Final project report

Initial Project Report

1. Introduction (5 points) – Text may be re-used from the proposal

a. Context and background

This is a trend analysis of flight, passenger, load factor and operating revenue data of USA domestic airlines. I was always interested in studying airline industry as airline has always been a struggling industry in terms of revenue because they depend a lot on uncertain and uncontrollable factors like jetline fuel prices, load factor, climatic conditions and a stiff competition. So, I wanted to do some trend analysis with airline data.

b. Goals of the project

Forecasting the number of flights, load_factor, passengers and operating revenue of domestic airlines in USA. I. The forecast of the number of passengers and load factor can be useful for the carriers to plan the number of flights they need to operate to cater to the demand. II. For passengers, it can be useful in planning their booking beforehand if they know how many flights are going to operate in the coming months and what is the load factor and they whether should wait till the last moment to book the flights. For example- during peak seasons, flights generally get booked very soon and if the forecast says for a particular season in the year, the number of operating flights is going to be reduced, they need to be wary of this and plan beforehand. From a passenger perspective, a more useful forecast would be the forecast of fare, the proportion of canceled flights. If I find that data, I will try to use it in the project. III. We can understand the relationship between the number of flights operating and the number of passengers. If a significant increase in the number of passengers is not followed by a significant increase in flights, it means that a lot of previous flights were not getting fully booked, and hence the operating revenue would also be lower or it could simply mean that the load factor has improved.

c. Data description: sources of data, time period(s) represented

Dataset	Frequency	# Data points	Start date	End date
1. Passengers	Monthly	213	10/2002	12/2021
2. Flights	Monthly	213	10/2002	12/2021
3. Operating revenue	Quarterly	87	Q1 2000	Q3 2021
4. Load factor	Monthly	213	10/2002	12/2021

Figure 1: data summary

For 3 datasets, the data was on monthly level. So I rolled it up on quarterly level and all my analysis are based on quarterly data.

2. Exploratory analysis (10 points) - For each time series:

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.1.3
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## v purrr   0.3.4
```

```
## Warning: package 'tibble' was built under R version 4.1.3
```

```
## Warning: package 'tidyr' was built under R version 4.1.3
```

```
## Warning: package 'readr' was built under R version 4.1.3
```

```
## Warning: package 'purrr' was built under R version 4.1.3
```

```
## Warning: package 'dplyr' was built under R version 4.1.3
```

```
## Warning: package 'forcats' was built under R version 4.1.3
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
flight <- read.csv('Flight.csv')
passengers <- read.csv('Passengers.csv')
load_factor <- read.csv('Load factor.csv')
op_rev <- read.csv('Operating Rev.csv')
head(flight)
```

```
##   Year Month DOMESTIC INTERNATIONAL  TOTAL
## 1 2002    10   815032         53708 868740
## 2 2002    11   766327         53279 819606
## 3 2002    12   781653         57219 838872
## 4 2003     1   785160         57667 842827
## 5 2003     2   690351         51259 741610
## 6 2003     3   797194         58926 856120
```

```r
head(passengers)
```

```
##   Year Month DOMESTIC INTERNATIONAL    TOTAL
## 1 2002    10 48054917       9578435 57633352
## 2 2002    11 44850246       9016535 53866781
## 3 2002    12 49684353      10038794 59723147
## 4 2003     1 43032450       9726436 52758886
## 5 2003     2 41166780       8283372 49450152
## 6 2003     3 49992700       9538653 59531353
```

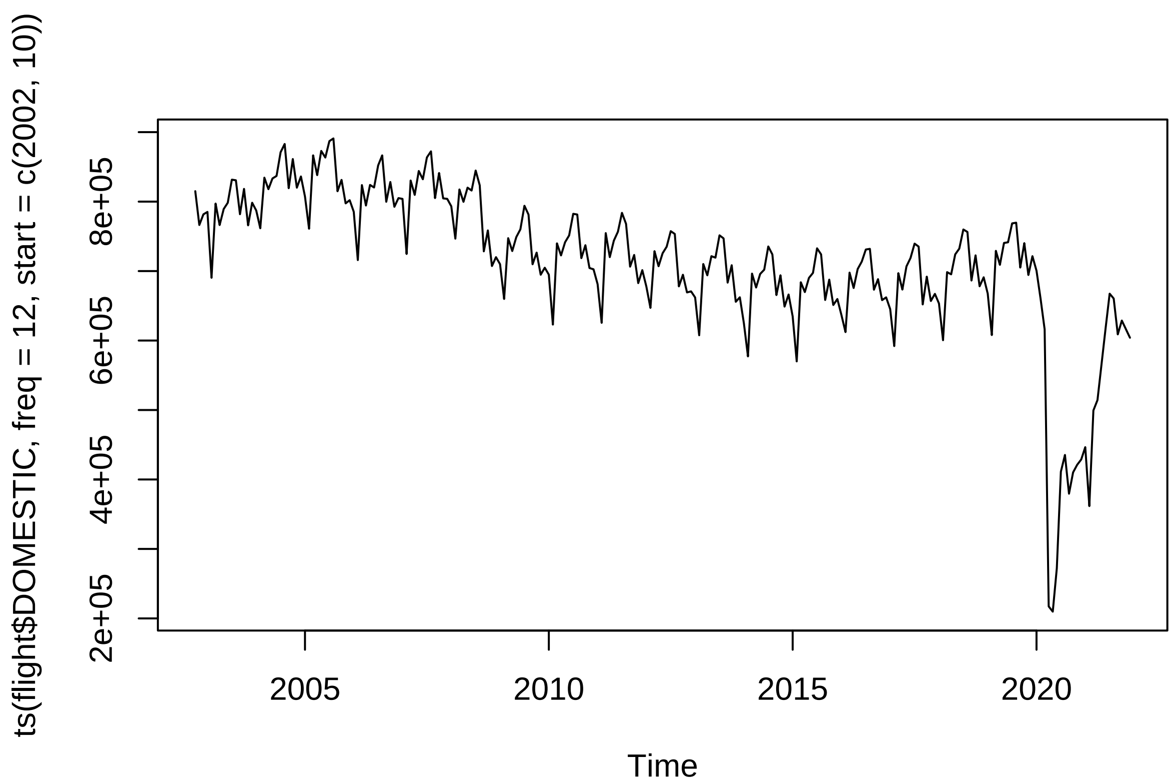
```r
head(load_factor)
```

```
##   Year Month DOMESTIC INTERNATIONAL TOTAL
## 1 2002    10       68            73    69
## 2 2002    11       67            70    67
## 3 2002    12       73            75    73
## 4 2003     1       64            72    66
## 5 2003     2       68            69    68
## 6 2003     3       73            71    72
```

```r
head(op_rev)
```

```
##   Year Quarter DOMESTIC LATIN.AMERICA ATLANTIC PACIFIC INTERNATIONAL    TOTAL
## 1 2000       1 23262183       1639679  2690229 2240370        370524 30202987
## 2 2000       2 25573067       1626743  3419032 2432840        390007 33441689
## 3 2000       3 25313087       1776933  3796968 2801208        370491 34058686
## 4 2000       4 24751473       1761932  3113734 2545414        372305 32544858
## 5 2001       1 23620312       1848448  2913460 2319491        409925 31111636
## 6 2001       2 23987256       1695829  3461580 2313437        343148 31801250
```

```r
plot(ts(flight$DOMESTIC,freq=12,start = c(2002,10)))
```



Data Cleaning

```
flight$Day <- 01
flight$Date<-as.Date(with(flight,paste(Year,Month,Day,sep="-")),"%Y-%m-%d")
flight$qtr<-substr(quarters(as.Date(flight$Date)), 2, 2)
passengers$Day <- 01
passengers$Date<-as.Date(with(passengers,paste(Year,Month,Day,sep="-")),"%Y-%m-%d")
passengers$qtr<-substr(quarters(as.Date(passengers$Date)), 2, 2)
load_factor$Day <- 01
load_factor$Date<-as.Date(with(load_factor,paste(Year,Month,Day,sep="-")),"%Y-%m-%d")
load_factor$qtr<-substr(quarters(as.Date(load_factor$Date)), 2, 2)
```

Converting to Time series object

```
freq <- 12
nfreq <- 4
flight_ts = ts(flight$DOMESTIC, freq = freq, start = c(2002,10),end=c(2019,12))
flight_ts <- aggregate(flight_ts, nfrequency=nfreq,mean)
start(flight_ts)
```

```
## [1] 2002    4
```

```
end(flight_ts)
```

```
## [1] 2019    4
```

```r
passengers_ts = ts(passengers$DOMESTIC, freq = freq, start = c(2002,10),end=c(2019,12))
passengers_ts <- aggregate(passengers_ts, nfrequency=nfreq,mean)
start(passengers_ts)
```

```
## [1] 2002    4
```

```r
end(passengers_ts)
```

```
## [1] 2019    4
```

```r
load_factor_ts = ts(load_factor$DOMESTIC, freq = freq, start = c(2002,10),end=c(2019,12))
load_factor_ts <- aggregate(load_factor_ts, nfrequency=nfreq,mean)
start(load_factor_ts)
```

```
## [1] 2002    4
```

```r
end(load_factor_ts)
```

```
## [1] 2019    4
```

```r
op_rev_ts = ts(op_rev$DOMESTIC, freq = 4, start = c(2000,01),end=c(2019,12))
op_rev_ts=window(op_rev_ts, start=c(2002, 04), end=c(2019,04))
start(op_rev_ts)
```

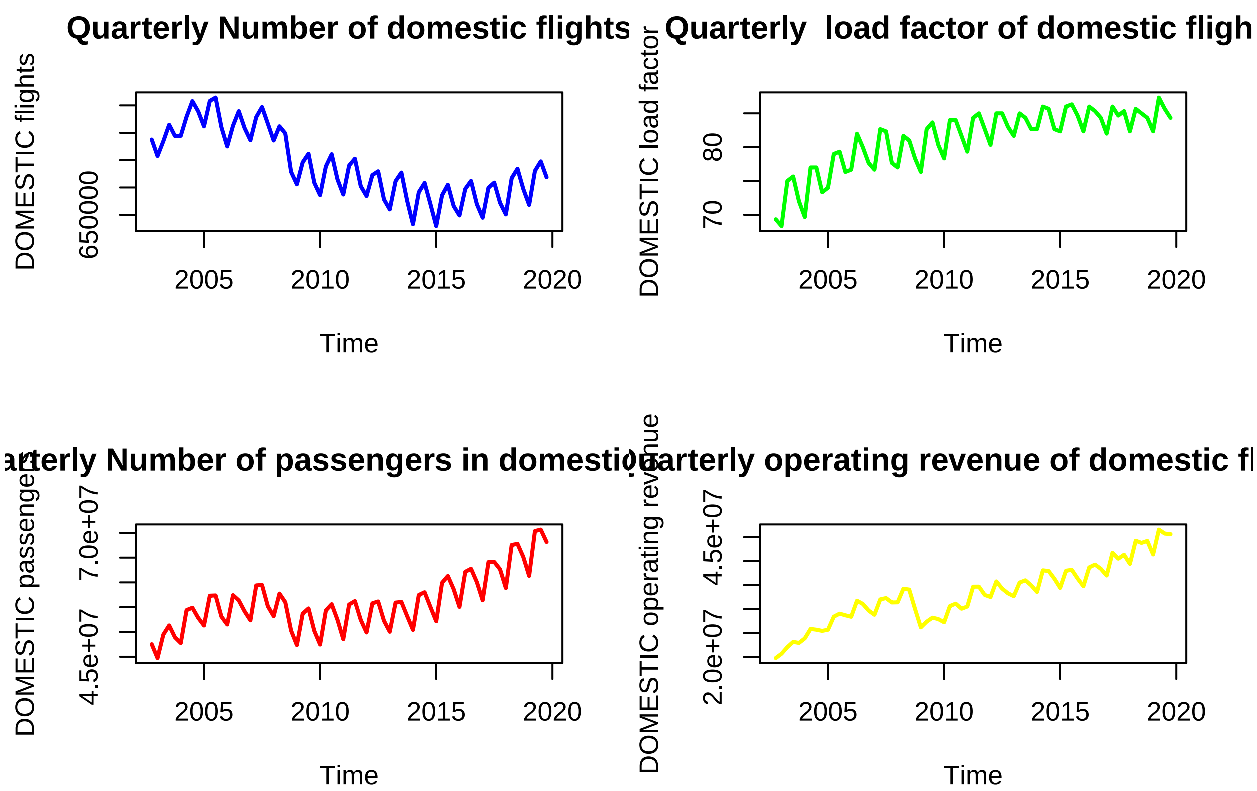
```
## [1] 2002    4
```

```r
end(op_rev_ts)
```

```
## [1] 2019    4
```

a. Plot the series

```r
par(mfcol=c(2,2))
plot(flight_ts, col="blue", lwd=2, ylab="DOMESTIC flights", main="Quarterly Number of domestic flights")
plot(passengers_ts, col="red", lwd=2, ylab="DOMESTIC passengers", main="Quarterly Number of passengers i
plot(load_factor_ts, col="green", lwd=2, ylab="DOMESTIC load factor", main="Quarterly  load factor of do
plot(op_rev_ts, col="yellow", lwd=2, ylab="DOMESTIC operating revenue", main="Quarterly operating revenu
```



b. Describe the following:

i. Missing or unusual values

There are no missing values in the data

ii. Changes in the series pattern

All the 4 time series seem to have seasonal components. And as it was observed in the flight time series, the series has a huge dip around the year 2020 which could be possibly due to covid. So, I excluded the covid period from my analysis so that the forecast is not affected.

Defined functions for stationarity tests, decomposition, plotting differenced series and seasonally adjust series

Stationarity tests

```
stationarity.test <- function(data,lag.length=25){
a<-Box.test(data, lag=lag.length, type="Ljung-Box") # test stationary signal
b<-kpss.test(data, null="Trend")
options(warn=-1)
c<-adf.test(data)
print(a)
print(b)
print(c)
if(b$p.value > 0.05)
  {
```

```
  cat('Series is stationary\n\n\n\n')
}
else{ cat('Series is not stationary\n\n\n\n')}
}
```

Decomposition

```
decomposition <- function(data){
fit <- stl(data, s.window = "periodic")
autoplot(fit, ts.colour = 'blue')}
```

Seasonally adjust data

```
adjust.seasonality <- function(data){
data_decompose <- decompose(data)
plot(data_decompose)
data_SA <- data - data_decompose$seasonal
return(data_SA)
}
```

```
library(ggplot2)

plot.diff <- function(data,lag,difference,s_lag,s_diff,title)
  {
  cbind("seasonal lag+ extra " = diff(diff(data,lag=s_lag,differences=s_diff),lag=lag,differences=diffe
        "only seasonal lag" = diff(data,lag=s_lag,differences=s_lag)) %>%
  autoplot(facets=TRUE) +
    xlab("Year") + ylab("") +ggtitle(title)
  #return(diff(data,lag=lag,differences=difference))
}
```

Since series are both seasonal and have trends, we need to perform differencing to remove trend and seasonality and then check the ACF to deduce the parameters.

c. Evaluate stationarity using a hypothesis test (R REQUIRED)

```
cat('Stationarity tests for flight data\n')
```

```
## Stationarity tests for flight data
```

```
stationarity.test(flight_ts)
```

```
## Warning in kpss.test(data, null = "Trend"): p-value smaller than printed p-value
```

```
##
##  Box-Ljung test
##
## data:  data
## X-squared = 443.75, df = 25, p-value < 2.2e-16
```

```
##
##
##  KPSS Test for Trend Stationarity
##
## data:  data
## KPSS Trend = 0.24918, Truncation lag parameter = 3, p-value = 0.01
##
##
##  Augmented Dickey-Fuller Test
##
## data:  data
## Dickey-Fuller = -2.4543, Lag order = 4, p-value = 0.3906
## alternative hypothesis: stationary
##
## Series is not stationary
```

```
cat('Stationarity tests for passenger data\n')
```

```
## Stationarity tests for passenger data
```

```
stationarity.test(passengers_ts)
```

```
##
##  Box-Ljung test
##
## data:  data
## X-squared = 191.14, df = 25, p-value < 2.2e-16
##
##
##  KPSS Test for Trend Stationarity
##
## data:  data
## KPSS Trend = 0.31891, Truncation lag parameter = 3, p-value = 0.01
##
##
##  Augmented Dickey-Fuller Test
##
## data:  data
## Dickey-Fuller = -1.9141, Lag order = 4, p-value = 0.6101
## alternative hypothesis: stationary
##
## Series is not stationary
```

```
cat('Stationarity tests for load factor data\n')
```

```
## Stationarity tests for load factor data
```

```
stationarity.test(load_factor_ts)
```

```
##
##  Box-Ljung test
```

```
##
## data:  data
## X-squared = 287.1, df = 25, p-value < 2.2e-16
##
##
##  KPSS Test for Trend Stationarity
##
## data:  data
## KPSS Trend = 0.38263, Truncation lag parameter = 3, p-value = 0.01
##
##
##  Augmented Dickey-Fuller Test
##
## data:  data
## Dickey-Fuller = -2.0683, Lag order = 4, p-value = 0.5474
## alternative hypothesis: stationary
##
## Series is not stationary
```

```
cat('Stationarity tests for operating revenue data\n')
```

```
## Stationarity tests for operating revenue data
```

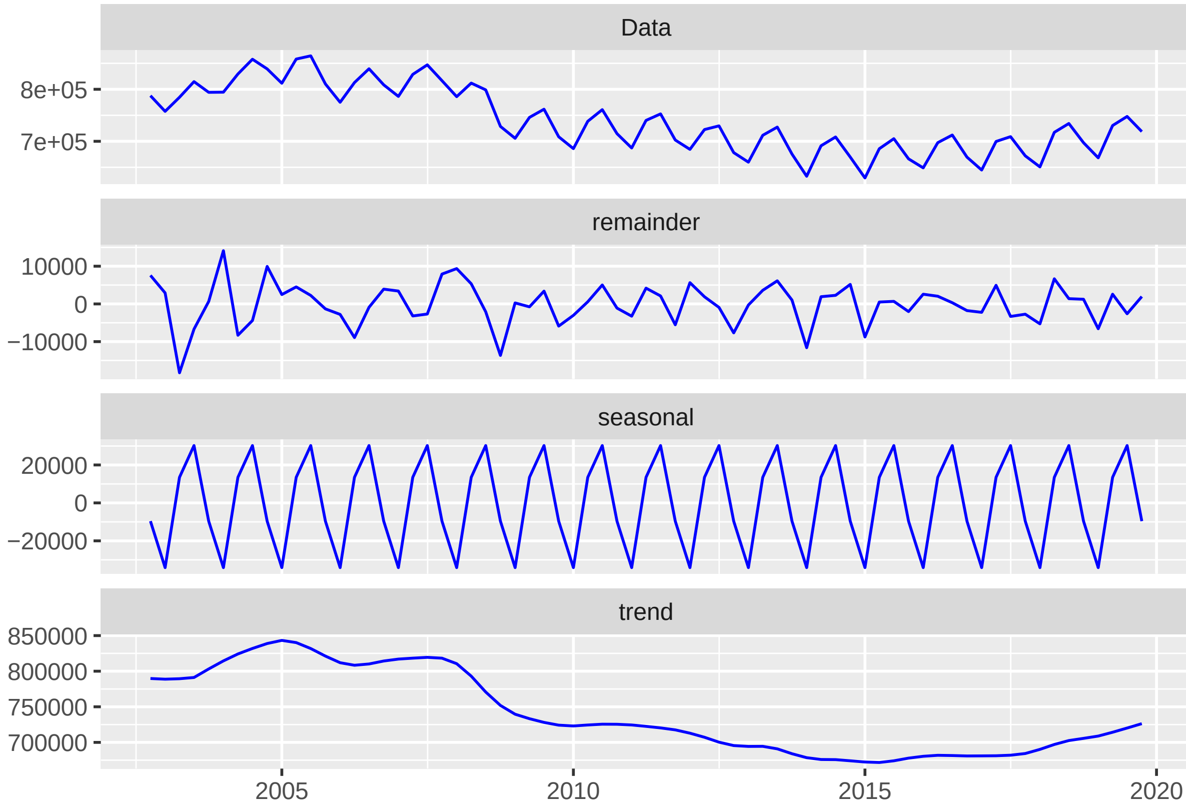
```
stationarity.test(op_rev_ts)
```

```
##
##  Box-Ljung test
##
## data:  data
## X-squared = 352.33, df = 25, p-value < 2.2e-16
##
##
##  KPSS Test for Trend Stationarity
##
## data:  data
## KPSS Trend = 0.14835, Truncation lag parameter = 3, p-value = 0.04804
##
##
##  Augmented Dickey-Fuller Test
##
## data:  data
## Dickey-Fuller = -3.9267, Lag order = 4, p-value = 0.01821
## alternative hypothesis: stationary
##
## Series is not stationary
```

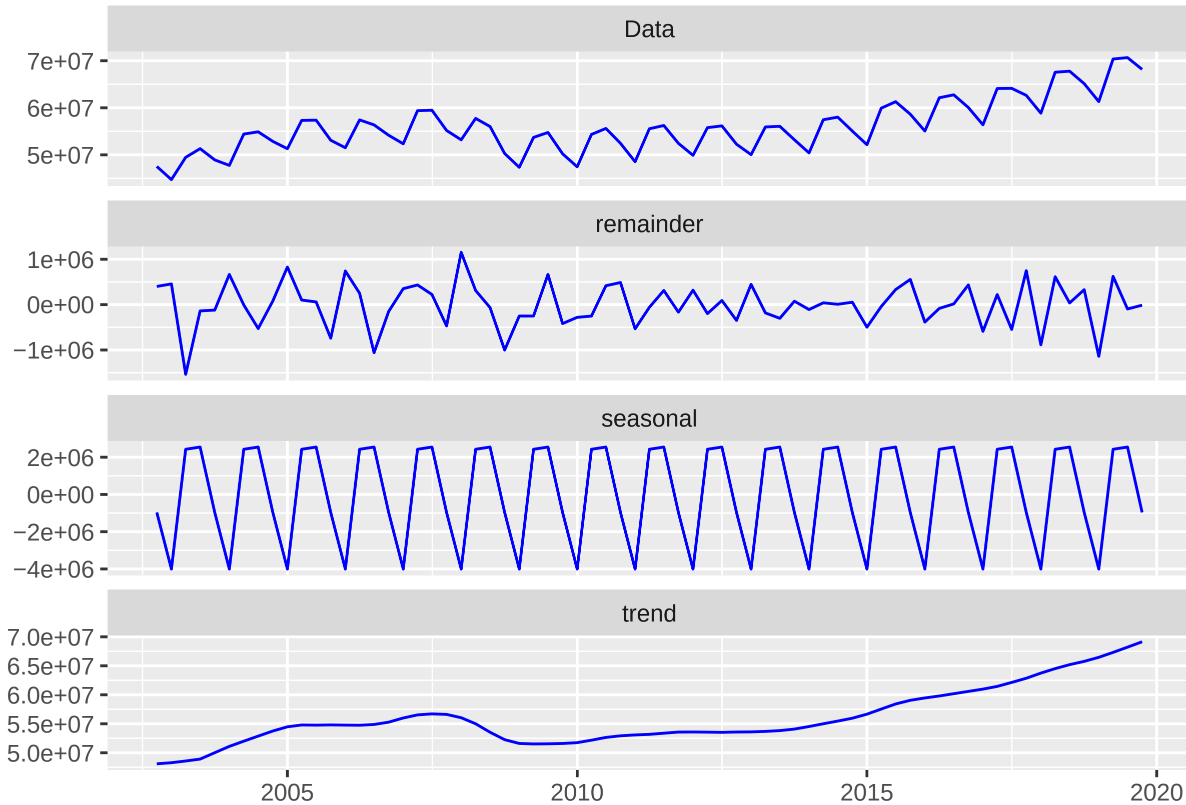
As it can be seen from the stationarity tests and also from the plots of the respective series, all the series are non-stationary.

d. Investigate seasonality using decomposition and/or spectral analysis

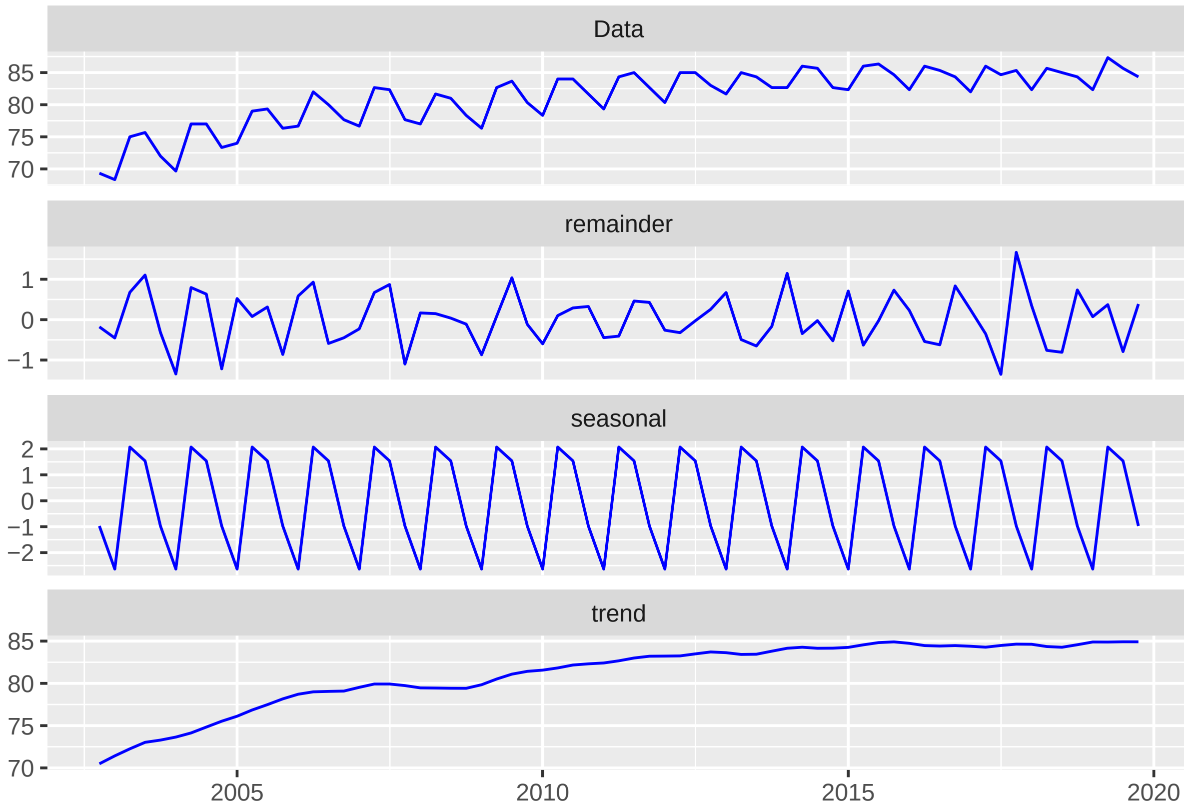
```
decomposition(flight_ts)
```



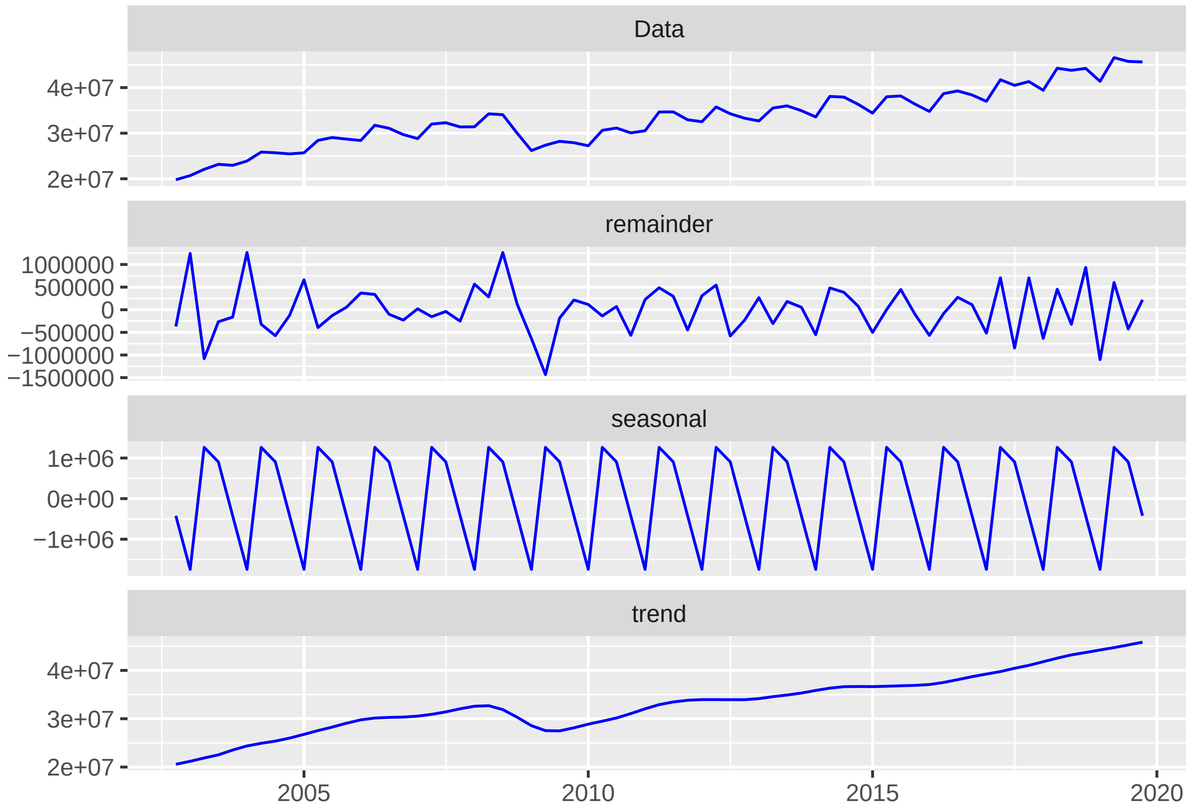

```
decomposition(passengers_ts)
```




```
decomposition(load_factor_ts)
```

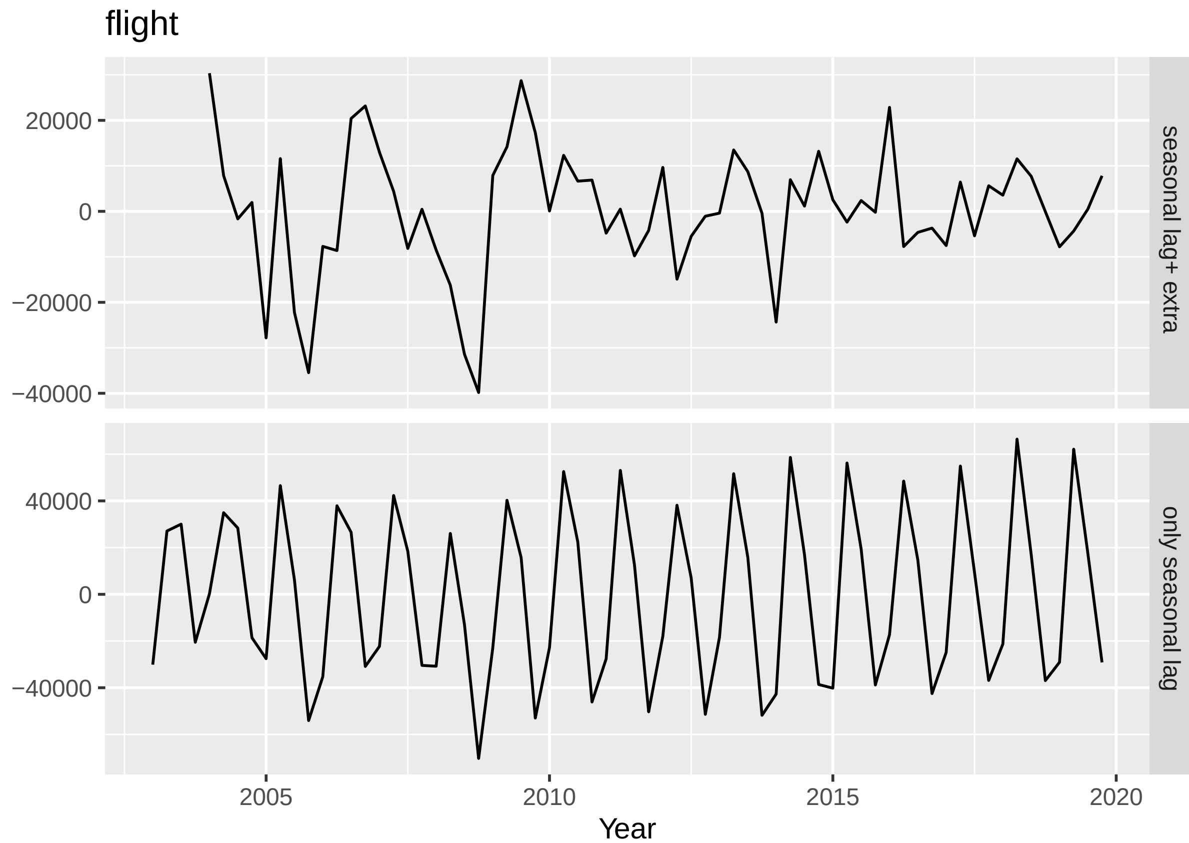



```
decomposition(op_rev_ts)
```

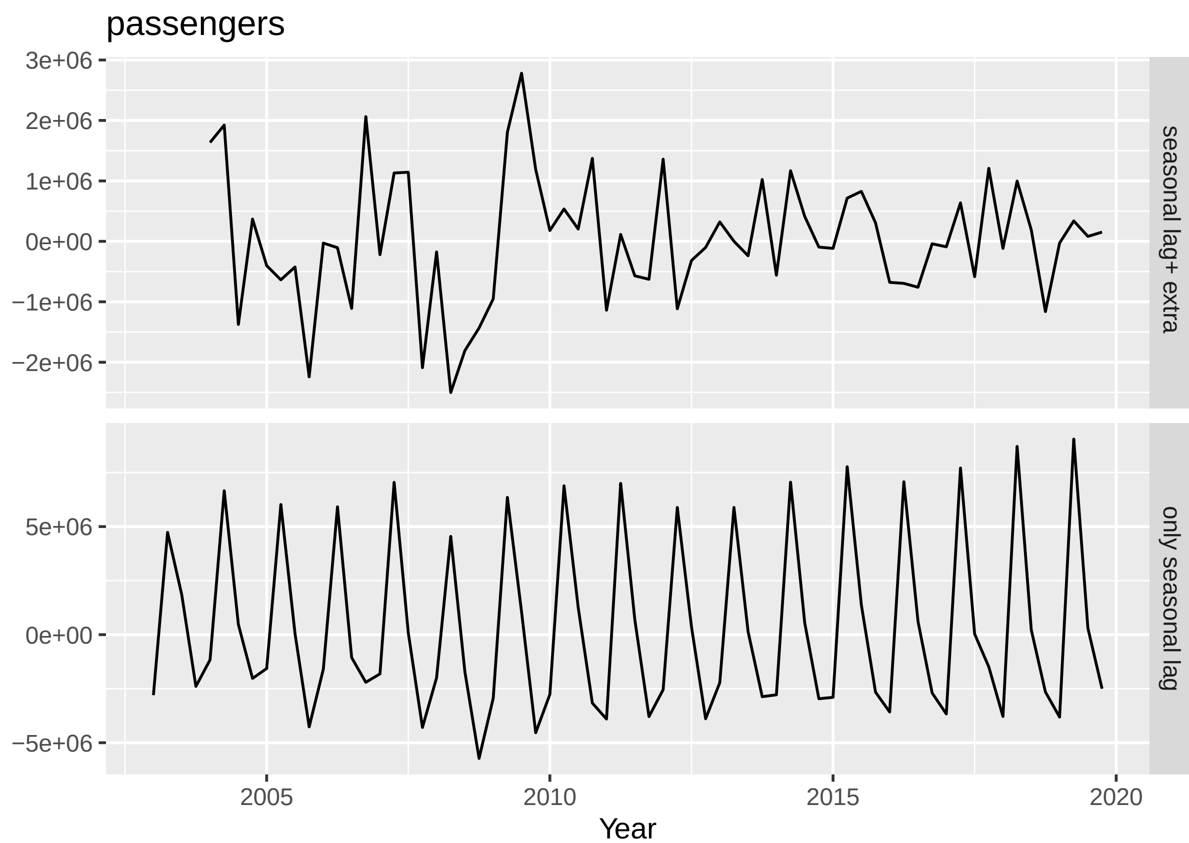


Performing differencing

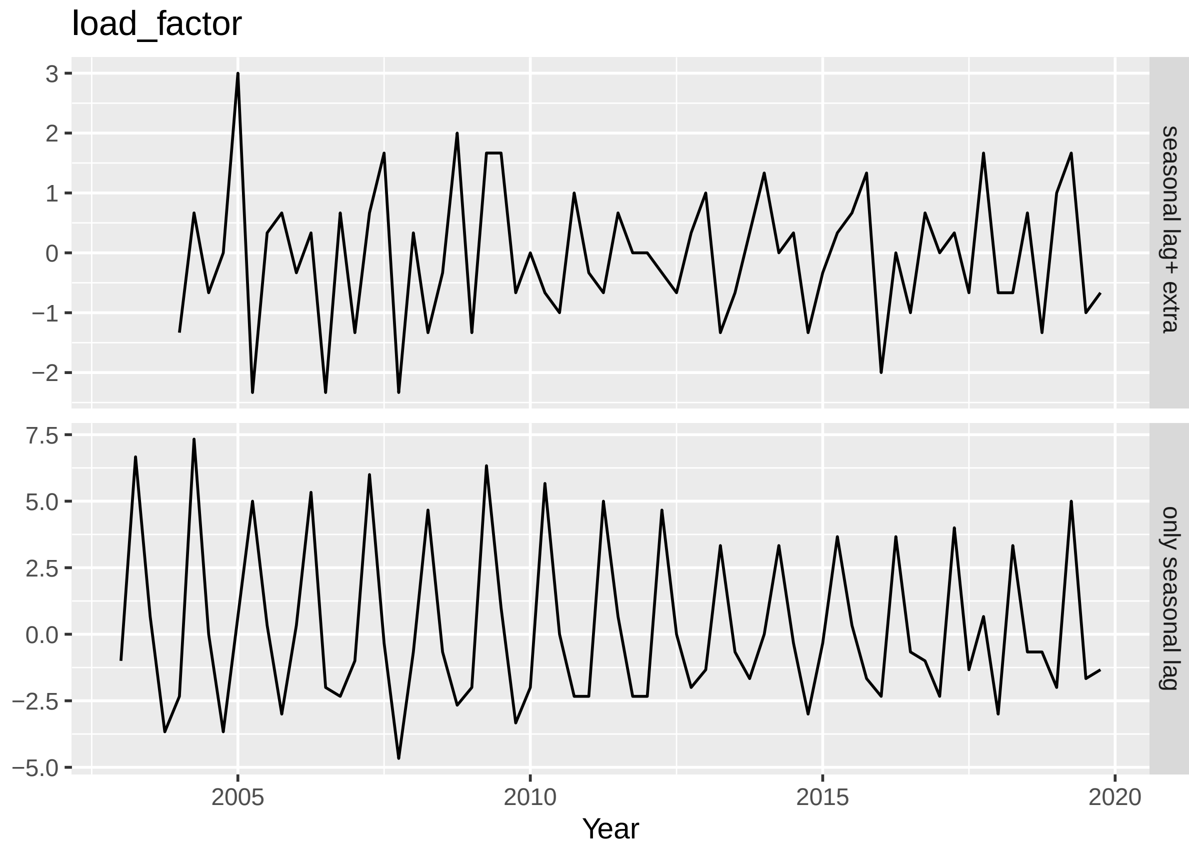
```
plot.diff(flight_ts,4,1,1,1,'flight')
```



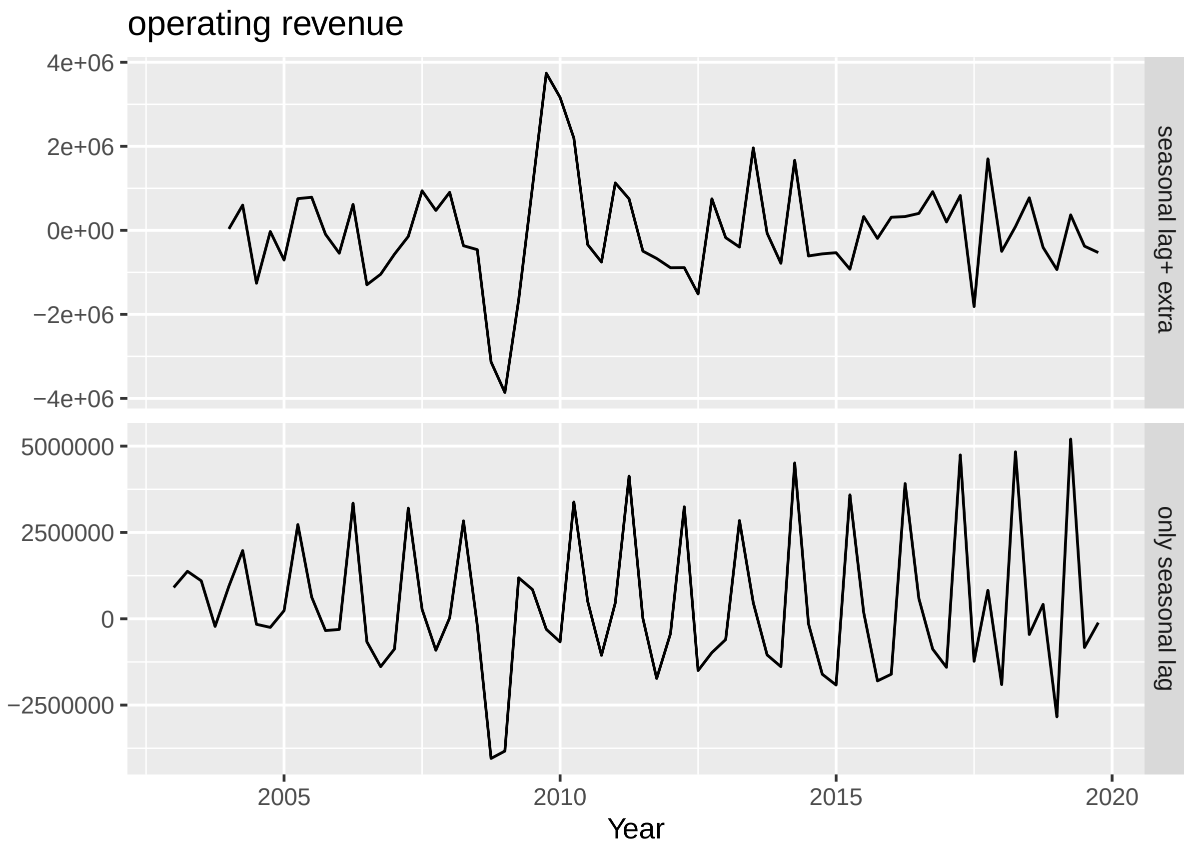

```
plot.diff(passengers_ts,4,1,1,1,'passengers')
```




```
plot.diff(load_factor_ts,4,1,1,1,'load_factor')
```




```
plot.diff(op_rev_ts,4,1,1,1,'operating revenue')
```



Stationarity check

```
cat('Stationarity tests for flight data\n')
```

```
## Stationarity tests for flight data
```

```
stationarity.test(diff(diff(flight_ts,4,1),1,1))
```

```
##
##  Box-Ljung test
##
## data:  data
## X-squared = 84.98, df = 25, p-value = 1.854e-08
##
##
##  KPSS Test for Trend Stationarity
##
## data:  data
## KPSS Trend = 0.02704, Truncation lag parameter = 3, p-value = 0.1
##
##
##  Augmented Dickey-Fuller Test
##
## data:  data
## Dickey-Fuller = -5.5004, Lag order = 3, p-value = 0.01
```

```
## alternative hypothesis: stationary
##
## Series is stationary
```

```r
cat('Stationarity tests for passenger data\n')
```

```
## Stationarity tests for passenger data
```

```r
stationarity.test(diff(diff(passengers_ts,4,1),1,1))
```

```
##
##  Box-Ljung test
##
## data:  data
## X-squared = 48.707, df = 25, p-value = 0.003066
##
##
##  KPSS Test for Trend Stationarity
##
## data:  data
## KPSS Trend = 0.032604, Truncation lag parameter = 3, p-value = 0.1
##
##
##  Augmented Dickey-Fuller Test
##
## data:  data
## Dickey-Fuller = -5.2668, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
##
## Series is stationary
```

```r
cat('Stationarity tests for load factor data\n')
```

```
## Stationarity tests for load factor data
```

```r
stationarity.test(diff(load_factor_ts,4,1))
```

```
##
##  Box-Ljung test
##
## data:  data
## X-squared = 59.448, df = 25, p-value = 0.0001245
##
##
##  KPSS Test for Trend Stationarity
##
## data:  data
## KPSS Trend = 0.081635, Truncation lag parameter = 3, p-value = 0.1
##
##
##  Augmented Dickey-Fuller Test
```

```
##
## data:  data
## Dickey-Fuller = -3.9205, Lag order = 3, p-value = 0.0188
## alternative hypothesis: stationary
##
## Series is stationary
```

```r
cat('Stationarity tests for operating revenue data\n')
```

```
## Stationarity tests for operating revenue data
```

```r
stationarity.test(diff(op_rev_ts,4,1))
```

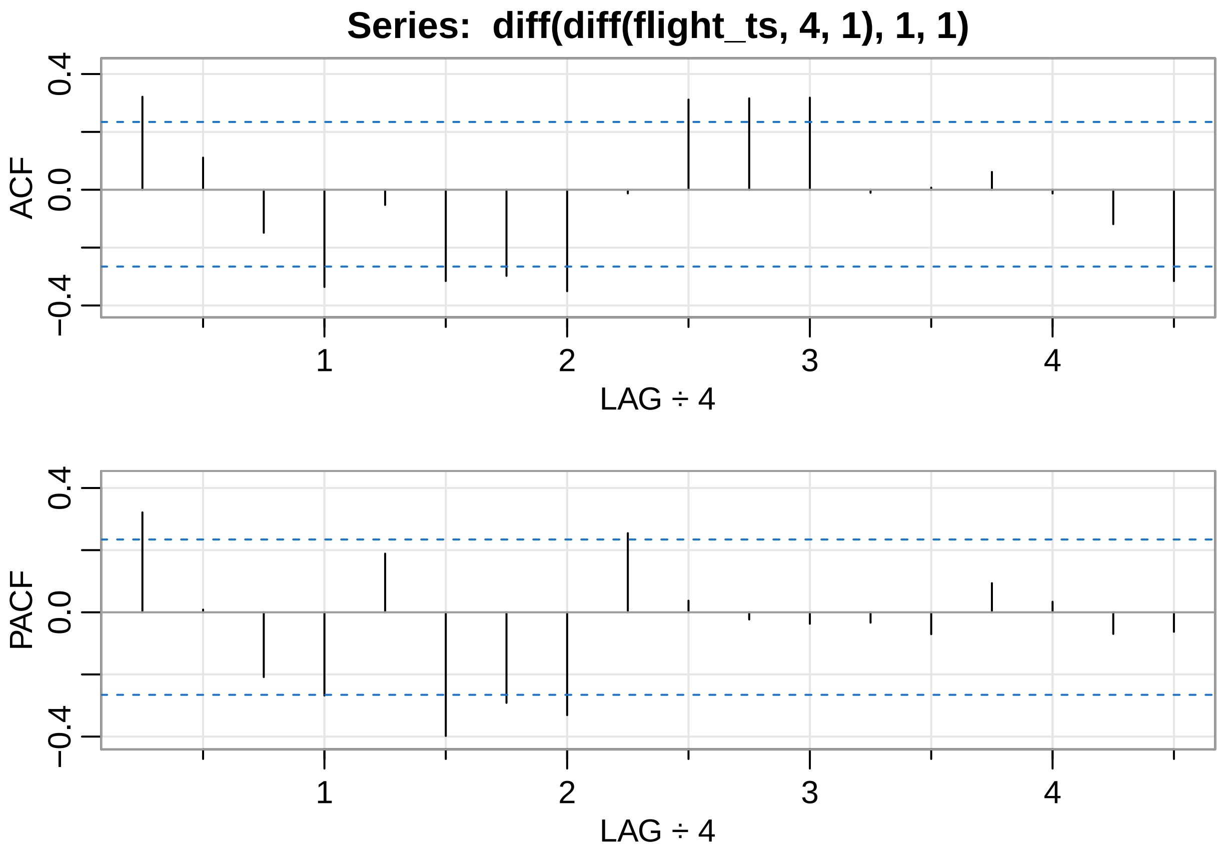
```
##
##  Box-Ljung test
##
## data:  data
## X-squared = 102.52, df = 25, p-value = 2.349e-11
##
##
##  KPSS Test for Trend Stationarity
##
## data:  data
## KPSS Trend = 0.10921, Truncation lag parameter = 3, p-value = 0.1
##
##
##  Augmented Dickey-Fuller Test
##
## data:  data
## Dickey-Fuller = -3.5388, Lag order = 3, p-value = 0.04552
## alternative hypothesis: stationary
##
## Series is stationary
```

After few trials, different differencing terms were identified to make the respective series stationary. Although the results of Box-Ljung,KPSS and ADF tests were displayed above, the decision of stationarity was made from KPSS results. First, seasonal differencing was performed to check if the series has become stationary. If not, further lag differences were performed until the series become stationary.

e. ACF/PACF

ACF and PACF for flight data

```r
acf2(diff(diff(flight_ts,4,1),1,1))
```



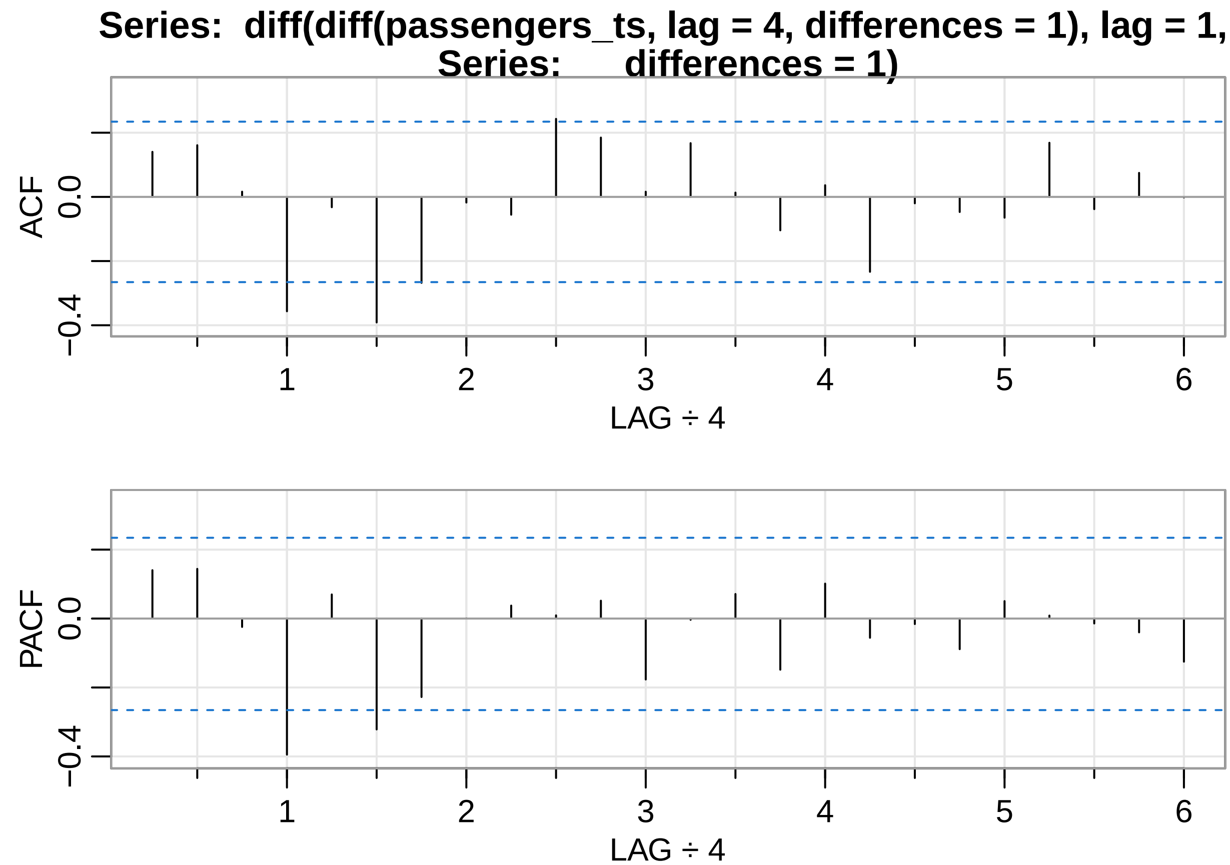

```
##      [,1] [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11] [,12]
## ACF  0.32 0.11 -0.15 -0.34 -0.05 -0.32 -0.30 -0.35 -0.01  0.31  0.32  0.32
## PACF 0.32 0.01 -0.21 -0.27  0.19 -0.40 -0.29 -0.33  0.25  0.04 -0.02 -0.04
##      [,13] [,14] [,15] [,16] [,17] [,18]
## ACF  -0.01  0.01  0.06 -0.01 -0.12 -0.32
## PACF -0.03 -0.07  0.09  0.03 -0.07 -0.06
```

The is one spike in the beginning of the ACF and PACF plots. So, an AR(1) MA(1) model can be used for the non seasonal term. For the seasonal term, The PACF has 2 significant spikes and then the correlation values decreases continuously. Whereas in ACF chart, there are multiple significant spikes. Hence, an AR(2) model for the seasonal term can used here.

ACF plots were plotted below after correct differencing was done.

ACF and PACF for passenger data

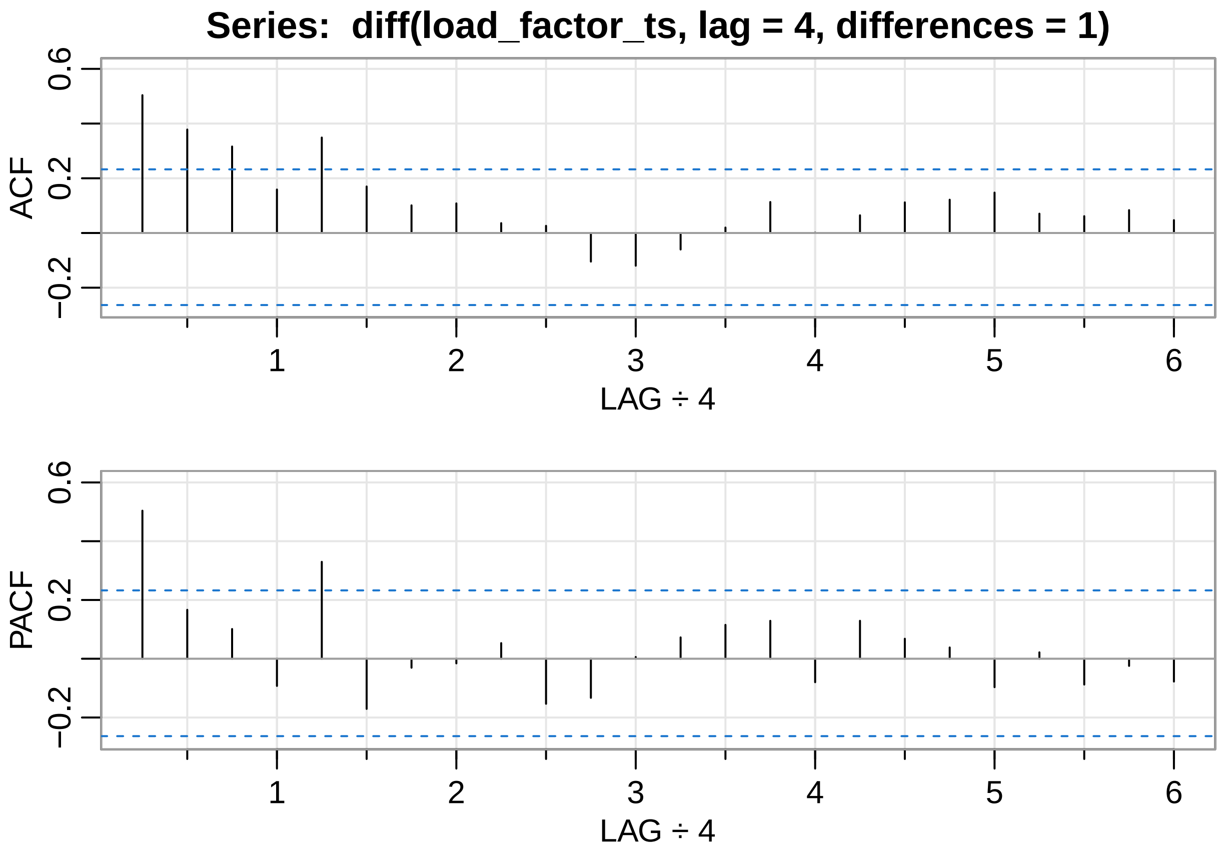
```
acf2(diff(diff(passengers_ts,lag=4,differences=1),lag=1,differences=1),max.lag=24)
```




```
##      [,1] [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11] [,12]
## ACF  0.14 0.16  0.02 -0.36 -0.03 -0.39 -0.27 -0.02 -0.06  0.24  0.18  0.02
## PACF 0.14 0.14 -0.02 -0.39  0.07 -0.32 -0.23  0.00  0.04  0.01  0.05 -0.18
##      [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## ACF   0.17  0.01 -0.10  0.04 -0.23 -0.02 -0.05 -0.06  0.17 -0.04  0.07  0.00
## PACF  0.00  0.07 -0.15  0.10 -0.06 -0.02 -0.09  0.05  0.01 -0.01 -0.04 -0.13
```

ACF and PACF for load factor data

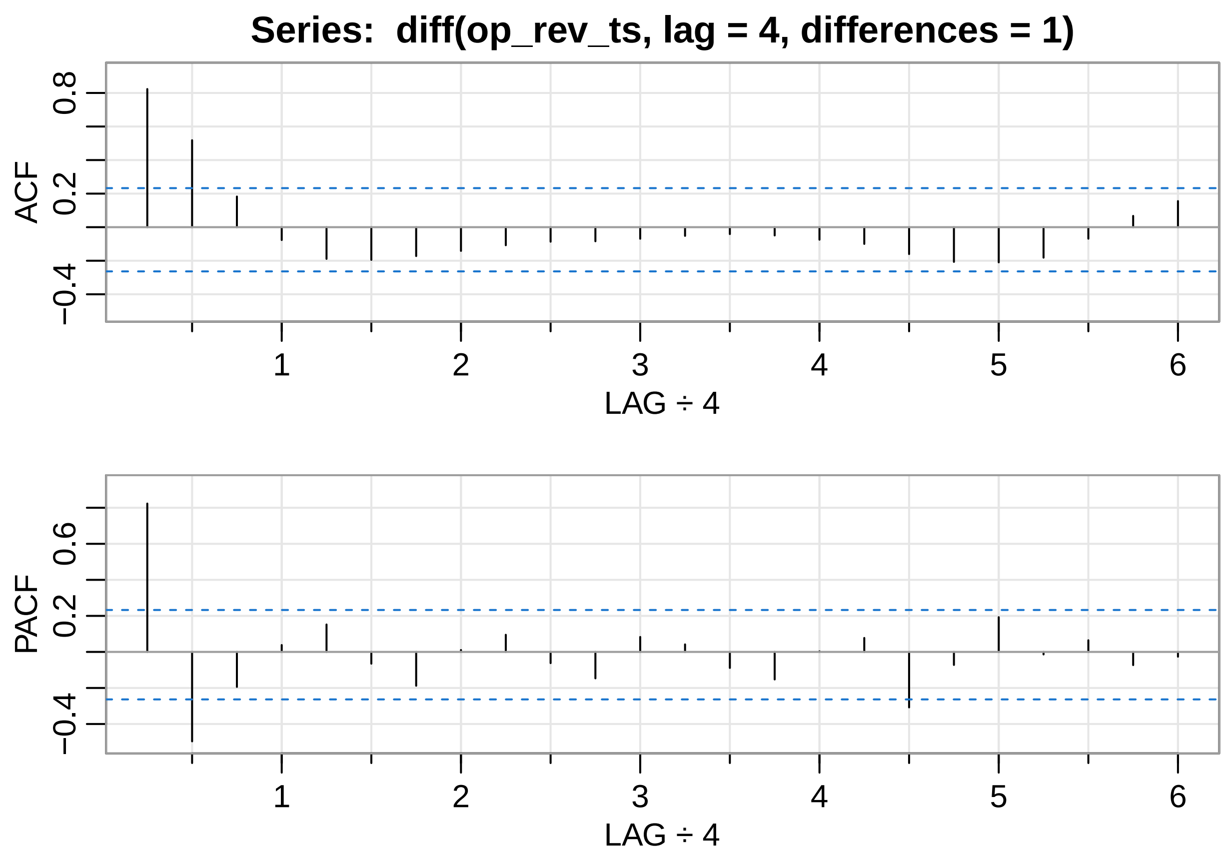
```
acf2(diff(load_factor_ts,lag=4,differences=1),max.lag=24)
```




```
##       [,1] [,2] [,3]  [,4] [,5]  [,6]  [,7]  [,8] [,9] [,10] [,11] [,12] [,13]
## ACF    0.5 0.38 0.32  0.16 0.35  0.17  0.10  0.11 0.04  0.03 -0.10 -0.12 -0.06
## PACF   0.5 0.17 0.10 -0.09 0.33 -0.17 -0.03 -0.02 0.05 -0.15 -0.13  0.01  0.07
##       [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## ACF    0.02  0.11  0.00  0.06  0.11  0.12  0.15  0.07  0.06  0.08  0.05
## PACF   0.12  0.13 -0.08  0.13  0.07  0.04 -0.10  0.02 -0.09 -0.02 -0.08
```

ACF and PACF for operating revenue data

```
acf2(diff(op_rev_ts,lag=4,differences=1),max.lag=24)
```




```
##       [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11] [,12]
## ACF  0.82  0.52  0.18 -0.08 -0.19 -0.19 -0.17 -0.14 -0.11 -0.09 -0.08 -0.07
## PACF 0.82 -0.50 -0.19  0.04  0.15 -0.07 -0.19  0.01  0.10 -0.06 -0.15  0.08
##      [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## ACF  -0.05 -0.04 -0.05 -0.07 -0.10 -0.16 -0.21 -0.21 -0.18 -0.07  0.07  0.16
## PACF  0.04 -0.09 -0.15  0.00  0.08 -0.31 -0.07  0.19 -0.01  0.06 -0.07 -0.03
```

3. ARIMA modeling

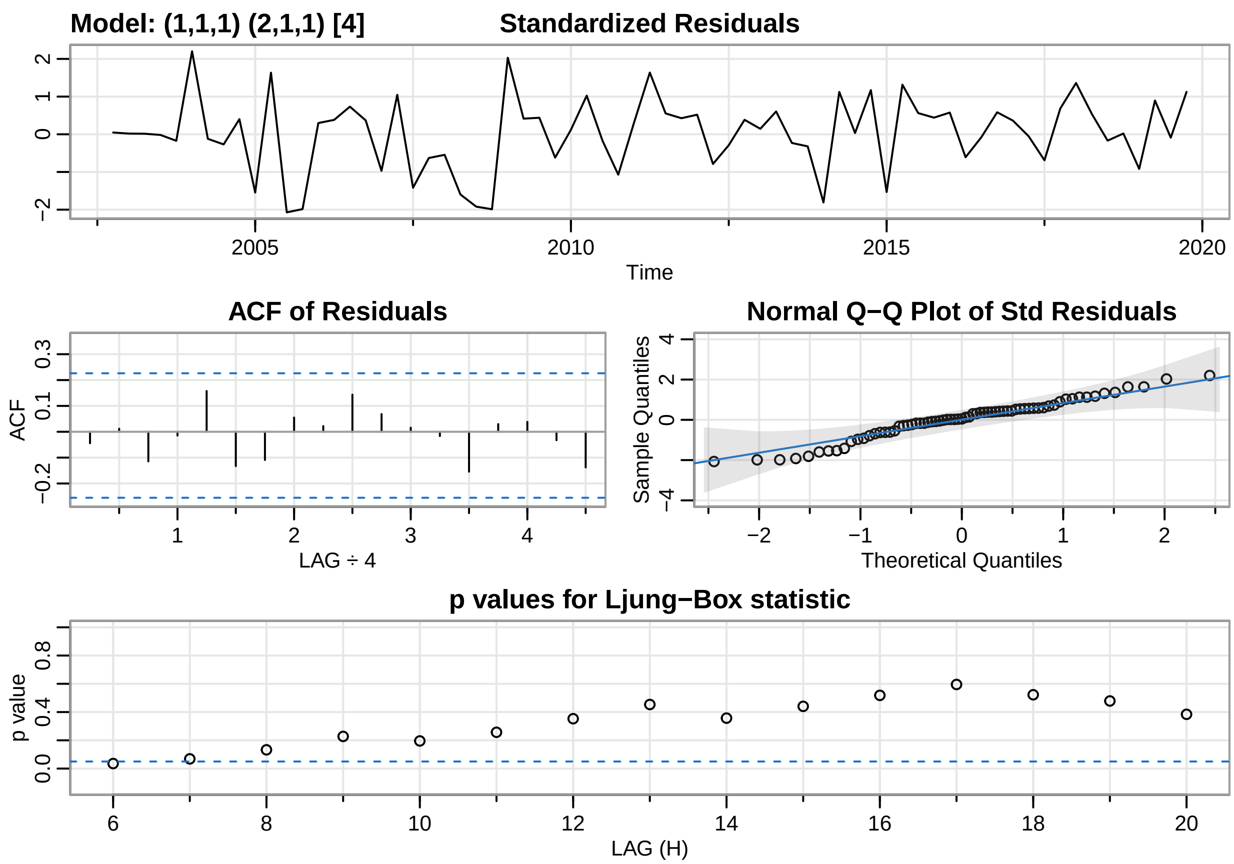
a. Fit at least one ARIMA model for each series

Sarima for flight data

```
sarima(flight_ts,1,1,1,2,1,1,4)
```

```
## initial  value 9.429339
## iter   2 value 9.193376
## iter   3 value 9.081937
## iter   4 value 9.077717
## iter   5 value 9.074069
## iter   6 value 9.071981
## iter   7 value 9.067588
## iter   8 value 9.065877
## iter   9 value 9.065299
```

```
## iter  10 value 9.065054
## iter  11 value 9.064801
## iter  12 value 9.064697
## iter  13 value 9.064694
## iter  13 value 9.064694
## iter  13 value 9.064694
## final  value 9.064694
## converged
## initial  value 9.244231
## iter   2 value 9.235356
## iter   3 value 9.221219
## iter   4 value 9.219842
## iter   5 value 9.217734
## iter   6 value 9.215219
## iter   7 value 9.214004
## iter   8 value 9.213788
## iter   9 value 9.213713
## iter  10 value 9.213570
## iter  11 value 9.213495
## iter  12 value 9.213477
## iter  13 value 9.213476
## iter  14 value 9.213476
## iter  15 value 9.213476
## iter  16 value 9.213476
## iter  16 value 9.213476
## iter  16 value 9.213476
## final  value 9.213476
## converged
```

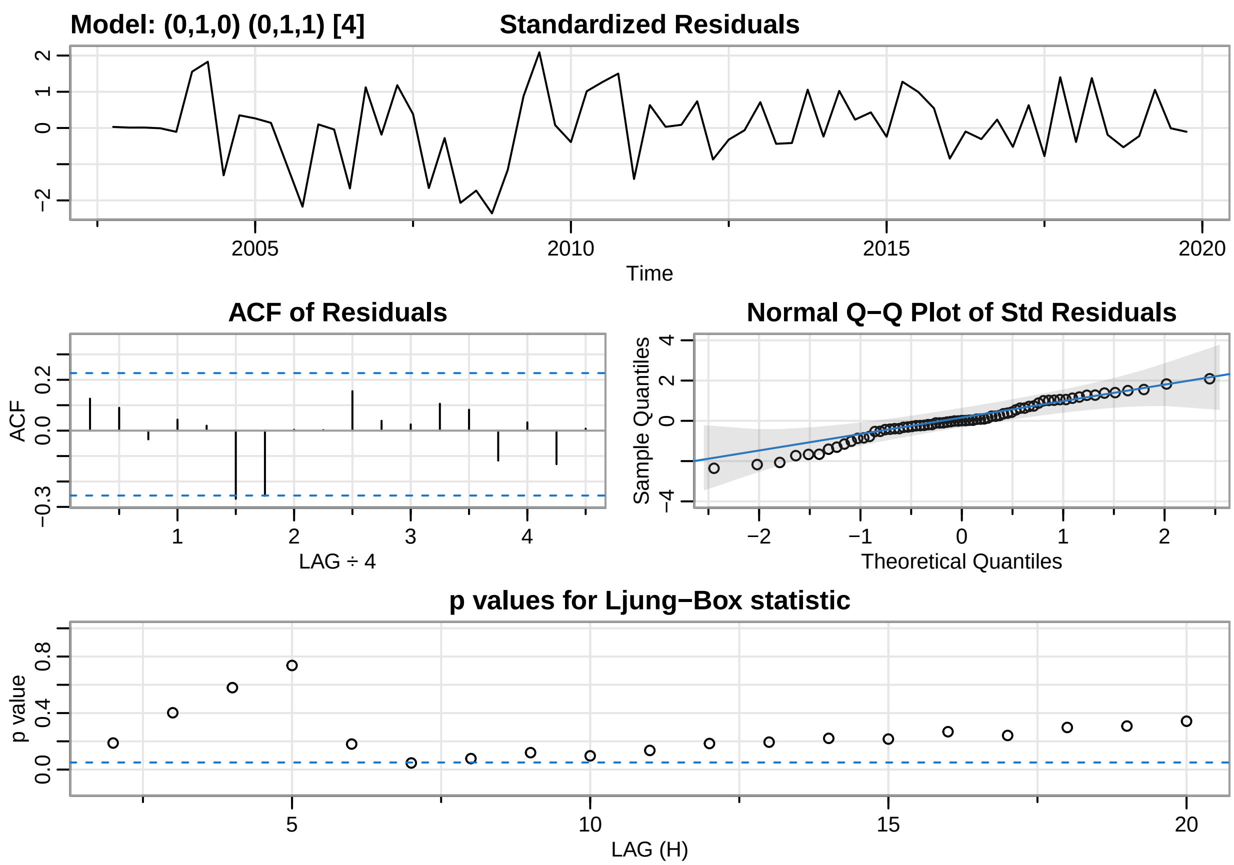



```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     include.mean = !no.constant, transform.pars = trans, fixed = fixed, optim.control = list(trace =
##         REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1      ma1     sar1     sar2     sma1
##       0.3582  -0.0400  -0.4360  -0.5930  -0.1459
## s.e.  0.2988   0.3041   0.1936   0.1242   0.2474
##
## sigma^2 estimated as 93733652:  log likelihood = -680.47,  aic = 1372.95
##
## $degrees_of_freedom
## [1] 59
##
## $ttable
##      Estimate     SE t.value p.value
## ar1    0.3582 0.2988  1.1991  0.2353
## ma1   -0.0400 0.3041 -0.1317  0.8957
## sar1  -0.4360 0.1936 -2.2515  0.0281
## sar2  -0.5930 0.1242 -4.7756  0.0000
## sma1  -0.1459 0.2474 -0.5896  0.5577
##
## $AIC
```

```
## [1] 21.45233
##
## $AICc
## [1] 21.46849
##
## $BIC
## [1] 21.65472
```

```
sarima(passengers_ts,0,1,0,0,1,1,4)
```

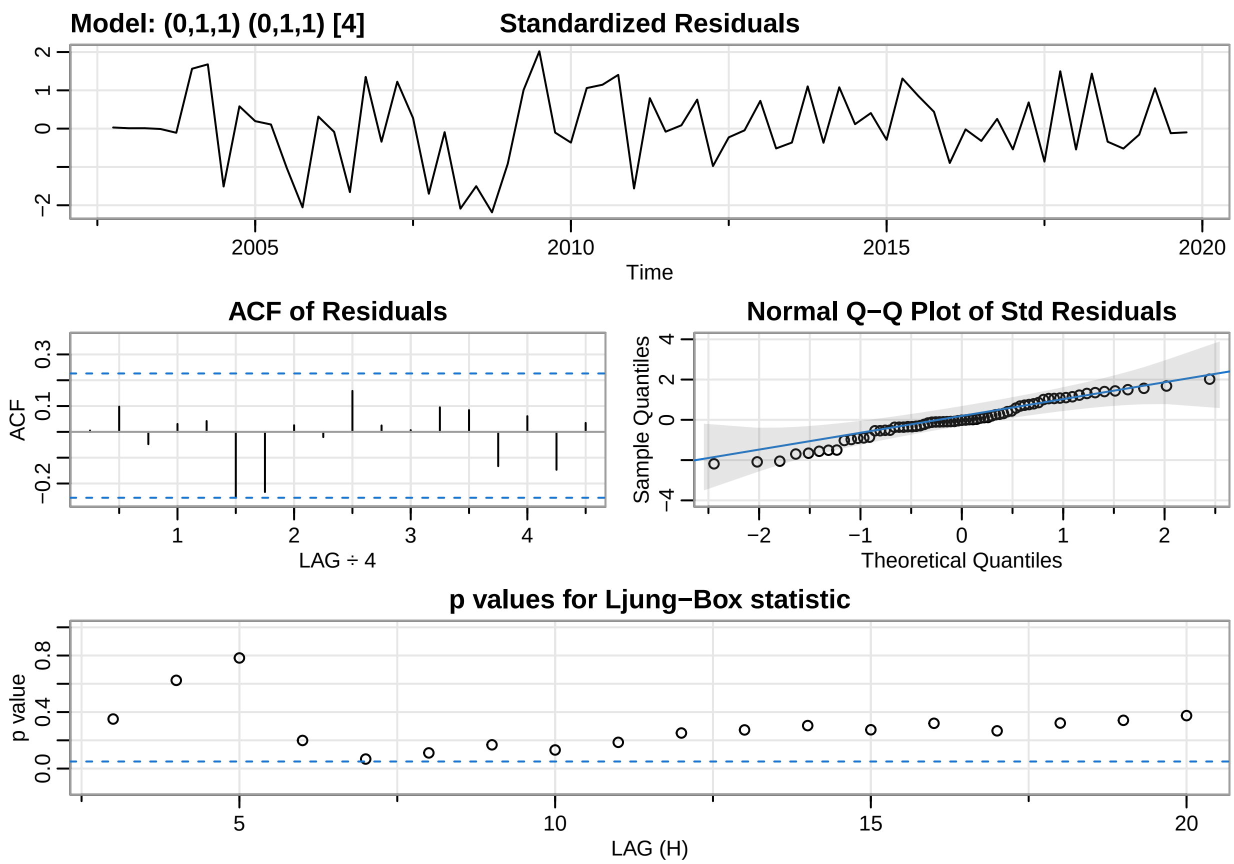
```
## initial  value 13.858357
## iter   2 value 13.777263
## iter   3 value 13.774210
## iter   4 value 13.773738
## iter   5 value 13.773733
## iter   5 value 13.773733
## iter   5 value 13.773733
## final  value 13.773733
## converged
## initial  value 13.764360
## iter   2 value 13.762424
## iter   3 value 13.762407
## iter   3 value 13.762406
## iter   3 value 13.762406
## final  value 13.762406
## converged
```



```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     include.mean = !no.constant, transform.pars = trans, fixed = fixed, optim.control = list(trace =
##         REPORT = 1, reltol = tol))
##
## Coefficients:
##          sma1
##       -0.5031
## s.e.   0.1266
##
## sigma^2 estimated as 8.83e+11:  log likelihood = -971.61,  aic = 1947.21
##
## $degrees_of_freedom
## [1] 63
##
## $ttable
##      Estimate     SE t.value p.value
## sma1  -0.5031 0.1266 -3.9731   2e-04
##
## $AIC
## [1] 30.42519
##
## $AICc
## [1] 30.4262
##
## $BIC
## [1] 30.49266
```

```
sarima(passengers_ts,0,1,1,0,1,1,4)
```

```
## initial  value 13.858357
## iter   2 value 13.768980
## iter   3 value 13.766409
## iter   4 value 13.765899
## iter   5 value 13.765894
## iter   5 value 13.765894
## iter   5 value 13.765894
## final  value 13.765894
## converged
## initial  value 13.756602
## iter   2 value 13.754746
## iter   3 value 13.754729
## iter   4 value 13.754728
## iter   4 value 13.754728
## iter   4 value 13.754728
## final  value 13.754728
## converged
```




```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     include.mean = !no.constant, transform.pars = trans, fixed = fixed, optim.control = list(trace =
##         REPORT = 1, reltol = tol))
##
## Coefficients:
##          ma1     sma1
##       0.1112  -0.4902
## s.e.  0.1105   0.1240
##
## sigma^2 estimated as 8.703e+11:  log likelihood = -971.11,  aic = 1948.23
##
## $degrees_of_freedom
## [1] 62
##
## $ttable
##      Estimate     SE t.value p.value
## ma1    0.1112 0.1105  1.0066  0.3180
## sma1  -0.4902 0.1240 -3.9546  0.0002
##
## $AIC
## [1] 30.44108
##
## $AICc
```

```
## [1] 30.44416
##
## $BIC
## [1] 30.54228
```

```
auto.arima(op_rev_ts,seasonal=TRUE,trace=TRUE)
```

```
##
##  ARIMA(2,0,2)(1,1,1)[4] with drift         : 1989.879
##  ARIMA(0,0,0)(0,1,0)[4] with drift         : 2078.766
##  ARIMA(1,0,0)(1,1,0)[4] with drift         : 1997.869
##  ARIMA(0,0,1)(0,1,1)[4] with drift         : 2028.881
##  ARIMA(0,0,0)(0,1,0)[4]                    : 2102.465
##  ARIMA(2,0,2)(0,1,1)[4] with drift         : 1987.765
##  ARIMA(2,0,2)(0,1,0)[4] with drift         : 1991.788
##  ARIMA(2,0,2)(0,1,2)[4] with drift         : 1992.532
##  ARIMA(2,0,2)(1,1,0)[4] with drift         : 1992.615
##  ARIMA(2,0,2)(1,1,2)[4] with drift         : 1992.548
##  ARIMA(1,0,2)(0,1,1)[4] with drift         : 1989.329
##  ARIMA(2,0,1)(0,1,1)[4] with drift         : 1989.434
##  ARIMA(3,0,2)(0,1,1)[4] with drift         : 1990.266
##  ARIMA(2,0,3)(0,1,1)[4] with drift         : 1990.279
##  ARIMA(1,0,1)(0,1,1)[4] with drift         : 1991.073
##  ARIMA(1,0,3)(0,1,1)[4] with drift         : 1991.681
##  ARIMA(3,0,1)(0,1,1)[4] with drift         : 1987.966
##  ARIMA(3,0,3)(0,1,1)[4] with drift         : Inf
##  ARIMA(2,0,2)(0,1,1)[4]                    : 1990.916
##
##  Best model: ARIMA(2,0,2)(0,1,1)[4] with drift
```

```
## Series: op_rev_ts
## ARIMA(2,0,2)(0,1,1)[4] with drift
##
## Coefficients:
##           ar1     ar2     ma1     ma2     sma1      drift
##       -0.1212  0.8213  1.2372  0.4339  -0.7852  332845.12
## s.e.   0.0938  0.1007  0.1262  0.1274   0.1342   65991.78
##
## sigma^2 = 9.104e+11:  log likelihood = -985.9
## AIC=1985.8   AICc=1987.77   BIC=2001.02
```

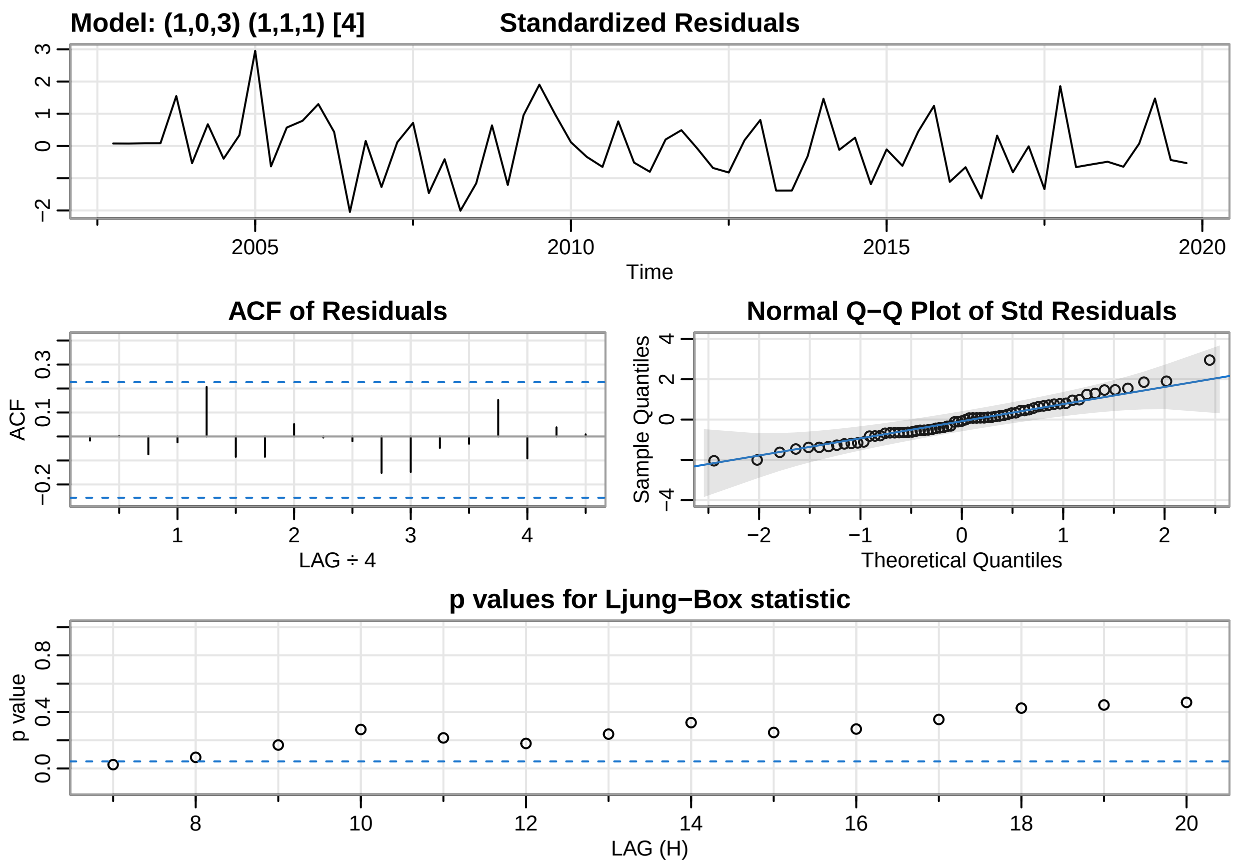
```
sarima(load_factor_ts,1,0,3,1,1,1,4)
```

```
## initial  value 0.128520
## iter   2 value 0.026603
## iter   3 value -0.029449
## iter   4 value -0.036504
## iter   5 value -0.038494
## iter   6 value -0.041420
## iter   7 value -0.052316
## iter   8 value -0.056543
## iter   9 value -0.070210
```

```
## iter  10 value -0.087612
## iter  11 value -0.090482
## iter  12 value -0.099668
## iter  13 value -0.104262
## iter  14 value -0.106706
## iter  15 value -0.109942
## iter  16 value -0.111127
## iter  17 value -0.118550
## iter  18 value -0.120669
## iter  19 value -0.123847
## iter  20 value -0.130839
## iter  21 value -0.133443
## iter  22 value -0.137901
## iter  23 value -0.142199
## iter  24 value -0.144472
## iter  25 value -0.147042
## iter  26 value -0.167793
## iter  27 value -0.168005
## iter  27 value -0.168005
## iter  28 value -0.170983
## iter  29 value -0.173331
## iter  30 value -0.173333
## iter  31 value -0.173740
## iter  32 value -0.173756
## iter  33 value -0.173870
## iter  34 value -0.173894
## iter  35 value -0.173970
## iter  36 value -0.174000
## iter  37 value -0.174056
## iter  38 value -0.174089
## iter  39 value -0.174135
## iter  40 value -0.174169
## iter  41 value -0.174209
## iter  42 value -0.174244
## iter  43 value -0.174259
## iter  44 value -0.174346
## iter  45 value -0.174368
## iter  46 value -0.174423
## iter  47 value -0.174450
## iter  48 value -0.174491
## iter  49 value -0.174520
## iter  50 value -0.174530
## iter  51 value -0.174590
## iter  52 value -0.174614
## iter  53 value -0.174654
## iter  54 value -0.174680
## iter  55 value -0.174685
## iter  56 value -0.174793
## iter  57 value -0.174806
## iter  58 value -0.174860
## iter  59 value -0.174880
## iter  60 value -0.174913
## iter  61 value -0.174937
## iter  62 value -0.174945
```

```
## iter  63 value -0.175034
## iter  64 value -0.175047
## iter  65 value -0.175088
## iter  66 value -0.175107
## iter  67 value -0.175116
## iter  68 value -0.175163
## iter  69 value -0.175180
## iter  70 value -0.175186
## iter  71 value -0.175223
## iter  72 value -0.175226
## iter  73 value -0.175368
## iter  74 value -0.175374
## iter  75 value -0.175418
## iter  76 value -0.175431
## iter  77 value -0.175535
## iter  78 value -0.175543
## iter  79 value -0.175572
## iter  80 value -0.175601
## iter  80 value -0.175601
## iter  81 value -0.175663
## iter  82 value -0.175683
## iter  83 value -0.175745
## iter  84 value -0.175767
## iter  85 value -0.175812
## iter  86 value -0.175838
## iter  87 value -0.175845
## iter  88 value -0.175905
## iter  89 value -0.175947
## iter  90 value -0.176066
## iter  91 value -0.176419
## iter  91 value -0.176419
## iter  92 value -0.177052
## iter  93 value -0.177215
## iter  93 value -0.177215
## iter  94 value -0.177215
## iter  94 value -0.177215
## iter  94 value -0.177215
## final  value -0.177215
## converged
## initial  value -0.026159
## iter   2 value -0.053777
## iter   3 value -0.064852
## iter   4 value -0.071695
## iter   5 value -0.072928
## iter   6 value -0.075161
## iter   7 value -0.076904
## iter   8 value -0.083286
## iter   9 value -0.088561
## iter  10 value -0.098058
## iter  11 value -0.101433
## iter  12 value -0.102668
## iter  13 value -0.107194
## iter  14 value -0.110590
## iter  15 value -0.112836
```

```
## iter  16 value -0.113197
## iter  17 value -0.113251
## iter  18 value -0.113308
## iter  19 value -0.113338
## iter  20 value -0.113357
## iter  21 value -0.113367
## iter  22 value -0.113370
## iter  23 value -0.113372
## iter  24 value -0.113377
## iter  25 value -0.113382
## iter  26 value -0.113384
## iter  27 value -0.113384
## iter  28 value -0.113384
## iter  29 value -0.113384
## iter  30 value -0.113384
## iter  31 value -0.113384
## iter  31 value -0.113384
## final  value -0.113384
## converged
```



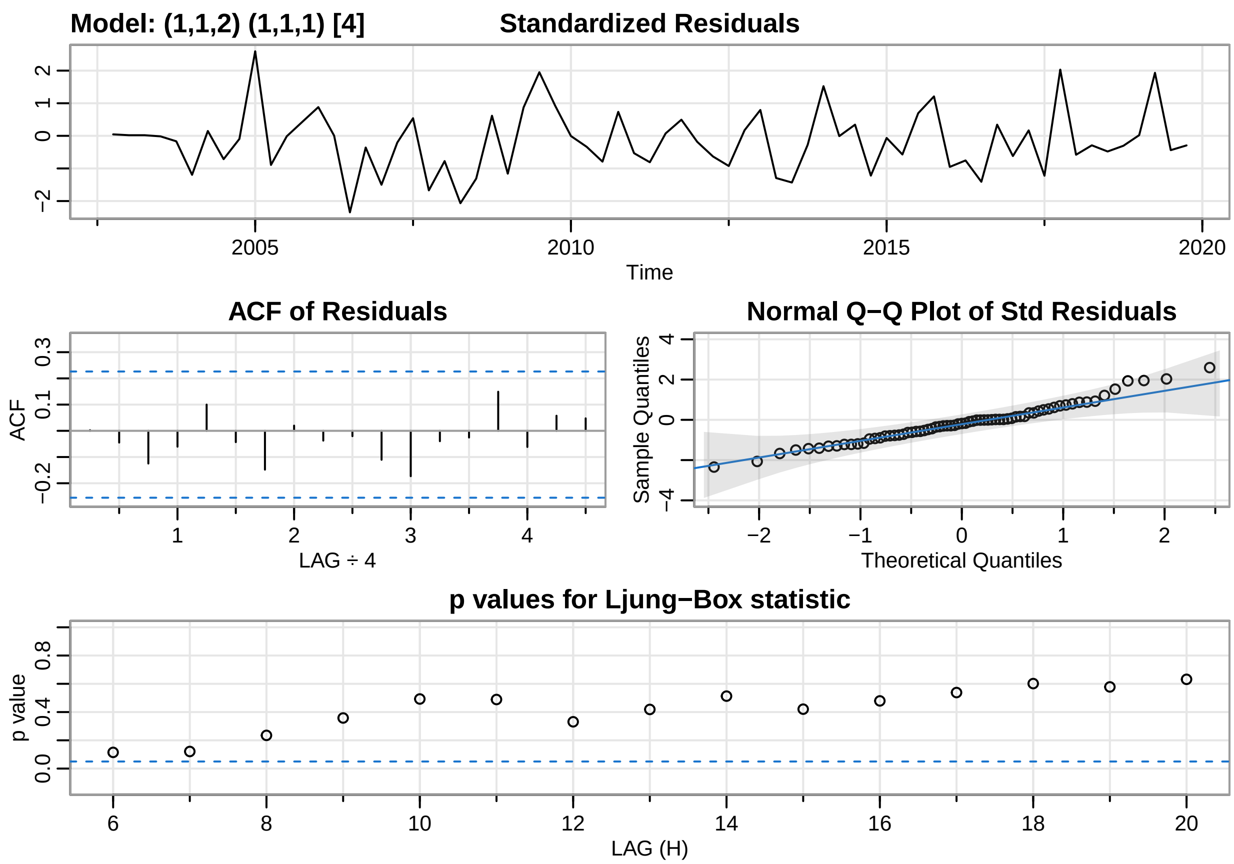

```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = constant, transform.pars = trans, fixed = fixed, optim.control = list(trace = trc,
```

```
##         REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1      ma1      ma2     ma3    sar1     sma1  constant
##       0.9598  -0.4556  -0.1962  0.1240  0.2764  -0.6961    0.2194
## s.e.  0.0507   0.1347   0.1416  0.1615  0.2342   0.1825    0.1134
##
## sigma^2 estimated as 0.7782:  log likelihood = -84.86,  aic = 185.72
##
## $degrees_of_freedom
## [1] 58
##
## $ttable
##          Estimate     SE t.value p.value
## ar1        0.9598 0.0507 18.9383  0.0000
## ma1       -0.4556 0.1347 -3.3824  0.0013
## ma2       -0.1962 0.1416 -1.3864  0.1709
## ma3        0.1240 0.1615  0.7680  0.4456
## sar1       0.2764 0.2342  1.1803  0.2427
## sma1      -0.6961 0.1825 -3.8145  0.0003
## constant   0.2194 0.1134  1.9342  0.0580
##
## $AIC
## [1] 2.857262
##
## $AICc
## [1] 2.887492
##
## $BIC
## [1] 3.124879
```

```
sarima(load_factor_ts,1,1,2,1,1,1,4)
```

```
## initial  value 0.063578
## iter   2 value 0.010360
## iter   3 value -0.097221
## iter   4 value -0.103965
## iter   5 value -0.107250
## iter   6 value -0.110989
## iter   7 value -0.111371
## iter   8 value -0.111553
## iter   9 value -0.111729
## iter  10 value -0.111845
## iter  11 value -0.111882
## iter  12 value -0.111885
## iter  12 value -0.111885
## final  value -0.111885
## converged
## initial  value -0.077000
## iter   2 value -0.078998
## iter   3 value -0.080475
## iter   4 value -0.080867
## iter   5 value -0.082366
## iter   6 value -0.087600
```

```
## iter   7 value -0.091914
## iter   8 value -0.092655
## iter   9 value -0.094073
## iter  10 value -0.097483
## iter  11 value -0.099649
## iter  12 value -0.103736
## iter  13 value -0.103908
## iter  14 value -0.104744
## iter  15 value -0.104787
## iter  16 value -0.104822
## iter  17 value -0.104827
## iter  18 value -0.104845
## iter  19 value -0.104848
## iter  20 value -0.104859
## iter  21 value -0.104877
## iter  22 value -0.104915
## iter  23 value -0.104958
## iter  24 value -0.104971
## iter  25 value -0.104975
## iter  26 value -0.104978
## iter  27 value -0.104979
## iter  28 value -0.104979
## iter  29 value -0.104983
## iter  30 value -0.104988
## iter  31 value -0.104996
## iter  32 value -0.105007
## iter  33 value -0.105031
## iter  34 value -0.105037
## iter  35 value -0.105038
## iter  36 value -0.105045
## iter  37 value -0.105049
## iter  38 value -0.105050
## iter  39 value -0.105050
## iter  40 value -0.105050
## iter  41 value -0.105050
## iter  42 value -0.105051
## iter  43 value -0.105051
## iter  44 value -0.105051
## iter  45 value -0.105051
## iter  45 value -0.105051
## final  value -0.105051
## converged
```

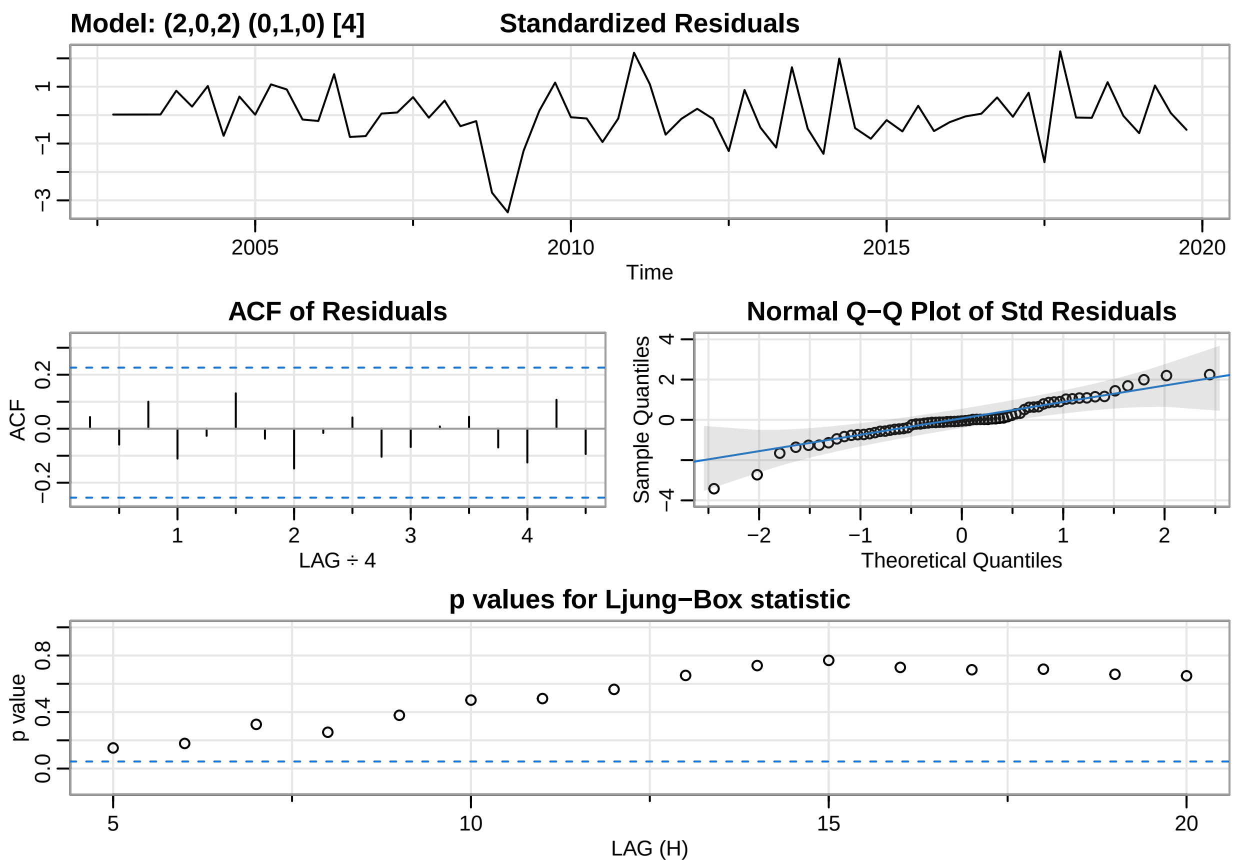



```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     include.mean = !no.constant, transform.pars = trans, fixed = fixed, optim.control = list(trace =
##         REPORT = 1, reltol = tol))
##
## Coefficients:
##           ar1     ma1      ma2    sar1     sma1
##       -0.7718  0.2869  -0.5613  0.3342  -0.6805
## s.e.   0.3575  0.3467   0.1780  0.2509   0.1967
##
## sigma^2 estimated as 0.7853:  log likelihood = -84.09,  aic = 180.18
##
## $degrees_of_freedom
## [1] 59
##
## $ttable
##      Estimate     SE t.value p.value
## ar1   -0.7718 0.3575 -2.1588  0.0349
## ma1    0.2869 0.3467  0.8275  0.4113
## ma2   -0.5613 0.1780 -3.1530  0.0025
## sar1   0.3342 0.2509  1.3319  0.1880
## sma1  -0.6805 0.1967 -3.4604  0.0010
##
## $AIC
```

```
## [1] 2.815274
##
## $AICc
## [1] 2.831438
##
## $BIC
## [1] 3.017669
```

```
sarima(op_rev_ts,2,0,2,0,1,0,4)
```

```
## initial  value 14.543912
## iter   2 value 14.160062
## iter   3 value 13.935120
## iter   4 value 13.885464
## iter   5 value 13.873991
## iter   6 value 13.870208
## iter   7 value 13.862707
## iter   8 value 13.846783
## iter   9 value 13.843467
## iter  10 value 13.831053
## iter  11 value 13.815420
## iter  12 value 13.813441
## iter  13 value 13.800454
## iter  14 value 13.797144
## iter  15 value 13.795923
## iter  16 value 13.795478
## iter  17 value 13.795341
## iter  18 value 13.795312
## iter  19 value 13.795310
## iter  19 value 13.795309
## final  value 13.795309
## converged
## initial  value 13.799491
## iter   2 value 13.799310
## iter   3 value 13.799166
## iter   4 value 13.799130
## iter   5 value 13.799076
## iter   6 value 13.799061
## iter   7 value 13.799059
## iter   8 value 13.799058
## iter   8 value 13.799058
## iter   8 value 13.799058
## final  value 13.799058
## converged
```




```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = constant, transform.pars = trans, fixed = fixed, optim.control = list(trace = trc,
##         REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1      ar2      ma1     ma2  constant
##       1.2112  -0.5578  -0.1169  0.3675  356713.4
## s.e.  0.2034   0.1555   0.1765  0.2422  107136.5
##
## sigma^2 estimated as 9.385e+11:  log likelihood = -989.17,  aic = 1990.34
##
## $degrees_of_freedom
## [1] 60
##
## $ttable
##             Estimate          SE t.value p.value
## ar1           1.2112      0.2034  5.9555  0.0000
## ar2          -0.5578      0.1555 -3.5875  0.0007
## ma1          -0.1169      0.1765 -0.6624  0.5102
## ma2           0.3675      0.2422  1.5172  0.1345
## constant 356713.4311 107136.4530  3.3295  0.0015
##
## $AIC
```

```
## [1] 30.62061
##
## $AICc
## [1] 30.63625
##
## $BIC
## [1] 30.82132
```

c. Evaluate fit of all models generated

After differencing acf of all the series were evaluated and the parameters were estimated. For flight series, with parameters p=1,d=1,q=1,P=2,D=1,Q=1, AIC was 1371.23. From the results, we saw that the ar1, ma1,sma1 terms are not significant.

For passenger series, with parameters p=0,d=1,q=0,P=0,D=1,Q=1, AIC was 1974.21. For this model sma1 term is significant which is the only term in the model. This model is very close to box-jenkins airplane model for passengers. I tried the box=jenkins model as well. The residual plots of this model were very similar to my model. However, the extra ma1 term is not significant.

For load factor series, with parameters p=1,d=0,q=3,P=1,D=1,Q=1, AIC was 185.73. From the results, we saw that the ma2,ma3 and sar1 terms are not significant.

For operating revenue series, with parameters p=2,d=0,q=2,P=0,D=1,Q=0, AIC was 1990.34. From the results, we saw that the ma1 and ma2 terms are not significant.

Since, there are insignificant terms in my models, we can try reducing few terms and check the residuals again.

With some trials,I found that for load factor series,with p=0,d=1,q=1 and P=0,D=1,Q=1, the residuals plots were significant and we will use that.

I found that for operating revenue series, with p=2,d=0,q=0 and P=0,D=1,Q=0, the residuals plots were significant and we will use that. Also, there are no insignificant terms.

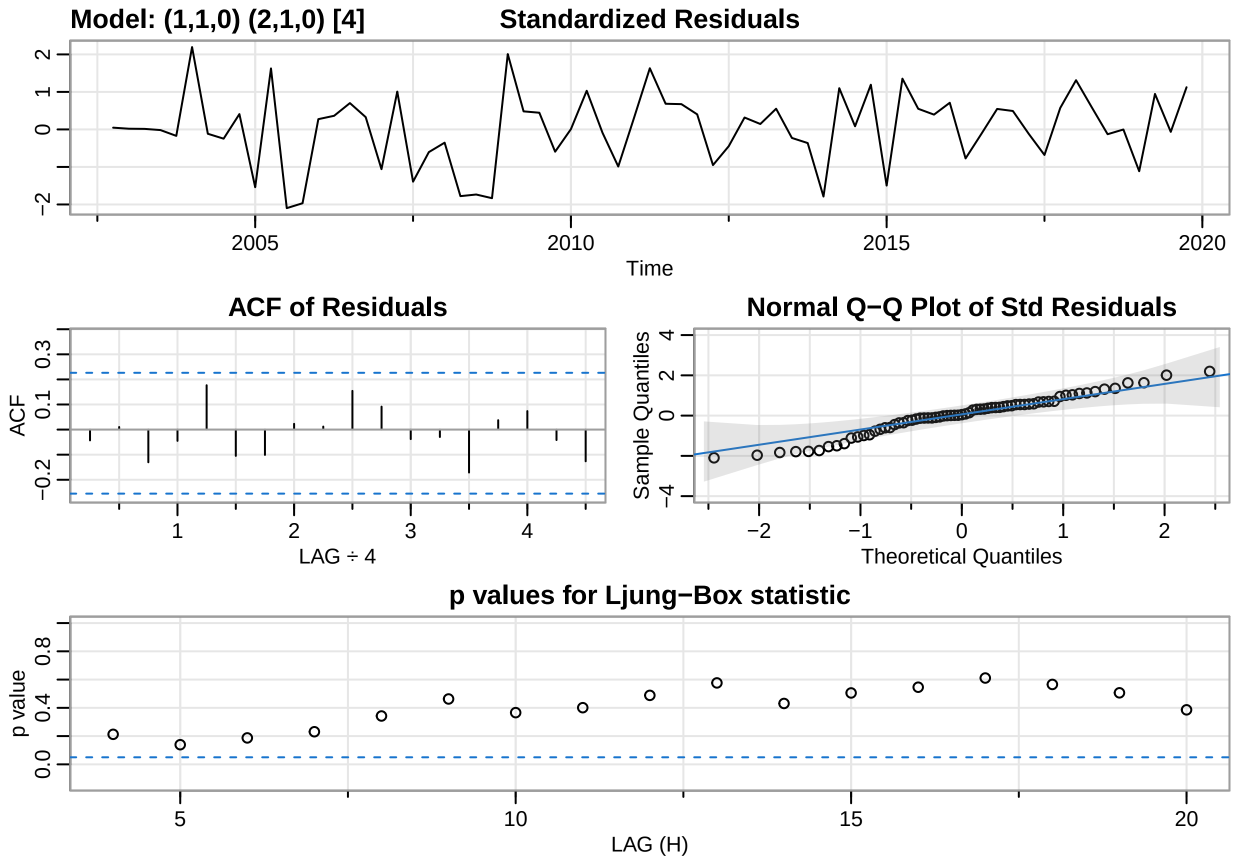
I found that for flight series, with p=1,d=1,q=0 and P=2,D=1,Q=1, the residuals plots were significant and we will use that. Also, now all the terms are significant.

Please find the new model and their residual plots below:

```
sarima(flight_ts,1,1,0,2,1,0,4)
```

```
## initial  value 9.429339
## iter   2 value 9.074311
## iter   3 value 9.072128
## iter   4 value 9.070758
## iter   5 value 9.070660
## iter   6 value 9.070602
## iter   7 value 9.070602
## iter   7 value 9.070602
## iter   7 value 9.070602
## final  value 9.070602
## converged
## initial  value 9.226472
## iter   2 value 9.220130
## iter   3 value 9.216861
## iter   4 value 9.215709
## iter   5 value 9.215698
```

```
## iter   5 value 9.215698
## iter   5 value 9.215698
## final  value 9.215698
## converged
```

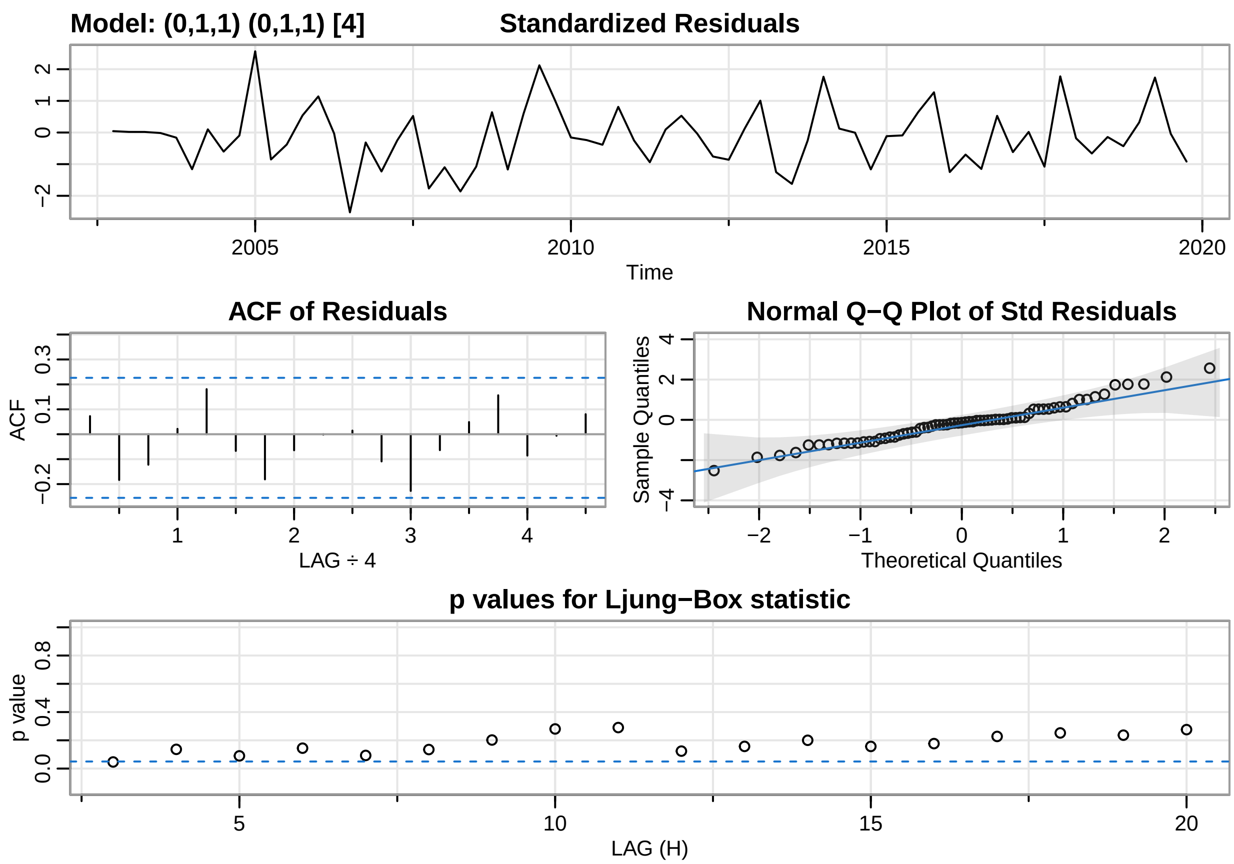



```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     include.mean = !no.constant, transform.pars = trans, fixed = fixed, optim.control = list(trace =
##         REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1     sar1     sar2
##       0.3179  -0.5301  -0.6256
## s.e.  0.1205   0.1014   0.1021
##
## sigma^2 estimated as 94188649:  log likelihood = -680.62,  aic = 1369.23
##
## $degrees_of_freedom
## [1] 61
##
## $ttable
##      Estimate     SE t.value p.value
## ar1    0.3179 0.1205  2.6372  0.0106
```

```
## sar1  -0.5301 0.1014 -5.2297  0.0000
## sar2  -0.6256 0.1021 -6.1294  0.0000
##
## $AIC
## [1] 21.39427
##
## $AICc
## [1] 21.40052
##
## $BIC
## [1] 21.5292
```

```
sarima(load_factor_ts,0,1,1,0,1,1,4)
```

```
## initial  value 0.103168
## iter   2 value -0.064897
## iter   3 value -0.084190
## iter   4 value -0.084989
## iter   5 value -0.085337
## iter   6 value -0.085343
## iter   7 value -0.085343
## iter   8 value -0.085343
## iter   8 value -0.085343
## iter   8 value -0.085343
## final  value -0.085343
## converged
## initial  value -0.079548
## iter   2 value -0.079601
## iter   3 value -0.079605
## iter   4 value -0.079605
## iter   4 value -0.079605
## iter   4 value -0.079605
## final  value -0.079605
## converged
```

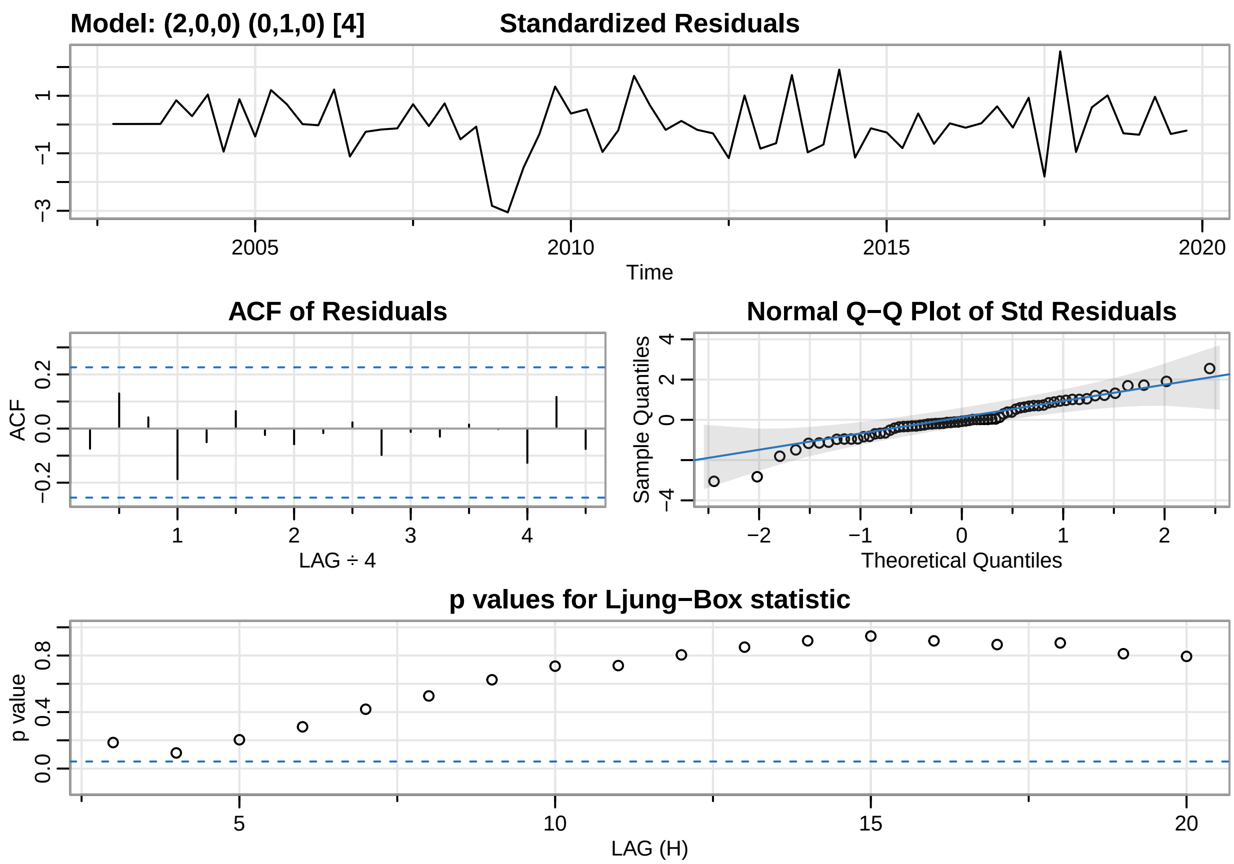



```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     include.mean = !no.constant, transform.pars = trans, fixed = fixed, optim.control = list(trace =
##         REPORT = 1, reltol = tol))
##
## Coefficients:
##           ma1     sma1
##       -0.5477  -0.4669
## s.e.   0.1192   0.1463
##
## sigma^2 estimated as 0.834:  log likelihood = -85.72,  aic = 177.43
##
## $degrees_of_freedom
## [1] 62
##
## $ttable
##      Estimate     SE t.value p.value
## ma1   -0.5477 0.1192 -4.5945  0.0000
## sma1  -0.4669 0.1463 -3.1907  0.0022
##
## $AIC
## [1] 2.772417
##
## $AICc
```

```
## [1] 2.77549
##
## $BIC
## [1] 2.873614
```

```r
sarima(op_rev_ts,2,0,0,0,1,0,4)
```

```
## initial  value 14.543912
## iter   2 value 14.422722
## iter   3 value 14.057184
## iter   4 value 13.905549
## iter   5 value 13.837453
## iter   6 value 13.823567
## iter   7 value 13.822376
## iter   8 value 13.822373
## iter   9 value 13.822373
## iter  10 value 13.822373
## iter  10 value 13.822373
## iter  10 value 13.822373
## final  value 13.822373
## converged
## initial  value 13.826858
## iter   2 value 13.826534
## iter   3 value 13.826423
## iter   4 value 13.826422
## iter   4 value 13.826422
## iter   4 value 13.826422
## final  value 13.826422
## converged
```




```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = constant, transform.pars = trans, fixed = fixed, optim.control = list(trace = trc,
##         REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1      ar2  constant
##       1.2325  -0.4948  362039.0
## s.e.  0.1054   0.1053  116404.3
##
## sigma^2 estimated as 9.957e+11:  log likelihood = -990.95,  aic = 1989.9
##
## $degrees_of_freedom
## [1] 62
##
## $ttable
##             Estimate          SE t.value p.value
## ar1           1.2325      0.1054 11.6980  0.0000
## ar2          -0.4948      0.1053 -4.6985  0.0000
## constant 362038.9612 116404.3111  3.1102  0.0028
##
## $AIC
## [1] 30.6138
##
```

```
## $AICc
## [1] 30.61985
##
## $BIC
## [1] 30.74761
```

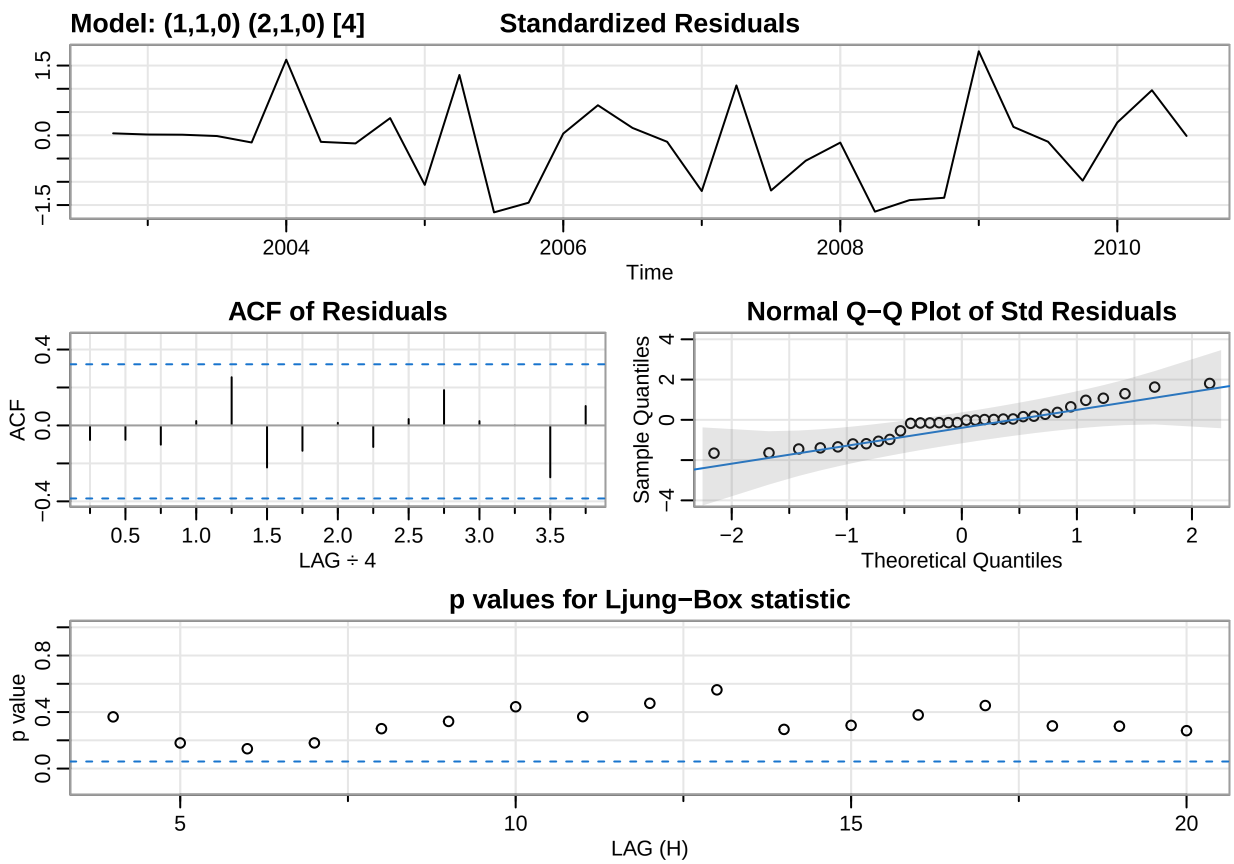
4. Additional analysis (15 points) – Include at least one of the following:

a. Forecasts from one or more series, including prediction intervals and evaluations of accuracy

Forecast for flight data

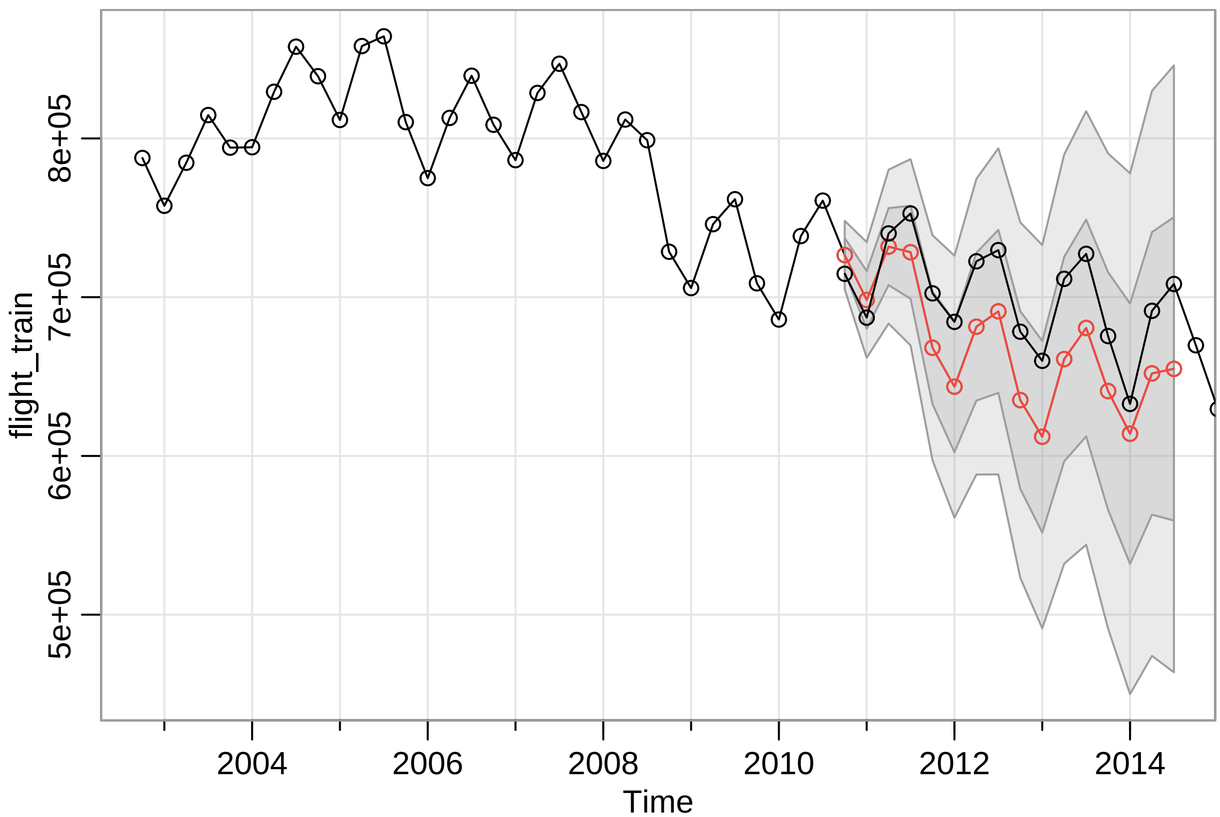
```
flight_train = window(flight_ts, end = c(2010, 3))
flight_test = window(flight_ts, start = c(2010, 4))
cc_flight= sarima(flight_train, 1,1,0,2,1,0,4)
```

```
## initial  value 9.793268
## iter   2 value 9.266260
## iter   3 value 9.239393
## iter   4 value 9.223831
## iter   5 value 9.221898
## iter   6 value 9.219591
## iter   7 value 9.219590
## iter   7 value 9.219590
## iter   7 value 9.219590
## final  value 9.219590
## converged
## initial  value 9.443504
## iter   2 value 9.430935
## iter   3 value 9.425133
## iter   4 value 9.420734
## iter   5 value 9.420598
## iter   6 value 9.420597
## iter   6 value 9.420597
## iter   6 value 9.420597
## final  value 9.420597
## converged
```




```
cc_flight_for = sarima.for(flight_train, n.ahead = 16,1,1,0,2,1,0,4)

lines(flight_test, type='o')
```




```
cc_flight_for$pred
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 2010                               726507.4
## 2011 698240.4 731834.8 728299.6 668119.9
## 2012 643628.0 681396.7 691074.9 635184.8
## 2013 612095.2 661002.9 680602.4 640863.2
## 2014 613993.9 652005.3 654863.7
```

```
cc_flight_for$se
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 2010                               10801.51
## 2011 18198.07 24240.75 29323.88 35443.58
## 2012 41259.79 46550.13 51362.48 55991.96
## 2013 60344.38 64429.33 68279.04 74840.53
## 2014 81998.66 88970.50 95566.10
```

```
accuracy(cc_flight_for$pred, flight_test)
```

```
##                ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
## Test set 31195.21 36905.81 34058.35 4.462302 4.870712 0.6617276 0.9816788
```

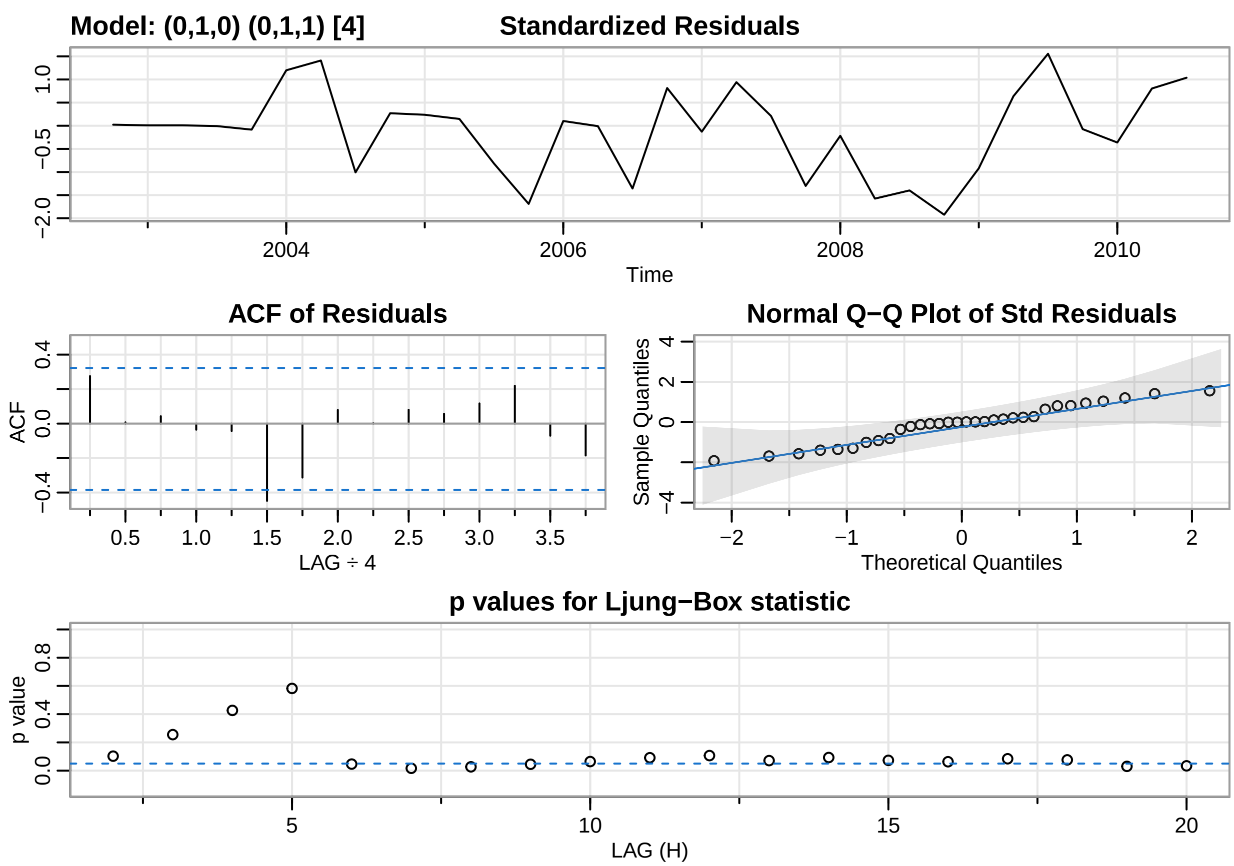
```
           ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
```

Test set 31195.21 36905.81 34058.35 4.462302 4.870712 0.6617276 0.9816788

Forecast for passenger data

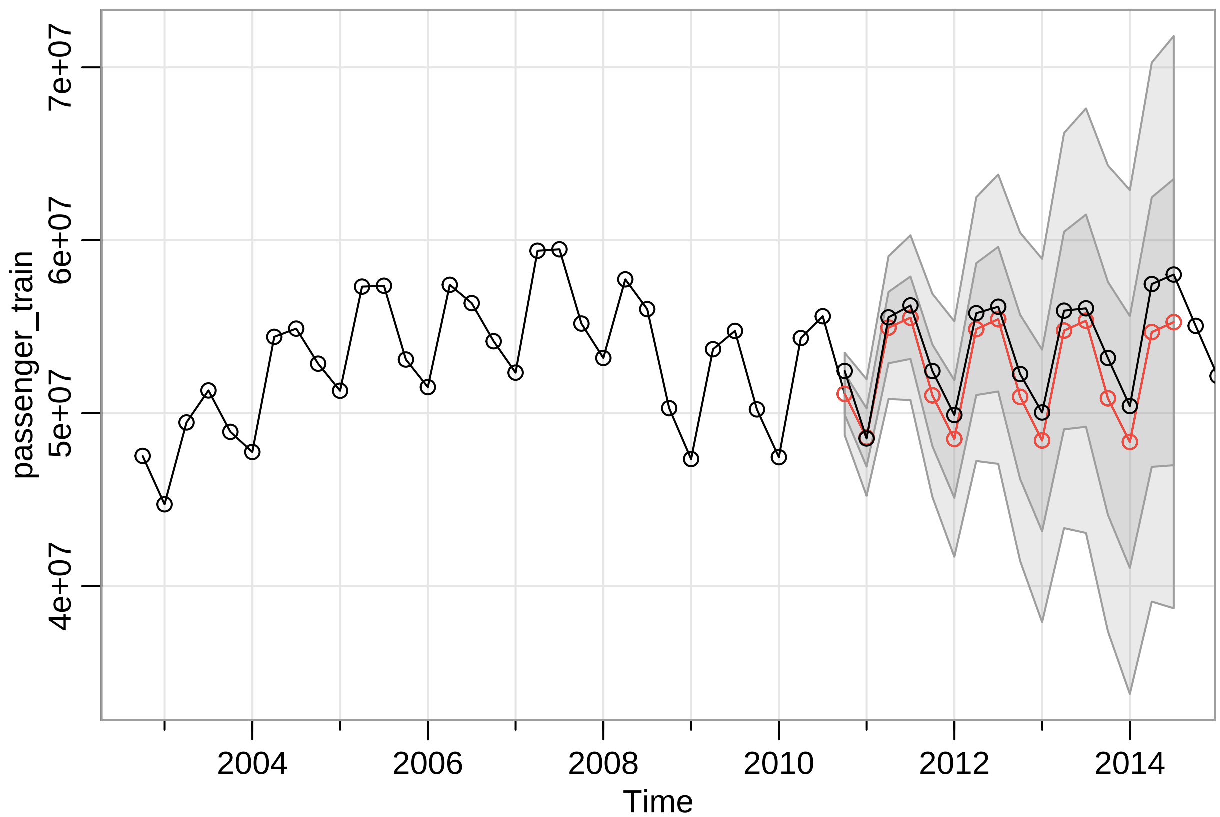
```r
passenger_train = window(passengers_ts, end = c(2010, 3))
passenger_test = window(passengers_ts, start = c(2010, 4))
cc_passenger = sarima(passenger_train, 0,1,0,0,1,1,4)
```

```
## initial  value 14.142820
## iter   2 value 14.030011
## iter   3 value 14.029859
## iter   4 value 14.029849
## iter   4 value 14.029849
## final  value 14.029849
## converged
## initial  value 14.020885
## iter   2 value 14.018149
## iter   3 value 14.017972
## iter   4 value 14.017972
## iter   4 value 14.017972
## final  value 14.017972
## converged
```




```r
cc_passenger_for = sarima.for(passenger_train, n.ahead = 16,0,1,0,0,1,1,4)

lines(passenger_test, type='o')
```




```r
cc_passenger_for$pred
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 2010                            51116411
## 2011 48593321 54946742 55520087 51030515
## 2012 48507425 54860846 55434191 50944619
## 2013 48421529 54774950 55348295 50858723
## 2014 48335633 54689054 55262399
```

```r
cc_passenger_for$se
```

```
##         Qtr1    Qtr2    Qtr3    Qtr4
## 2010                         1191268
## 2011 1684652 2063246 2382418 2937669
## 2012 3403427 3812707 4182123 4747120
## 2013 5251590 5711676 6137369 6734941
## 2014 7283560 7793656 8272357
```

```r
accuracy(cc_passenger_for$pred, passenger_test)
```

```
##               ME    RMSE     MAE      MPE     MAPE      ACF1 Theil's U
## Test set 1360008 1564227 1366996 2.520655 2.535053 0.5256521 0.3733236
```

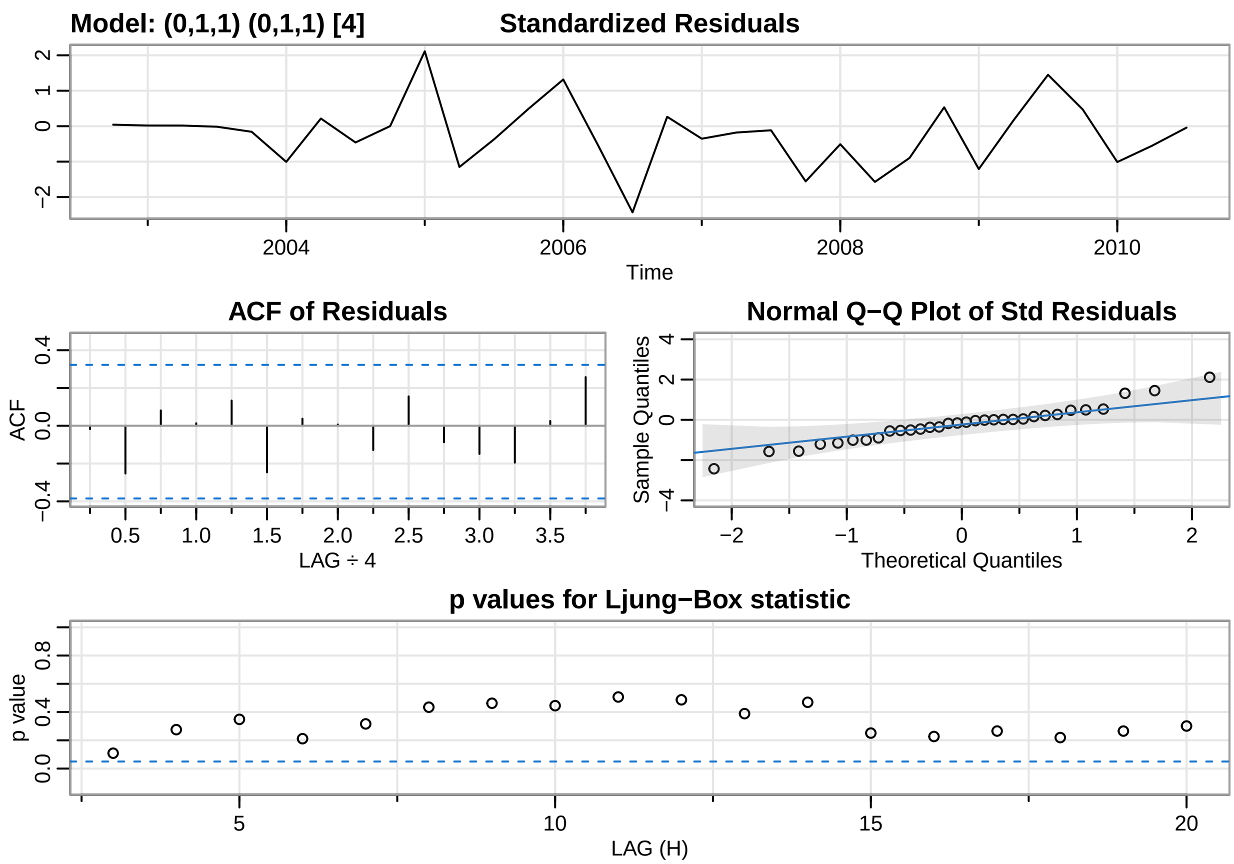
```
          ME    RMSE     MAE      MPE     MAPE      ACF1 Theil's U
```

Test set 1360008 1564227 1366996 2.520655 2.535053 0.5256521 0.3733236

Forecast for load factor data

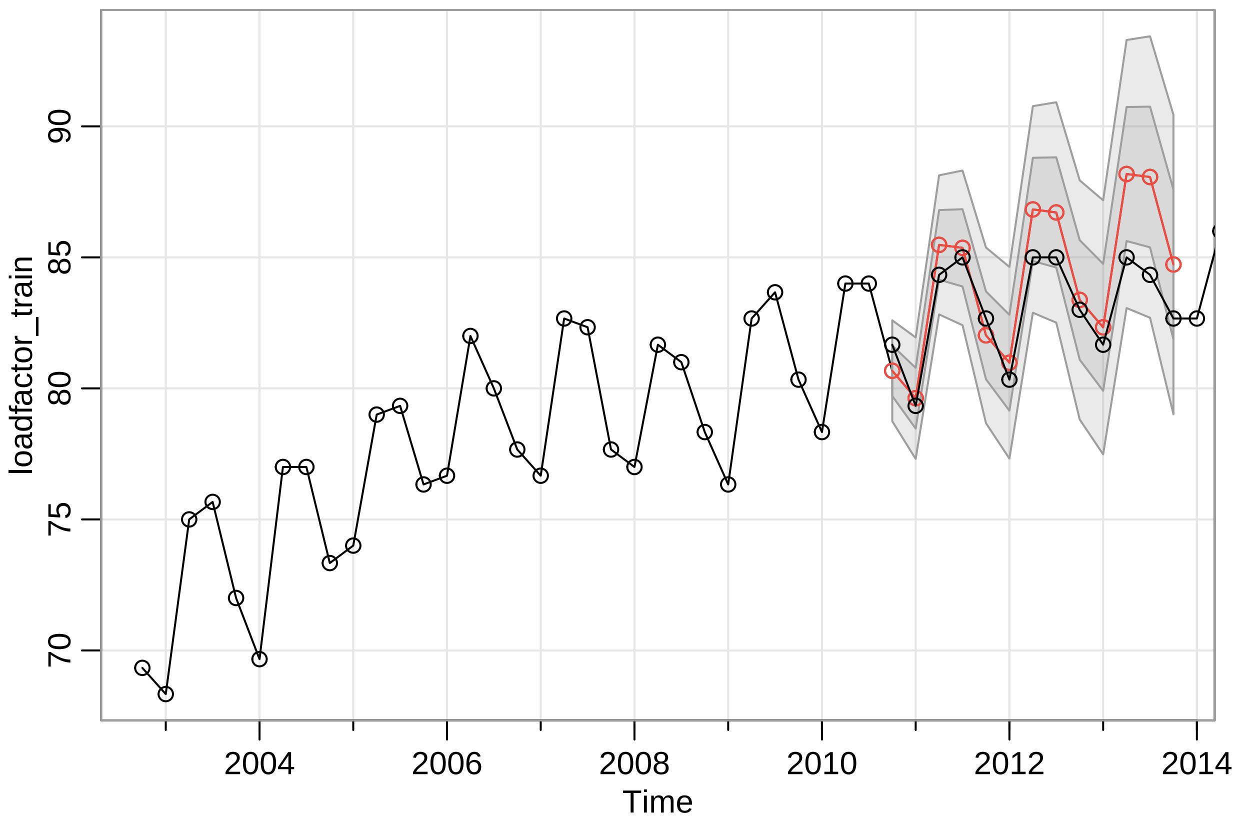
```
loadfactor_train = window(load_factor_ts, end = c(2010, 3))
loadfactor_test = window(load_factor_ts, start = c(2010, 4))
cc_load = sarima(loadfactor_train, 0,1,1,0,1,1,4)
```

```
## initial  value 0.302506
## iter   2 value 0.039202
## iter   3 value 0.028033
## iter   4 value 0.025819
## iter   5 value 0.025805
## iter   6 value 0.025803
## iter   7 value 0.025803
## iter   7 value 0.025803
## iter   7 value 0.025803
## final  value 0.025803
## converged
## initial  value 0.046554
## iter   2 value 0.045179
## iter   3 value 0.044345
## iter   4 value 0.043879
## iter   5 value 0.043817
## iter   6 value 0.043807
## iter   7 value 0.043806
## iter   8 value 0.043806
## iter   9 value 0.043805
## iter  10 value 0.043805
## iter  11 value 0.043805
## iter  11 value 0.043805
## final  value 0.043805
## converged
```




```
cc_load_for = sarima.for(loadfactor_train, n.ahead = 13,0,1,1,0,1,1,4)

lines(loadfactor_test, type='o')
```




```
loadfactor_test
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 2010                            81.66667
## 2011 79.33333 84.33333 85.00000 82.66667
## 2012 80.33333 85.00000 85.00000 83.00000
## 2013 81.66667 85.00000 84.33333 82.66667
## 2014 82.66667 86.00000 85.66667 82.66667
## 2015 82.33333 86.00000 86.33333 84.66667
## 2016 82.33333 86.00000 85.33333 84.33333
## 2017 82.00000 86.00000 84.66667 85.33333
## 2018 82.33333 85.66667 85.00000 84.33333
## 2019 82.33333 87.33333 85.66667 84.33333
```

```
cc_load_for$pred
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 2010                            80.67255
## 2011 79.62613 85.47697 85.36268 82.02445
## 2012 80.97803 86.82887 86.71458 83.37635
## 2013 82.32993 88.18077 88.06648 84.72825
```

```
cc_load_for$se
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 2010                            0.962678
## 2011 1.159241 1.327000 1.475811 1.675718
## 2012 1.829202 1.970768 2.102825 2.281664
## 2013 2.423540 2.557556 2.684892 2.855804
```

```r
accuracy(cc_load_for$pred, loadfactor_test)
```

```
##                 ME     RMSE      MAE       MPE     MAPE      ACF1 Theil's U
## Test set -1.105081 1.721169 1.356825 -1.312091 1.618885 0.4764248 0.6524814
```

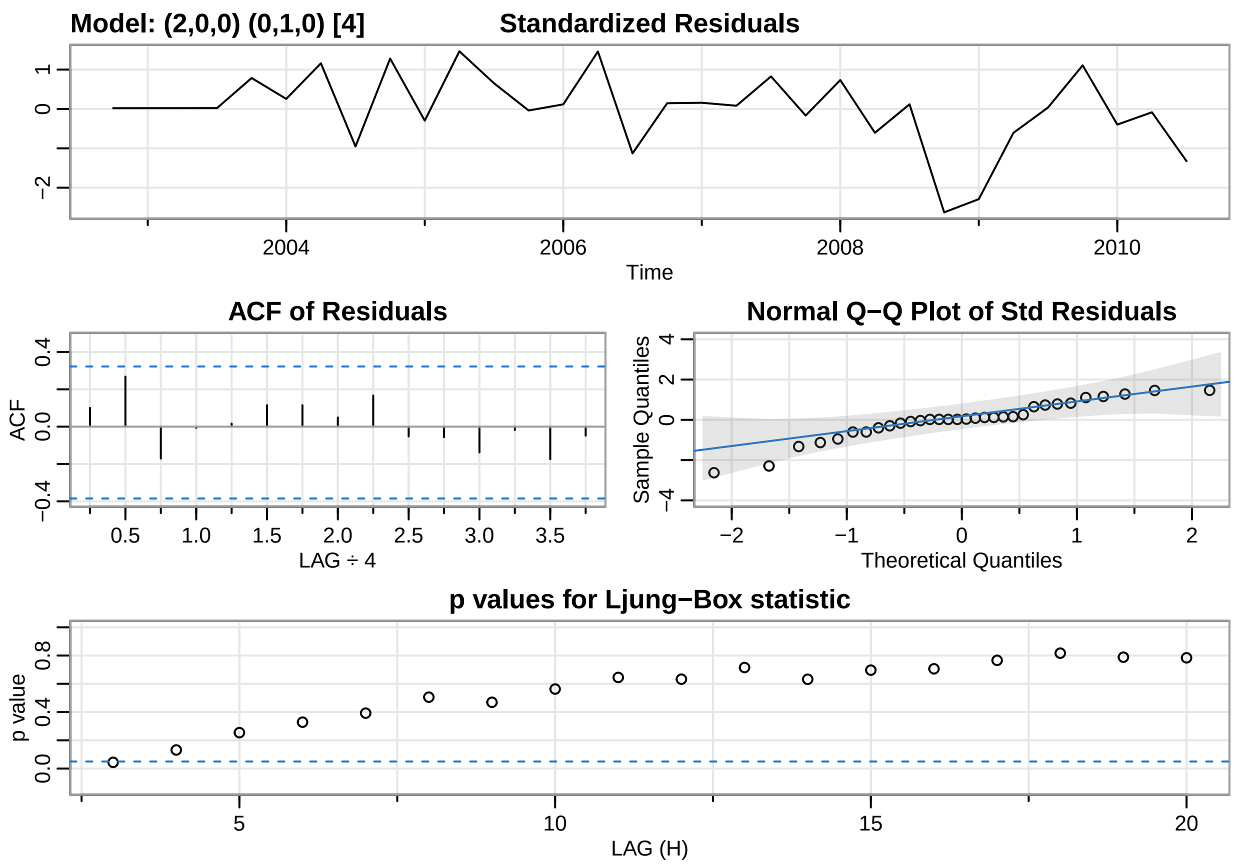
```
            ME     RMSE      MAE       MPE     MAPE      ACF1 Theil's U
```

Test set -1.105081 1.721169 1.356825 -1.312091 1.618885 0.4764248 0.6524814

Forecast for operating revenue data

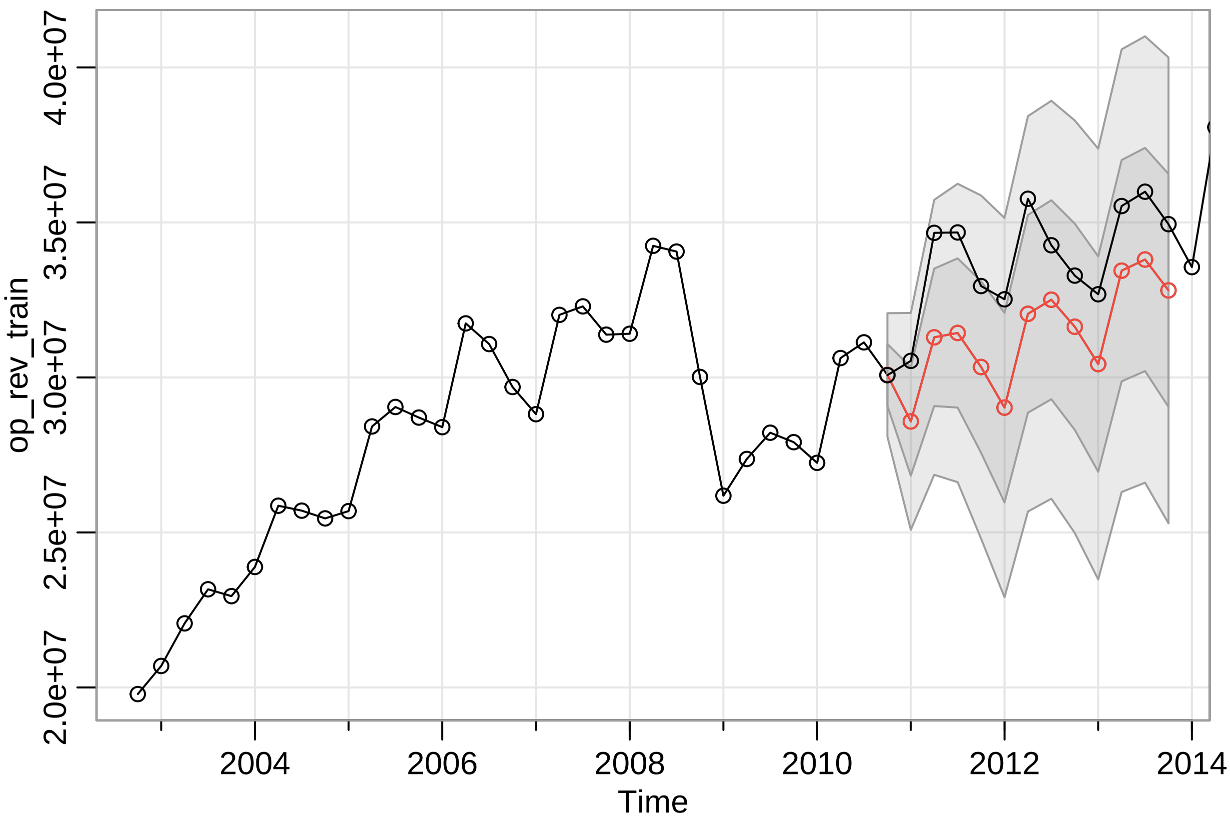
```r
op_rev_train = window(op_rev_ts, end = c(2010, 3))
op_rev_test = window(op_rev_ts, start = c(2010, 4))
cc_op_re = sarima(op_rev_train, 2,0,0,0,1,0,4)
```

```
## initial  value 14.868693 
## iter   2 value 14.729414
## iter   3 value 14.334569
## iter   4 value 14.096422
## iter   5 value 13.943609
## iter   6 value 13.859475
## iter   7 value 13.844423
## iter   8 value 13.836155
## iter   9 value 13.835042
## iter  10 value 13.834231
## iter  11 value 13.833790
## iter  12 value 13.833785
## iter  12 value 13.833785
## iter  12 value 13.833785
## final  value 13.833785 
## converged
## initial  value 13.861209 
## iter   2 value 13.860636
## iter   3 value 13.859104
## iter   4 value 13.859049
## iter   5 value 13.859036
## iter   6 value 13.859034
## iter   6 value 13.859034
## iter   6 value 13.859034
## final  value 13.859034 
## converged
```




```
cc_op_re_for = sarima.for(op_rev_train, n.ahead = 13,2,0,0,0,1,0,4)

lines(op_rev_test, type='o')
```




```
op_rev_test
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 2010                            30072585
## 2011 30534144 34663890 34676558 32947504
## 2012 32520037 35763241 34263076 33283857
## 2013 32682301 35527896 35990506 34944094
## 2014 33558448 38071207 37924357 36317906
## 2015 34400448 37988917 38170710 36372331
## 2016 34766496 38683010 39267618 38390204
## 2017 36984571 41730343 40500338 41323753
## 2018 39417877 44253860 43798660 44215557
## 2019 41375991 46580348 45748634 45636832
```

```
cc_op_re_for$pred
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 2010                            30081486
## 2011 28580419 31293257 31434161 30334722
## 2012 29025756 32051298 32505889 31634927
## 2013 30430656 33443308 33804079 32806864
```

```
cc_op_re_for$se
```

```
##            Qtr1      Qtr2      Qtr3      Qtr4
## 2010                                995723.5
## 2011 1749268.6 2217066.5 2405938.8 2767170.5
## 2012 3058731.3 3188558.9 3209344.6 3326637.4
## 2013 3474477.0 3569799.1 3599743.3 3757227.5
```

```
accuracy(cc_op_re_for$pred, op_rev_test)
```

```
##               ME    RMSE     MAE      MPE     MAPE      ACF1 Theil's U
## Test set 2341759 2527950 2343128 6.886783 6.891337 0.2830676  1.333473
```

ME RMSE MAE MPE MAPE ACF1 Theil's U

Test set 2341759 2527950 2343128 6.886783 6.891337 0.2830676 1.333473

5. Summary and implications

There were both seasonal and trend components in all the time series analysed here. Some of the series became stationary after just differencing the seasonal term. However, some series required additional differencing to make the series stationary.

After many trials, best model parameters were determined and were used for forcasting.

Conclusion:

1. After analyzing the trend components of each series, I found that the number of passengers, load factor and operating revenue increase over the time. However, the number flights has a decreasing trend.Let's break it down a bit.

Period till 2010:

i. The flights series has a decreasing trend. ii.The number of passengers are fairly constant over the time or we can say the series has a slightly increasing trend.
ii. The load factor exhibits a strong increasing trend.
iii. The operating revenue also increases over the time.

This proves our initial hypothesis that if increase in passengers is not followed by increase in number of flights and the operating revenue still increases over time it means that load factor must have improved. The airline services did not need to operate more number of flights as less people were travelling earlier. But as more people started travelling later, more seats were booked and increased the load factor. So even with less flights, the operating revenue increased. But we cannot correlate load factor with revenue as we know that the airline ticket prices also increased over the time which would be another important factor for increase in revenue apart from other factors.

Period from 2011:

i. The flights series has a slight decreasing trend initially but later increases. ii.The number of passengers exhibits a strong increasing trend.
ii. The load factor exhibits a fairly constant trend.
iii. The operating revenue also increases over the time.

Now, this tells us a different story. As we hypothesized, number of flights increases as more and more people start travelling. This makes sense, as after a certain time, the load factor would saturate and in order to increase revenue, airline services need to operate more flights for growing number of people. Now, obviously they can keep increasing prices keeping the number of flights same but this is not a viable option and this defeats the whole purpose of the airline services if they only let rich people to travel.

The saturation of the load factor is also evident in the trend plots.

2. All the series have a seasonal components confirming what we have experienced ourselves. Some months or quarters see more flights getting booked or flights being overbooked. The final models used for forecasting yielded good results and the MAPE of all the models were low.