Laboratory 3 Forecasting with Regression Including Smoothing Techniques such as Moving Averages and Exponential Smoothing

Purpose and Alignments Written Report

ANA 535 Forecasting

Name KOHEI NISHITANI Student ID# 1122867

Date May 2nd , 2025

# Table of Contents

* Introduction - 3
* Background - 3
* Data - 4
* Methods and Procedures - 4
* Results - 11
* Conclusions - 14
* References - 14
* Attachment or Appendices - 14

# Introduction

Lab3 shifts its focus from diagnosing seasonality and stationery to producing usable forecast for Amtrak dataset. Building the foundation of periodicity insights in Lab2, this Lab3 exercise will enable researcher to familiarize time-series regression, moving-average smoothing, ETS and Holt-winters smoothing. Those methodologies translate observed patterns into predictable patterns. The workflow at this lab begins with exploratory data analysis. The target variable passenger-miles are inspected to see if it has normal distribution or any other skewness, then it go through log-transformation because of observed skewness. After this preliminary inspection, a tslm() regression spanning 1991‑2024 establishes a first fitted‑versus‑actual benchmark. Then showing how model fitting can be changed with different training windows —1991‑1997, 1991‑2004, 1991‑2016, and 1991‑2020—to illustrate how structural breaks, most apparent the 2020 COVID‑19 collapse, Then demonstrate other methodologies like ETS and Holt-Winter’s model.

**Background**

The previous lab exercise established all necessary data cleaning and exploratory analysis to ensure reliable forecasting results. We cleaned the Amtrak passenger‑miles data, sorting, filtering and then analyzed it using frequency‑domain methods including Fast Fourier Transform and periodograms and STL decomposition to reveal both strong annual patterns. Consistent patterns were successfully identified and eliminated in periodic data during Lab 2, but that exercise did not proceed to generate meaningful forecasted results. The absence of regression analysis and smoothing techniques as well as accuracy evaluation means decision‑making professionals still need quantitative projections together with uncertainty understanding for future demand.

This Lab 3 closes that gap. The exercise applies time-series regression and moving-average filters alongside Holt-Winters, ETS, exponential-smoothing models to generate forecasts from the seasonally adjusted dataset produced in Lab 2. The statistical evaluation comparing multiple prediction periods before and after the COVID pandemic with multiple different training period and the use of basic forecasting methods in Lab 3 supplies crucial information for Amtrak managers to make schedule changes based on changing travel patterns.

**Data**

Data from Amtrak source enables the exploration of time-series analysis periodicity finding and adjustment through this exercise. The Amtrak dataset covers a time span from January 1991 to June 2024 holding about four hundred monthly records. The records contain a month field along with Ridership, PassengerMiles and RidersReported fields. For modelling purposes, the target variable is PassengerMiles. Preliminary checks in Lab 2 confirmed strong annual seasonality and a long‑run upward trend.

**Methods and Procedures**

A histogram shows strong left-skew at initial EDA(figure 1). Applying a log transform change the distribution to see if it made distribution more Normal curved distribution, satisfying the normality assumption used in later regression and smoothing models. However, this log transformation keeps the left-skewness after the transformation, so it doesn’t solve the skewness. Instead of log transformation, square transform changes the distribution more normally curved distribution.

Given square transformation, Using the transformed series, three polynomial time-trend regressions were fitted with ggplot2::geom\_smooth(), cubic fit shows better fitting than other type of regressions(figure 2).

A group of graphs showing different types of lines

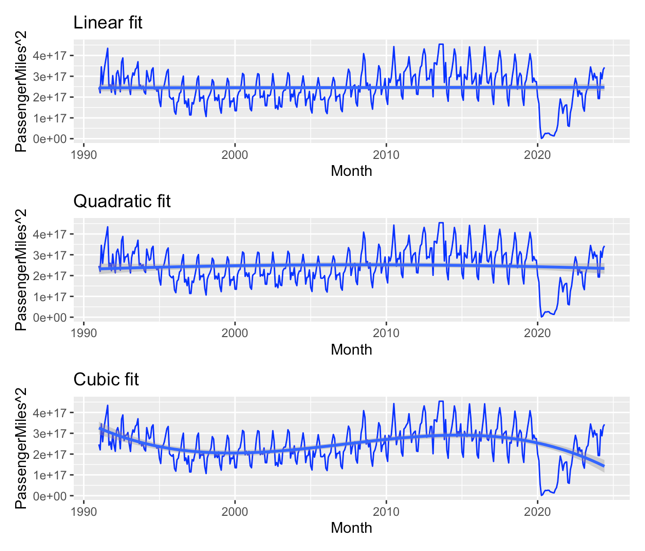
AI-generated content may be incorrect. 

Figure. 1 Figure. 2

After EDA four model variants were applied in a single model() call: (1) linear + trend, (2) linear + season(), (3) quadratic + season, and (4) cubic + season(figure 3). For all period, the accuracy statistics ranked the cubic-season specification best (i.e., lowest MAPE). However, this cubic fitting doesn’t capture the drop well(figure 4), so re-estimated on four historical windows—1991-1997, 1991-2004, 1991-2016, and 1991-2020—to capture the best MAPE with best fitting(figure 5-12). Across four historical windows, cubic fitting has least MAPE among other fittings, as anticipated cubic fitting at 1991-2020 has largest MAPE than other periods.

All period Best fitting model Cubic MAPE:17.3

A graph with numbers and lines

AI-generated content may be incorrect.A graph of a passenger miles per month

AI-generated content may be incorrect.

Figure. 3 Figure. 4

1991-1997 Best fitting model Cubic: MAPE 3.56

A graph showing different colored lines

AI-generated content may be incorrect.A graph of a graph showing the number of miles per month

AI-generated content may be incorrect.

Figure. 5 Figure. 6

1991-2004 Best fitting model Cubic: MAPE 4.15

A graph showing a number of different colored lines

AI-generated content may be incorrect.A graph of an airplane passenger miles per month

AI-generated content may be incorrect.

Figure. 7 Figure. 8

1991-2016 Best fitting model Cubic: MAPE 4.43

A graph showing a number of different colored lines

AI-generated content may be incorrect.A graph of a passenger miles per month

AI-generated content may be incorrect.

Figure. 9 Figure. 10

1991-2020 Best fitting model Cubic: MAPE 14.0

A graph with colorful lines

AI-generated content may be incorrect.A graph of a passenger miles per month

AI-generated content may be incorrect.

Figure. 11 Figure. 12

Because of MAPE and the challenge of COVID sudden dropping, analysis focuses on 1991-2016 window instead of rest of it.(figure 13) Then gg\_tsresiduals reveals residual of histogram, time-plot, and ACF of the residuals(figure 14). Overall, the residuals appear well-behaved for real-world data: the ACF drops off quickly, indicating little remaining autocorrelation. A spike at lag 12 hints at residual annual seasonality, but it is minor; a further seasonal difference does not seem warranted, so we proceed with the model as is. Then we decomposed the same window with decompose(), subtracted the seasonal and trend components(figure 15&16), and examined ACF/PACF after one and two seasonal differences(figure 17&18). The second difference offered no significant improvement(figure 19). As parallel exercise, using the natural-log series produced identical ACF behaviour, confirming that the original (variance-stabilised) scale was adequate(figure 20&21). We therefore retained the cubic-season specification without additional transformations for subsequent forecasting work.

A graph showing a line of miles

AI-generated content may be incorrect.A graph of different types of data

AI-generated content may be incorrect.

Figure. 13 Figure. 14

A graph of different types of time

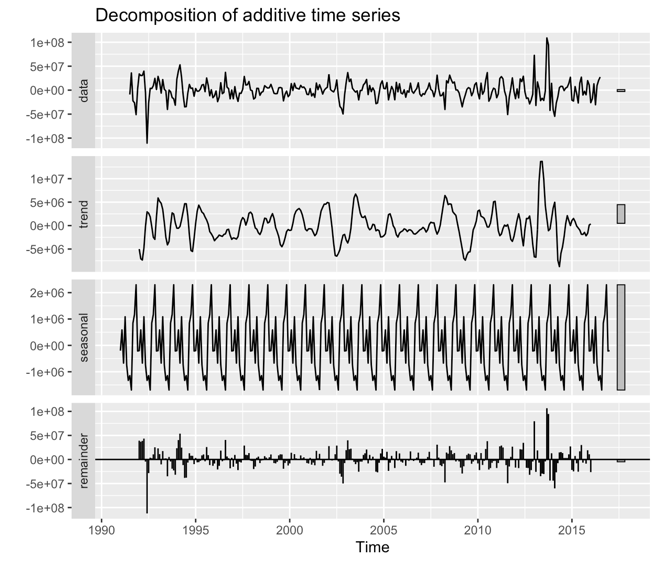
AI-generated content may be incorrect.

Figure. 15 Figure. 16

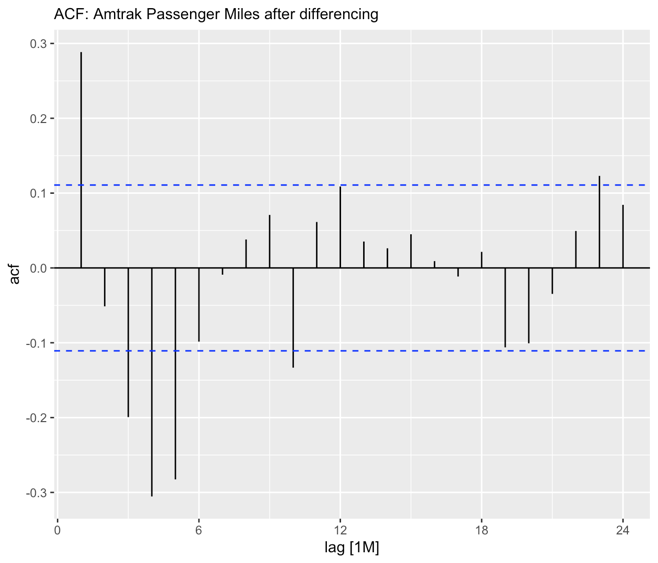
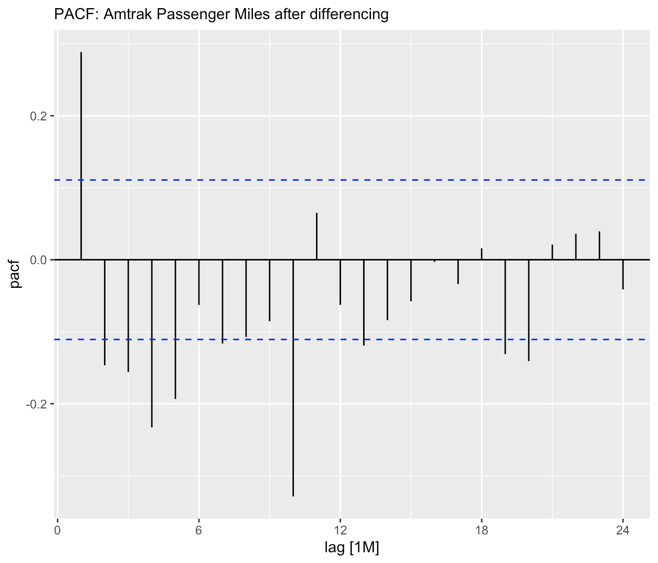


Figure. 17 Figure. 18

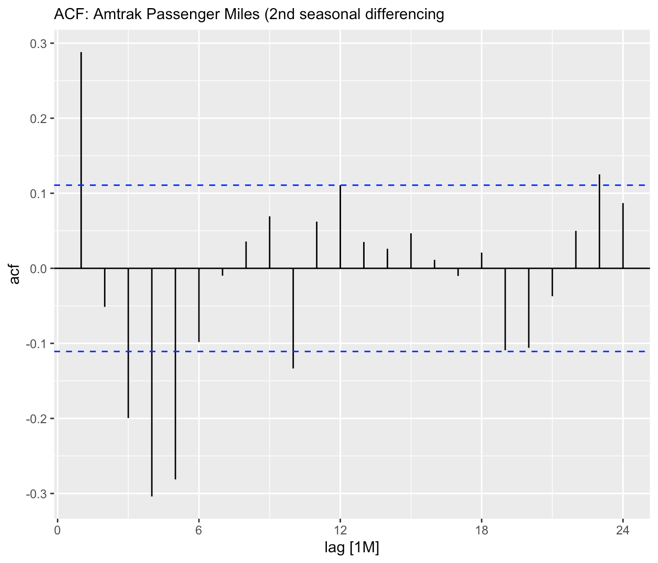


Figure 19

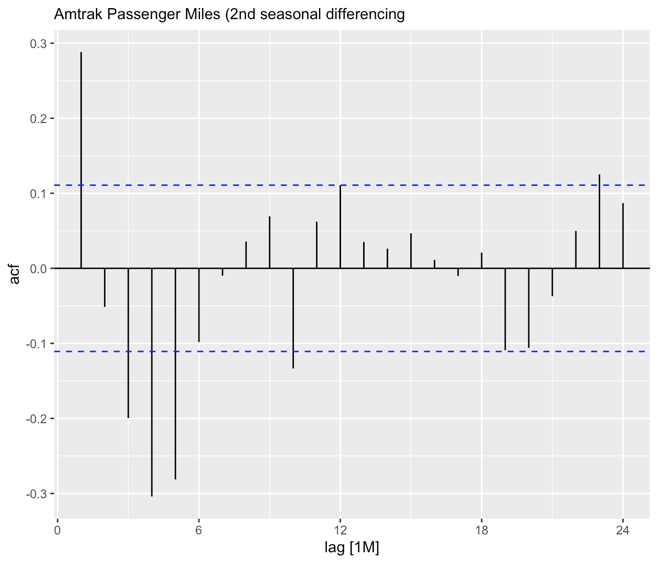
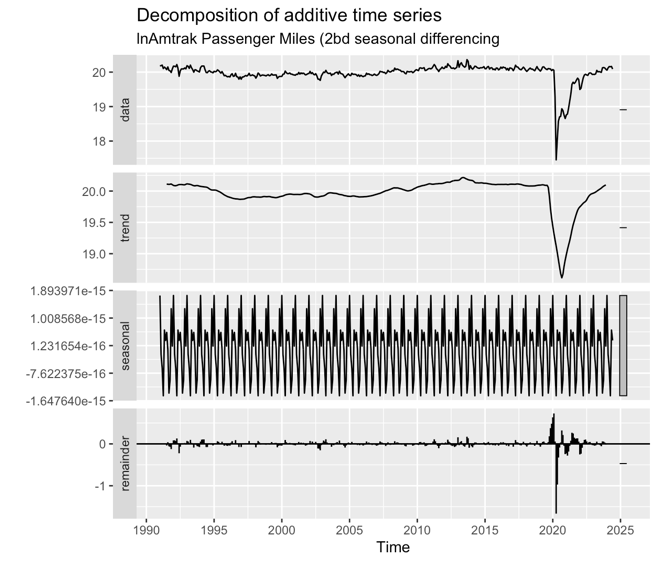


Figure. 20 Figure. 21

Results

Because two rounds of differencing did not remove the annual signal enough, a short moving-average filter was introduced to remove the remaining seasonal noise and to provide a simple benchmark forecast. The analysis first focused on the deseasonalised, detrended series for 1991-2016, a span chosen to avoid pandemic distortions. A window length had to balance noise reduction against information loss: although the dominant cycle is yearly, a 12-month average would erase sub-annual variation, so a three-month window was selected(figure 22). Both centred and trailing versions were calculated, plotted alongside the original line(figure 23), and visually inspected to show how the centred filter tracks the peaks while the trailing filter lags turning points. To demonstrate other approach(Shumeli's book, Practicval Time Series Forecasting with R.), 1991-2004 was partitioned into a training and validation segments(figure 24&25). A 12-month trailing average was fitted to the training data and its final value was projected flat across the hold-out period; the resulting overlay, labelled “Training,” “Validation,” and “Future,” demonstrates how smoothing flattens volatility yet fails to anticipate sudden shifts because moving-average need to be de-seasonalized. This lab3 also introduce other methods like naïve forecast, ETS and Holt-Winter’s model. ETS fit automatically defaulted to an M N A / M N M structure—no trend, only repeating seasonality—so its forecasts behave like a seasonal-naïve baseline(Figure 26&27). Holt-Winters was therefore introduced next because its additive and multiplicative forms allow both level and seasonal amplitude to evolve. At this exercise, Holt-Winter model(Additive and Multiplicative) is applied to 1991-2019 and 2020-2024 window, each models shows almost same MAPE, however shows almost identical MAPE values. Yet the training window makes a clear difference: Holt-Winters models calibrated on the full 1991-2019 history start from a higher level and recover quickly, whereas those trained only on 2020-2024 begin lower and lag the rebound(Figure 28). The contrast quantifies the COVID-era shift and confirms that multiplicative Holt-Winters—while marginally more accurate—remains highly sensitive to the chosen calibration period, highlighting the need to decide whether pre-pandemic behaviour is still relevant for forward planning.

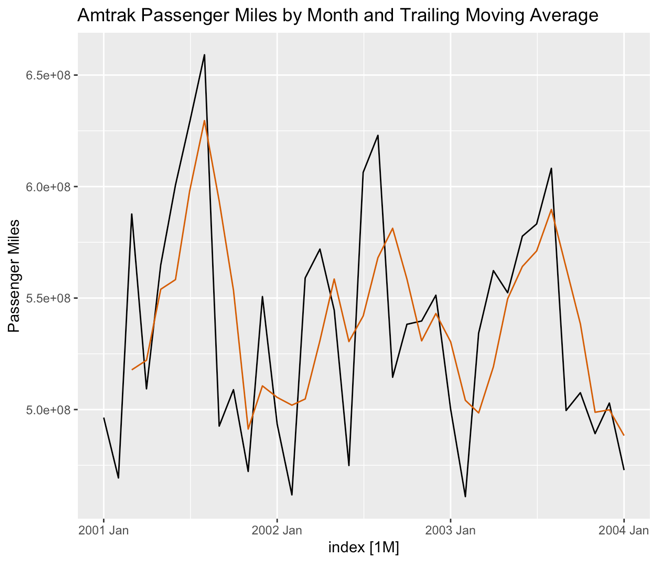
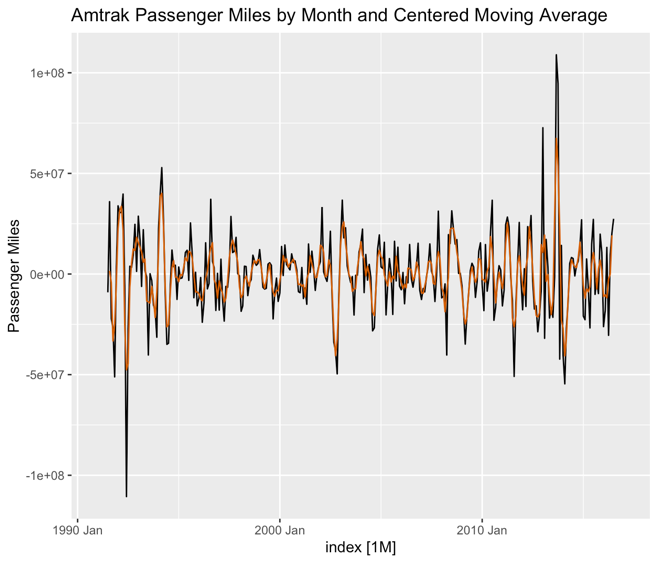


Figure. 22 Figure. 23

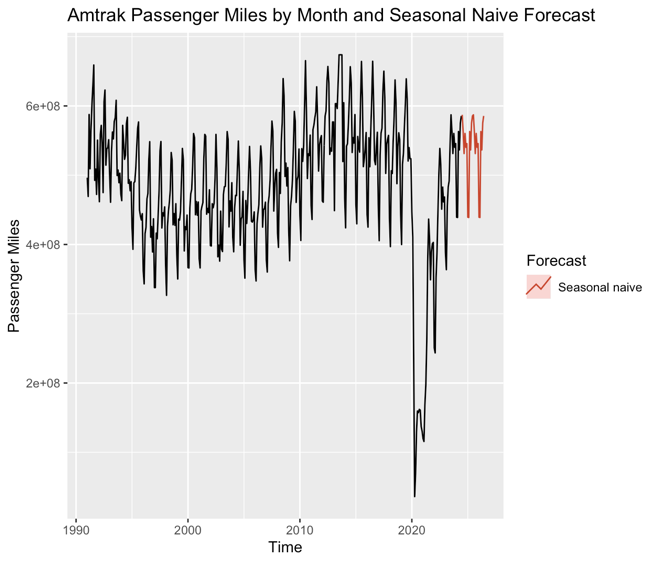
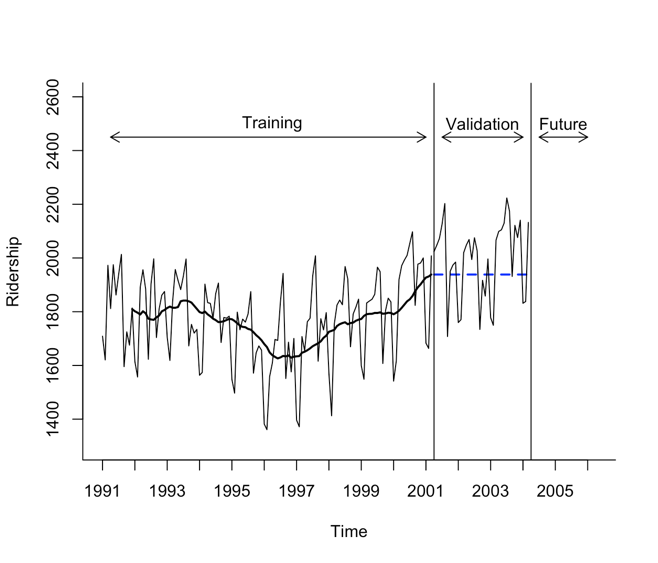


Figure. 24 Figure. 25

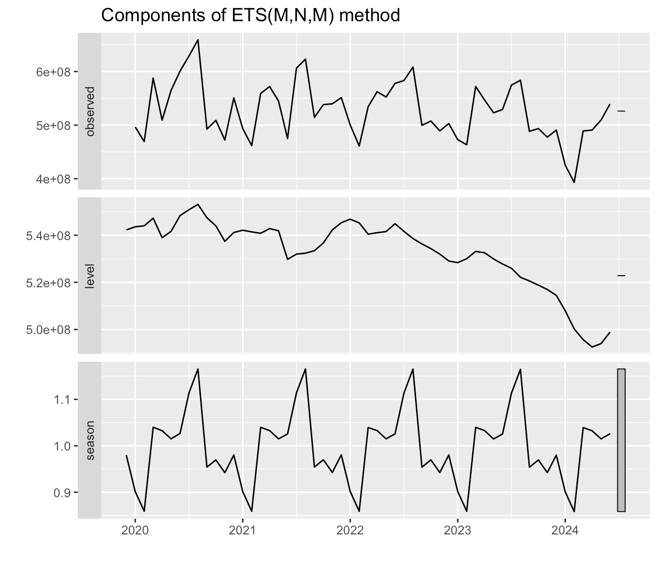
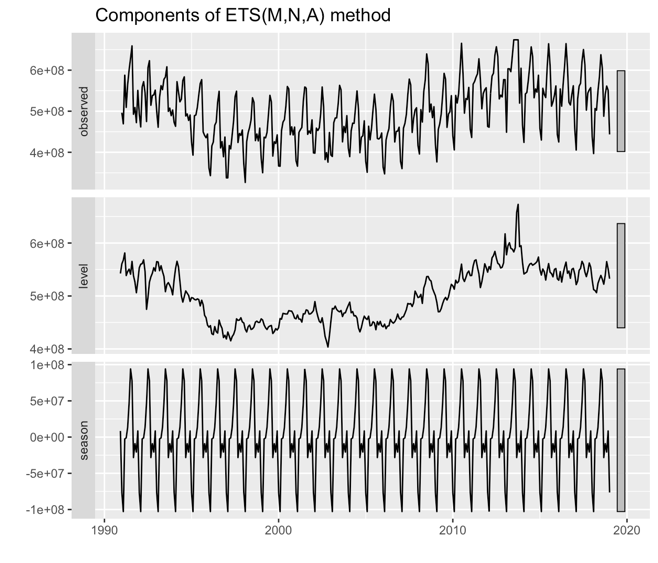


Figure. 26 Figure. 27

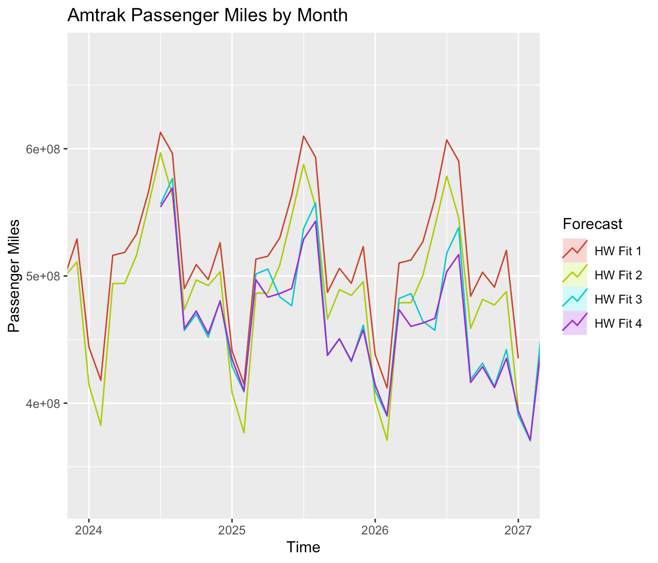


Figure 28

Conclusions

This lab3 demonstrates Three important aspects about time-series forecasting modeling. Sometime log transform fails to resolve issues; alternative power transforms along with visual checks must be used whenever distribution fits poorly. A successful forecasting process starts by removing every remainder of seasonality using twice-differencing and short moving-average filters. The length of data used for calibration influences Holt-Winters predictions because removing pre-2020 information caused the forecasts to remain lower and recover more slowly which shows that optimal historical data selection is vital for producing dependable forecasts.

**References**

**Galit Shmueli. (2016) Practical Time Series Forecasting with R: A Hands-On Guide**

**Script**

#Laboratory 3 Forecasting with regression including smoothing techniques

#such as moving averages and exponential smoothing.

#

#I've left out some plots. This is because the fits are so

#overdriven that we cannot see much in these plots.

#

#There are also a number of plots that you don't necessarily need but are

#good to consider in interpreting the results of the analysis.

#

#Load all the necessary libraries

#

library(xlsx)

library(fpp3)

library(dplyr)

library(tidyverse)

library(ggplot2)

library(tsibble)

library(tsibbledata)

library(fable)

library(feasts)

library(forecast)

library(lubridate)

library(zoo)

#

#Set the working directory and read in the data - if you are not continuing with

#the data already loaded!

#

setwd("/Users/nkohei/Workspace/McDaniel-Repository/535/lab3")

Amtrak <- read.csv("Amtrak1991-2024.csv")

colnames(Amtrak) <- c("Month", "Ridership", "PassengerMiles", "RidersReported")

View(Amtrak)

str(Amtrak)

Amtrak

#

#Take care of the date format and data type and produce the results from Lab#1

#

Amtrak$Month <- mdy(Amtrak$Month)

str(Amtrak)

par(mfrow = c(1,1))

ggplot(Amtrak, aes(x=Month, y=PassengerMiles)) +

geom\_line(color = "blue") +

stat\_smooth(aes(y = PassengerMiles), method = "lm", formula = y ~ x + I(x^2))

#

#I'm not sure what syntax this is having a problem with. If you run these lines

#alone it works. If you run the entire script all at one time this following lines

#do not produce a plot. Don't worry too much about this.

#

ggplot(Amtrak, aes(x=Month, y=PassengerMiles)) +

geom\_line(color = "blue") +

stat\_smooth(aes(y = PassengerMiles), method = "lm", formula = y ~ x + I(x^2) + I(x^3))

#

#It is always good practice to tie your work back to previous work either you or

#someone else has completed. This is part of "reproducible research". So, let's

#also take a couple minutes to consider ridership too.

#

#Or, we can start using more complicated commands that allow us to illustrate

#more complicated situations.

#

Amtrak.Ridership.ts <- ts(Amtrak$Ridership, start = c(1991, 1), end = c(2024, 6), frequency = 12)

str(Amtrak.Ridership.ts)

plot(Amtrak.Ridership.ts, xlab = "Time", ylab = "Ridership", bty = "l")

#

#Or, we can start using more complicated commands that allow us to illustrate

#more complicated situations.

#

Amtrak.Ridership2.lm <- tslm(Amtrak.Ridership.ts ~ trend + I(trend^2))

plot(Amtrak.Ridership.ts, xlab = "Time", ylab = "Ridership", bty = "l")

lines(Amtrak.Ridership2.lm$fitted, lwd = 2)

Amtrak.Ridership3.lm <- tslm(Amtrak.Ridership.ts ~ trend + I(trend^2) + I(trend^3))

plot(Amtrak.Ridership.ts, xlab = "Time", ylab = "Ridership", bty = "l")

lines(Amtrak.Ridership3.lm$fitted, lwd = 2)

#

#If these plots look the same as those you got in the first laboratory you are

#good to go to Laboratory #3!

#

#

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Step 1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#

#Always begin with some exploratory data analysis including verifying assumptions.

#Let's consider a histogram of the dependent variable "Passenger Miles". What

#does it look like? Is it normal in distribution? How would you transform it

#to make it more normal.

#

par(mfrow = c(3,1))

h <- hist(Amtrak$PassengerMiles, breaks = 40, density = 80,

main = "Histogram of Amtrak Passenger Miles")

xfit <- seq(min(Amtrak$PassengerMiles), max(Amtrak$PassengerMiles), length = 100)

yfit <- dnorm(xfit, mean = mean(Amtrak$PassengerMiles), sd = sd(Amtrak$PassengerMiles))

yfit <- yfit \* diff(h$mids[1:2]) \* length(Amtrak$PassengerMiles)

lines(xfit, yfit, lwd = 2)

#

#We can see that the distribution is left-skewed. If you don't remember how to

#handle this from ANA 500/510 you can review that at the website,

#https://www.statisticshowto.com/probability-and-statistics/skewed-distribution/

# and https://statacumen.com/teach/S4R/PDS\_book/skewed-left-distributions.html

#And, you can use the "square," the "cube root," or the "log" transform to try

#to make the distribution more normal. Let's start with the square.

#

y = (Amtrak$PassengerMiles)^2

h <- hist(y, breaks = 40, density = 80,

main = "Histogram of square transformation Amtrak Passenger Miles")

xfit <- seq(min(y), max(y), length = 100)

yfit <- dnorm(xfit, mean = mean(y), sd = sd(y))

yfit <- yfit \* diff(h$mids[1:2]) \* length(y)

lines(xfit, yfit, lwd = 2)

#

#That kind-of, sort-of looks better. It is probably as good as you can get with

#real-world data. It looks a bit like a bimodal distribution now

#but we'll try it and see if we get some better results with regard to the model(s)

#we build. (Note what happens if you change the formula for "y" to take a log

#transorm of Amtrak$PassengerMiles! You will want to report in Laboratory #3

#whether or not taking the log makes a difference, or at least a difference that

#makes things sufficiently better during the analysis to overcome the requirement

#to transform back to real-world values when we look at the results. If you do

#not understand this please ask your insructor!)

#

#

y <- log(Amtrak$PassengerMiles)

# --- histogram with fitted Normal --------------------------------------

h <- hist(y, breaks = 40, density = 80,

main = "Histogram of log(Amtrak Passenger Miles)")

xfit <- seq(min(y), max(y), length = 100)

yfit <- dnorm(xfit, mean = mean(y), sd = sd(y)) \*

diff(h$mids[1:2]) \* length(y)

lines(xfit, yfit, lwd = 2)

#So what happens now if we generate a time plot and add a trend line for a

#linear, a quadratic and a cubic fit?

#

par(mfrow = c(3,1))

ggplot(Amtrak, aes(x=Month, y=(Amtrak$PassengerMiles)^2)) +

geom\_line(color = "blue") +

stat\_smooth(aes(y = Amtrak$PassengerMiles^2), method = "lm",

formula = y ~ x + I(x^1))

ggplot(Amtrak, aes(x=Month, y=(Amtrak$PassengerMiles)^2)) +

geom\_line(color = "blue") +

stat\_smooth(aes(y = Amtrak$PassengerMiles^2), method = "lm",

formula = y ~ x + I(x^2))

ggplot(Amtrak, aes(x=Month, y=(Amtrak$PassengerMiles)^2)) +

geom\_line(color = "blue") +

stat\_smooth(aes(y = Amtrak$PassengerMiles^2), method = "lm",

formula = y ~ x + I(x^2) +I(x^3))

#install.packages("patchwork")

library(patchwork)

# base layer for convenience

base <- ggplot(Amtrak, aes(Month, PassengerMiles^2)) +

geom\_line(colour = "blue")

# three fitted-trend variants

p1 <- base + geom\_smooth(method = "lm", formula = y ~ x) +

labs(title = "Linear fit")

p2 <- base + geom\_smooth(method = "lm", formula = y ~ x + I(x^2)) +

labs(title = "Quadratic fit")

p3 <- base + geom\_smooth(method = "lm", formula = y ~ x + I(x^2) + I(x^3)) +

labs(title = "Cubic fit")

# stack them vertically

(p1 / p2 / p3) # “/” puts plots on top of one another

#

#Consider the linear, a quadratic, and a cubic fit. As I expected neither the

#linear or the quadratic fits the data very well. But here are the plots

#for you to consider.

#

#

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Step 2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#

#

#What about iid residuals? First we have to build the model to get residuals

#so let's take a first look at this.

#

#First let's copy over what we did in Laboratory #1. Here we generate the "fitted

#values" and we zoomed into take a closer look at a few years. Note that here we

#are using the tslm() command from the forecast package.

#

Amtrak.ts <- ts(Amtrak$PassengerMiles, start = c(1991, 1), end = c(2024, 6), frequency = 12)

str(Amtrak.ts)

plot(Amtrak.ts, xlab = "Time", ylab = "Passenger Miles", bty = "l", main="Raw time-series data")

library(forecast)

Amtrak.lm <- tslm(Amtrak.ts ~ trend + I(trend^2))

Amtrak.lm

par(mfrow = c(2,1))

plot(Amtrak.ts, xlab = "Time", ylab = "Passenger Miles", bty = "l", ylim = c(0,1000000000), main='')

lines(Amtrak.lm$fitted, lwd = 2, col = "red")

Amtrak.ts.zoom <- window(Amtrak.ts, start = c(1997, 1), end = c(2000, 12))

plot(Amtrak.ts.zoom, xlab = "Time", ylab = "Passenger Miles", bty = "l")

lines(Amtrak.lm$fitted, lwd = 2, col = "red")

#

#This reproduces what we had in Lab #1. We see the same seasonality in the zoomed

#in plot that we did before.

#

#What we want is a plot of fitted versus actuals as seen in Figure 7.6 in

#Section 7.2 of the textbook. Let's use the TSLM() command

#to do this now. To use this method we'll first generate a tsibble for the

#Amtrak data.

#

#

#Here (again) is a plot of the actual values but now using the syntax associated

#with a tsibble.

#

Amtrak.tsb <- as\_tsibble(Amtrak.ts)

Amtrak.tsb

autoplot(Amtrak.tsb) +

labs(y = "Passenger Miles", x = "Time")

#

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Step 3 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#

#Since we have the data in a tsibble now we can do more with less, so to speak.

#That is, we can embed commands within commands to have 1 line per model, linear

#quadratic, and cubic.

#

#We'll generate the fitted values using the TSLM() command. Notice that now the

#TSLM() command is all upper case letters because we are using the fable package.

#Because the "signal" (that is, we can think of the Amtrak data as a signal that

#is acquired monthly) is relatively complex let's try a quadratic and a cubed

#fit to see what we can get. Once the models are built then we will plot them.

#

par(mfrow = c(2,1))

fit\_tslm <- Amtrak.tsb |>

model(trend\_fit = TSLM(value ~ trend()),

fit\_tslm\_season = TSLM(value ~ trend() + season()),

fit\_tslm2 = TSLM(value ~ trend() + season() + I(trend()^2)),

fit\_tslm3 = TSLM(value ~ trend() + season() + I(trend()^2) + I(trend()^3)))

report(fit\_tslm)

augment(fit\_tslm) |>

autoplot(.resid) +

labs(x = "Time", y = "", title = "Residuals")

fit\_tslm %>%

accuracy() %>%

arrange(MAPE)

#

#It looks like the cubic fit is the best in terms of accuracy based on MAPE and.

#just looking at the variance of the residuals around zero. So let's plot that

#with the actual values. We'll focus mostly on the cubic model from here on but

#there is some back and forth so you get plots and build intuition about the

#different models.

#

#Based on what you see, what do you think an appropriate period for a forecast

#would be?

#

fit\_tslm <- Amtrak.tsb |>

model(fit\_tslm3 = TSLM(value ~ trend() + season() + I(trend()^2) + I(trend()^3)))

# model(fit\_tslm3 = TSLM(value ~ trend() + season()+I(trend()^2) ))

augment(fit\_tslm) |>

ggplot(aes(x = index)) +

geom\_line(aes(y = value, colour = "Data")) +

geom\_line(aes(y = .fitted, colour = "Fitted")) +

labs(y = NULL,

title = "Amtrak Passenger Miles per Month"

) +

scale\_colour\_manual(values=c(Data="black",Fitted="#D55E00")) +

guides(colour = guide\_legend(title = NULL))

#

#This does not catch all the drops, e.g. the fitted values do not catch the slight

#drop around 2008 due to the financial meltdown or the large drop due to the

#shutdowns during Covid. But it does catch longer-term increases and drops in

#passenger miles. Does this change what you believe the appropriate period for

#a forecast should be?

#

#Build a number of models for different periods of time in order to generate an

#overall piecewise model of the data. Here is the first period, from 1991 through

#the end of 1997 (that is why the data run to 01/01/1998).

#

#It looks like the passenger miles per month is decreasing for this time period.

#

Amtrak.ts.91.97 <- ts(Amtrak$PassengerMiles, start = c(1991, 1), end = c(1998, 1), frequency = 12)

Amtrak.tsb.91.97 <- as\_tsibble(Amtrak.ts.91.97)

fit\_tslm\_91\_97 <- Amtrak.tsb.91.97 |>

model(trend\_fit = TSLM(value ~ trend()),

fit\_tslm\_season = TSLM(value ~ trend() + season()),

fit\_tslm2 = TSLM(value ~ trend() + season() + I(trend()^2)),

fit\_tslm3 = TSLM(value ~ trend() + season() + I(trend()^2) + I(trend()^3)))

report(fit\_tslm)

augment(fit\_tslm\_91\_97) |>

# filter(.model == "fit\_tslm3") |>

ggplot(aes(x = index)) +

geom\_line(aes(y = value, colour = "Data")) +

geom\_line(aes(y = .fitted, colour = "Fitted")) +

labs(y = NULL,

title = "Amtrak Passenger Miles per Month"

) +

scale\_colour\_manual(values=c(Data="black",Fitted="#D55E00")) +

guides(colour = guide\_legend(title = NULL))

#

#Let's look at the residuals, i.e. Actuals versus Fitted. It looks like the

#3rd order fit is best.

#

augment(fit\_tslm\_91\_97) |>

autoplot(.resid) +

labs(x = "Time", y = "", title = "Residuals: 91-97")

#

#Consider the numeric values associated with measures of accuracy for this period.

#

fit\_tslm\_91\_97 %>%

accuracy() %>%

arrange(MAPE)

#

#Now plot it.

#

fit\_tslm\_91\_97 <- Amtrak.tsb.91.97 |>

model(fit\_tslm3 = TSLM(value ~ trend() + season() + I(trend()^2) + I(trend()^3)))

report(fit\_tslm)

augment(fit\_tslm\_91\_97) |>

ggplot(aes(x = index)) +

geom\_line(aes(y = value, colour = "Data")) +

geom\_line(aes(y = .fitted, colour = "Fitted")) +

labs(y = NULL,

title = "Amtrak Passenger Miles per Month Cubic fitting 91-97"

) +

scale\_colour\_manual(values=c(Data="black",Fitted="#D55E00")) +

guides(colour = guide\_legend(title = NULL))

#

#The cubic fit doesn't quite capture everything but it isn't too bad either.

#

#Here are the commands for the second period 1991 through and including 2004.

#Here I've commented out commands that consider the linear, quadratic and cubic

#models together and left only commands for plotting the cubic fit.

#

Amtrak.ts.91.04 <- ts(Amtrak$PassengerMiles, start = c(1991, 1), end = c(2005, 1), frequency = 12)

Amtrak.tsb.91.04 <- as\_tsibble(Amtrak.ts.91.04)

fit\_tslm\_91\_04 <- Amtrak.tsb.91.04 |>

model(trend\_fit = TSLM(value ~ trend()),

fit\_tslm\_season = TSLM(value ~ trend() + season()),

fit\_tslm2 = TSLM(value ~ trend() + season() + I(trend()^2)),

fit\_tslm3 = TSLM(value ~ trend() + season() + I(trend()^2) + I(trend()^3)))

report(fit\_tslm\_91\_04)

augment(fit\_tslm\_91\_04) |>

autoplot(.resid) +

labs(x = "Time", y = "", title = "Residuals: 91-04")

fit\_tslm\_91\_04 %>%

accuracy() %>%

arrange(MAPE)

fit\_tslm\_91\_04 <- Amtrak.tsb.91.04 |>

model(fit\_tslm3 = TSLM(value ~ trend() + season() + I(trend()^2) + I(trend()^3)))

report(fit\_tslm\_91\_04)

augment(fit\_tslm\_91\_04) |>

ggplot(aes(x = index)) +

geom\_line(aes(y = value, colour = "Data")) +

geom\_line(aes(y = .fitted, colour = "Fitted")) +

labs(y = NULL,

title = "Amtrak Passenger Miles per Month: 91-04"

) +

scale\_colour\_manual(values=c(Data="black",Fitted="#D55E00")) +

guides(colour = guide\_legend(title = NULL))

#

#It looks like there is more variance in the Residuals for this period 1991 - 2004

#than there was for the first period 1991-1997. The MAPE goes from 3.56 to 4.15

#so numerically the results match our intuition looking at the plot. If you

#want to see all the measures of accuracy for all models just remove the

#"hashtags" that comment out the lines below.

#

#

#Here are the commands for a third period 1991 through and including 2016.

#

Amtrak.ts.91.17 <- ts(Amtrak$PassengerMiles, start = c(1991, 1), end = c(2017, 1), frequency = 12)

Amtrak.tsb.91.17 <- as\_tsibble(Amtrak.ts.91.17)

fit\_tslm\_91\_17 <- Amtrak.tsb.91.17 |>

model(trend\_fit = TSLM(value ~ trend()),

fit\_tslm\_season = TSLM(value ~ trend() + season()),

fit\_tslm2 = TSLM(value ~ trend() + season() + I(trend()^2)),

fit\_tslm3 = TSLM(value ~ trend() + season() + I(trend()^2) + I(trend()^3)))

report(fit\_tslm\_91\_17)

augment(fit\_tslm\_91\_17) |>

autoplot(.resid) +

labs(x = "Time", y = "", title = "Residuals: 91-17")

fit\_tslm\_91\_17 %>%

accuracy() %>%

arrange(MAPE)

fit\_tslm\_91\_17 <- Amtrak.tsb.91.17 |>

model(fit\_tslm3 = TSLM(value ~ trend() + season() + I(trend()^2) + I(trend()^3)))

report(fit\_tslm)

augment(fit\_tslm\_91\_17) |>

ggplot(aes(x = index)) +

geom\_line(aes(y = value, colour = "Data")) +

geom\_line(aes(y = .fitted, colour = "Fitted")) +

labs(y = NULL,

title = "Amtrak Passenger Miles per Month: 91-17"

) +

scale\_colour\_manual(values=c(Data="black",Fitted="#D55E00")) +

guides(colour = guide\_legend(title = NULL))

#

#Here are the commands for the fourth period 1991 through and including 2020.

#

Amtrak.ts.91.20 <- ts(Amtrak$PassengerMiles, start = c(1991, 1), end = c(2021, 1), frequency = 12)

Amtrak.tsb.91.20 <- as\_tsibble(Amtrak.ts.91.20)

fit\_tslm\_91\_20 <- Amtrak.tsb.91.20 |>

model(trend\_fit = TSLM(value ~ trend()),

fit\_tslm\_season = TSLM(value ~ trend() + season()),

fit\_tslm2 = TSLM(value ~ trend() + season() + I(trend()^2)),

fit\_tslm3 = TSLM(value ~ trend() + season() + I(trend()^2) + I(trend()^3)))

report(fit\_tslm\_91\_20)

augment(fit\_tslm\_91\_20) |>

autoplot(.resid) +

labs(x = "Time", y = "", title = "Residuals: 91-20")

fit\_tslm\_91\_20 %>%

accuracy() %>%

arrange(MAPE)

fit\_tslm\_91\_20 <- Amtrak.tsb.91.20 |>

model(fit\_tslm3 = TSLM(value ~ trend() + season() + I(trend()^2) + I(trend()^3)))

report(fit\_tslm\_91\_20)

augment(fit\_tslm\_91\_20) |>

ggplot(aes(x = index)) +

geom\_line(aes(y = value, colour = "Data")) +

geom\_line(aes(y = .fitted, colour = "Fitted")) +

labs(y = NULL,

title = "Amtrak Passenger Miles per Month: 91-20"

) +

scale\_colour\_manual(values=c(Data="black",Fitted="#D55E00")) +

guides(colour = guide\_legend(title = NULL))

fit\_tslm\_91\_20 %>%

accuracy() %>%

arrange(MAPE)

#

#As anticipated, the accuracy is further decreased as shown by the measures of

#error such as MAPE = 14.0.

#

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Step 4 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#

#Generate a time plot of the period of interest 1991 - 2016 and add

#the appropriate trend line.

#

Amtrak.ts.91.17 <- ts(Amtrak$PassengerMiles, start = c(1991, 1), end = c(2017, 1), frequency = 12)

Amtrak.tsb.91.17 <- as\_tsibble(Amtrak.ts.91.17)

par(mfrow = c(1,1))

autoplot(Amtrak.tsb.91.17, value) +

geom\_smooth(method = "lm",

formula = y ~ x + I(x^2) + I(x^3),

colour = "#D55E00") +

labs(title = "Amtrak Passenger Miles (1991-2016)",

y = "Passenger Miles")

# wrong

# ggplot(Amtrak, aes(x=Month, y=PassengerMiles)) +

# geom\_line(color = "blue") +

# stat\_smooth(aes(y = PassengerMiles), method = "lm", formula = y ~ x + I(x^2) + I(x^3))

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Step 5 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#

#Use gg\_tsresiduals to consider additional diagnostics for the model

#from 1991 through and including 2016.

#

fit\_tslm\_91\_17 <- Amtrak.tsb.91.17 |>

model(fit\_tslm3 = TSLM(value ~ trend() + season() + I(trend()^2) + I(trend()^3)))

report(fit\_tslm)

fit\_tslm\_91\_17 |> gg\_tsresiduals()

#

#This presents some interesting results about the residuals. In addition to a

#graphical presentation of the distribution of the residuals it also includes

#a plot of the ACF. The residuals look ok for real-world data.

# The ACF decreases rapidly as desired.

# However, there is just a hint that there may still be some

#annual periodicity (lag = 12) in the ACF. If this were more pronounced I'd

#recommend doing another difference. In this case I think it would be ok to

#leave that alone and proceed.

#

Amtrak.tsb.91.17 |>

ACF() |> autoplot() +

labs(subtitle = "Amtrak Passenger Miles")

#First we'll remove the seasonality in the data. As usual with R, there are a lot

#of different ways to do this. We'll use a couple of them so you get familiar

#with different methods. We only want to use the years 1991 through and including 2016.

#This is sort of a convoluted way to do that, i.e. we'll stop the "window" for the

#period at 01/01/2017 or through the end of 2016.

#

Amtrak.ts.91.16 <- ts(Amtrak$PassengerMiles, start = c(1991, 1), end = c(2017, 1), frequency = 12)

Amtrak.comp.91.16 <- decompose(Amtrak.ts.91.16)

autoplot(Amtrak.comp.91.16)

str(Amtrak.comp.91.16)

#

#Now detrend and/or deseason the data by differencing as required

#

Amtrak.ts.91.16\_desea <- Amtrak.comp.91.16$x - Amtrak.comp.91.16$seasonal

Amtrak.ts.91.16\_detren <- Amtrak.ts.91.16\_desea - Amtrak.comp.91.16$trend

Amtrak.ts.91.16\_de <- decompose(Amtrak.ts.91.16\_detren)

autoplot(Amtrak.ts.91.16\_de)

#

#This is the plot of the decomposed differenced data, i.e. the plot of the data

#after the trend and seasonality have been subtracted. So, let's look at the ACF

#again.

#

par(mfrow = c(2,1))

Amtrak.tsb.91.16\_de <- as\_tsibble(Amtrak.ts.91.16\_detren)

Amtrak.tsb.91.16\_de |>

ACF() |> autoplot() +

labs(subtitle = "ACF: Amtrak Passenger Miles after differencing")

Amtrak.tsb.91.16\_de |>

PACF() |> autoplot() +

labs(subtitle = "PACF: Amtrak Passenger Miles after differencing")

#

#It looks like there is still some seasonality in the data so we'll take a

#second differencing to remove that.

#

Amtrak.ts.91.16\_desea2 <- Amtrak.ts.91.16\_de$x - Amtrak.ts.91.16\_de$seasonal

autoplot(Amtrak.ts.91.16\_desea2)

Amtrak.tsb.91.16\_desea2 <- as\_tsibble(Amtrak.ts.91.16\_desea2)

Amtrak.tsb.91.16\_desea2 |>

ACF() |> autoplot() +

labs(subtitle = "ACF: Amtrak Passenger Miles (2nd seasonal differencing")

#

#Taking a second difference to remove residual seasonality does not appear

#to have made any difference. Let's go back to the start and look at what

#happens if we had taken the natural log of the data to begin with. We won't

#do all the steps again. We only want to see the effect of taking a log of

#the original data. We'll still use Passenger Miles as the variable of interest.

#

lnAmtrakPassengerMiles <- log(Amtrak$PassengerMiles)

lnAmtrakMonth <- Amtrak$Month

lnAmtrak <- data.frame(lnAmtrakMonth, lnAmtrakPassengerMiles)

lnAmtrak.ts <- ts(lnAmtrak$lnAmtrakPassengerMiles, start = c(1991, 1),

end = c(2024, 6), frequency = 12)

plot(lnAmtrak.ts, xlab = "Time", ylab = "Passenger Miles", bty = "l")

#

#Generate a time series for the natural log of Passenger Miles and consider its

#decomposition.

#

lnAmtrak.comp <- decompose(lnAmtrak.ts)

autoplot(lnAmtrak.comp)

#

#Do a first differencing to remove the seasonality in the data, decompose that

#data and plot it.

#

lnAmtrak\_desea <- lnAmtrak.comp$x - lnAmtrak.comp$seasonal

lnAmtrak\_desea.comp <- decompose(lnAmtrak\_desea)

autoplot(lnAmtrak\_desea.comp) +

labs(subtitle = "lnAmtrak Passenger Miles (1st seasonal differencing")

#

#Do a second differencing on the seasonal component to remove any remaining

#seasonality and plot.

#

lnAmtrak\_desea2 <- lnAmtrak\_desea.comp$x - lnAmtrak\_desea.comp$seasonal

lnAmtrak\_desea2.comp <- decompose(lnAmtrak\_desea2)

autoplot(lnAmtrak\_desea2.comp)+

labs(subtitle = "lnAmtrak Passenger Miles (2bd seasonal differencing")

#

#Compute the ACF of the twice differences Amtrak Passenger Miles Data and plot.

#

lnAmtrak\_desea2.comp <- as\_tsibble(Amtrak.ts.91.16\_desea2)

lnAmtrak\_desea2.comp |>

ACF() |> autoplot() +

labs(subtitle = "Amtrak Passenger Miles (2nd seasonal differencing")

#

#Ok, after initially taking the natural log of the variable Passenger Mile the ACF

#still looks the same as it did without taking the natural log. There are still

#some things going on but all are +/-0.3 or less than 0.3. That is, we have

#demonstrated that initially taking the natural log of this variable,

#Amtrak$PassengerMiles does not make any difference in the results of the data

#analysis. Let's just go back to where we were and not worry about taking the

#natural log of the variable of interest in this case.

#

#

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Step 6 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#

#

#Generate a 3-month centered moving average model for the Amtrak data (all years).

#Do we need to deseasonalize the data? Not necessarily but a moving average will

#suppress seasonality in data. From one point of view this is a good thing becase

#we've seen that differencing twice was not sufficient to remove all the

#seasonality from the Amtrak data.

#

#For the moving averages start with the deseasonalized data. Don't forget to go back

#to the regular data after that!!! The ARIMA and related models take trend and

#seasonality into consideration. You only need to detrend and deseasonalize data

#for some types of smoothing such as moving averages!

#

#

#How do you choose the amount of time to generate the moving average over? It

#isn't that different from beginning to consider the time period for validation.

#Shumeli suggests that you use the period associated with the seasonality in the

#data. In this case a strong periodicity is annually or w, the size of the window

#over which we'll take the average, would equal 12, i.e. we'll take averages across

#12-month periods. That is probably not a good choice to start with because that

#would smooth out all the other features in the data too. Let's start with a

#3-month moving average.

#

#Note that I've gone back to a period from 1991 to 2016 removing any effects from

#the Covid lock-downs.

#

Amtrak\_3MA <- Amtrak.tsb.91.16\_de

#

#Now work on the moving average smoothing beginning with the centered moving

#average

#

Amtrak\_3MA.19.16 <- Amtrak\_3MA |>

mutate(

'3-MA' = slider::slide\_dbl(value, mean,

.before = 1, .after = 1, .complete = TRUE)

)

Amtrak\_3MA.19.16 |>

autoplot(Amtrak\_3MA$value) +

geom\_line(aes(y = `3-MA`), colour = "#D55E00") +

labs(y = "Passenger Miles",

title = "Amtrak Passenger Miles by Month and Centered Moving Average")

#

#Let's zoom into a few years to see exactly how the 3-month moving average

#is handling the fit.

#

str(Amtrak.ts.91.16\_detren)

Amtrak\_MAZoom\_2001\_03 <- window(Amtrak.ts.91.16\_detren, start = c(2001,1),

end = c(2004,1), frequency = 12)

str(Amtrak\_MAZoom\_2001\_03)

Amtrak.tsb\_2001\_03 <- as\_tsibble(Amtrak\_MAZoom\_2001\_03)

Amtrak\_3MA\_2001\_03 <- Amtrak.tsb\_2001\_03 |>

mutate(

'3-MA' = slider::slide\_dbl(value, mean,

.before = 1, .after = 1, .complete = TRUE)

)

Amtrak\_3MA\_2001\_03 |>

autoplot(Amtrak.tsb\_2001\_03$value) +

geom\_line(aes(y = Amtrak\_3MA\_2001\_03$`3-MA`), colour = "#D55E00") +

labs(y = "Passenger Miles",

title = "Amtrak Passenger Miles by Month and Centered Moving Average")

#

#Ok, you can see how the moving average "smoothes" out the peaks and troughs in

#the data. Using the centered moving average, the fit follows the values in the data

#as they increase and decrease.

#

#

#Let's look at those same zoomed in years to see exactly how the 3-month trailing

#moving average is handling the fit.

#

Amtrak.ts\_2001\_03 <- ts(Amtrak$PassengerMiles, start = c(2001, 1), end = c(2004, 1), freq = 12)

Amtrak.tsb\_2001\_03 <- as\_tsibble(Amtrak.ts\_2001\_03)

Amtrak\_3MATR\_2001\_03 <- Amtrak.tsb\_2001\_03 |>

mutate(

'3-MA' = slider::slide\_dbl(value, mean,

.before = 2, .after = 0, .complete = TRUE)

)

Amtrak\_3MATR\_2001\_03 |>

autoplot(Amtrak.tsb\_2001\_03$value) +

geom\_line(aes(y = Amtrak\_3MATR\_2001\_03$`3-MA`), colour = "#D55E00") +

labs(y = "Passenger Miles",

title = "Amtrak Passenger Miles by Month and Trailing Moving Average")

#

#Now you can see that the up/down movement of values in the fit trails the actual values.

#

#

#Let's look at a specific example that incorporates moving average smoothing and a

#prescribed (3 year) validation period. So I don't mix up anything I'll go back to

#the original Amtrak Ridership values but divide by 1000. to get them scaled a little.

#I'll zoom into the years between 1991 and 2004 (one of the periods we considered before).

#

setwd("/Users/nkohei/Workspace/McDaniel-Repository/535/lab3")

Amtrak <- read.csv("Amtrak1991-2024.csv")

colnames(Amtrak) <- c("Month", "Ridership", "PassengerMiles", "RidersReported")

View(Amtrak)

str(Amtrak)

Amtrak

#

#Take care of the date format and data type and produce the results from Lab#1

#

Amtrak$Month <- mdy(Amtrak$Month)

str(Amtrak)

#

#Start the code to plot the time series and moving average.

#

Amtrak$Ridership <- Amtrak$Ridership/1000.

head(Amtrak)

ridership.ts <- ts(Amtrak$Ridership, start = c(1991, 1), end = c(2004, 3), freq = 12)

#

#This is an example from Shumeli's book, Practicval Time Series Forecasting with R.

#

nValid <- 36

nTrain <- length(ridership.ts) - nValid

train.ts <- window(ridership.ts, start = c(1991, 1), end = c(1991, nTrain))

valid.ts <- window(ridership.ts, start = c(1991, nTrain + 1), end = c(1991, nTrain + nValid))

ma.trailing <- rollmean(train.ts, k = 12, align = "right")

last.ma <- tail(ma.trailing, 1)

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

# Figure 5-3

plot(train.ts, ylim = c(1300, 2600), ylab = "Ridership", xlab = "Time", bty = "l", xaxt = "n", xlim = c(1991,2006.25), main = "")

axis(1, at = seq(1991, 2006, 1), labels = format(seq(1991, 2006, 1)))

lines(ma.trailing, lwd = 2)

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

lines(valid.ts)

lines(c(2004.25 - 3, 2004.25 - 3), c(0, 3500))

lines(c(2004.25, 2004.25), c(0, 3500))

text(1996.25, 2500, "Training")

text(2002.75, 2500, "Validation")

text(2005.25, 2500, "Future")

arrows(2004 - 3, 2450, 1991.25, 2450, code = 3, length = 0.1, lwd = 1,angle = 30)

arrows(2004.5 - 3, 2450, 2004, 2450, code = 3, length = 0.1, lwd = 1,angle = 30)

arrows(2004.5, 2450, 2006, 2450, code = 3, length = 0.1, lwd = 1, angle = 30)

#

#This is an interesting plot. You can see that using a window of 12 months (or

#annually) really smooths out a lot of the fluctuations in the Amtrak data. This

#might or might not be a good thing. If you want to keep the information about

#periodicities smaller than annual then it is bad. You will have "smoothed" out

#or lost all that information. If you only are interested in a general idea of

#whether or not the ridership is increasing or decreasing then it is probably ok.

#

#

#Other things to keep in mind is that these simple methods, like moving averages,

#don't work with data that includes a trend or seasonality. You need to remove

#that from the data prior to applying a moving average. If you use a trailing

#moving average with w = 1 you will get the same result as a naive forecast.

#

#

#Now consider the seasonal naive method of "simple" forecasting

#

AmtrakSNaive <- window(Amtrak.ts, start=c(1991, 1), end = c(2024, 6))

autoplot(AmtrakSNaive) +

autolayer(snaive(AmtrakSNaive), h = 36,

series = "Seasonal naive", PI=FALSE) +

ggtitle("Amtrak Passenger Miles by Month and Seasonal Naive Forecast") +

xlab("Time") + ylab("Passenger Miles") +

guides(colour = guide\_legend(title = "Forecast"))

#

#A naive forecast just uses the last period values as the forecast for future

#periods. It will never change. Obviously this is "usually" a bad way

#of forecasting. But, where values almost never change and keeping things

#as simple as possible then the Naive forecast can be pretty good.

#

#

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Step 7 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#

#

#Let's look more at Exponential Smoothing. First, we'll build a model using the

# ets() command. Remember that usually these acronyms have the same or similar

#meanings. ets is error-trend-seasonality. stl is seasonality-trend-level. However,

#even though the acronyms are similar the commands/functions do very different

#things. For example, the ets() command can be used to build a model but not to

#make a forecast (predictions).

#

#Remember that you only need to go from 1991 through and including 2016 for this

#step and then add your prediction.

#

Amtrak1991\_2019 <- ts(Amtrak$PassengerMiles, start = c(1991, 1), end = c(2019, 1), frequency = 12)

fit\_ets <- ets(Amtrak1991\_2019)

fit\_ets

autoplot(fit\_ets)

fit\_ets$method

fit\_ets\_fcast <- forecast(fit\_ets, h=60) %>%

autoplot() +

ylab("Amtrak Passenger Miles by Month (91-19)") +

labs(title = "Amtrak Passenger Miles per Month (91-19)")

autoplot(ets(Amtrak1991\_2019))

# 60 か月先までの予測付きプロット

ets(Amtrak1991\_2019) |>

forecast(h = 60) |>

autoplot() +

labs(title = "Amtrak Passenger Miles per Month (91-19)")

#

#This looks a bit different. As indicated by the automatic labeling of the plot,

#the components of the ets model (by the plot title) are error = multiplicative,

#trend = null, and seasonality = additive.

#

#

#Just to see what is there, build a model (using the same exponential smoothing)

#for the years 2021 through and including the months available for 2024.

#

Amtrak2024 <- ts(Amtrak$PassengerMiles, start = c(2020, 1), end = c(2024, 6), frequency = 12)

fit\_ets2 <- ets(Amtrak2024)

ret <- forecast(fit\_ets2)

fit\_ets2

autoplot(fit\_ets2)

fit\_ets2\_fcast <- forecast(fit\_ets2, h=60) %>%

autoplot() +

ylab("Amtrak Passenger Miles by Month (20-24)")

print(fit\_ets2\_fcast)

#Now the automatic labeling indicates that the error is multiplicative, the trend

#is null and the seasonality is multiplicative.

#

#It looks like there is a slightly decreasing trend but the change in values are very

#small. There is definitely still the seasonality as we saw before the Covid

#pandemic.

#

#

#This is all nice but we want to save the values of a forecast. So, we'll modify

#what was outlined in fpp2 to store the output from the forecast above and then plot

#as the following: (Note that the plots look like naive forecasts as they should.)

#

fit\_ets\_fcast <- forecast(fit\_ets, h = 72)

autoplot(fit\_ets$fitted) +

autolayer(fit\_ets\_fcast$mean, series = "Forecast") +

ylab("Passenger Miles")

fit\_ets2\_fcast <- forecast(fit\_ets2, h = 72)

autoplot(fit\_ets2$fitted) +

autolayer(fit\_ets2\_fcast$mean, series = "Forecast") +

ylab("Passenger Miles")

#

#Well the naive forecast seems a little off. There is no real variation in the

#level even though the years prior look like there should be. The biggest part of the

#problem is that there are multiple seasonalities and we've been trying to apply a

#simple moving average. That is, we have not yet applied

#the Holt-Winter's method to build a model which is required when seasonality is

#present!

#

#

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Step 8 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#

#

#Let's look at the Holt-Winter's models. We'll forecast or predict the years

#following 2019.

#

Amtrak1991\_2019 <- ts(Amtrak$PassengerMiles, start = c(1991, 1), end = c(2019, 1), frequency = 12)

fit1\_hw <- hw(Amtrak1991\_2019, seasonal = "additive", h = 96)

fit2\_hw <- hw(Amtrak1991\_2019, seasonal = "multiplicative", h = 96)

fit1\_hw

fit2\_hw

autoplot(Amtrak1991\_2019) +

autolayer(fit1\_hw, series = "HW additive forecasts", PI = FALSE) +

autolayer(fit2\_hw, series = "HW multiplicative forecasts", PI = FALSE) +

xlab("Time") +

ylab("Passenger Miles") +

ggtitle("Amtrak Passenger Miles by Month (91-19)") +

guides(colour = guide\_legend(title="Forecast"))

accuracy(fit1\_hw)

accuracy(fit2\_hw)

#

#Now you can see the slight decrease we expected from the slightly decreasing level

#in the decomposition from the ets() model. You can also see that the multiplicative

#forecast is trending lower than the additive forecast as time goes on.

#

#Now build a model (using the same Holt-Winter's exponential smoothing) for the

#years 2021 through and including the months available for 2024.

#

Amtrak2024 <- ts(Amtrak$PassengerMiles, start = c(2020, 1), end = c(2024, 6), frequency = 12)

fit3\_hw <- hw(Amtrak2024, seasonal = "additive", h=36)

fit4\_hw <- hw(Amtrak2024, seasonal = "multiplicative", h=36)

fit3\_hw

fit4\_hw

autoplot(Amtrak2024) +

autolayer(fit3\_hw, series = "HW additive forecasts", PI = FALSE) +

autolayer(fit4\_hw, series = "HW multiplicative forecasts", PI = FALSE) +

xlab("Time") +

ylab("Passenger Miles") +

ggtitle("Amtrak Passenger Miles by Month (20-24)") +

guides(colour = guide\_legend(title="Forecast"))

accuracy(fit3\_hw)

accuracy(fit4\_hw)

#

#That actually looks pretty good! Now let's plot the first predictions with these

#actuals and predictions to see what it all looks like. We'll plot it all and then

#restrict the x-axis to 2020 through 2024.

#

autoplot(Amtrak1991\_2019) +

autolayer(fit1\_hw, series = "HW Fit 1", PI = FALSE) +

autolayer(fit2\_hw, series = "HW Fit 2", PI = FALSE) +

autolayer(fit3\_hw, series = "HW Fit 3", PI = FALSE) +

autolayer(fit4\_hw, series = "HW Fit 4", PI = FALSE) +

xlab("Time") +

ylab("Passenger Miles") +

ggtitle("Amtrak Passenger Miles by Month") +

coord\_cartesian(xlim = c(2024, 2027)) +

guides(colour = guide\_legend(title="Forecast"))

#

#This is interesting! It looks like Fit 1 and Fit 2 which were the HW fit for all the

#data from 1991 through 2024 are somewhat higher than Fit 3 and Fit 4 which only

#used the data from 2020 to 2024 to make the forecast. It also looks like Fit 3 and

#Fit 4 lag or are somewhat behind the forecasts using all the data. I would make

#the conclusion that the years prior to 2020 do effect the forecast!

#

#Now we could compute the difference between what the forecast was before the onset

#of Covid with what the actual was to determine if there was a difference. We will

#leave that for a future exercise and move on.

#

#Below are the Holt-Winter's forecasts for the years between 1991 and 2016,

#or before the onset of the Covid pandemic. This includes the HW\_linear trend and

#the HW\_seasonal additive and multiplicative.

#

Amtrak1991\_2016 <- ts(Amtrak$PassengerMiles, start = c(1991, 1), end = c(2017, 1), frequency = 12)

fit\_hw <- hw(Amtrak1991\_2016)

fit\_hwA <- hw(Amtrak1991\_2016, seasonal = "additive")

fit\_hwM <- hw(Amtrak1991\_2016, seasonal = "multiplicative")

summary(fit\_hw)

summary(fit\_hwA)

summary(fit\_hwM)

#

#If you haven't caught it yet, not designating a "method" defaults to the HW-

#additive method.

#

autoplot(Amtrak1991\_2016) +

autolayer(fit\_hwA, series = "HW additive forecasts", PI=FALSE) +

autolayer(fit\_hwM, series = "HW multiplicative forecasts", PI=FALSE) +

xlab("Time") +

ylab("Passenger Miles") +

ggtitle("Amtrak Passenger Miles by Month") +

guides(colour = guide\_legend(title = "Forecast"))

autoplot(Amtrak1991\_2016) +

autolayer(fit\_hwA, series = "HW additive forecasts", PI=FALSE) +

# autolayer(fit\_hwM, series = "HW multiplicative forecasts", PI=FALSE) +

xlab("Time") +

ylab("Passenger Miles") +

ggtitle("Amtrak Passenger Miles by Month") +

guides(colour = guide\_legend(title = "Forecast"))

autoplot(Amtrak1991\_2016) +

# autolayer(fit\_hwA, series = "HW additive forecasts", PI=FALSE) +

autolayer(fit\_hwM, series = "HW multiplicative forecasts", PI=FALSE) +

xlab("Time") +

ylab("Passenger Miles") +

ggtitle("Amtrak Passenger Miles by Month") +

guides(colour = guide\_legend(title = "Forecast"))

accuracy(fit\_hwA)

accuracy(fit\_hwM)

#

#To be thorough I am including use of the ets() method from the Forecast

#package. In order to get a prediction we need to add a couple more lines

#of code. See the code below. This is an example from Shumeli's book,

#Practicval Time Series Forecasting with R.

#

diff.twice.ts <- diff(diff(ridership.ts, lag = 12), lag = 1)

nValid <- 36

nTrain <- length(diff.twice.ts) - nValid

train.ts <- window(diff.twice.ts, start = c(1992, 2), end = c(1992, nTrain + 1))

valid.ts <- window(diff.twice.ts, start = c(1992, nTrain + 2), end = c(1992, nTrain + 1 + nValid))

ses <- ets(train.ts, model = "ANN", alpha = 0.2)

ses.pred <- forecast(ses, h = nValid, level = 0)

plot(ses.pred, ylim = c(-250, 300), ylab = "Ridership (Twice-Differenced)",

xlab = "Time", bty = "l", xaxt = "n", xlim = c(1991,2006.25), main = "",

flty = 2)

axis(1, at = seq(1991, 2006, 1), labels = format(seq(1991, 2006, 1)))

lines(ses.pred$fitted, lwd = 2, col = "blue")

lines(valid.ts)

lines(c(2004.25 - 3, 2004.25 - 3), c(-250, 350))

lines(c(2004.25, 2004.25), c(-250, 350))

text(1996.25, 275, "Training")

text(2002.75, 275, "Validation")

text(2005.25, 275, "Future")

arrows(2004 - 3, 245, 1991.5, 245, code = 3, length = 0.1, lwd = 1,angle = 30)

arrows(2004.5 - 3, 245, 2004, 245, code = 3, length = 0.1, lwd = 1,angle = 30)

arrows(2004.5, 245, 2006, 245, code = 3, length = 0.1, lwd = 1, angle = 30)

ses.opt <- ets(train.ts, model = "ANN")

ses.opt.pred <- forecast(ses.opt, h = nValid, level = 0)

ses.opt

accuracy(ses.pred, valid.ts)

accuracy(ses.opt.pred, valid.ts)