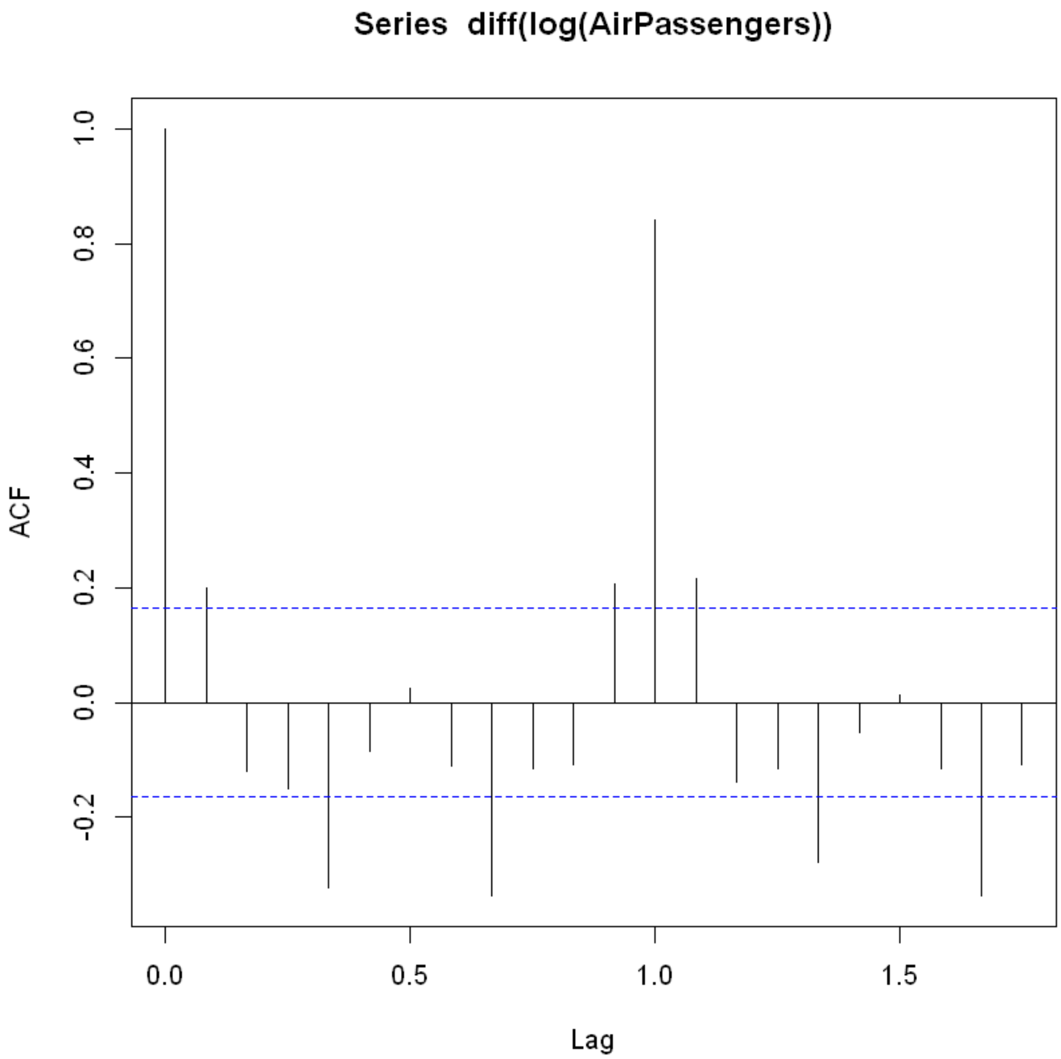
. Autocorrelation is the linear dependence of a variable with itself at two points in time Auto correlation function, as the word suggests, auto-correlation, means it is really correlation on itself.With time series we just have a single stream of values, or in other words there is just the X no Y.

. So suppose your time series is like this X = 3,5,6,6,7,4,5,6,7,2,3,4,. correlation between 4 and 3, 3 and 2, 2 and 7 (lag 1) will be say y1; correlation between 4 and 2, 3 and 7, 2 and 6 (lag 2) will be say y2; and so on at lags 3 = y3, lag 4 = y4.

[23]:
```
acf(AirPassengers)
```

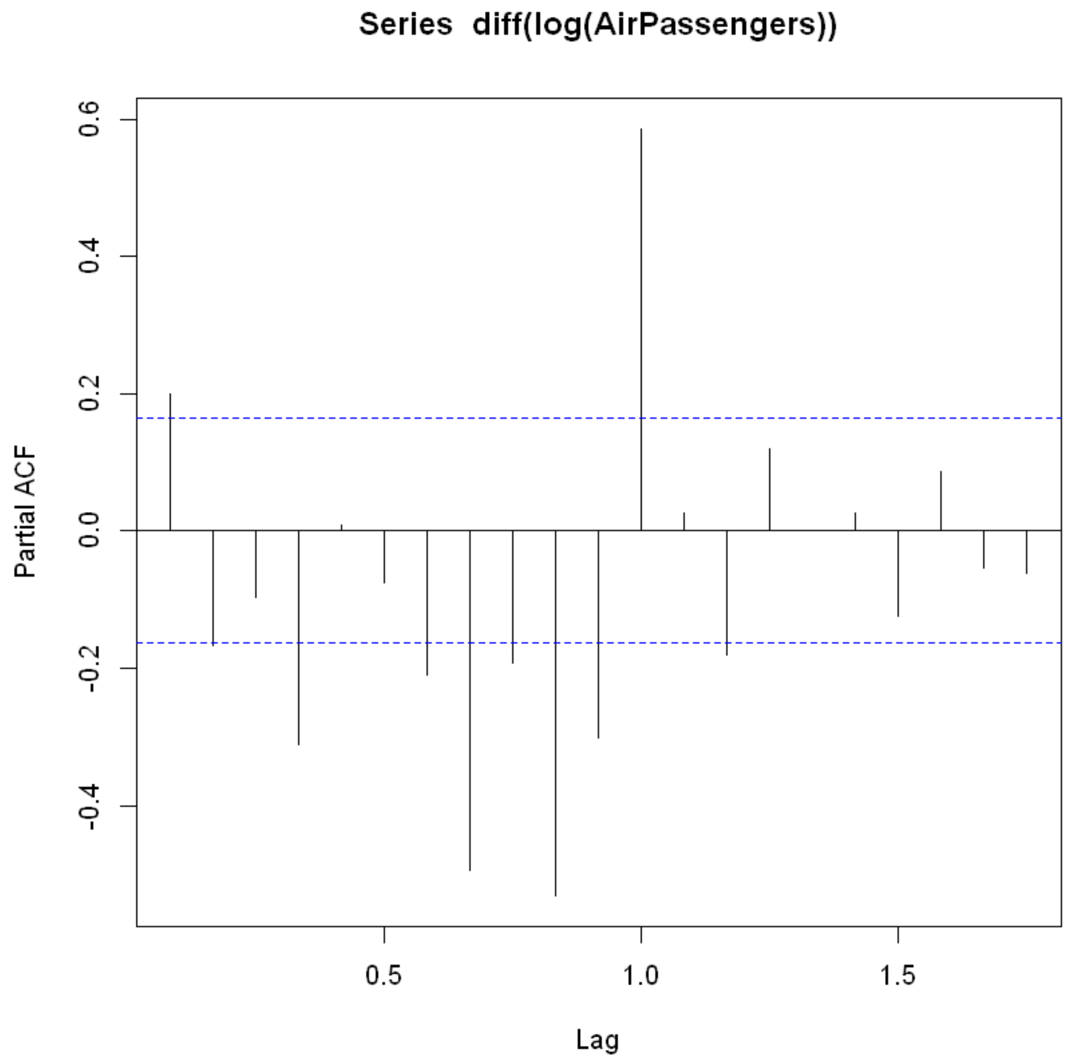


[24]:
```
acf(diff(log(AirPassengers)))
```



Partial auto correlation function, as the word suggests, is partial not complete. Here again we are plot the correlations at various lags 1,2,3 BUT after adjusting for the effects of intermediate numbers.

[25]:
```
pacf(diff(log(AirPassengers)))
```

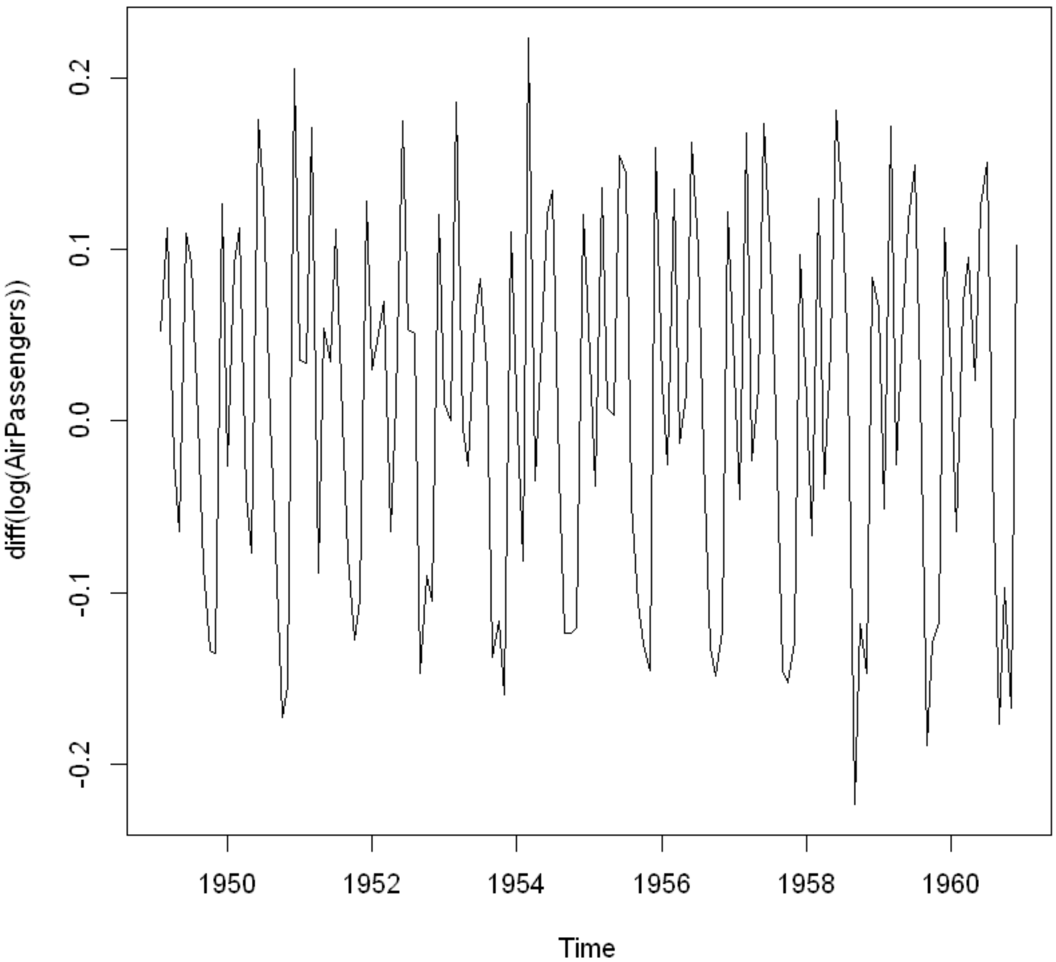


It determine value of p (value we got as 0)

d is number of time you do the differentiations to make the mean

We do diff only one time so value of d is 1

[26]:
```
plot(diff(log(AirPassengers)))
```



0.6 ARIMA MODEL PREDICTION

[27]:
```
fit <- arima(log(AirPassengers),c(0,1,1),seasonal =
 ↪list(order=c(0,1,1),period=12))
```

[28]:
```
fit
```

```
Call:
arima(x = log(AirPassengers), order = c(0, 1, 1), seasonal = list(order = c(0,
    1, 1), period = 12))

Coefficients:
```

```
         ma1      sma1
      -0.4018  -0.5569
s.e.   0.0896   0.0731

sigma^2 estimated as 0.001348:  log likelihood = 244.7,  aic = -483.4
```

0.6.1 Predict for next 10 years

[29]:
```
pred <- predict(fit,n.ahead=10*12) #10 years * 12 months
pred
```

$pred A Time Series: 10 × 12

	Jan	Feb	Mar	Apr	May	Jun	Jul
1961	6.110186	6.053775	6.171715	6.199300	6.232556	6.368779	6.50729
1962	6.206435	6.150025	6.267964	6.295550	6.328805	6.465028	6.60354
1963	6.302684	6.246274	6.364213	6.391799	6.425054	6.561277	6.69979
1964	6.398933	6.342523	6.460463	6.488048	6.521304	6.657526	6.79604
1965	6.495183	6.438772	6.556712	6.584297	6.617553	6.753776	6.89229
1966	6.591432	6.535022	6.652961	6.680547	6.713802	6.850025	6.98854
1967	6.687681	6.631271	6.749210	6.776796	6.810051	6.946274	7.08478
1968	6.783930	6.727520	6.845460	6.873045	6.906301	7.042523	7.18103
1969	6.880180	6.823769	6.941709	6.969294	7.002550	7.138773	7.27728
1970	6.976429	6.920019	7.037958	7.065544	7.098799	7.235022	7.37353

$se A Time Series: 10 × 12

	Jan	Feb	Mar	Apr	May	Jun
1961	0.03671562	0.04278291	0.04809072	0.05286830	0.05724856	0.0613167
1962	0.09008475	0.09549708	0.10061869	0.10549195	0.11014981	0.1146185
1963	0.14650643	0.15224985	0.15778435	0.16313118	0.16830825	0.1733307
1964	0.20896657	0.21513653	0.22113442	0.22697386	0.23266679	0.2382237
1965	0.27748210	0.28408309	0.29053414	0.29684503	0.30302451	0.3090804
1966	0.35174476	0.35876289	0.36564634	0.37240257	0.37903840	0.3855600
1967	0.43142043	0.43883816	0.44613258	0.45330963	0.46037481	0.4673331
1968	0.51620376	0.52400376	0.53168935	0.53926541	0.54673651	0.5541068
1969	0.60582584	0.61399203	0.62205103	0.63000694	0.63786363	0.6456247
1970	0.70005133	0.70856907	0.71698563	0.72530453	0.73352910	0.7416624

The above output prediction value are in logarithemic part,convert them to original form we need to transform them. 2.718 is e value and round them to 0 decimal

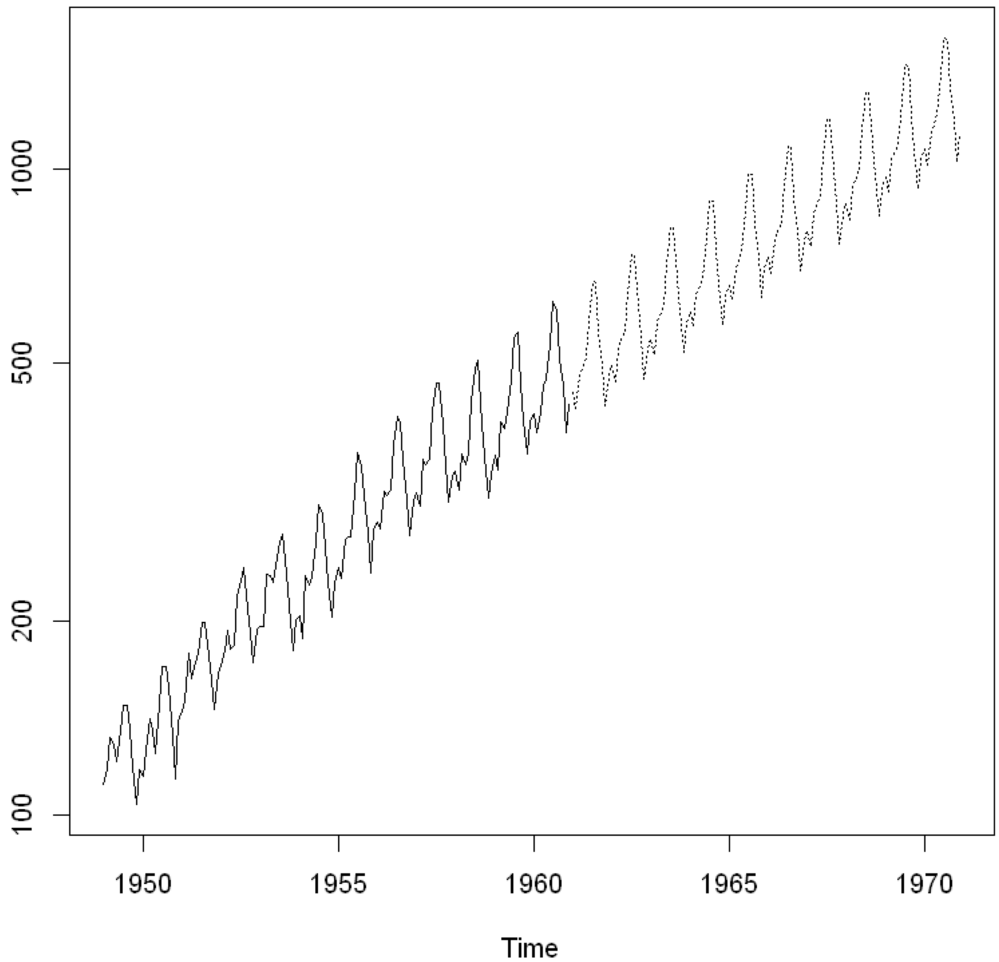
[30]:
```
pred1<-round(2.718^pred$pred,0)
pred1
```

A Time Series: 10×12

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1961	450	425	479	492	509	583	670	667	558	497	430	477
1962	496	468	527	542	560	642	737	734	614	547	473	525
1963	546	516	580	597	617	707	812	808	676	602	521	578
1964	601	568	639	657	679	778	894	890	745	663	573	637
1965	661	625	703	723	748	857	984	980	820	730	631	701
1966	728	688	775	796	823	943	1083	1079	903	804	695	772
1967	802	758	853	877	906	1039	1193	1188	994	885	765	850
1968	883	834	939	965	998	1143	1313	1308	1094	975	843	935
1969	972	919	1034	1063	1099	1259	1446	1440	1205	1073	928	1030
1970	1070	1012	1138	1170	1210	1386	1592	1585	1326	1181	1021	1134

line type (lty) can be specified using either text ("blank", "solid", "dashed", "dotted", "dotdash", "longdash", "twodash") or number (0, 1, 2, 3, 4, 5, 6). Note that lty = "solid" is identical to lty=1.

[31]:
```
ts.plot(AirPassengers,pred1,log="y",lty=c(1,3))
```



0.6.2 Get only 1961 values

[32]:
```
data1<-head(pred1,12)
data1
```

A Time Series: 1 × 12

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1961	450	425	479	492	509	583	670	667	558	497	430	477

0.6.3 Predicted outcome in 1960 row

[33]:
```
predicted_1960 <- round(data1)#head of Predicted
predicted_1960
```

A Time Series: 1 × 12

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1961	450	425	479	492	509	583	670	667	558	497	430	477

0.6.4 Actual outcome in 1960 row

[34]:
```
original_1960 <- tail(AirPassengers,12) #tail of original
original_1960
```

A Time Series: 1 × 12

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1960	417	391	419	461	472	535	622	606	508	461	390	432

Lets Test this Model we are going to take a dataset till 1959, and then we predict value of 1960, then validate that 1960 from alredy existing value we have it in dataset

0.6.5 Recreate model till 1959

[35]:
```
datawide <- ts(AirPassengers, frequency = 12, start=c(1949,1), end=c(1959,12))
datawide
```

A Time Series: 11 × 12

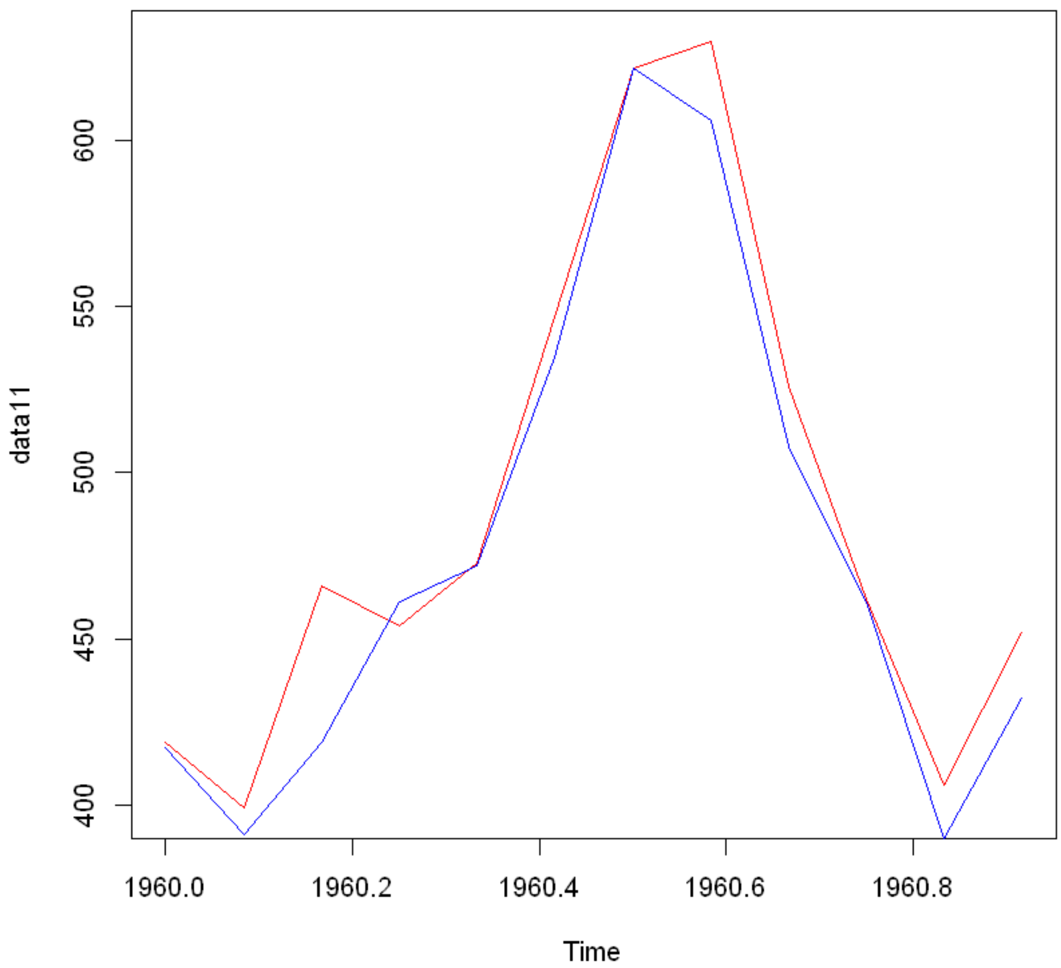
	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1949	112	118	132	129	121	135	148	148	136	119	104	118
1950	115	126	141	135	125	149	170	170	158	133	114	140
1951	145	150	178	163	172	178	199	199	184	162	146	166
1952	171	180	193	181	183	218	230	242	209	191	172	194
1953	196	196	236	235	229	243	264	272	237	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

0.6.6 give op of 1960 to 1970

[36]:
```
fit1 <- arima(log(datawide),c(0,1,1),seasonal = list(order=c(0,1,1),period=12))
pred <- predict(fit1,n.ahead=10*12)  # predictfor now 1960 to 1970
pred1<-2.718^pred$pred
pred1
```

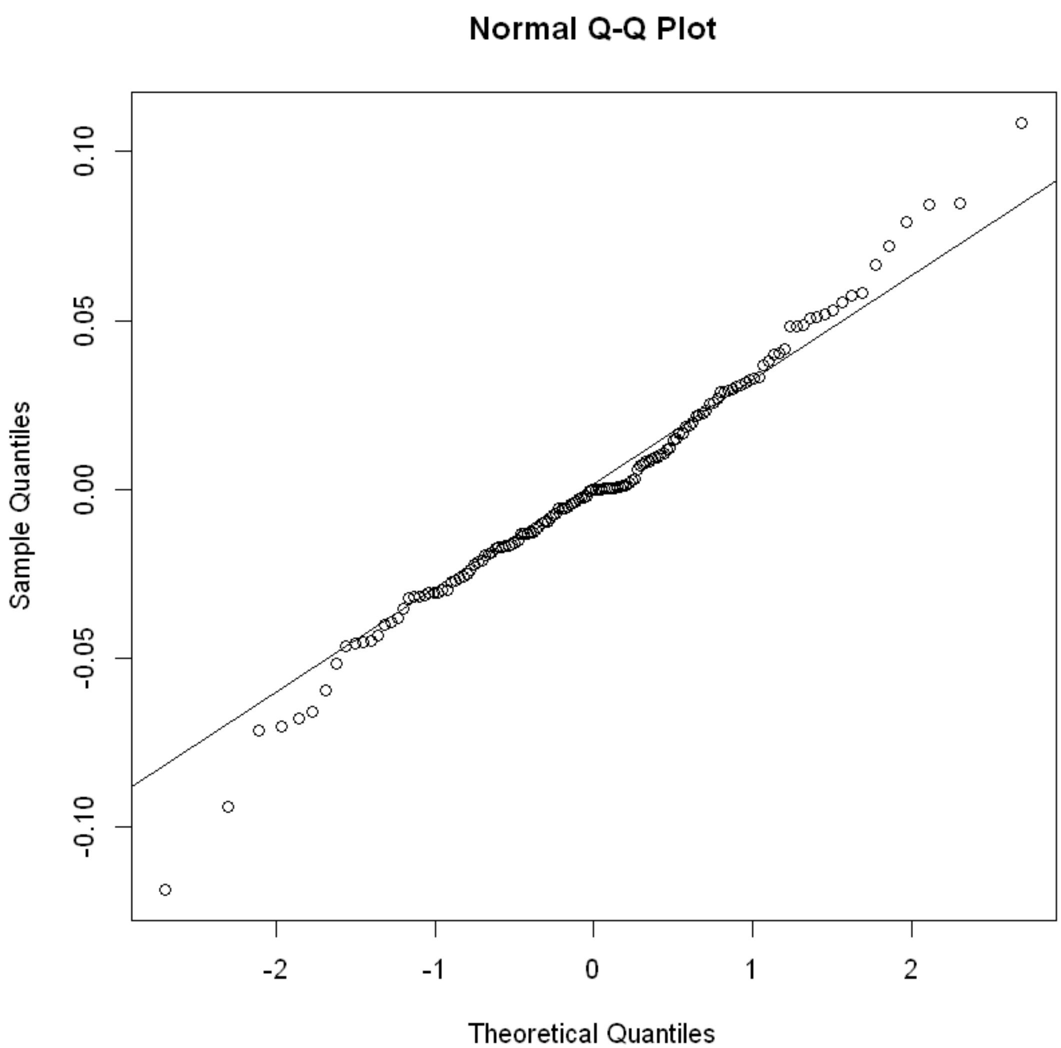
A Time Series: 10 × 12		Jan	Feb	Mar	Apr	May	Jun	Jul
	1960	419.0628	398.6732	466.2820	454.1188	472.9611	546.7614	621.8017
	1961	469.5742	446.7270	522.4849	508.8558	529.9692	612.6649	696.7502
	1962	526.1740	500.5730	585.4623	570.1903	593.8487	686.5121	780.7325
	1963	589.5961	560.9092	656.0306	638.9179	665.4278	769.2603	874.8376
	1964	660.6626	628.5180	735.1048	715.9294	745.6347	861.9826	980.2855
	1965	740.2952	704.2760	823.7102	802.2235	835.5093	965.8811	1098.443
	1966	829.5262	789.1655	922.9956	898.9189	936.2168	1082.3030	1230.843
	1967	929.5126	884.2870	1034.2482	1007.2695	1049.0631	1212.7576	1379.202
	1968	1041.5507	990.8740	1158.9107	1128.6801	1175.5113	1358.9366	1545.444
	1969	1167.0934	1110.3083	1298.5992	1264.7248	1317.2007	1522.7351	1731.723

[39]:
```
data11=round(head(pred1,12),0) #head of Predicted
data22=round(tail(AirPassengers,12),0) #tail of original
plot(data11,col="red", type="l")
lines(data22,col="blue")
```



0.6.7 CHECK NORMALITY USING Q-Q PLOT

[42]:
```
qqnorm(residuals(fit))
qqline(residuals(fit))
```



[ ]: