ADS 506: Monthly Amtrak Ridership Forecasting Final Project

Team 3: Jimmy Nguyen, Luke Awino

11/27/2021

Contents

Libraries

```
library(astsa)
library(readr)
library(forecast)
library(zoo)
library(xts)
library(pander)
library(tidyverse)
library(tseries)
library(lubridate)

knitr::opts_chunk$set(warning = FALSE, message = FALSE)
```

Data Set

Code:

```
# Load the data set from CSV file
df <- read_csv("../Data/Amtrak Ridership Data.csv")


# Rename columns
names(df)[1] <- 'Dates'
names(df[2]) <- 'Number_of_Passengers'


# First 12 months in 1991
head(df, n = 12) %>%
  pander(style = "grid", caption = "First 12 Months - 1991")
```

Table 1: First 12 Months - 1991

Dates	Number of Passengers
Jan-91	1708917
Feb-91	1620586
Mar-91	1972715
Apr-91	1811665
May-91	1974964
Jun-91	1862356
Jul-91	1939860
Aug-91	2013264
Sep-91	1595657
Oct-91	1724924
Nov-91	1675667
Dec-91	1813863

Data Exploration

Code:

```
# convert to time series object
df<- ts(data = df[,2], start = c(1991,1),
        end = c(2013,5), frequency = 12)

print("Starting Year and Month: ")
```

```
## [1] "Starting Year and Month: "
```

```
start(df)
```

```
## [1] 1991    1
```

```
print("Final Year and Month: ")
```

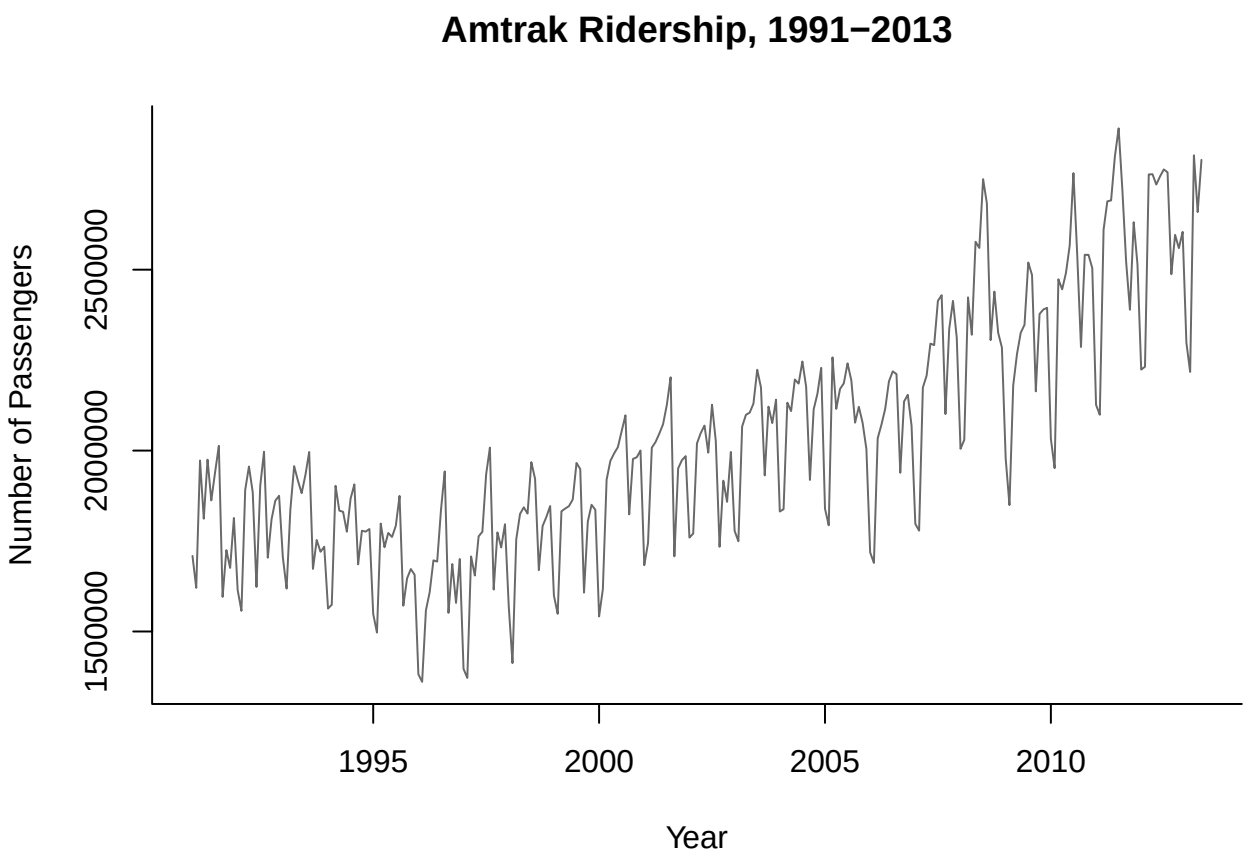
```
## [1] "Final Year and Month: "
```

```
end(df)
```

```
## [1] 2013    5
```

Code:

```
# Make a quick time-series plot
plot(df, xlab = "Year", ylab = "Number of Passengers",
     bty = "l", col = "grey41",
     main = "Amtrak Ridership, 1991-2013")
```



Data Pre-processing

Smoothing Methods (Moving Average)

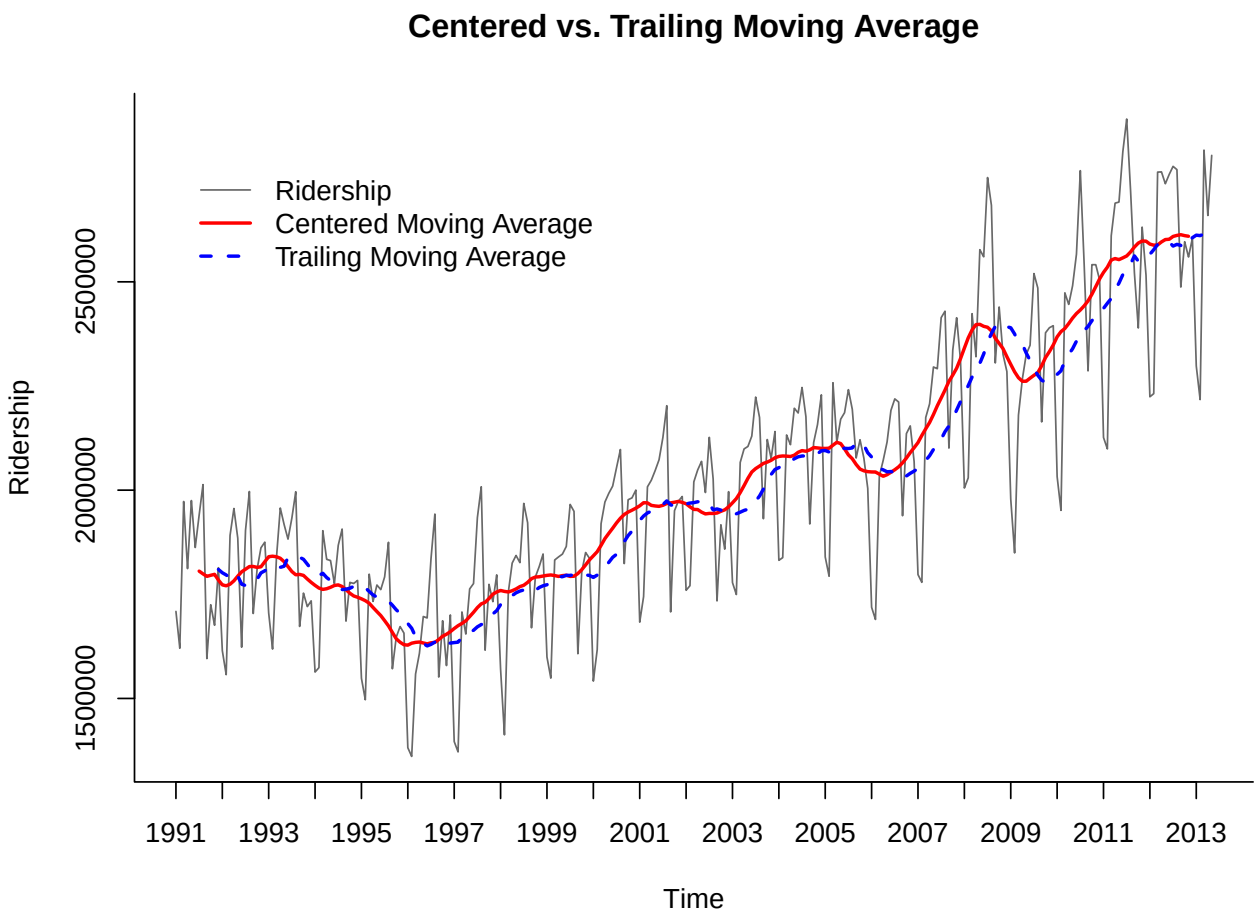
At the beginning of our data exploration, we were able to see there are components that change over time. Thus, we will need to look into data-driven methods because it deals data without a predetermined structure.

Moving Average

- This method is a simple smoother, it contains the average values across a time window, w, specified by the user. Two types of moving averages:
- a centered-moving average: useful for visualizing trends since averaging can suppress seasonality and noise
- a trailing moving average: useful for forecasting

Code:

```
# Trailing Average
ma.trailing <- rollmean(df, k = 12, align = "right")

# Centered-Average
ma.centered <- ma(df, order = 12)

# Original Data
plot(df, ylab = "Ridership", xlab = "Time",
     bty = "l", xaxt = "n", col = 'grey41',
     main = "Centered vs. Trailing Moving Average")

# Labels
axis(1, at = seq(1991, 2013.50, 1), labels = format(seq(1991, 2013.50, 1)))

# Centered average lines
lines(ma.centered, lwd = 2, col = 'red')

# trailing moving average
lines(ma.trailing, lwd = 2, lty = 2, col ='blue')

# legend
legend(1991,2800000, c("Ridership","Centered Moving Average",
                       "Trailing Moving Average"), lty=c(1,1,2),
lwd=c(1,2,2), bty = "n", col = c("grey41", 'red', 'blue'))
```



Analysis:

Since the goal is to suppress seasonality in the data to visualize the trend, we should choose the length of a seasonal cycle. *The Amtrak ridership data indicates a choice of* $w = 12$.

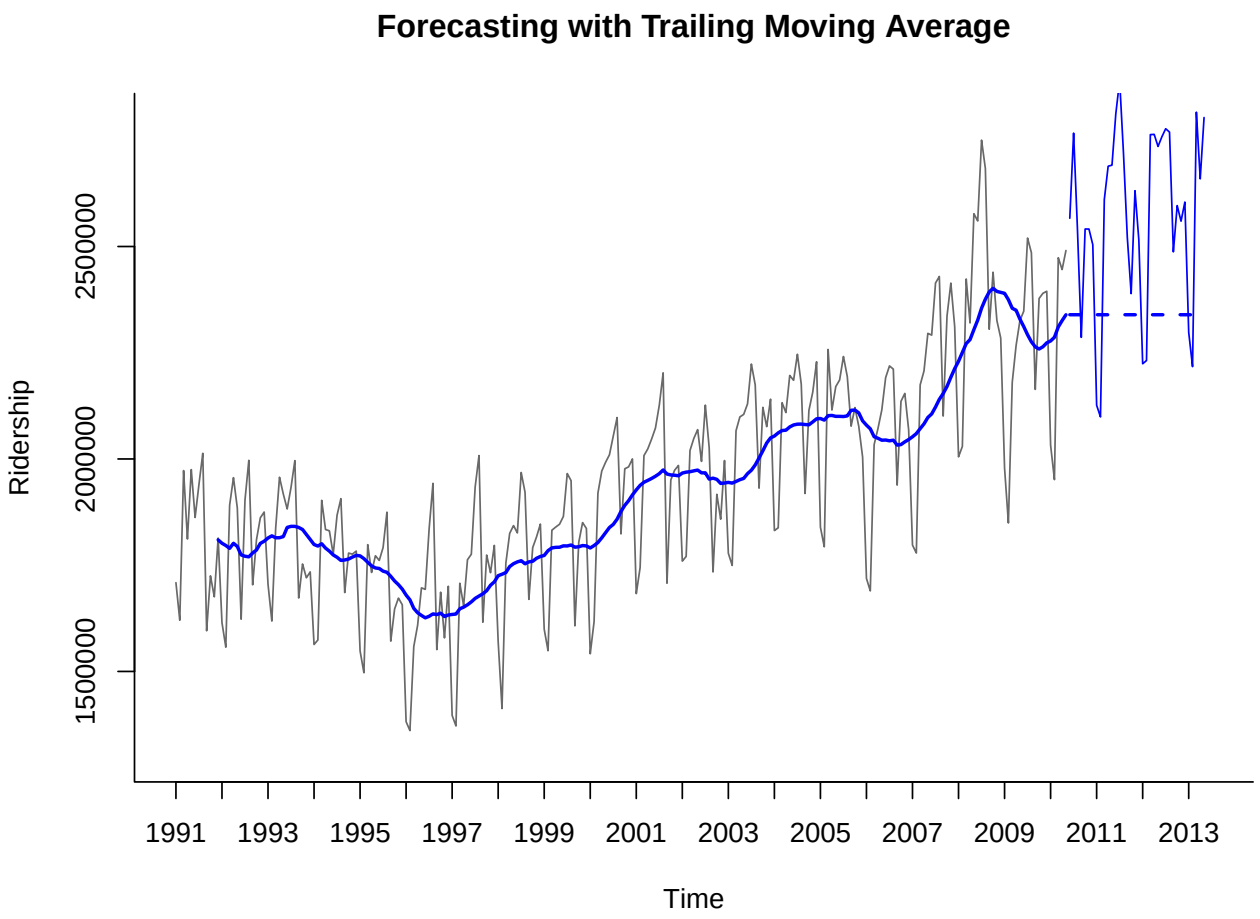
- This figure shows somewhat of a global U-shape, but the moving average looks to increase as the year passes.
- However, since centered moving averages uses data both in the past and future of a given time point, they cannot be used for forecasting because the future is typically unknown.

Trailing Moving Average

Therefore, trailing moving averages is the better approach here where the window of width is placed over the most recent available values.

```
# Validation Data
nValid <- 36

# Training data
nTrain <- length(df) - nValid

# time window for training data
train.ts <- window(df, start = c(1991, 1), end = c(1991, nTrain))

# time window for validation data
valid.ts <- window(df, start = c(1991, nTrain + 1), end = c(1991, nTrain + nValid))
```

```r
# trailing moving average
ma.trailing <- rollmean(train.ts, k = 12, align = "right")

# last trailing moving average
last.ma <- tail(ma.trailing, 1)

# prediction by trailing moving average
ma.trailing.pred <- ts(rep(last.ma, nValid), start = c(1991, nTrain + 1),
end = c(1991, nTrain + nValid), freq = 12)

# plot
plot(train.ts, ylim = c(1300000, 2800000), ylab = "Ridership",
     xlab = "Time", bty="l", xaxt = "n", col = 'grey41',
xlim = c(1991,2013.50), main = "Forecasting with Trailing Moving Average")

# labels
axis(1, at = seq(1991, 2013.50, 1), labels = format(seq(1991, 2013.50, 1)))

# training data
lines(ma.trailing, lwd = 2, col = "blue")

# validation data
lines(valid.ts, col = 'blue')

# predictions
lines(ma.trailing.pred, lwd = 2, col = "blue", lty = 2)
```



Analysis:

- The first thing to notice is that the forecasts for all the months in the validation period denoted in blue and blue dashes are identical because this method is not roll-forward next month forecasts. It is clear that the trailing moving average forecaster is inadequate for the Amtrak monthly forecast task. The reason why is because it does not capture the seasonality in the data. The forecaster predicted seasons with high ridership with lower ridership and seasons with low ridership with high ridership. This occurs because the moving average lags behind when forecasting a time series with a trend. Therefore, over-forecasting and under-forecasting in the presence of increasing and decreasing trends. So, between the smoothing methods of moving averages, it should only be use for forecasting when a series lack seasonality and trend, which is not true here for the Amtrak ridership data.
- However, there are other approaches for removing trends and seasonality, such as regression models or differencing.
- Then we can use the moving average to forecast a de-trended and de-seasonalized series.

Differencing

Differencing is a popular method for removing trend or seasonality patterns by taking the difference between two values in a series.

- For example, the lag-1 difference takes the difference between every two consecutive values ($y_t - y_{t-1}$).
- Meanwhile, differencing at lag-k means to subtract the value from k-periods back ($y_t - y_{t-k}$).

Dickey-Fuller Test

However, before using differencing as a pre-processing step, we should run a Dickey-Fuller test to see if differencing is actually needed. In other words, this test also check if the time series is stationary or not.

Code:

```
# Running the Dickey-Fuller Test on the original data without any pre-processing
adf.test(df)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  df
## Dickey-Fuller = -3.8062, Lag order = 6, p-value = 0.01915
## alternative hypothesis: stationary
```

Analysis:

This may be a biased Dickey-Fuller test where we have a type 1 error. There is definitely visible seasonality happening in the data.
Since the test rejects the null hypothesis that the series is non-stationary, in this case the series was actually non-stationary. Therefore, we will ignore that it ever happened because we will need to perform a second order difference due to a trend and seasonality pattern in the series. - Alternative approaches without differencing can be aggregating the monthly data into a coarser level such as yearly ridership as the total sum instead.

Detrending

Detrending can be used by the lag-1 difference of a series. This would remove the somewhat U-shape of Amtrak ridership series. An advantage of difference is there are no assumptions that the trend is global.

- For quadratic or exponential trends, one more step of lag-1 differencing must be applied to remove the trend.

Deseasonalizing

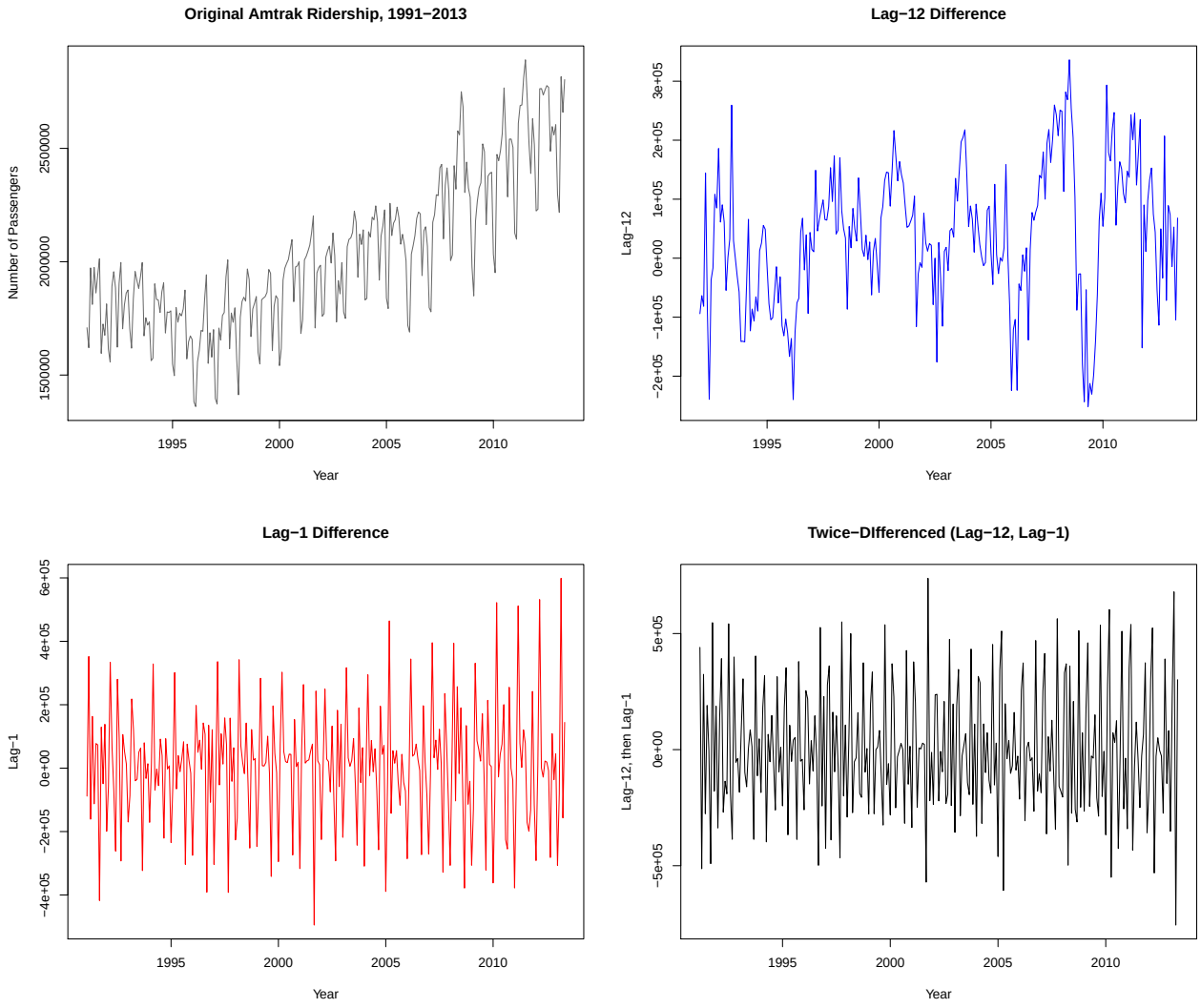
We can remove the seasonality of the Amtrak ridership data by using a lag-12 difference series. THis will remove the monthly pattern

Removing seasonality and trend

- When both of these components exist, we can apply differencing twice to the series.
- Since the Amtrak ridership data has both trend and seasonality, we will perform the doube differencing method in order to de-trend and deseasonalize it.

Code:

```
par(mfrow=c(2,2))

# Original
plot(df, xlab = "Year", ylab = "Number of Passengers",
     main = "Original Amtrak Ridership, 1991-2013", col = 'grey41')

# lag-12 difference
plot(diff(df, lag = 12), xlab = "Year", ylab = "Lag-12",
     main = "Lag-12 Difference", col = 'blue')

# lag-1 difference
plot(diff(df), xlab = "Year", ylab = "Lag-1",
     main = "Lag-1 Difference", col = 'red')


# Double Differencing
plot(diff(diff(df, s = 12)), xlab = "Year", ylab = "Lag-12, then Lag-1",
     main = "Twice-DIfferenced (Lag-12, Lag-1)", col = 1)
```



Analysis:

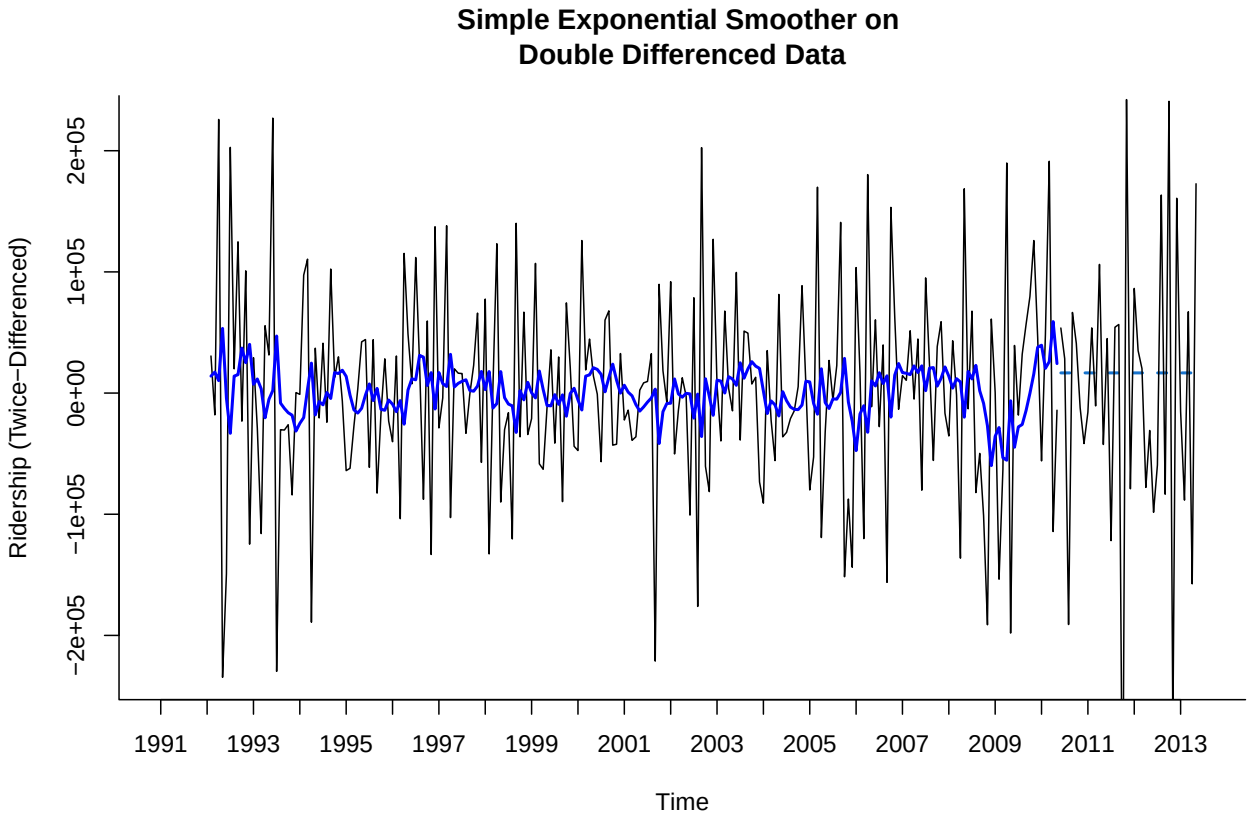
- The lag-1 difference plot on the bottom left contains no visible trend compared to the original series above.
- If we are dealing with daily data ridership, then we could remove a seasonal pattern of lag-7 differences. However, since we are have monthly data, we are using a lag-12 difference series as shown in the top right figure where the monthly pattern is absent.
- Lastly, since there are both a seasonality and trend, the double differencing effect on the bottom right panel is a series without trend or monthly seasonality.

Simple Exponential Smoothing

Exponential smoothing works very similar to forecasting with a moving average, except it takes the weighted average over all the past values of a series. By doing so, the weights will decrease exponentially into the past. This is valuable because we give weight to recent information more than the older information. This method is also very popular due to its low computation costs, easy automation, and good performance. However, it is important to note that using exponential smoothing for forecasting assumes no trend or seasonality in a series. So the idea is similar to before with moving averages, by first removing the trend and seasonality, then apply the exponential smoothing forecaster.

Code:

```
# remove trend and seasonality by doing double-differencing
diff_twice <- diff(diff(df, lag = 12), lag = 1)

# Number of validation data
nValid <- 36

# number of training data
nTrain <- length(diff_twice) - nValid

# specified time window for training data
train.ts <- window(diff_twice, start = c(1992, 2), end = c(1992, nTrain + 1))

# specified time window for validation data
valid.ts <- window(diff_twice, start = c(1992, nTrain + 2),
                   end = c(1992, nTrain + 1 + nValid))

# Additive, no trend, no seasonality model using a constant (learning rate) of 0.2
ses <- ets(train.ts, model = "ANN", alpha = 0.2)

# make predictions using model
ses.pred <- forecast(ses, h = nValid, level = 0)

# plot predictions
plot(ses.pred, ylab = "Ridership (Twice-Differenced)", xlab = "Time",
bty = "l", xaxt = "n", xlim = c(1991,2013.50), main = "Simple Exponential Smoother on
Double Differenced Data", flty = 2)

# labels
axis(1, at = seq(1991, 2013, 1), labels = format(seq(1991, 2013, 1)))


# validaiton data
lines(valid.ts)


# model from training and validation data
lines(ses.pred$fitted, lwd = 2, col = "blue")
```



Analysis:

For forecasting with simple exponential smoothing, the data trained on was the double-difference ridership data that contains no seasonality or trend. Then we fit the simple exponential smoothing model to the training data set with $\alpha = 0.2$ as default. This smoothing model is under the *ets* framework using the model with additive (A), no trend (N), and no seasonality (N), denoted as "ANN". The forecasts for the validation set for each month remained as the same value similar to the moving average forecast model. This would mean that the simple exponential forecaster or ANN model is also inadequate for the monthly forecasting task. The reason why is because it also does not capture the seasonality in the data.

- Since both moving average and simple exponential smoothing models should only be used for forecasting a time series without trend or seasonality, one solution attempted with these models were de-trending or de-seasonalizing the data.
- Another solution is to use a more complex and sophisticated exponential smoothing that is able to model data with both trend and seasonality.