AirTimeSeriesForecasting

Dr. B

Friday, May 15, 2015

## Start the Process

# Clear the environment  
rm(list=ls())  
  
# Turn off scientific notations for numbers  
options(scipen = 999)   
options(digits=8)  
  
# Set locale  
Sys.setlocale("LC\_ALL", "English")

## [1] "LC\_COLLATE=English\_United States.1252;LC\_CTYPE=English\_United States.1252;LC\_MONETARY=English\_United States.1252;LC\_NUMERIC=C;LC\_TIME=English\_United States.1252"

# Set seed for reproducibility  
set.seed(2345)  
  
# load the libraries  
library(forecast)

## Loading required package: zoo  
##   
## Attaching package: 'zoo'  
##   
## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric  
##   
## Loading required package: timeDate  
## This is forecast 5.9

library(TTR)

## Loading required package: xts

library(fpp)

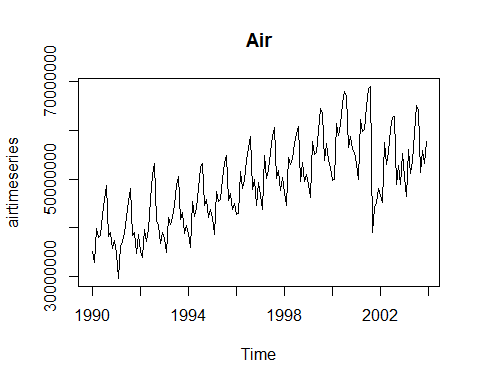
## Loading required package: fma  
## Loading required package: tseries  
## Loading required package: expsmooth  
## Loading required package: lmtest

## Load the Data and Plot

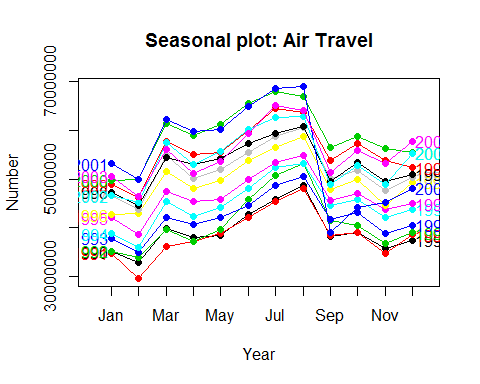
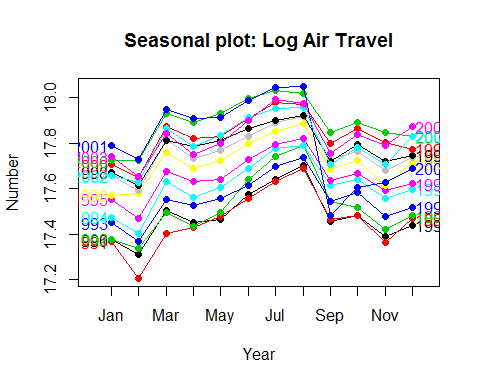
Load January 1990 to April 2004: Air Revenue Per Mile per (000)

# Load the data and make it into time series  
air <- scan("D:/Data/911air.txt")  
airtimeseries <- ts(air, frequency=12, start=c(1990,1))  
#air <- scan("D:/Data/SampleTimeSeries.txt")  
#airtimeseries <- ts(air, frequency=12, start=c(2010,1))

Plot the timeseries data:



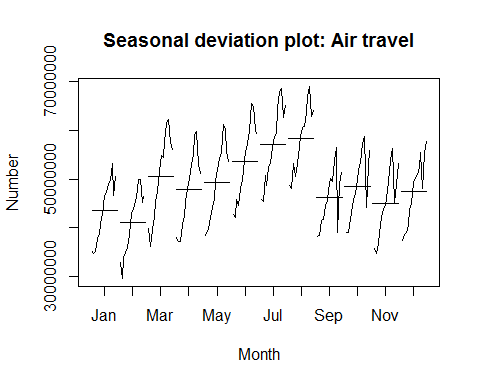
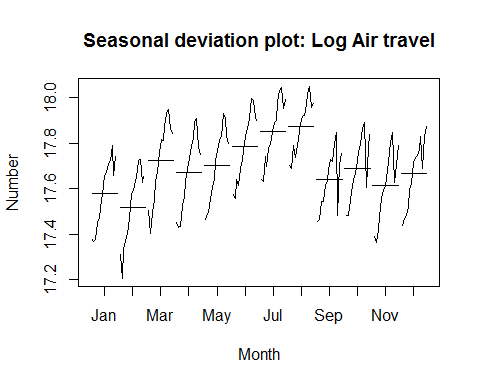
## Seasonal plots

A seasonal plot is similar to a time plot except that the data are plotted against the individual "seasons" in which the data were observed. An example is given below showing the antidiabetic drug sales.  

These are exactly the same data shown earlier, but now the data from each season are overlapped. A seasonal plot allows the underlying seasonal pattern to be seen more clearly, and is especially useful in identifying years in which the pattern changes.

In this case, it is clear that there is a large jump in sales in January each year. Actually, these are probably sales in late December as customers stockpile before the end of the calendar year, but the sales are not registered with the government until a week or two later. The graph also shows that there was an unusually low number of sales in March 2008 (most other years show an increase between February and March). The small number of sales in June 2008 is probably due to incomplete counting of sales at the time the data were collected.

## Seasonal subseries plots

An alternative plot that emphasises the seasonal patterns is where the data for each season are collected together in separate mini time plots.  

The horizontal lines indicate the means for each month. This form of plot enables the underlying seasonal pattern to be seen clearly, and also shows the changes in seasonality over time. It is especially useful in identifying changes within particular seasons. In the Antidiabetic Drug Sales example, the plot is not particularly revealing; but in some of the other examples, this plot is the most useful way of viewing seasonal changes over time.

## Decomposing Time Series

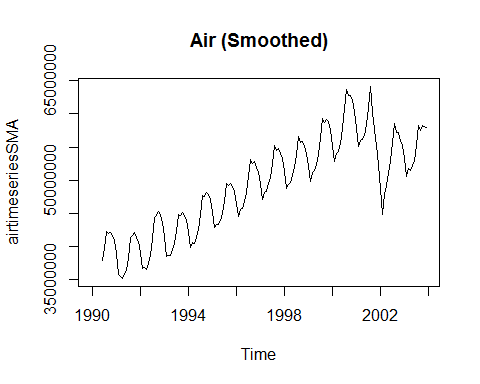
Decomposing a time series means separating it into its various components, which are usually a trend component and an irregular component, and if it is a seasonal time series, a seasonal component.

### Decomposing Non-Seasonal Data

A non-seasonal time series consists of a trend component and an irregular component. Decomposing the time series involves trying to separate the time series into these components, that is, estimating the the trend component and the irregular component. To estimate the trend component of a non-seasonal time series that can be described using an additive model, it is common to use a smoothing method, such as calculating the simple moving average of the time series. The SMA() function in the "TTR" package can be used to smooth time series data using a simple moving average.

To use the SMA() function, you need to specify the order (span) of the simple moving average, using the parameter "n". For example, to calculate a simple moving average of order 5, we set n=5 in the SMA() function.

#   
n=6  
airtimeseriesSMA <- SMA(airtimeseries,n=n)  
plot.ts(airtimeseriesSMA, main="Air (Smoothed)")

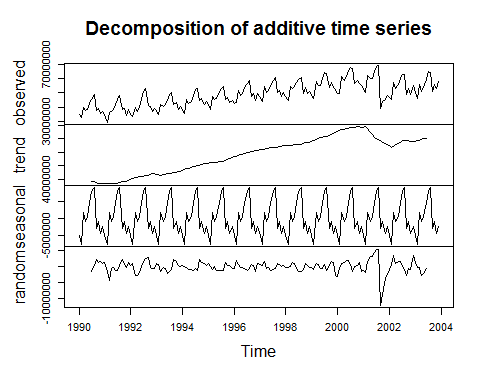


### Decomposing Seasonal Data

A seasonal time series consists of a trend component, a seasonal component and an irregular component. Decomposing the time series means estimating these three components. To estimate the trend component and seasonal component of a seasonal time series that can be described using an additive model, we can use the "decompose()" function in R. This function estimates the trend, seasonal, and irregular components of a time series that can be described using an additive model.

The function "decompose()" returns a list object as its result, where the estimates of the seasonal component, trend component and irregular component are stored in named elements of that list objects, called "seasonal", "trend", and "random" respectively.

#   
airtimeseriescomponents <- decompose(airtimeseries)  
plot(airtimeseriescomponents)

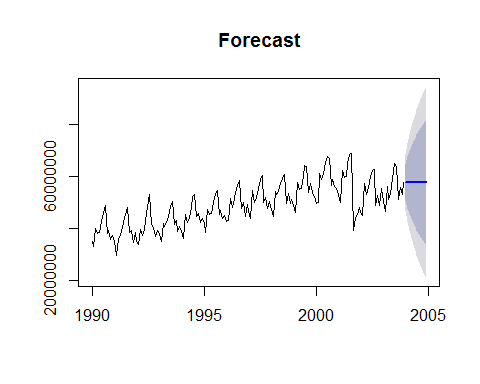


## Forecasting Methods

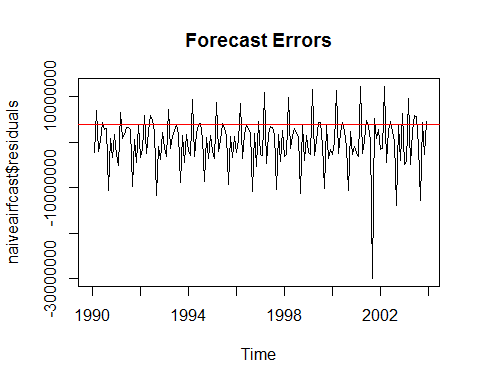
### Naive Method

This method is only appropriate for time series data. All forecasts are simply set to be the value of the last observation.

# naive(y, h) or rwf(y,h)  
# y contains the time series  
# h is the forecast horizon  
naiveairfcast <- naive(airtimeseries,12)  
plot(naiveairfcast,main="Forecast")



plot(naiveairfcast$residuals, main="Forecast Errors")  
MANE <-accuracy(naiveairfcast)[,'MAE']  
abline(h=MANE,col="red")



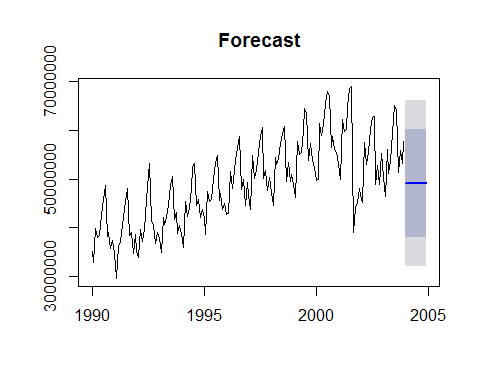
accuracy(naiveairfcast)

## ME RMSE MAE MPE MAPE MASE  
## Training set 135582.82 5457958.5 4049155.7 -0.33661789 8.4887371 1.4612963  
## ACF1  
## Training set -0.24384502

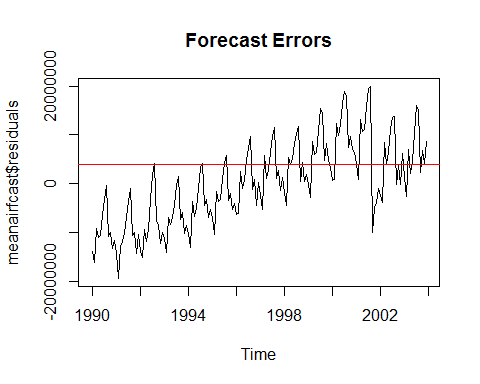
### Average method

The forecasts of all future values are equal to the mean of the historical data.

# meanf(y, h)   
# y contains the time series  
# h is the forecast horizon  
meanairfcast <- meanf(airtimeseries,12)  
plot(meanairfcast,main="Forecast")



plot(meanairfcast$residuals,main="Forecast Errors")  
abline(h=MANE,col="red")



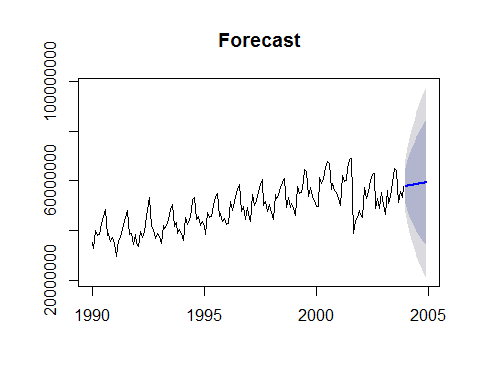
accuracy(meanairfcast)

## ME RMSE MAE MPE MAPE  
## Training set 0.000000003545454 8541605.3 7044880.1 -3.1677606 15.016544  
## MASE ACF1  
## Training set 2.5424206 0.78606545

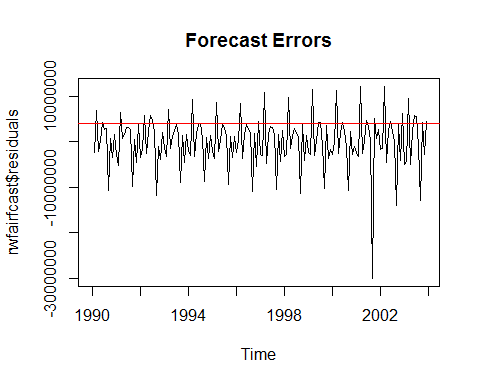
### Drift method

A variation on the naïve method is to allow the forecasts to increase or decrease over time, where the amount of change over time (called the drift) is set to be the average change seen in the historical data. This is equivalent to drawing a line between the first and last observation, and extrapolating it into the future.

#rwf(y,h, drift=TRUE)  
# y contains the time series  
# h is the forecast horizon  
rwfairfcast <- rwf(airtimeseries,12, drift=TRUE)  
plot(rwfairfcast, main="Forecast")



plot(rwfairfcast$residuals, main="Forecast Errors")  
abline(h=MANE,col="red")



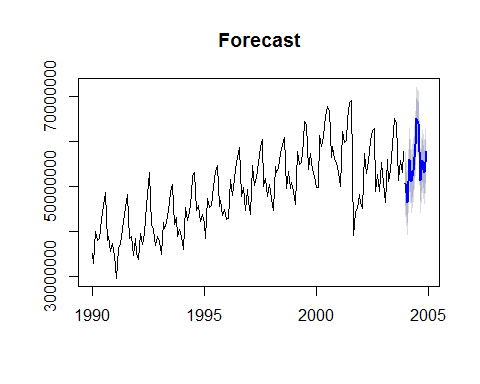
accuracy(rwfairfcast)

## ME RMSE MAE MPE MAPE  
## Training set -0.00000000062447976 5456274.2 4028908 -0.62117598 8.4624248  
## MASE ACF1  
## Training set 1.4539891 -0.24384502

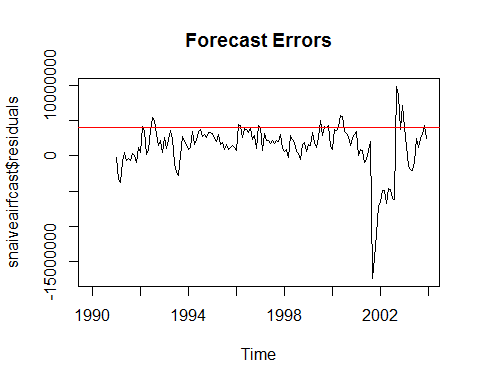
### Seasonal naive method

A similar method is useful for highly seasonal data. In this case, we set each forecast to be equal to the last observed value from the same season of the year (e.g., the same month of the previous year).

# naive(y, h) or rwf(y,h)  
# y contains the time series  
# h is the forecast horizon  
snaiveairfcast <- snaive(airtimeseries,12)  
plot(snaiveairfcast, main="Forecast")



plot(snaiveairfcast$residuals, main="Forecast Errors")  
abline(h=MANE,col="red") #MANE from naive



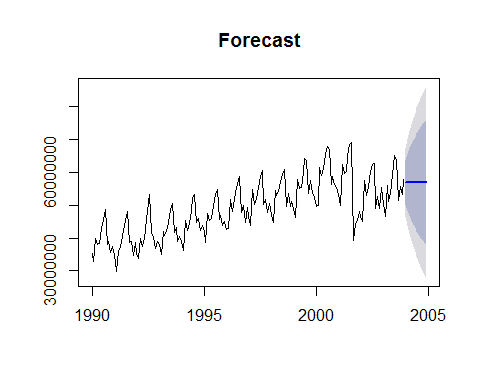
accuracy(snaiveairfcast)

## ME RMSE MAE MPE MAPE MASE  
## Training set 1233080.2 3670853.4 2770934.1 2.3513434 5.6461282 1  
## ACF1  
## Training set 0.71885486

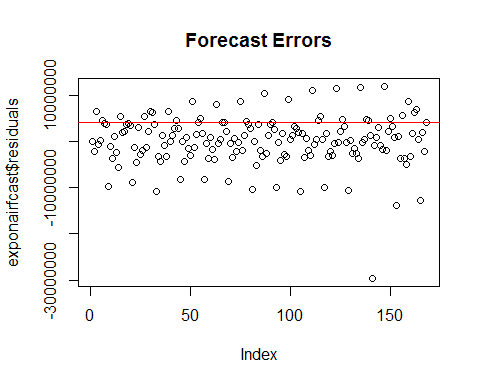
### Simple exponential smoothing

The simplest of the exponentially smoothing methods is naturally called "simple exponential smoothing" (SES).

exponairfcast<- ses(airtimeseries, alpha=0.8, initial="simple", h=12)  
plot(exponairfcast, main="Forecast")



plot(exponairfcast$residuals, main="Forecast Errors")  
abline(h=MANE,col="red") #MANE from naive



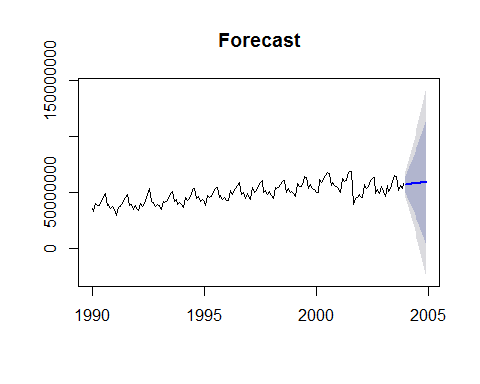
accuracy(exponairfcast)

## ME RMSE MAE MPE MAPE MASE  
## Training set 162351.13 5271788.8 3789047.6 -0.34437844 7.9349435 1.3674261  
## ACF1  
## Training set -0.064973685

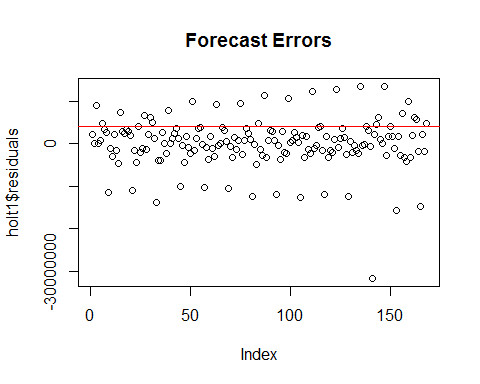
### Holt's linear trend method

Holt (1957) extended simple exponential smoothing to allow forecasting of data with a trend. This method involves a forecast equation and two smoothing equations (one for the level and one for the trend):

holt1 <- holt(airtimeseries, alpha=0.8, beta=0.2, initial="simple", h=12)   
holt2 <- holt(airtimeseries, alpha=0.8, beta=0.2, initial="simple", exponential=TRUE, h=12)   
holt3 <- holt(airtimeseries, alpha=0.8, beta=0.2, damped=TRUE, initial="simple", h=12)   
plot(holt1, main="Forecast")



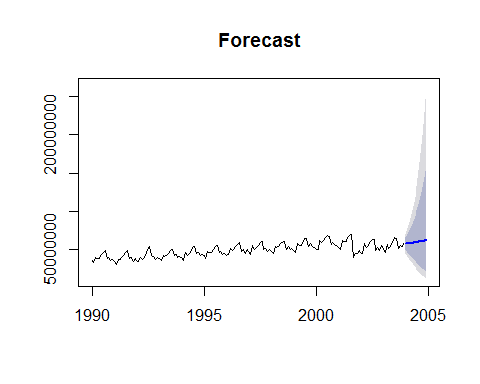
plot(holt1$residuals, main="Forecast Errors")  
abline(h=MANE,col="red") #MANE from naive



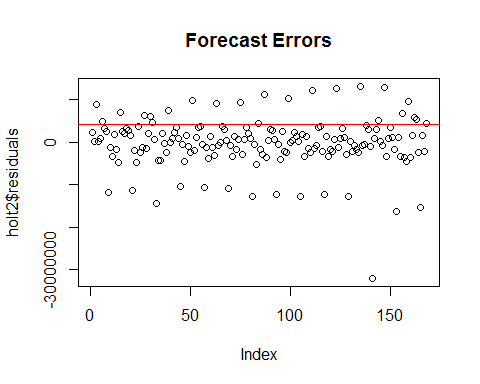
accuracy(holt1)

## ME RMSE MAE MPE MAPE MASE  
## Training set 87840.253 5735280.7 3863914.2 -0.32193713 8.1433855 1.3944447  
## ACF1  
## Training set -0.055357563

plot(holt2, main="Forecast")



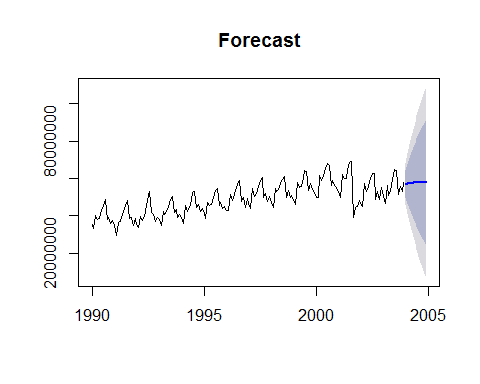
plot(holt2$residuals, main="Forecast Errors")  
abline(h=MANE,col="red") #MANE from naive



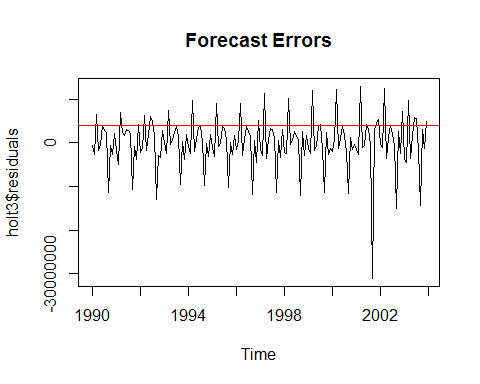
accuracy(holt2)

## ME RMSE MAE MPE MAPE MASE  
## Training set -234532.3 5759568.1 3847852.1 -0.98700757 8.1351565 1.388648  
## ACF1  
## Training set -0.046444452

plot(holt3, main="Forecast")



plot(holt3$residuals, main="Forecast Errors")  
abline(h=MANE,col="red") #MANE from naive



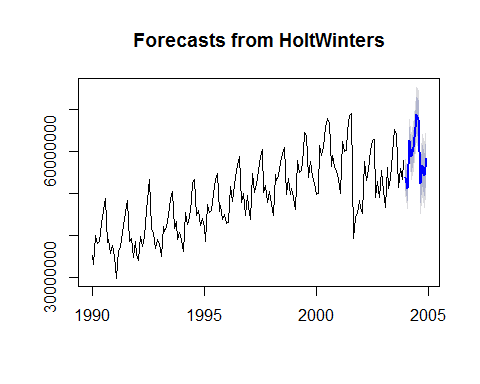
accuracy(holt3)

## ME RMSE MAE MPE MAPE MASE  
## Training set 72140.519 5546752.8 3794161.6 -0.41015765 7.9859113 1.3692717  
## ACF1  
## Training set -0.12818391

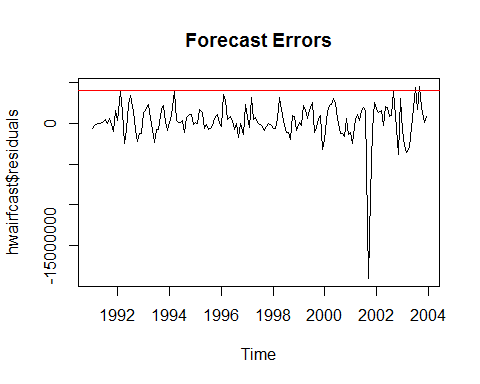
### Holt-Winters exponential smoothing

If you have a time series that can be described using an additive model with increasing or decreasing trend and seasonality, you can use Holt-Winters exponential smoothing to make short-term forecasts. Holt-Winters exponential smoothing estimates the level, slope, and seasonal component at the current time point.

airHWforecast <- HoltWinters(airtimeseries)  
hwairfcast <- forecast.HoltWinters(airHWforecast, h=12)  
plot(hwairfcast)



plot(hwairfcast$residuals,main="Forecast Errors")   
abline(h=MANE,col="red") #MANE from naive

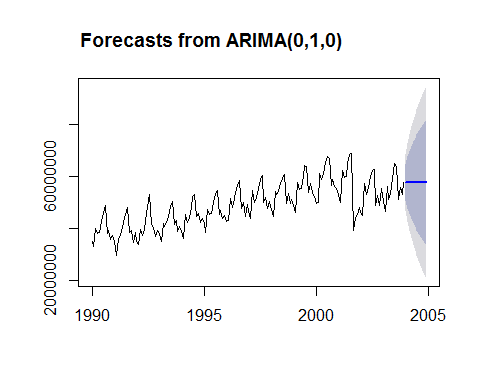


accuracy(hwairfcast)

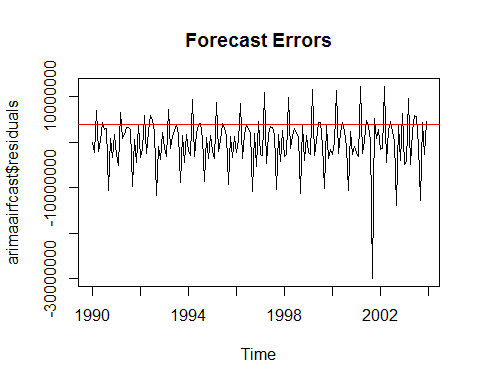
## ME RMSE MAE MPE MAPE MASE  
## Training set 252762.99 2344940.6 1426766.2 0.58119644 2.9345077 0.50741727  
## ACF1  
## Training set 0.35271844

### ARIMA Model

# fit an ARIMA(0,1,1) model  
arimaairtimeseries <- arima(airtimeseries, order=c(0,1,0))   
arimaairfcast<- forecast.Arima(arimaairtimeseries, h=12)  
plot(arimaairfcast)



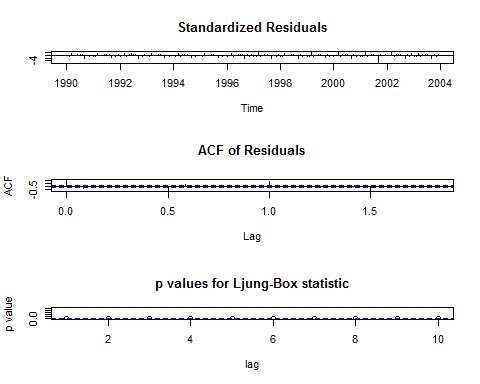
plot(arimaairfcast$residuals,main="Forecast Errors")   
abline(h=MANE,col="red") #MANE from naive



accuracy(arimaairfcast)

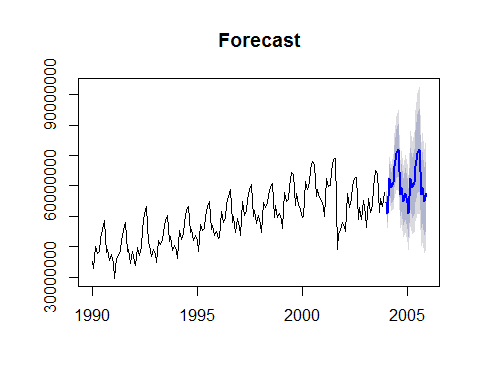
## ME RMSE MAE MPE MAPE MASE  
## Training set 134985.03 5441691 4025262.8 -0.33401898 8.4388041 1.4526736  
## ACF1  
## Training set -0.24379811

tsdiag(arimaairtimeseries)

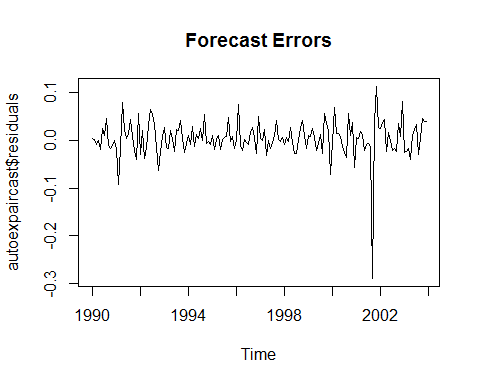


### Automated forecasting using an exponential model

# Automated forecasting using an exponential model  
autoexpaircast<-ets(airtimeseries)  
plot(forecast(autoexpaircast), main="Forecast")



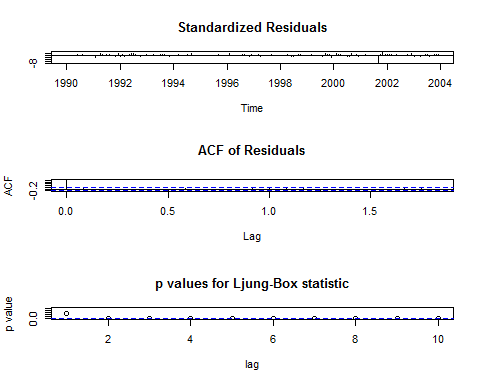
plot(autoexpaircast$residuals, main="Forecast Errors")  
abline(h=MANE,col="red") #MANE from naive



accuracy(autoexpaircast)

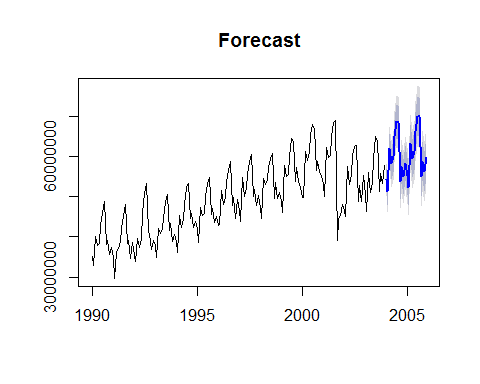
## ME RMSE MAE MPE MAPE MASE  
## Training set 140705.79 1853211.9 1144660.7 0.20720425 2.3985129 0.28269119  
## ACF1  
## Training set 0.046441684

tsdiag(autoexpaircast)

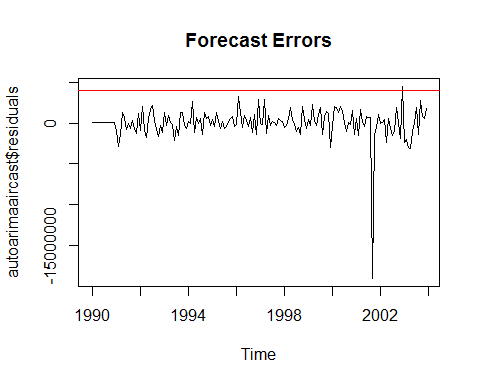


### Automated ARIMA Model

# Automated forecasting using an ARIMA model  
autoarimaaircast<-auto.arima(airtimeseries)  
plot(forecast(autoarimaaircast), main="Forecast")



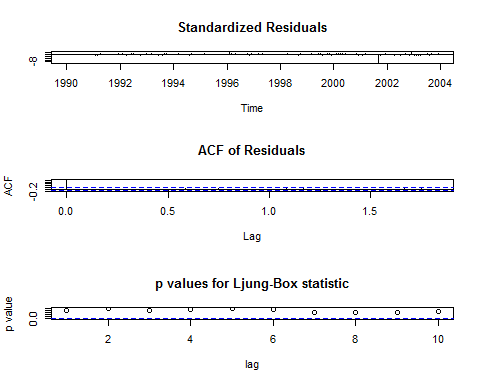
plot(autoarimaaircast$residuals, main="Forecast Errors")  
abline(h=MANE,col="red") #MANE from naive



accuracy(autoarimaaircast)

## ME RMSE MAE MPE MAPE MASE  
## Training set -332.2908 1932945.8 1054067.7 -0.13064969 2.2206098 0.2603179  
## ACF1  
## Training set 0.018238102

tsdiag(autoarimaaircast)



### Summary

A summary of Training Set MASE for the all methods covered:

accuracy(naiveairfcast)[,'MASE']

## [1] 1.4612963

accuracy(meanairfcast)[,'MASE']

## [1] 2.5424206

accuracy(rwfairfcast)[,'MASE']

## [1] 1.4539891

accuracy(snaiveairfcast)[,'MASE']

## [1] 1

accuracy(exponairfcast)[,'MASE']

## [1] 1.3674261

accuracy(holt1)[,'MASE']

## [1] 1.3944447

accuracy(holt2)[,'MASE']

## [1] 1.388648

accuracy(holt3)[,'MASE']

## [1] 1.3692717

accuracy(hwairfcast)[,'MASE']

## [1] 0.50741727

accuracy(arimaairfcast)[,'MASE']

## [1] 1.4526736

accuracy(autoexpaircast)[,'MASE']

## [1] 0.28269119

accuracy(autoarimaaircast)[,'MASE']

## [1] 0.2603179

This is an [R Markdown document](http://rmarkdown.rstudio.com). Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents.