Practical no 9

Aim: Demonstration of Time-series forecasting

Theory:

Time series forecasting is a method in the statistics field to analyze historical data with a time component and

create a prediction based on it.

Some classic examples of time series forecasting methods are Moving Average, ARIMA, and Exponential

Smoothing. These methods have been used for a long time and are still useful now because of how easy it is

for users to explain the result — although with less accurate prediction.

Whether you need a classical approach or a machine learning-driven model, many have developed Python

packages to access all these methods. Some of the famous packages are Statsmodel, pmdarima, and sktime.

However, the forecasting model is not limited to only the ones that are listed above because many great

packages are worthy of consideration.

1. StatsForecast – It is a Python package that provides a collection of univariate time-series Forecasting

models. What is unique about StatsForecast is the model provides fast training and is optimized for

high-accuracy models. Also, the package offers several benchmarks we could use when training

various models.

2. PyAF - PyAF or Python Automatic Forecasting is an open-source Python package to automatically

develop time-series forecasting models (either univariate or with exogenous data). The model was

built on top of Scikit-Learn and Pandas, so expect familiar APIs. The package also offers various

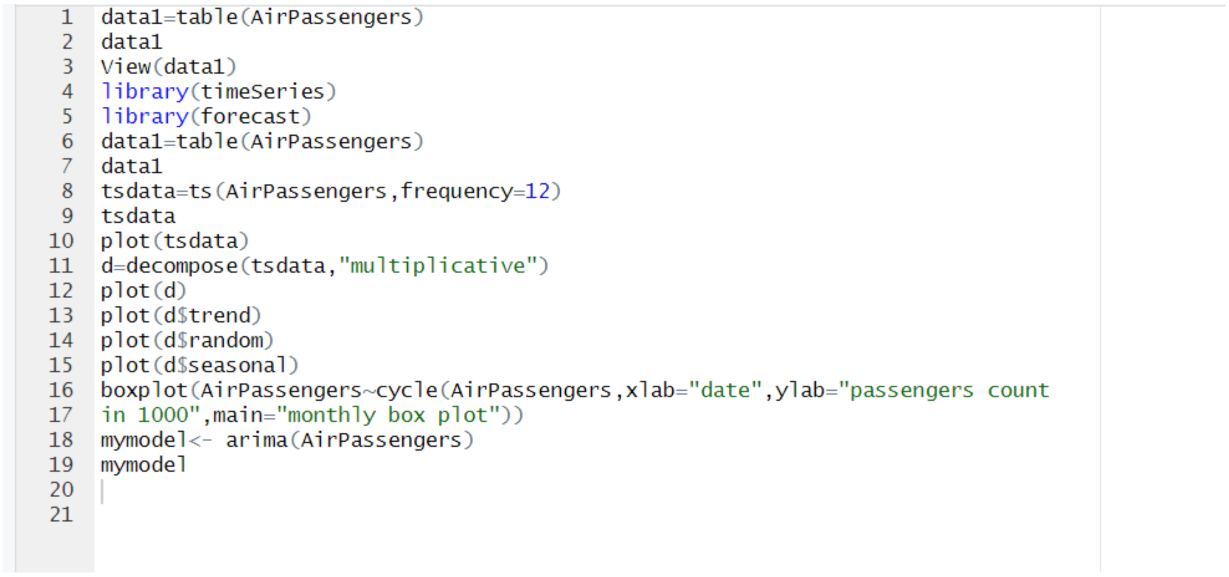
models to use in a few lines as much as possible.

3. NeuralProphet - It is a Python Package for developing a time-series model based on Facebook Prophet

but with Neural Network architecture. The package is based on PyTorch and could easily be used with

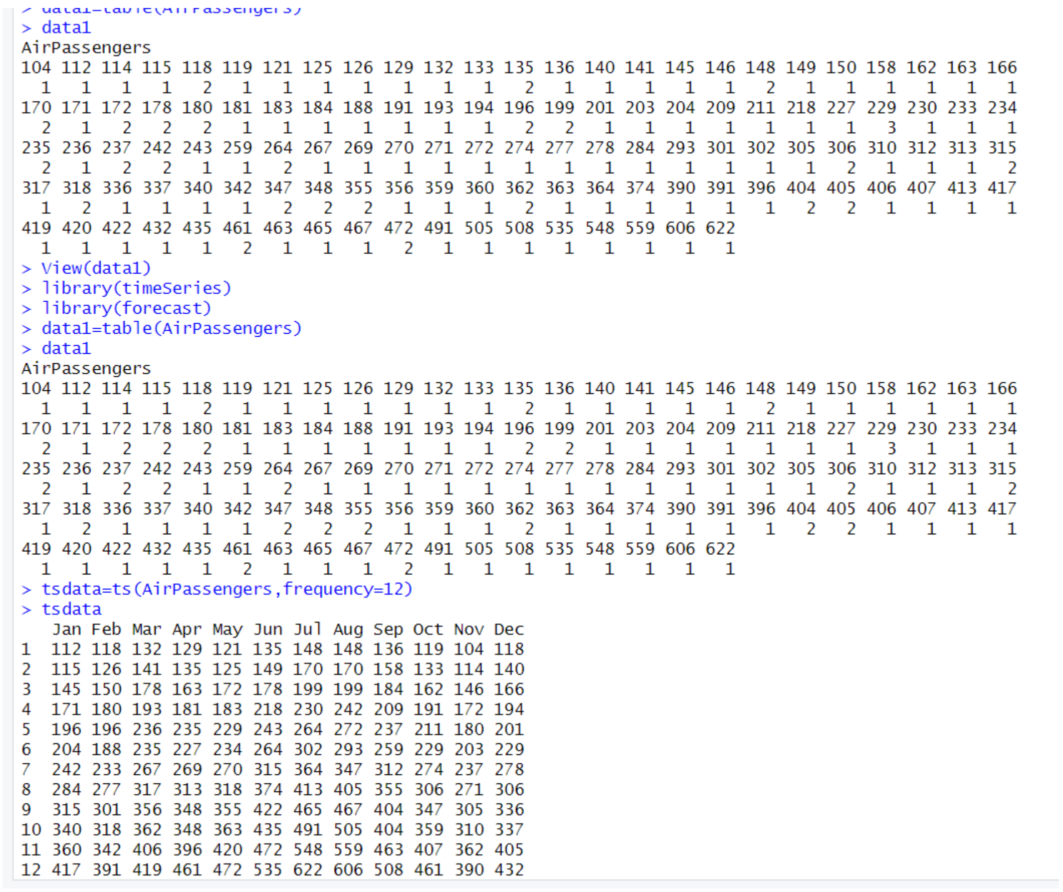
as few lines as possible.

Code:

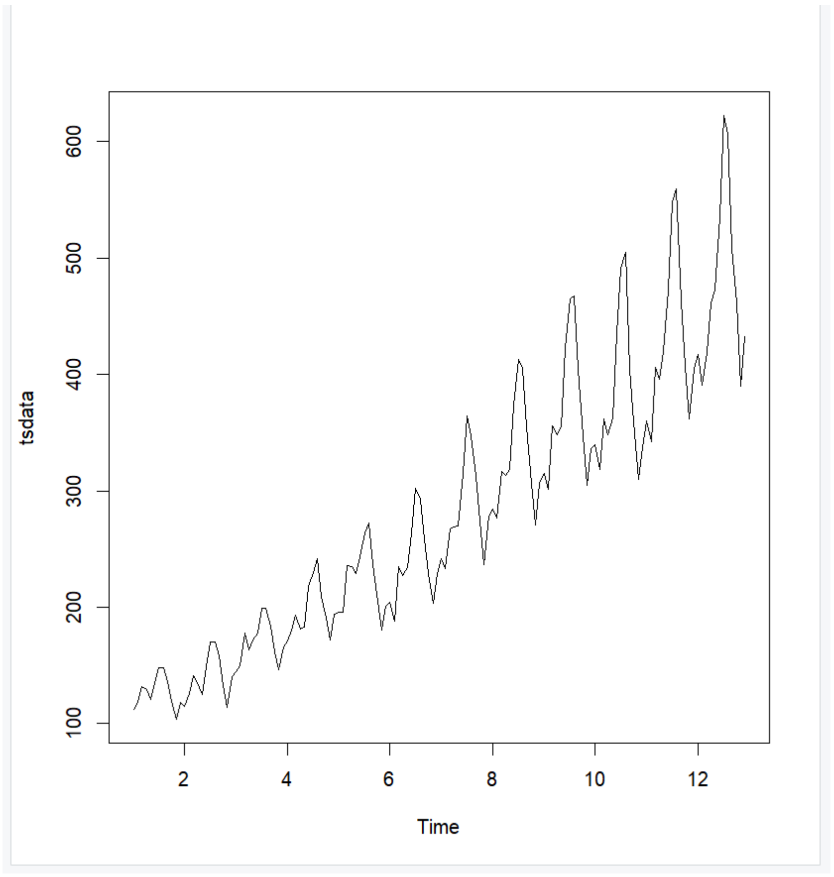


```
data1=table(AirPassengers)
data1
View(data1)
library(timeSeries)
library(forecast)
data1=table(AirPassengers)
data1
tsdata=ts(AirPassengers,frequency=12)
tsdata
plot(tsdata)
d=decompose(tsdata,"multiplicative")
plot(d)
plot(d$trend)
plot(d$random)
plot(d$seasonal)
boxplot(AirPassengers~cycle(AirPassengers,xlab="date",ylab="passengers count
in 1000",main="monthly box plot"))
mymodel<- arima(AirPassengers)
mymodel
```

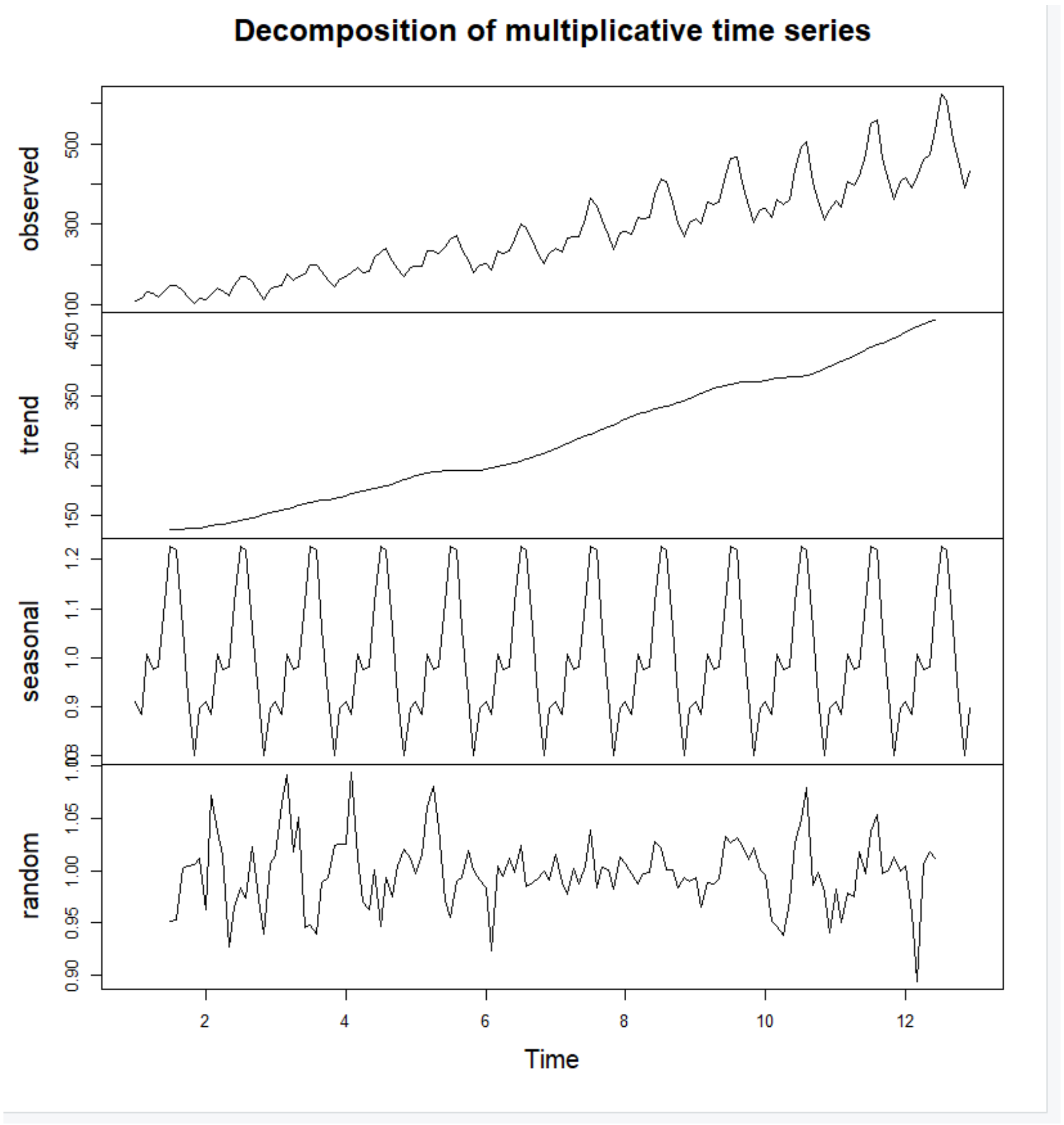
Output:

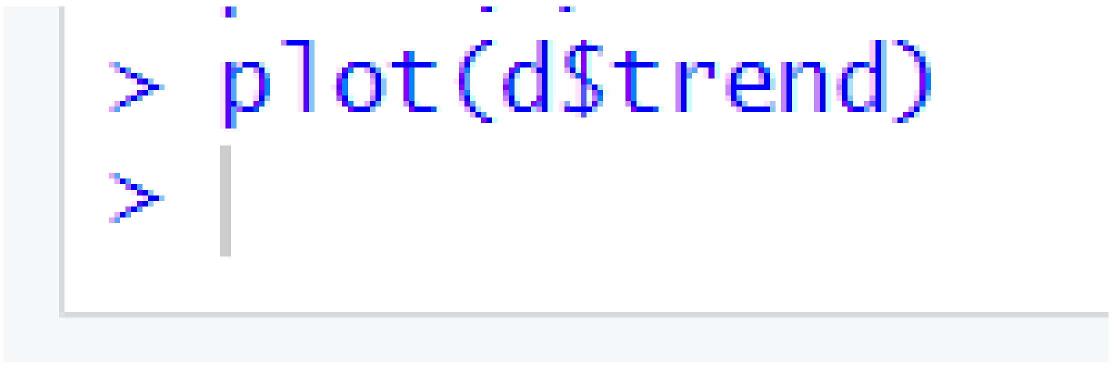


```
> data1=table(AirPassengers)
> data1
AirPassengers
104 112 114 115 118 119 121 125 126 129 132 133 135 136 140 141 145 146 148 149 150 158 162 163 166 
  1   1   1   1   2   1   1   1   1   1   1   1   2   1   1   1   1   1   2   1   1   1   1   1   1 
170 171 172 178 180 181 183 184 188 191 193 194 196 199 201 203 204 209 211 218 227 229 230 233 234 
  2   1   2   2   2   1   1   1   1   1   1   1   2   2   1   1   1   1   1   1   1   3   1   1   1 
235 236 237 242 243 259 264 267 269 270 271 272 274 277 278 284 293 301 302 305 306 310 312 313 315 
  2   1   2   2   1   1   2   1   1   1   1   1   1   1   1   1   1   1   1   1   2   1   1   1   2 
317 318 336 337 340 342 347 348 355 356 359 360 362 363 364 374 390 391 396 404 405 406 407 413 417 
  1   2   1   1   1   1   2   2   2   1   1   1   2   1   1   1   1   1   1   2   2   1   1   1   1 
419 420 422 432 435 461 463 465 467 472 491 505 508 535 548 559 606 622 
  1   1   1   1   1   2   1   1   1   2   1   1   1   1   1   1   1   1 
> View(data1)
> library(timeSeries)
> library(forecast)
> data1=table(AirPassengers)
> data1
AirPassengers
104 112 114 115 118 119 121 125 126 129 132 133 135 136 140 141 145 146 148 149 150 158 162 163 166 
  1   1   1   1   2   1   1   1   1   1   1   1   2   1   1   1   1   1   2   1   1   1   1   1   1 
170 171 172 178 180 181 183 184 188 191 193 194 196 199 201 203 204 209 211 218 227 229 230 233 234 
  2   1   2   2   2   1   1   1   1   1   1   1   2   2   1   1   1   1   1   1   1   3   1   1   1 
235 236 237 242 243 259 264 267 269 270 271 272 274 277 278 284 293 301 302 305 306 310 312 313 315 
  2   1   2   2   1   1   2   1   1   1   1   1   1   1   1   1   1   1   1   1   2   1   1   1   2 
317 318 336 337 340 342 347 348 355 356 359 360 362 363 364 374 390 391 396 404 405 406 407 413 417 
  1   2   1   1   1   1   2   2   2   1   1   1   2   1   1   1   1   1   1   2   2   1   1   1   1 
419 420 422 432 435 461 463 465 467 472 491 505 508 535 548 559 606 622 
  1   1   1   1   1   2   1   1   1   2   1   1   1   1   1   1   1   1 
> tsdata=ts(AirPassengers,frequency=12)
> tsdata
   Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1  112 118 132 129 121 135 148 148 136 119 104 118
2  115 126 141 135 125 149 170 170 158 133 114 140
3  145 150 178 163 172 178 199 199 184 162 146 166
4  171 180 193 181 183 218 230 242 209 191 172 194
5  196 196 236 235 229 243 264 272 237 211 180 201
6  204 188 235 227 234 264 302 293 259 229 203 229
7  242 233 267 269 270 315 364 347 312 274 237 278
8  284 277 317 313 318 374 413 405 355 306 271 306
9  315 301 356 348 355 422 465 467 404 347 305 336
10 340 318 362 348 363 435 491 505 404 359 310 337
11 360 342 406 396 420 472 548 559 463 407 362 405
12 417 391 419 461 472 535 622 606 508 461 390 432
```

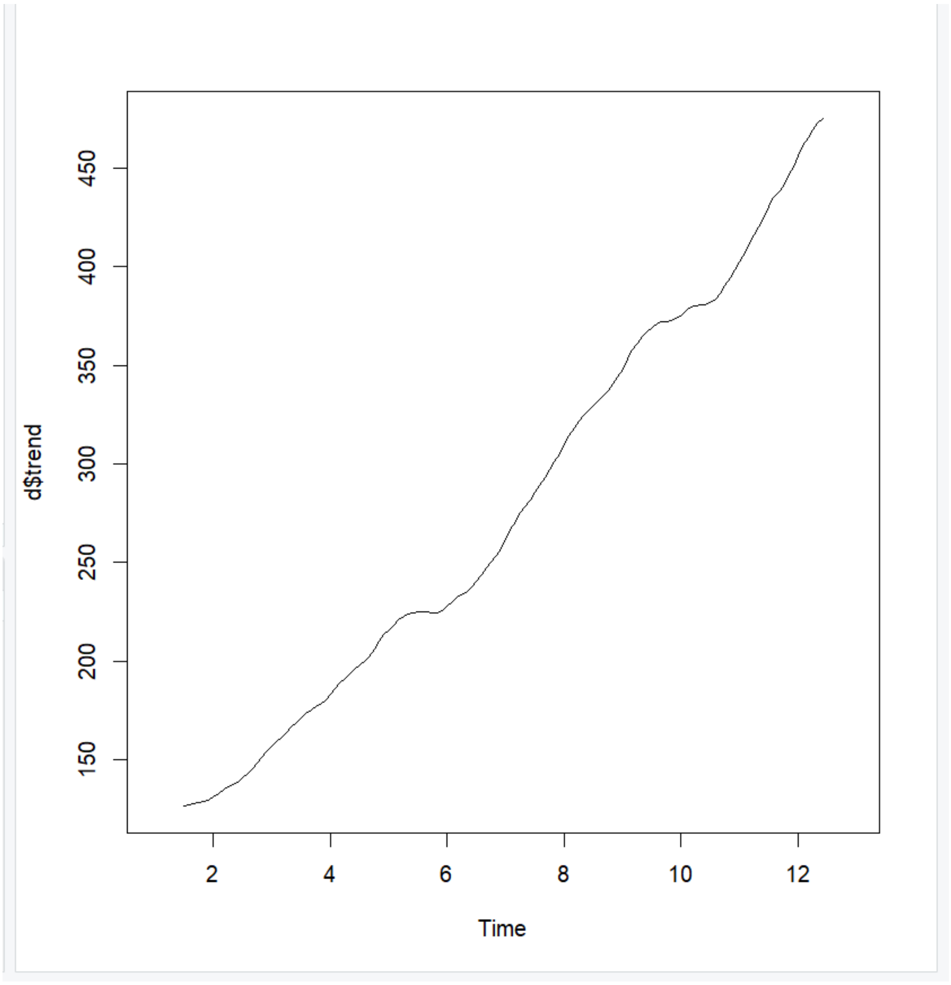


```
> d=decompose(tsdata,"multiplicative")
> plot(d)
>
```

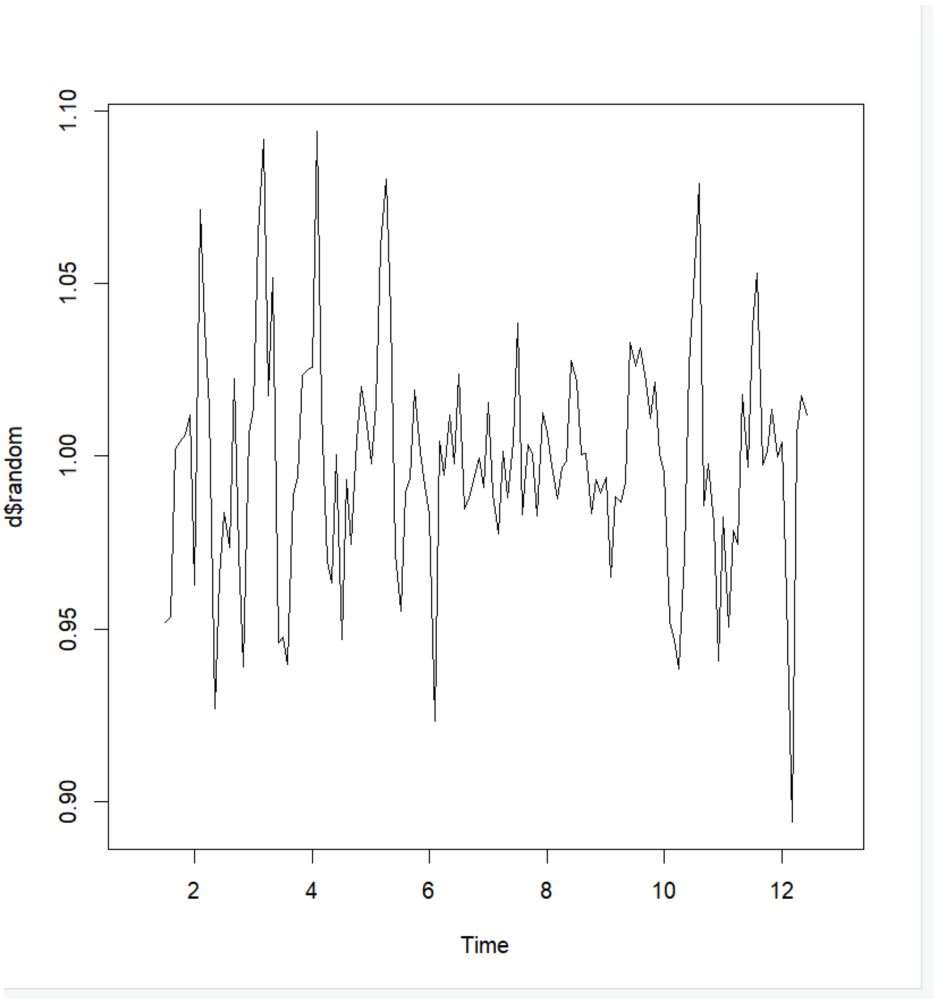

Decomposition of multiplicative time series
observed
trend
seasonal
random
Time

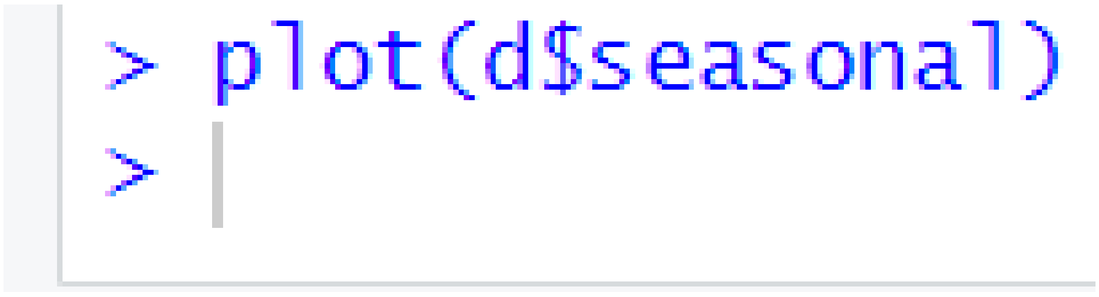


```
> plot(d$trend)
> 
```

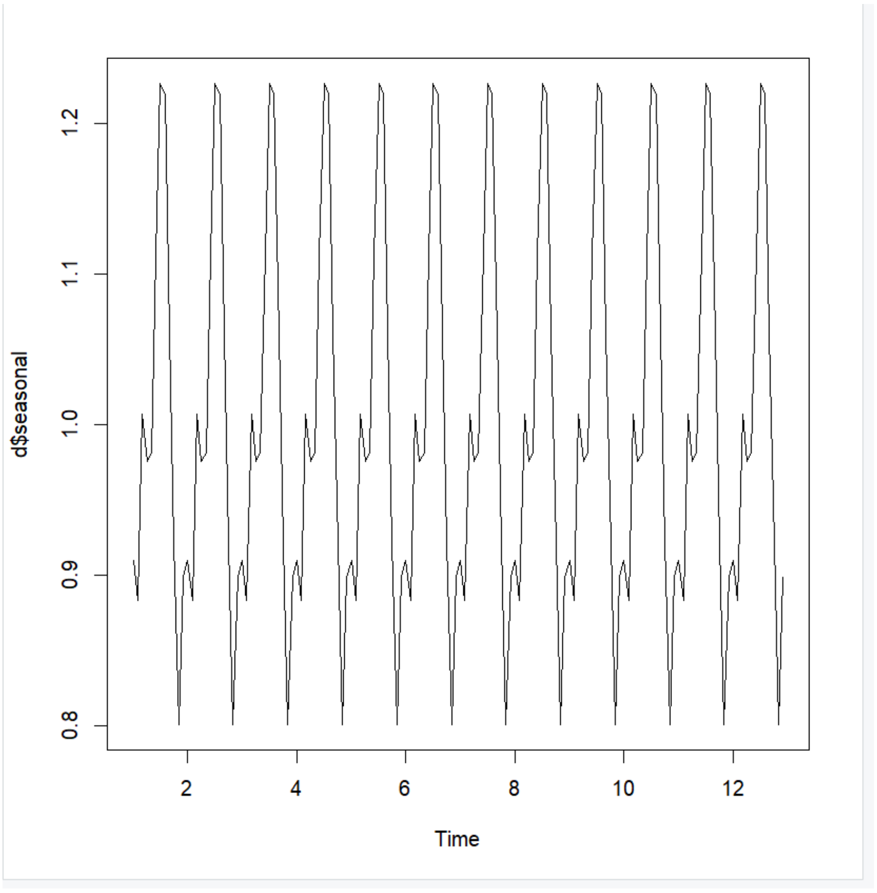


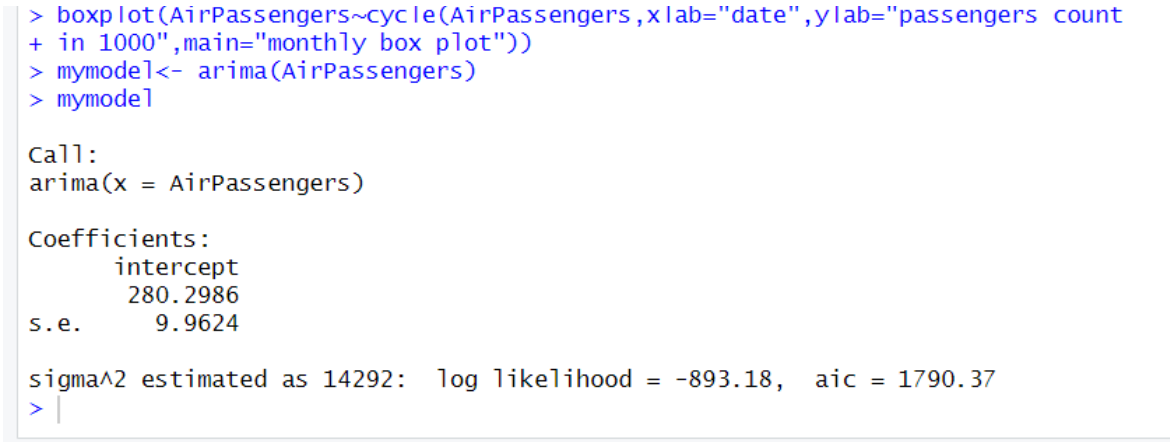
```
> plot(d$random)
>
```





```
> plot(d$seasonal)
> 
```



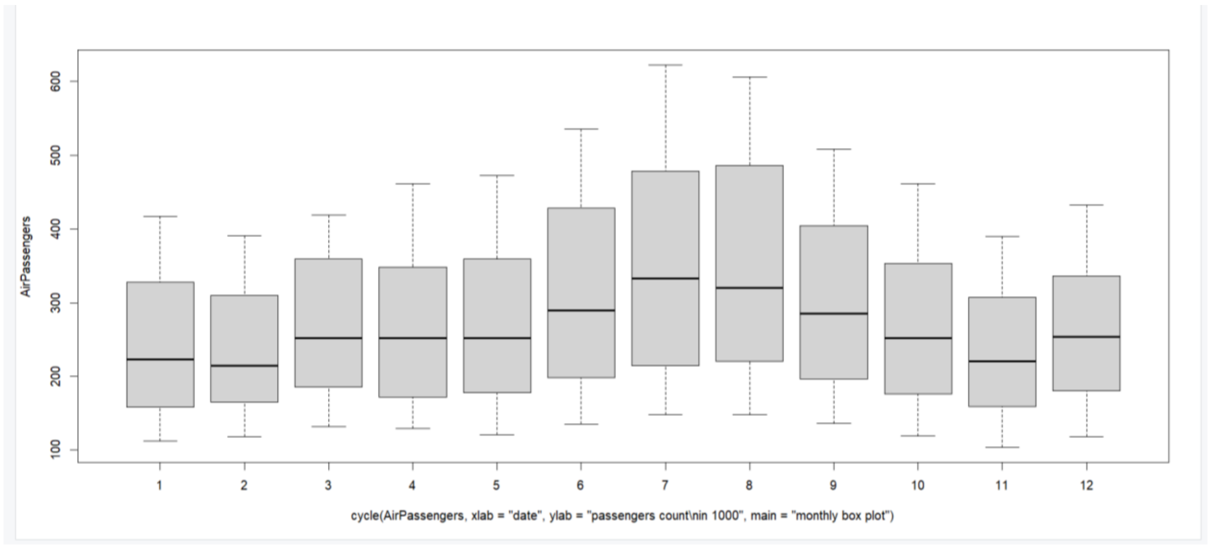


```
> boxplot(AirPassengers~cycle(AirPassengers,xlab="date",ylab="passengers count
+ in 1000",main="monthly box plot"))
> mymodel<- arima(AirPassengers)
> mymodel

Call:
arima(x = AirPassengers)

Coefficients:
      intercept
       280.2986
s.e.     9.9624

sigma^2 estimated as 14292:  log likelihood = -893.18,  aic = 1790.37
> |
```



Conclusion: Hence we have successfully learnt and performed Time-series forecasting.