Week 6 Topics

1. Chapter 6 – Regression-Based Models

Regression models for capturing trend

Linear regression is a popular forecasting method, using suitable predictors to capture trend and/or seasonality as well as other patterns. This week notes shows how a linear regression model can be set up to capture different types of trend in a time series. The model, which is estimated from the training period, can then produce forecasts on future data by inserting the relevant predictor information into the estimated regression equation.

We describe different types of regression trend models for capturing common trend shapes (linear, exponential, and polynomial). Linear regression can be used to fit a global trend that applies to the entire series and will apply in the forecasting period. A linear trend means that the values of the series increase or decrease linearly in time, whereas an exponential trend captures an exponential increase or decrease. We can also use more flexible functions, such a quadratic functions or higher order polynomials, to capture more complex trend shapes.

## Linear Trend

To create a linear regression model that captures a time series with a global linear trend, the output variable (y) is set as the time series measurement or some function of it, and the predictor (x) is set as a time index t. Let us consider Amtrak ridership example: fitting a linear trend to the Amtrak ridership data. The first step is to create a new column that is a time index t = 1,2,3, … This will serve as our predictor. Here is a snapshot of the first few rows for the two corresponding columns (y and t). As you remember from previous chapters of our textbook, *a seasonal pattern in a time series means that observations that fall in some seasons have consistently higher or lower values than those that fall in other seasons*. Examples are day-of-week patterns, monthly patterns, and quarterly patterns. The Amtrak ridership monthly time series, as can be seen in the figure 1 time plot, exhibits strong monthly seasonality (with highest traffic during summer months).

### Table 1: Amtrak Ridership

|  |  |  |
| --- | --- | --- |
| Month | Ridership | t (Time Index) |
| Jan-91 | 1709 | 1 |
| Feb-91 | 1621 | 2 |
| Mar-91 | 1973 | 3 |
| Apr-91 | 1812 | 4 |
| May-91 | 1975 | 5 |
| Jun-91 | 1862 | 6 |
| Jul-91 | 1940 | 7 |
| Aug-91 | 2013 | 8 |
| Sep-91 | 1596 | 9 |
| Oct-91 | 1725 | 10 |
| Nov-91 | 1676 | 11 |
| Dec-91 | 1814 | 12 |
| Jan-92 | 1615 | 13 |
| Feb-92 | 1557 | 14 |

Before fitting the linear regression, we partition the ridership time series into training and validation periods. Here we keep the last 3 years of data as the validation period. Next, to fit a linear relationship between Ridership and Time, we set the output variable (y) as the Amtrak ridership and the predictor as the time index *t* in the regression model:

Where  is the Ridership at time point *t* and ε is the standard noise term in a linear regression. Thus, we are modeling three of the four time-series components: level (β0), trend (β1), and noise (ε). Seasonality is not modeled.

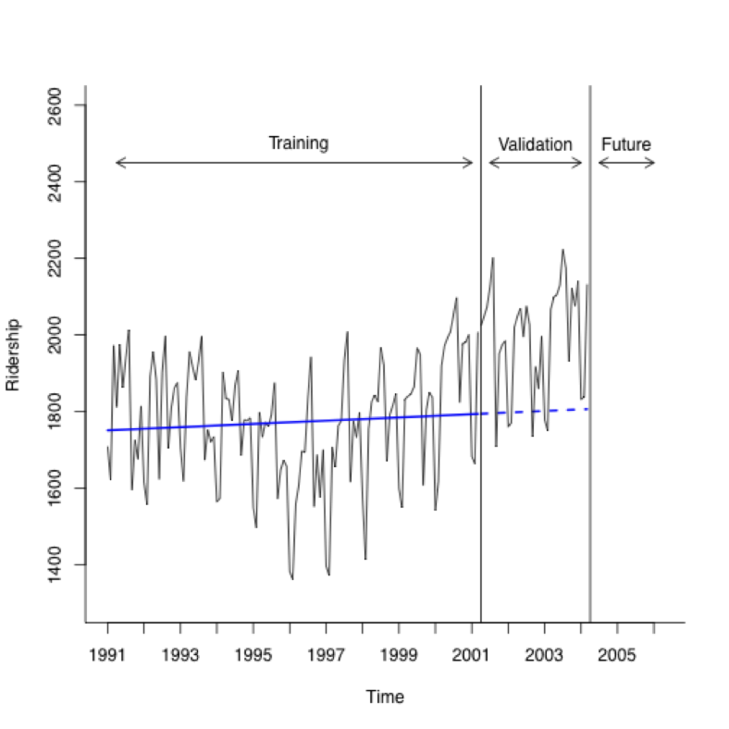
The next step is to use the linear trend model (also a linear regression model) to make forecasts in the validation period. The figure 1 depicts the actual time series, and the linear model predictions in the training and validation periods.

Figure 1: Amtrak ridership series with linear trend model predictions for training (smooth blue line) and validation (dashed blue line)

The following R code fits a linear regression model with a linear trend, and plots the predicted values for the training and validation sets overlaid on the original series:

### The following R Codes are used in figure 6.2 on page 120.

### The following R code fits a linear regression model with a linear trend, and plots the predicted values for the training and validation sets overlaid on the original series

library (forecast)

library(zoo)

Amtrak.data <- read.csv("Amtrak data.csv")

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

nValid <- 36 #number of observations in the validation partition

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))

train.lm <- tslm(train.ts ~ trend) #build the model (seasonality is not includes)

# Figure 6.2

plot(train.ts, xlab = "Time", ylab = "Ridership", ylim = c(1300, 2300), bty = "l")

lines(train.lm$fitted, lwd = 2, col = “blue”)

# Table 6.1

summary(train.lm)

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 1750.3595 29.0729 60.206 <2e-16 \*\*\*

trend 0.3514 0.4069 0.864 0.39

**Note 1**: beware of examining only the coefficients and their statistical significance for making decisions about the trend, can be misleading. A significant coefficient for trend does not mean that a linear fit is adequate. An insignificant coefficient does not mean that there is no trend in the data. In our example, the slope coefficient (0.3514) is insignificant (p- value=0.39), yet there may be a trend in the data (often once we control for seasonality). To determine suitability of any trend shape, look at the time plot of the (de-seasonalized) time series with the trend overlaid; examine the residual time plot; and look at performance measures on the validation period.

## Exponential Trend

Several alternative trend shapes are useful and easy to fit via a linear regression model. Recall Excel’s Trend line and other plots that help to assess the type of trend in the data. One such shape is an exponential trend. An exponential trend implies a multiplicative increase/decrease of the series over time  . Exponential trends are popular in sales data, where they reflect percentage growth.

which is equal to

Then we call and we know

Therefore, to fit an exponential trend, simply replace the output variable y with log(y) and fit the linear regression:

In the Amtrak example, for instance, we would fit a linear regression of log(*Ridership*) as the dependent variable and the index variable t as the predictor.

The R code for fitting this model is:

train.lm.expo.trend <- tslm(train.ts ~ trend, lambda = 0)

[In R, setting lambda = 0 in the function tslm() indicates an exponential trend. For a linear trend, we set lambda=1, which is the default.]

**Note 2**: We use “log” to denote the natural logarithm (base e = 2.71828…). Excel uses the function =LN. In R, use the function log (). I use mathematical version of it which is

**Note 3**: As in the general case of linear regression, when comparing the predictive accuracy of models that have a different output variable, such as comparing a linear model trend (with y) and an exponential model trend (with log(y)), it is essential to compare forecast or forecast errors on the same scale. An exponential trend model will produce forecasts in logarithmic scale, and the forecast errors reported by many software packages (e.g. XLMiner) will therefore be of the form

## Polynomial Trend

Another nonlinear trend shape that is easy to fit via linear regression is a polynomial trend, and in particular, a quadratic relationship of the form:

This is done by creating an additional predictor *t2* (the square of t) and fitting a multiple linear regression with the two predictors *t* and t2. For the Amtrak ridership data, we have already seen a U-shaped trend in the data. We therefore fit a quadratic model to the training period. In R we can do this using the code:  
  
train.lm.poly.trend <- tslm(train.ts ~ trend + I(trend^2))

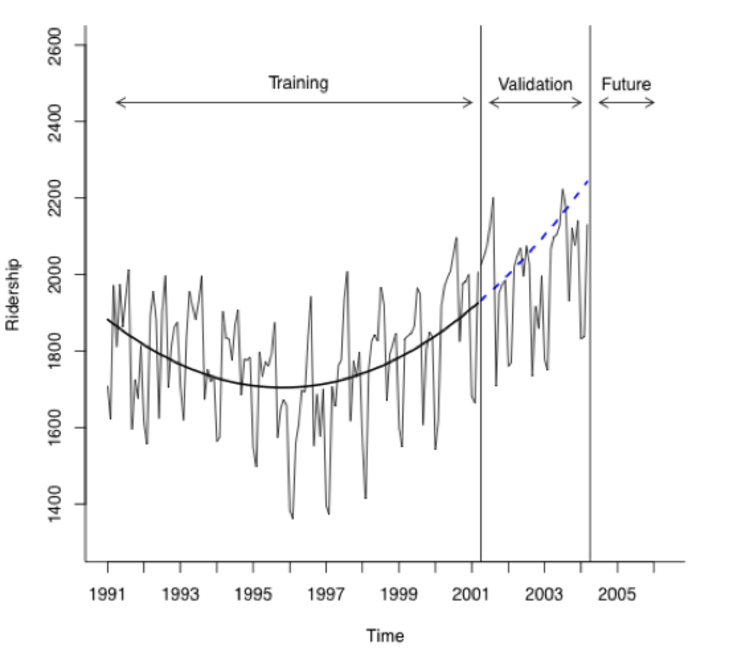
Figure 2 and 3 are the pair of plots for the quadratic fit (Figure 2: actual + forecasts; Figure 3: forecast errors). We can see that this shape captures the pattern in the trend in the training period. The forecast errors now exhibit only seasonality and no trend

Figure 2: Amtrak ridership series with quadratic trend model: actual and forecasts

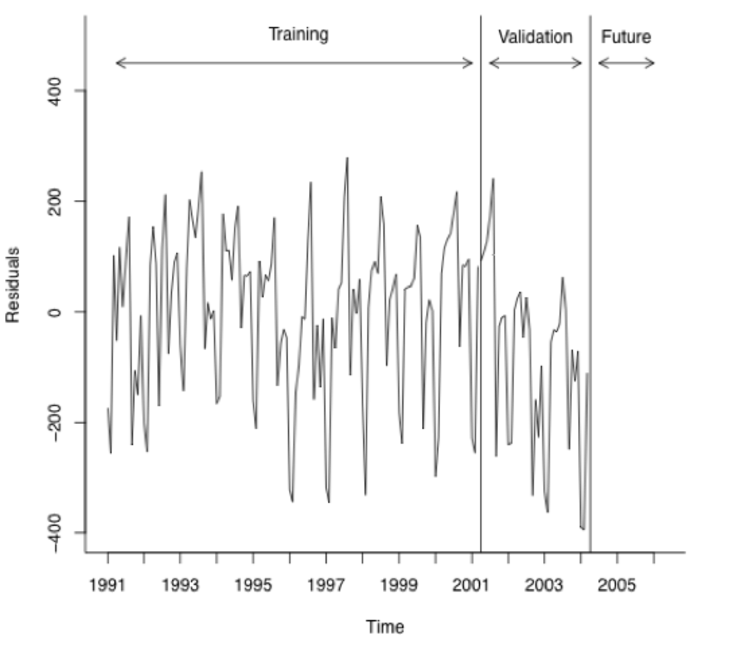


Figure 3: forecast errors

### The following R Codes are used in figure 2 and 3. (exponential trend)

library(forecast)

library(zoo)

Amtrak.data <- read.csv("Amtrak data.csv")

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

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))

train.lm.expo.trend <- tslm(train.ts ~ trend, lambda = 0)

train.lm.expo.trend.pred <- forecast(train.lm.expo.trend, h = nValid, level = 0)

train.lm.linear.trend <- tslm(train.ts ~ trend, lambda = 1)

train.lm.linear.trend.pred <- forecast(train.lm.linear.trend, h = nValid, level = 0)

# **Figures 6-3/6-5**

plot(train.lm.expo.trend.pred, ylim = c(1300, 2600), ylab = "Ridership", 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(train.lm.expo.trend.pred$fitted, lwd = 2, col = "blue") # Added in 6-5

lines(train.lm.linear.trend.pred$fitted, lwd = 2, col = "black", lty = 3)

lines(train.lm.linear.trend.pred$mean, lwd = 2, col = "black", lty = 3)

lines(train.ts)

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)

BoxCox(train.ts, 0)

In general, any type of trend shape can be fit as long as it has a mathematical representation. However, the underlying assumption is that this shape is applicable throughout the period of data that we currently have as well as during the validation period and future. Do not choose an overly complex shape, because although it will fit the training period well, it will likely be overfitting the data. To avoid overfitting, always examine performance on the validation period and refrain from choosing overly complex trend patterns

Regression models for capturing seasonality

A seasonal pattern in a time series means that observations that fall in some seasons have consistently higher or lower values than those that fall in other seasons. Examples are day-of-week patterns, monthly patterns, and quarterly patterns. The Amtrak ridership monthly time series, as can be seen in the time plot, exhibits strong monthly seasonality (with highest traffic during summer months).

### Additive Seasonality

|  |  |  |
| --- | --- | --- |
| Month | Ridership | Season |
| Jan-91 | 1709 | Jan |
| Feb-91 | 1621 | Feb |
| Mar-91 | 1973 | Mar |
| Apr-91 | 1812 | Apr |
| May-91 | 1975 | May |
| Jun-91 | 1862 | Jun |
| Jul-91 | 1940 | Jul |
| Aug-91 | 2013 | Aug |
| Sep-91 | 1596 | Sep |
| Oct-91 | 1725 | Oct |
| Nov-91 | 1676 | Nov |
| Dec-91 | 1814 | Dec |
| Jan-92 | 1615 | Jan |
| Feb-92 | 1557 | Feb |
| Mar-92 | 1891 | Mar |
| Apr-92 | 1956 | Apr |
| May-92 | 1885 | May |

The most common way to capture seasonality in a regression model is by creating a new categorical variable that denotes the season for each observation. This categorical variable is then turned into dummy variables, which in turn are included as predictors in the regression model. To illustrate this, we created a new Month column for the Amtrak ridership data, as shown below (I show the first seventeen rows).

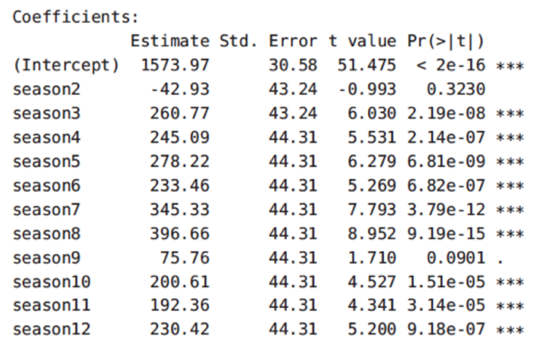
In order to include the season categorical variable as a predictor in a regression model for y (e.g., Ridership), we turn it into dummy variables. For m seasons, we create m-1 dummy variables, which are binary variables that take on the value 1 if the record falls in that particular season, and 0 otherwise. The mth season does not require a dummy, since it is identified when all the m-1 dummies take on zero values.

As with the trend models, before fitting the linear regression, we partition the ridership time series into training and validation periods. Here we keep the last 3 years of data as the validation period.

Next, to fit an additive seasonality model for Ridership. We set the output variable (y) as the Amtrak ridership and use the 11-month dummy variables as predictors. In R, the tslm() function automatically performs this dummy coding so we can directly use the column “season”:

train.lm.season <- tslm(train.ts ~ season)

The estimated regression model looks like this:



In the following regression function, we see that R fit the regression model with 11 dummies dropping season1 (January):

and we are modeling three of the four time-series components: level (β0), seasonality (β1 thru β11) and noise ε. Trend is not modeled.

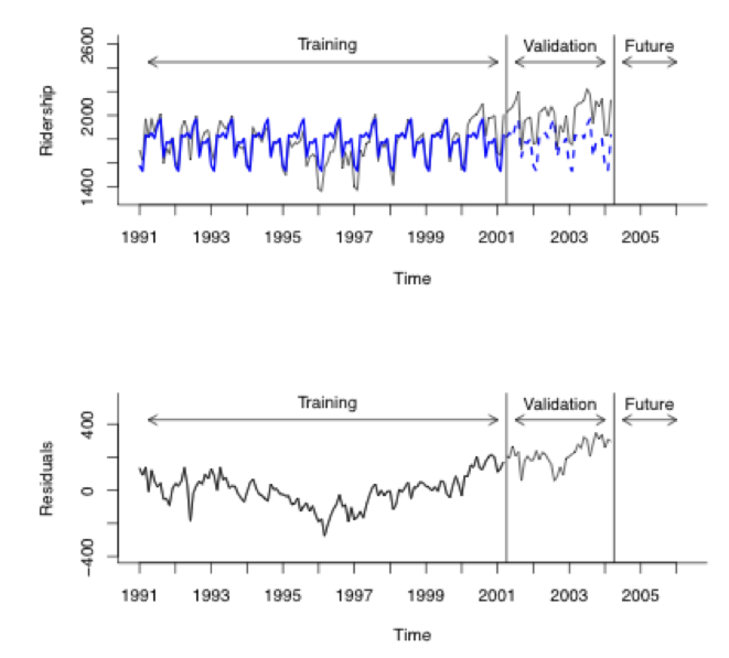
The next step is to use the regression model to make forecasts in the validation period. The figure 4 depicts the actual time series, and the regression model predictions in the training and validation periods.

Figure 4 Amtrak ridership series with additive seasonality regression model. Top: predictions for training (smooth blue line) and validation (dashed blue line). Bottom: forecast errors. <Table 6.3 &. figure 6.6>

### R Codes are used to generate table 6.3 &. figure 6.6 (page 127 & 132).

library(forecast)

library(zoo)

Amtrak.data <- read.csv("Amtrak data.csv")

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

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))

# Table 6.3

train.lm.season <- tslm(train.ts ~ season)

summary(train.lm.season)

train.lm.season.pred <- forecast(train.lm.season, h = nValid, level = 0)

# Figure 6-6 (correction Figure 6.6)

par(mfrow = c(2,1))

plot(train.lm.season.pred, ylim = c(1300, 2625), ylab = "Ridership", 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(train.lm.season.pred$fitted, lwd = 2, col = "blue")

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, 2600, "Training")

text(2002.75, 2600, "Validation")

text(2005.25, 2600, "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)

plot(train.lm.season.pred$residuals, ylim = c(-400, 550), ylab = "Residuals", 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(train.ts - train.lm.season.pred$fitted)

lines(valid.ts - train.lm.season.pred$mean)

lines(c(2004.25 - 3, 2004.25 - 3), c(-500, 3500))

lines(c(2004.25, 2004.25), c(-500, 3500))

text(1996.25, 525, "Training")

text(2002.75, 525, "Validation")

text(2005.25, 525, "Future")

arrows(2004 - 3, 425, 1991.25, 425, code = 3, length = 0.1, lwd = 1,angle = 30)

arrows(2004.5 - 3, 425, 2004, 425, code = 3, length = 0.1, lwd = 1,angle = 30)

arrows(2004.5, 425, 2006, 425, code = 3, length = 0.1, lwd = 1, angle = 30)

### Multiplicative Seasonality

When seasonality is added as described above (create a categorical seasonal variable, then create dummy variables from it, and then regress on *yt*), it captures *additive seasonality*. This means that the average value of y in a certain season is higher or lower by a fixed amount compared to another season. For example, in the Amtrak ridership, the coefficient for August (396.66) indicates that the average number of passengers in August is higher by 396,660 passengers compared to the average in January (the reference category). Using regression models, we can also capture *multiplicative seasonality*, where average values on a certain season are higher or lower by a *fixed percentage* compared to another season. To fit multiplicative seasonality, we use the same model as above, except that we use *log(yt)*as the output variable. To do this in R, we include the argument lambda = 0 in the tslm() function:

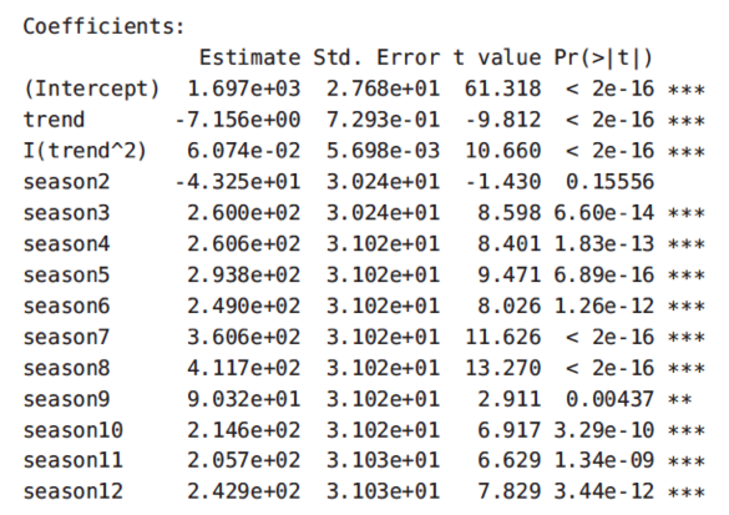
train.lm.season <- tslm(train.ts ~ season, lambda = 0)

### Model with Trend and Seasonality

### We can create regression models that capture both trend and seasonality by including predictors of both types. Our exploration of the Amtrak ridership data indicates that the series has a quadratic trend and monthly seasonality. We therefore fit a model with 13 predictors: 11 dummy variables for month, and *t* and *t2* for trend. In R we can do this as follows:

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

The estimated model is shown below, with the pair of performance charts (actual and forecasted values, and forecast errors). We see that the training forecast errors no longer contain trend or seasonality (figure 5). In the validation period the seasonality is captured, but the model’s trend seems to over-forecast. The model is depicted in figure 6.6 (page 130 textbook and summary (below) is on page 129.



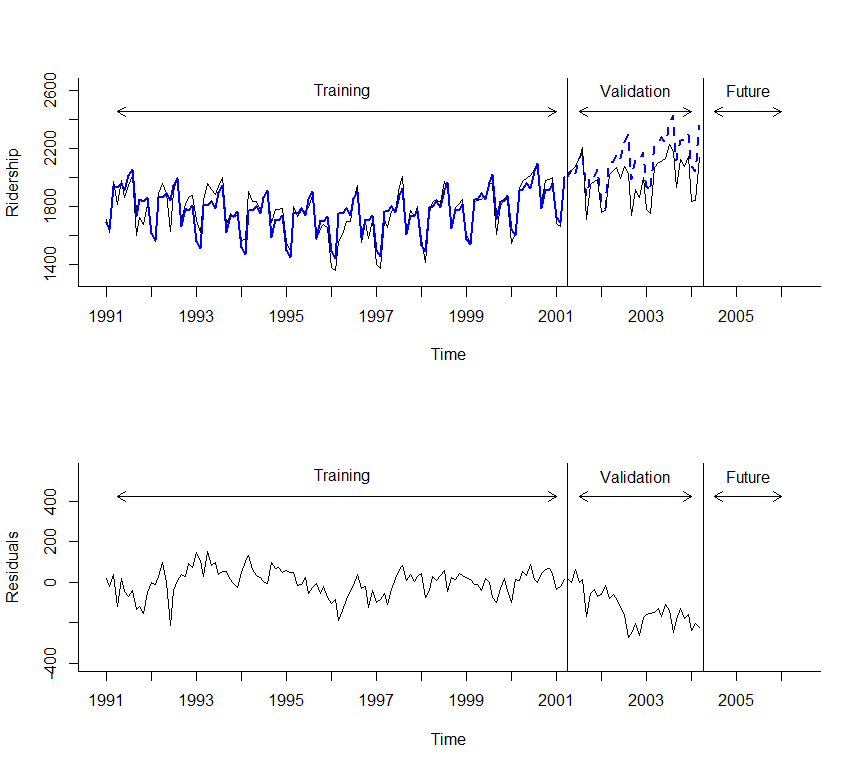


Figure 5: Amtrak ridership series with quadratic trend and additive seasonality regression model. Top: predictions for training (smooth blue line) and validation (dashed blue line). Bottom: forecast errors

### R Codes are used to generate table 6.4 (page 129) & figure 6.7 (page 130)

library(forecast)

library(zoo)

Amtrak.data <- read.csv("Amtrak data.csv")

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

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))

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

train.lm.trend.season.pred <- forecast(train.lm.trend.season, h = nValid, level = 0)

# Table 6-4

summary(train.lm.trend.season)

# Figure 6-7

par(mfrow = c(2,1))

plot(train.lm.trend.season.pred, ylim = c(1300, 2625), ylab = "Ridership", 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(train.lm.trend.season.pred$fitted, lwd = 2, col = "blue")

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, 2600, "Training")

text(2002.75, 2600, "Validation")

text(2005.25, 2600, "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)

plot(train.lm.trend.season.pred$residuals, ylim = c(-400, 550), ylab = "Residuals", 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(train.ts - train.lm.trend.season.pred$fitted)

lines(valid.ts - train.lm.trend.season.pred$mean)

lines(c(2004.25 - 3, 2004.25 - 3), c(-500, 3500))

lines(c(2004.25, 2004.25), c(-500, 3500))

text(1996.25, 525, "Training")

text(2002.75, 525, "Validation")

text(2005.25, 525, "Future")

arrows(2004 - 3, 425, 1991.25, 425, code = 3, length = 0.1, lwd = 1,angle = 30)

arrows(2004.5 - 3, 425, 2004, 425, code = 3, length = 0.1, lwd = 1,angle = 30)

arrows(2004.5, 425, 2006, 425, code = 3, length = 0.1, lwd = 1, angle = 30)

Smoothly Transitioning Seasonality

When the seasonal pattern transitions smoothly from one season to the next, we can use continuous mathematical functions to approximate the seasonal pattern, such as including sinusoidal functions as predictors in the regression model. For example, the Centers for Disease Control and Prevention in the United States use a regression model for modeling the percent of weekly deaths attributed to pneumonia & influenza in 122 cities. The model includes a quadratic trend as well as sine and cosine functions for capturing the smooth seasonality pattern. In particular, they use the following regression model:

The trend terms *t* and *t2* accommodate long-term linear and curvilinear changes in the background proportion of pneumonia & influenza deaths arising from factors such as population growth or improved disease prevention or treatment (CDC data). The sine and cosine terms capture the yearly periodicity of weekly data (with 52.18\* weeks per year). This regression model is then fitted to five years of data to create a “baseline” against which new weekly mortality is compared, called the ’Serfling method’. To fit this type of model with linear, quadratic, sine, and cosine terms to the monthly Amtrak ridership data, we add two predictors to quadratic trend model. In R, the code would be:

\*\*tslm(train.ts ~ trend + I(trend^2) + I(sin(2\*pi\*trend/12)) + I(cos(2\*pi\*trend/12)))

Figure 6.8 (textbook page 131) shows how the seasonality is forecasted using sinusoidal function for the CDC data. I used the same method to forecast the Amtrak ridership. The figure 6 shows the forecast and below is forecast summary.

tslm(formula = train.ts ~ trend + I(sin(2 \* pi \* trend/12)) + I(cos(2 \* pi \* trend/12)))

Residuals:

Min 1Q Median 3Q Max

-318.34 -96.30 8.04 89.73 307.89

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 1748.0335 25.0477 69.788 < 2e-16 \*\*\*

trend 0.4229 0.3507 1.206 0.23015

I(sin(2 \* pi \* trend/12)) -48.2666 17.5260 -2.754 0.00681 \*\*

I(cos(2 \* pi \* trend/12)) -106.5245 17.6773 -6.026 1.93e-08 \*\*\*

---

Signif. codes: 0 ë\*\*\*í 0.001 ë\*\*í 0.01 ë\*í 0.05 ë.í 0.1 ë í 1

Residual standard error: 137.9 on 119 degrees of freedom

Multiple R-squared: 0.2762, Adjusted R-squared: 0.2579

F-statistic: 15.14 on 3 and 119 DF, p-value: 2.088e-08

\*In every 4 years, there are 3 years of 365 days and 1 year 366. There are 52.14 weeks in the first three years and 52.28 weeks in the 4th year.

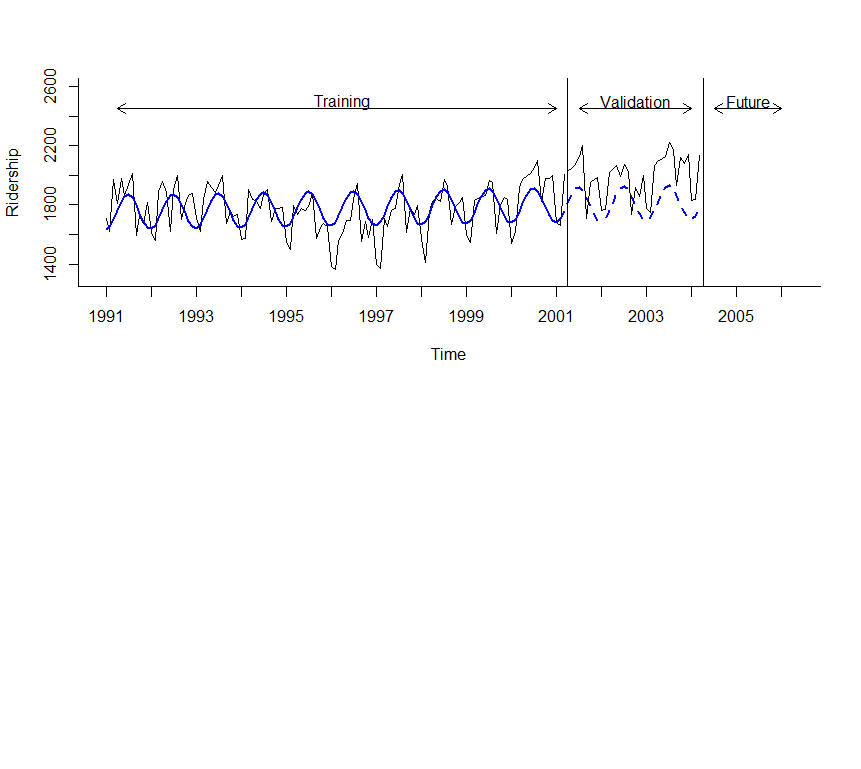
\*\*The R I() function would return a vector of values of the result of math function within parenthesis.

Figure 6: Amtrak forecast of the validation dataset using “Serfling method”.

### R Codes are used to generate figure 6

library(forecast)

library(zoo)

Amtrak.data <- read.csv("Amtrak data.csv")

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

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))

#Table 6.5

train.lm.trig <- tslm(train.ts ~ trend + I(sin(2\*pi\*trend/12)) + I(cos(2\*pi\*trend/12)))

train.lm.trig.pred <- forecast(train.lm.trig, h = nValid, level = 0)

summary(train.lm.trig)

#fig 6 in my notes

plot(train.lm.trig.pred, ylim = c(1300, 2600), ylab = "Ridership", 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(train.lm.trig.pred$fitted, lwd = 2, col = "blue")

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)

### Regression function (regression model)

Regression mode is a mathematical function with predictors as input and estimated observation value as the output. In previous sections of this chapter we saw some of them. I will repeat them and then add those missed. The function has a general form but should be customized based on characteristics of the dataset.

1. Dataset with linear trend and no seasonality

Where

1. Dataset with quadratic trend and no seasonality

1. Dataset with linear trend and additive seasonality (assuming our season is M and we have N seasons)

Note: if we are forecasting an observation in a specific season say, M*i ,* only this season is one and the rest are all zeros

1. Dataset with quadratic trend and additive seasonality
2. Dataset with exponential trend and no seasonality

I will omit all types of multiplicative seasonality in this course.