Week 2 Topics

Stationary and Non-Stationary Time Series

# Stationarity in Time Series

Stationarity signifies that the statistical properties of time series, such as mean, variance, and covariance, remain constant over time, which is the fundamental assumption for many time series modeling techniques. It simplifies the complex dynamics within the data, making it more flexible to analysis, modeling, and forecasting.

There are two main types of Stationarity present in time series data.

* Strick Stationary
* Weak Stationary

These two types are already explained in the first class meeting. For the first one at any point of time dataset statistical property is the same. Here I repeat the meaning of the second one.

A time series is said to have Weak Stationarity if the **mean** and **variance** of the data remain constant over time, and the covariance between any two data points is a function of their time lag. It’s a more practical condition, as it allows for minor fluctuations in the data while preserving the essential statistical properties. Many real-world time series data can be approximated as weakly stationarity, making it a widely used assumption in time series analysis. This is also known as second-order stationarity or covariance stationarity.

Here are main properties of Stationary Time series data.

* Constant Mean
* Constant Variance
* Auto-covariance

The Measures of the relationship between data points at different time lags. In a stationary time series, this function exhibits a consistent pattern. This property is used in Autoregressive and ARIMA models

* Independence of observations
* Lack of significant Trend and Seasonality

Andrew Tate, Further reading, December 1, 2023: https://hex.tech/blog/stationarity-in-time-series/#:~:text=Non%2Dstationary%20data%20can%20lead,than%20non%2Dstationary%20time%20series.

Chapter 3 – Performance Evaluation of Forecasting Models

At first glance, we might think it best to choose a model that generates the best forecasts on the data series at hand. However, when we use the same data both to develop the forecasting model and to assess its performance, we introduce bias. This is because when we choose a model among a set of models that works best with the data, this model’s superior performance comes from two sources:

1. a superior model
2. chance aspects of the data that happen to match the chosen model better than they match other models

The latter is a particularly serious problem with techniques that do not impose linear or other structure on the data, and thus end up overfitting the data. Overfitting means that the model is not only fitting the systematic component of the data, but also the noise. An over-fitted model is therefore likely to perform poorly on new data.

To address the problem of overfitting, an important preliminary step before applying any forecasting method is data partitioning, where the series is split into two periods. We develop our forecasting model or method using only one of the periods. After we have a model, we try it out on another period and see how it performs. In particular, we can measure the forecast errors, which are the differences between the predicted values and the actual values, as described in the next section.

Data Partitioning

Normally to perform supervised learning you need two types of data sets:

* In one dataset (your "gold standard") you have the input data together with correct/expected output. This dataset is usually properly prepared either by humans or by collecting some data in semi-automated way. But it is important that you have the expected output for every data row here, because you need for supervised learning.
* The data you are going to apply your model to. In many cases this is the data where you are interested for the output of your model and thus you don't have any "expected" output here yet.

While performing machine learning you do the following:

* Training phase: you present your data from your "gold standard" and train your model, by pairing the input with expected output.
* Validation/Test phase: in order to estimate how well your model has been trained (that is dependent upon the size of your data, the value you would like to predict, input etc.) and to estimate model properties (mean error for numeric predictors, classification errors for classifiers, recall and precision for IR-models etc.)
* Application phase: now you apply your freshly developed
* .model to the real-world data and get the results. Since you normally don't have any reference value in this type of data (otherwise, why would you need your model?), you can only speculate about the quality of your model output using the results of your validation phase.

The validation phase is often split into two parts:

* In the first part you just look at your models and select the best performing approach using the validation data ( = validation)
* Then you estimate the accuracy of the selected approach (=test).

Hence a possible separation to (T/V/T) 50/25/25

In case if you don't need to choose an appropriate model from several competing approaches, you can just re-partition your set that you basically have only training set and test set, without performing the validation of your trained model. Without Validation set, data is partitioned into, say the (T/T or T/V) 70/30).

# Dataset types

1. Cross-sectional

Cross-sectional data, or a cross section of a population under study, in statistics and econometrics is a type of data collected by observing many different subjects (e.g. people, organization, countries, or regions) at the one point or period of time. For analyst, generally time is an irrelevant matter for analysis. Analysis of observations in a cross-sectional data usually consists of comparing the differences among selected attributes.

For example, if we want to estimate current house value in a geographic area such as a county, we could collect randomly a sample of 1,000 houses from that area (also known as a cross section of that area entire house population), measure their characteristics such as number of rooms and size under construction, and calculate what percentage of that sample is considered as expensive. This cross-sectional sample provides us with a snapshot of that population, at that one point in time. Note that we do not know based on one cross-sectional sample if houses price are increasing or decreasing in that particular county; we can only describe the current situation. All data types we used in our data mining and predictive class were cross-sectional

1. Time series

As we discussed on first class meeting, time series data is a collection of observations obtained through repeated measurements over time. Plot the points on a graph, and one of the (generally the x) axes would always be time.

Time series data is everywhere, since time is a part of everything that is observable. In today world, sensors and systems are constantly recording stream of time series data. And as you will learn in this course, such data has numerous applications across various organizations. All dataset we will use in this course are time series

# Data Partitioning for Time Series Data

1. Temporal partitioning
2. Joining partitions before forecasting
3. Choosing the validation period
4. Roll-Forward Validation

Naïve forecasts – The Benchmark

Although it is tempting to apply “sophisticated” forecasting methods, remember to consider *naïve forecasts*. A naïve forecast is simply the most recent value of the series. In other words, at time t, the k-step-ahead naïve forecast is given by the formula:

In the case of a seasonal series, the seasonal naïve forecast is the value from the most recent identical season (e.g. forecast December using last December’s value). For a series with M seasons, we can write the formula:

. . .

Let’s assume we have month for seasonal period. Since there are 12 months in a year then M = 12.

Now, we want to forecast for t = 13 (that is January of the next year) then

There is a more general formula for calculating the naïve forecast value for series with seasonality.

For k period ahead forecasting we calculate the integer value of the J = then the forecasting formula can be written as

Here I am going to get the value of the Y subscript for t = 12, k = 1, and M = 12:

J = and t + k – M (J+1) = 12 + 1 - 12(0+1) = 1 so (past January value)

And for we have j = 1 and t + k – M(1+1) = 12+15 – 24 = 3 and (past March value)

The underlying logic is that the most recent information is likely to be the most relevant for forecasting the future. Naïve forecasts are used for two purposes:

1. As the actual forecasts of the series. Naïve forecasts, which are simple to understand and easy to implement, can sometimes achieve sufficiently useful accuracy levels. Following the principle of “the simplest method that does the job”, naïve forecasts are a serious contender.
2. As a baseline. When evaluating the predictive performance of a certain method, it is important to compare it to some baseline. Naïve forecasts should always be considered as a baseline, and the comparative advantage of any other methods considered should be clearly shown.

As with forecasts in general, the predictive performance of naïve forecasts is evaluated on the validation period, and we can examine the corresponding forecast error distribution and create prediction intervals.

Note: Data partitioning is performed not only with time series data, but also with cross sectional data. However, such partitioning is done in a slightly different way.

Evaluating forecast accuracy

# Training and test sets

It is important to evaluate forecast accuracy using genuine forecasts. Consequently, the size of the residuals is not a reliable indication of how large true forecast errors are likely to be. The accuracy of forecasts can only be determined by considering how well a model performs on new data that were not used when fitting the model.

As we discussed in the section above, when choosing models, it is common practice to separate the available data into two portions, training and validation or test data, where the training data is used to estimate any parameters of a forecasting method and the validation data is used to evaluate its accuracy. Because the test data is not used in determining the forecasts, it should provide a reliable indication of how well the model is likely to forecast on new data.



**Validation Data**

The size of the test set is typically about 20% of the total sample, although this value depends on how long the sample is and how far ahead you want to forecast. As we discussed, the validation data should ideally be at least as large as the maximum forecast horizon required. The following points should be noted.

* A model which fits the training data well will not necessarily forecast well.
* A perfect fit can always be obtained by using a model with enough parameters.
* Over-fitting a model to data is just as bad as failing to identify a systematic pattern in the data.

Some references describe the validation set as the “hold-out set” because these data are “held out” of the data used for fitting. Other references call the training set the “in-sample data” and the validation dataset the “out-of-sample data”.

# Forecast errors

A forecast “error” is the difference between an observed value and its forecast. Here “error” does not mean a mistake, it means the unpredictable part of an observation. It can be written as

where the training data is given by {y1,…,yn} and the validation data is given by {yn+1,yn+2,…,yn+v}.

# Residual

Note that forecast errors are different from residuals in two ways. First, residuals are calculated on the training set while forecast errors are calculated on the validation set. Second, residuals are based on one-step forecasts while forecast errors can involve multi-step forecasts. We can measure forecast accuracy by summarizing the forecast errors in different ways.

**Scale-dependent errors**

The forecast errors are on the same scale as the data. Accuracy measures that are based only on ***et*** = ***yt – Ft*** are therefore scale-dependent and cannot be used to make comparisons between series that involve different units. The most used scale-dependent metrics are based on absolute errors or on squared errors:

* Mean Error ME or MD.

Formula:

Function: ME = mean(et)

* Mean Absolute Error MAE or mean absolute deviation Error

Formula:

Function: MAE = mean(|et|)

* Geometric Mean Absolute Error

Formula: GMAE = (vth root of product of v numbers)

Function: GMAE = gmean(|et|)

* Mean Square Error

Formula:

Function:

* Root Mean Squared Error

Formula:

Function:

**Percentage errors**

The percentage error is given by Pt= 100(et/yt). Percentage errors have the advantage of being unit-free, and so are frequently used to compare forecast performances between data sets. The most commonly used measure is:

* Mean absolute percentage error: MAPE = mean(|pt|)

Measurements based on percentage errors have the disadvantage of being infinite or undefined if there are zero values in a series, as is frequent for intermittent data. Moreover, percentage errors can have an extremely skewed distribution when actual values are close to zero. With intermittent-demand data, it is impossible to use the MAPE because of the occurrences of zero periods of demand.

Mean Absolut Scaled Error (MASE) is an alternative to percentages for the calculation of scale-independent measurements involves scaling the error based on the in-sample MAE from Naïve forecast method. Using the Naïve method, we generate one-period-ahead forecasts from each data point in the *training* dataset. Accordingly, a scaled error is defined as:

# Forecast Accuracy vs. Profitability

Evaluating Forecast Uncertainty

# Distribution of Forecast Error

The forecast errors are the key to estimating uncertainty. Popular forecasting methods such as regression models and smoothing methods that we will see in this course produce forecast errors that are not necessarily normally distributed. Forecast errors might not be independent, and not even be symmetrical around zero. Plotting a histogram of the forecast errors is therefore very useful. We can learn about the expected distribution of forecast errors, and the chances of obtaining forecast errors of different directions and magnitudes. For example, a left-tailed histogram indicates that the model is more likely to over-predict (negative errors) and the magnitude of errors is larger for over-prediction. This is not necessarily a problem: asymmetries might be useful in some contexts, where, for instance we prefer a high chance of small over-prediction but do not mind large under-predictions.

# Prediction intervals

Reporting a single number as a forecast is called a *point forecast* (or *point prediction*). While useful, it is highly unlikely that the point forecast will be equal to the actual value. The performance charts and metrics give us an idea about the expected magnitude of errors, but these are either aggregated (such as RMSE and MAPE metrics) or are for individual periods in the past (as in the performance charts). To convey the uncertainty about forecasting each future period, it is highly advisable to report a *forecast interval* (also called *prediction interval*) along with the point forecast. For example, assuming that the forecast errors are normally distributed, a 95% prediction interval for the k-step forecast is . Where is an estimate of the standard deviation of the k-step forecast distribution.

More generally, a prediction interval can be written as where the multiplier C depends on the coverage probability. In this course we usually calculate 80% intervals and 95% intervals, although any percentage may be used. The following table gives the value of C for a range of coverage probabilities assuming normally distributed forecast errors.

|  |  |
| --- | --- |
| Percentage | Multiplier |
| 50 | 0.67 |
| 55 | 0.76 |
| 60 | 0.84 |
| 65 | 0.93 |
| 70 | 1.04 |
| 75 | 1.15 |
| 80 | 1.28 |
| 85 | 1.44 |
| 90 | 1.64 |
| 95 | 1.96 |
| 96 | 2.05 |
| 97 | 2.17 |
| 98 | 2.33 |
| 99 | 2.58 |

The value of prediction intervals is that they express the uncertainty in the forecasts. If we only produce point forecasts, there is no way of telling how accurate the forecasts are. However, if we also produce prediction intervals, then it is clear how much uncertainty is associated with each forecast. For this reason, point forecasts can be of almost no value without the accompanying prediction intervals.

**One-step prediction intervals.**

When forecasting one step ahead, the standard deviation of the forecast distribution is almost the same as the standard deviation of the residuals. (In fact, the two standard deviations are identical if there are no parameters to be estimated, as is the case with the Naïve method. For forecasting methods involving parameters to be estimated, the standard deviation of the forecast distribution is slightly larger than the residual standard deviation, although this difference is often ignored.)

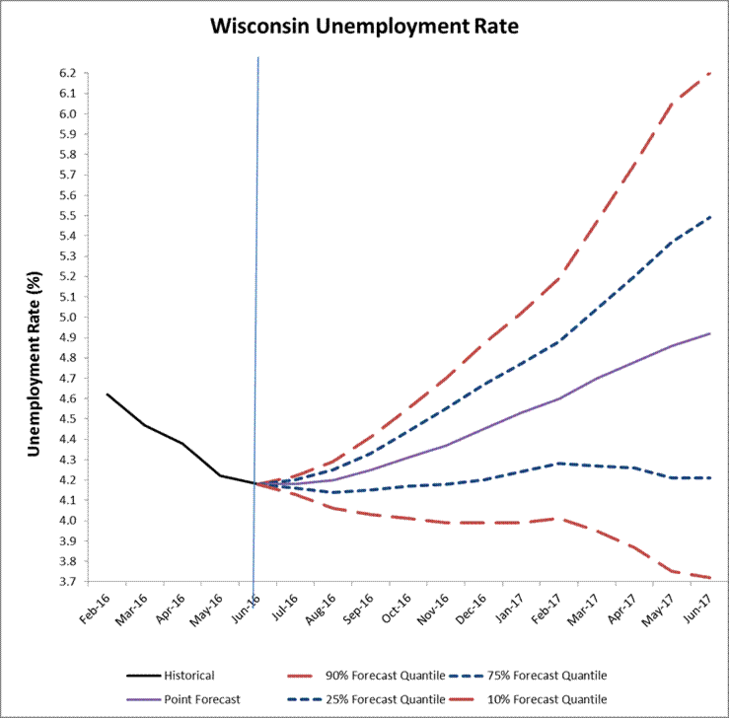
For example, consider a Naïve forecast for the Google stock price data goog200 (shown in Figure 3.5). The last value of the observed series is 531.48, so the forecast of the next value of the GSP is 531.48. The standard deviation of the residuals from the naïve method is 6.21. Hence, a 95% prediction interval for the next value (k = 1) of the GSP is

**Multi-step prediction intervals**

A common feature of prediction intervals is that they increase in length as the forecast horizon increases. The further ahead we forecast, the more uncertainty is associated with the forecast, and thus the wider the prediction intervals. That is,  usually increases with k (although there are some non-linear forecasting methods that do not have this property).

To produce a prediction interval, it is necessary to have an estimate of . As already noted, for one-step forecasts (k=1), the residual standard deviation provides a good estimate of the forecast standard deviation σ1. For multi-step forecasts, a more complicated method of calculation is required. These calculations assume that the residuals are uncorrelated.

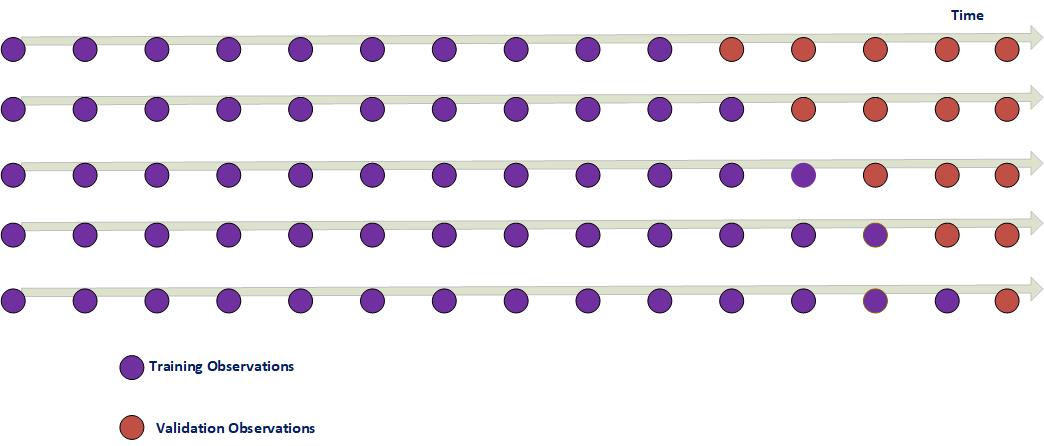
Finally, some forecasting models have different levels of forecast certainty for different future periods, typically with decreasing certainty as we forecast further into the future. When such a model’s prediction intervals at a specific level of certainty are contiguous in time, they form a prediction cone. The following figure illustrates a forecasting cone for unemployment rates in Wisconsin in the next 12 months. These forecasts and forecast cones are updated monthly.



Point forecasts, 50% and 80% prediction cones for monthly unemployment rates in Wisconsin for the next 12 months.

Advanced Partitioning: Roll-Foreword Validation

A more sophisticated version of training/validation sets is time series cross-validation. In this procedure, there are a series of validation sets, each consisting of a single (or one-less observation, textbook version) observation. The corresponding training set consists only of observations that occurred prior to the observation that forms the validation set. Thus, no future observations can be used in constructing the forecast. Since it is not possible to obtain a reliable forecast based on a small training set, the earliest observations are not considered as validation sets.

The following diagram illustrates the series of training and test sets, where the blue observations form the training sets, and the red observations form the validation sets.

The forecast accuracy is computed by averaging over the validation sets. This procedure is sometimes known as “evaluation on a rolling forecasting origin” because the “origin” at which the forecast is based rolls forward in time.

Chapter 3 figures R-Code.

Figure 3-1

##\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Figure 3.1 Page 47 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

library(forecast)

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

plot(Amtrak.ts , ylim = c(1300, 2600), ylab = "Passenger Count", xlab = "Date", bty = "l", xaxt = "n", xlim = c(1991,2006.25), col="dark blue")

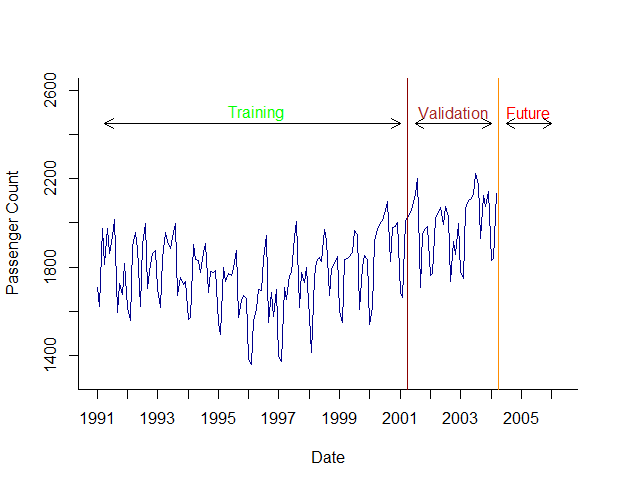
axis(1, at = seq(1991, 2006, 1), labels = format(seq(1991, 2006, 1), digits = 2))

lines(c(2004.25 -3 , 2004.25 -3), c(0, 3500), col="dark red")

lines(c(2004.25, 2004.25), c(0, 3500), col = "dark orange")

text(1996.25, 2500, "Training", col = "green")

text(2002.75, 2500, "Validation", col = "brown")

text(2005.25, 2500, "Future", col = "red")

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)

Figure 3-2

##\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Figure 3.2 Page 49 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

##We assume the dataset is read and transformed into time series forma (see figure 3.1 code)

#data partitioning

stepsAhead <- 36 #three years, 36 month

length(Amtrak.ts)

train.length <- length(Amtrak.ts) - stepsAhead

train.ts <- window(Amtrak.ts, start = c(1991, 1), end = c(1991, train.length))

valid.ts <- window(Amtrak.ts, start = c(1991, train.length + 1), end = c(1991, train.length + stepsAhead))

#Use regression function for time sereis tslm() to forcasting model with quadratic curve.

Amtrak.lm <- tslm(train.ts ~ trend + I(trend^2)) # assuming a quadratic curve trend

Amtrak.lm.pred <- forecast(Amtrak.lm, h = stepsAhead, level = 0)#forcasting the validation data (h mean horizon)

## Generating the vizualization

plot(Amtrak.lm.pred, ylim = c(1300, 2600),ylab = "Passenger Count", xlab = "Date", bty = "l", xaxt = "n", xlim = c(1991,2006.25), main = "", flty = 2, col = "red")

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

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

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)

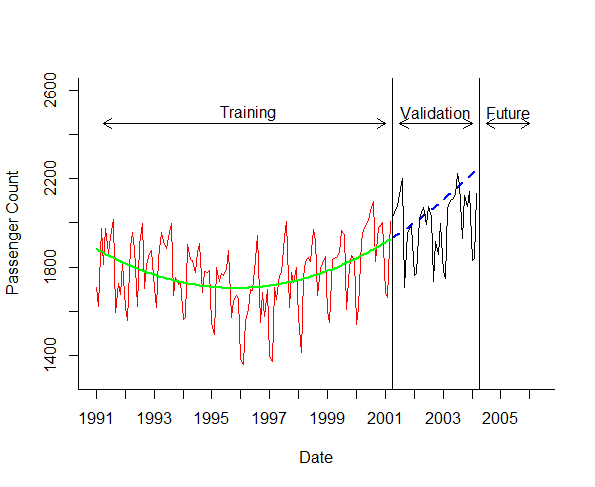


Figure 3-3

##\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Figure 3.3 Page 57 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

##We assume the dataset is read and transformed into time series forma (see figure 3.1 code)

#data oartitioning

stepsAhead <- 36 #three years, 36 month

length(Amtrak.ts)

train.length <- length(Amtrak.ts) - stepsAhead

train.ts <- window(Amtrak.ts, start = c(1991, 1), end = c(1991, train.length))

valid.ts <- window(Amtrak.ts, start = c(1991, train.length + 1), end = c(1991, train.length + stepsAhead))

#Use regression function for time sereis tslm() to forcasting model with quadratic curve.

Amtrak.lm <- tslm(train.ts ~ trend + I(trend^2)) # assuming a quadratic curve trend

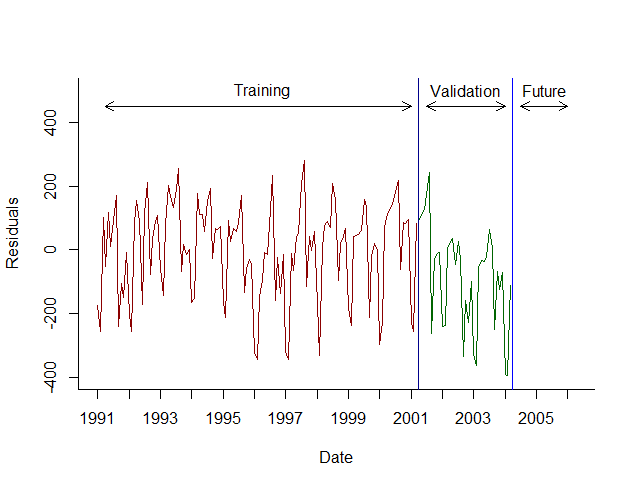
Amtrak.lm.pred <- forecast(Amtrak.lm, h = stepsAhead, level = 0)

plot(Amtrak.lm.pred$residuals, ylim = c(-400, 500), ylab = "Residuals", xlab = "Date", bty = "l", xaxt = "n", xlim = c(1991,2006.25), main = "", col = "dark red"); axis(1, at = seq(1991, 2006, 1), labels = format(seq(1991, 2006, 1)))

lines(valid.ts - Amtrak.lm.pred$mean, lwd = 1, col="dark green"); lines(c(2004.25 - 3, 2004.25 - 3), c(-500, 3500), col = "dark blue")

lines(c(2004.25, 2004.25), c(-500, 3500),col="blue")

text(1996.25, 500, "Training"); text(2002.75, 500, "Validation"); text(2005.25, 500, "Future")

arrows(2004 - 3, 450, 1991.25, 450, code = 3, length = 0.1, lwd = 1,angle = 30); arrows(2004.5 - 3, 450, 2004, 450, code = 3, length = 0.1, lwd = 1,angle = 30); arrows(2004.5, 450, 2006, 450, code = 3, length = 0.1, lwd = 1, angle = 30

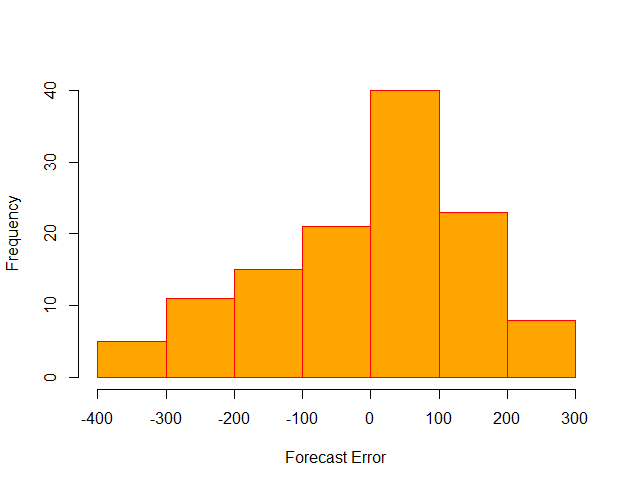
Notes: As you see, the first section of the code is the same as figure 2. We just plot the residual of the validation forecast.

Here I just put the code for names of the output of the trend forecast model and the plot of the histogram of the forecast error (residuals). Check the codes for figure 3-2 and 3-3.

names(ridership.lm.pred)

[1] "model" "mean" "lower" "upper" "level" "x" "series" "method" "newdata" "residuals" "fitted"

Figure 3-4

##\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Figure 3.4 Page 58 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#using the code for the figure 3.3 page 58

hist(Amtrak.lm.pred$residuals, ylab = "Frequency", xlab = "Forecast Error", bty = "l", main = "", col = "orange", border = "red")

Figure 3-5

The code for this figure is not different than the ones for figure 3-2. Except the forecast is not a point forecast but rather a probability forecast with confidence level of 95%. See the *level* parameter in the *forecast* function.

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Figure 3.5 Page 59 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

##We assume the dataset is read and transformed into time series forma (see figure 3.1 code)

#Use regression function for time sereis tslm() to forcasting model with quadratic curve. Note the level of acuracy is 95%

Amtrak.lm <- tslm(train.ts ~ trend + I(trend^2)) # assuming a quadratic curve trend

Amtrak.lm.pred <- forecast(Amtrak.lm, h = stepsAhead, level = 95)

plot(Amtrak.lm.pred, ylim = c(1300, 2600), ylab = "Passenger Count", xlab = "Time", bty = "l", xaxt = "n", xlim = c(1991,2006.25), main = "", flty = 2, col="dark red")

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

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

lines(valid.ts)

lines(c(2004.25 - 3, 2004.25 - 3), c(0, 3500), col = "dark blue")

lines(c(2004.25, 2004.25), c(0, 3500), col="blue")

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)

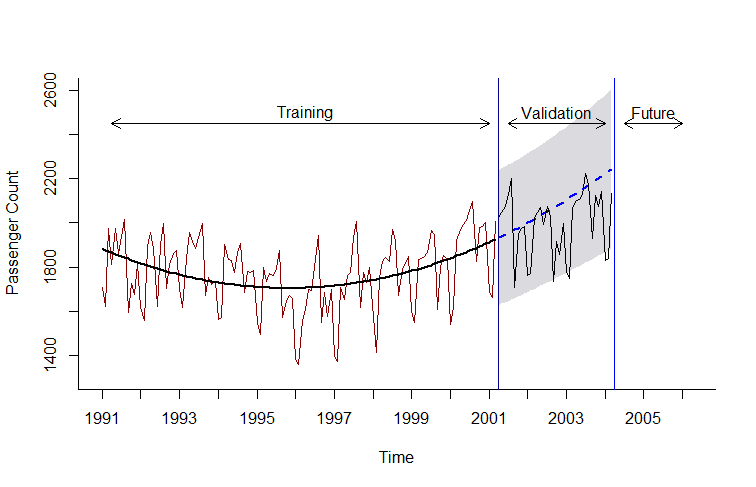


Figure 3-6

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Figure 3.6 Page 62 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

tumblr.df<-read.csv("Tumblr.csv")

people.ts<-ts(tumblr.df$People.Worldwide)/1000000

people.ets.AAN<-ets(people.ts, model="AAN") #model 1

people.ets.MMN<-ets(people.ts, model="MMN", damped = FALSE) #model 2

people.ets.MMdN<-ets(people.ts, model="MMN", damped = TRUE) #model 3

people.ets.AAN.pred<-forecast(people.ets.ANN, h = 115, level = c(0.2, 0.4, 0.6, 0.8))

people.ets.MMN.pred<-forecast(people.ets.MMN, h = 115, level = c(0.2, 0.4, 0.6, 0.8))

people.ets.MMdN.pred<-forecast(people.ets.MMdN, h = 115, level = c(0.2, 0.4, 0.6, 0.8))

par(mfrow=c(1,3)) # three graph in one row

plot(people.ets.AAN.pred, xlab="Month", ylab="People (in million)", ylim = c(0, 1000))

plot(people.ets.MMN.pred, xlab="Month", ylab="People (in million)", ylim = c(0, 1000))

plot(people.ets.MMdN.pred, xlab="Month", ylab="People (in million)", ylim = c(0, 1000))

# Examine the lower and upper limits of the MMN model's prediction cones.

people.ets.MMN.pred$lower

people.ets.MMN.pred$upper

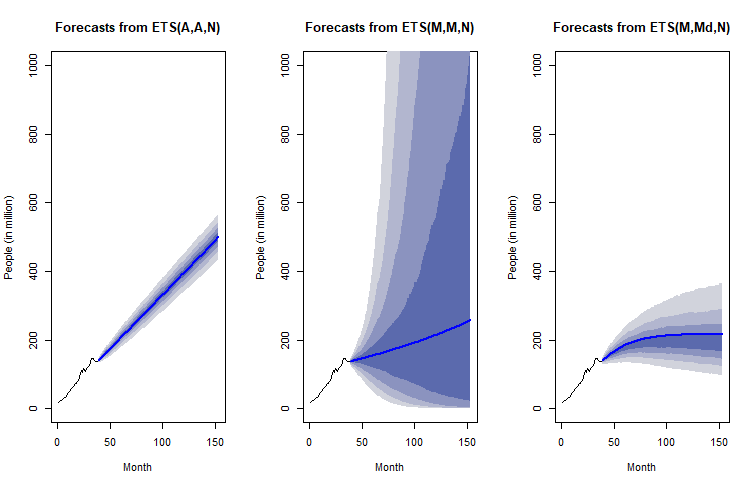


Figure 3-7

##\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Figure 3.7 Page 63 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

library(forecast)

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

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

fixed.Valid <- 36

fixed.Train <- length(Amtrak.ts) - fixed.Valid

train.ts <- window(Amtrak.ts, start = c(1991, 1), end = c(1991, fixed.Train))

valid.ts <- window(Amtrak.ts, start = c(1991, fixed.Train + 1), end = c(1991, fixed.Train + fixed.Valid))

naive.fixed <- naive(train.ts, h = fixed.Valid)

naive.roll <- ts(Amtrak.data$Ridership[fixed.Train:(fixed.Train + fixed.Valid - 1)], start = c(1991, fixed.Train + 1), end = c(1991, fixed.Train + fixed.Valid), freq = 12)

plot(train.ts, ylim = c(1300, 2600), ylab = "Passenger Count", xlab = "Date", bty = "l", xaxt = "n", xlim = c(1991,2006.25), main = "", col="dark red")

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

lines(naive.fixed$mean, lwd = 2, col = "blue", lty = 2)

lines(naive.roll, lwd = 2, col = "red", lty = 2)

lines(valid.ts, col="green")

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

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

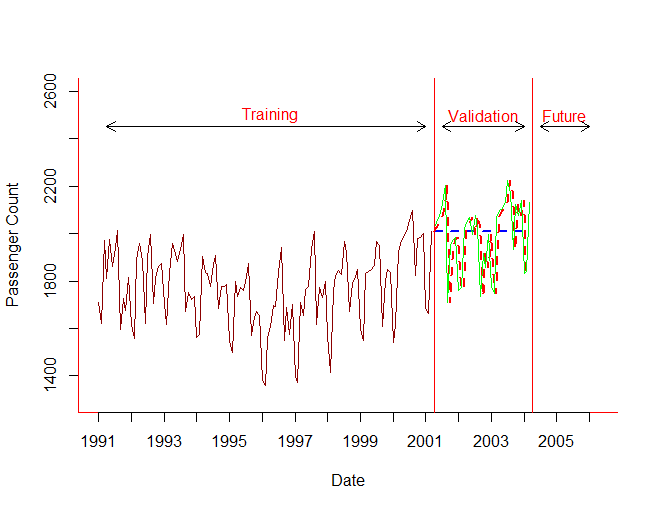
text(1996.25, 2500, "Training")

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)

R-Code on page 64. Naïve forecasting and evaluation measures for table 3-3

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

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

#you can chose values between 1 and 36 to see effect of roll-forward validation policy (I picked 1, 2, and 36)

n<-1

fixed.Valid <- 36

fixed.Train <- length(Amtrak.ts) - fixed.Valid

stepsAhead <-n

error<-rep(0,fixed.Valid - stepsAhead+1)

percent.error<-rep(0,fixed.Valid - stepsAhead + 1)

for (j in fixed.Train:(fixed.Train + fixed.Valid - stepsAhead)){

train.ts <- window(Amtrak.ts, start = c(1991, 1), end = c(1991, j))

valid.ts <- window(Amtrak.ts, start = c(1991, j + stepsAhead), end = c(1991, j + stepsAhead))

naive.pred<-naive(train.ts, h = stepsAhead)

error[j - fixed.Train + 1]<-valid.ts - naive.pred$mean[stepsAhead]

percent.error[j - fixed.Train + 1]<-error[j - fixed.Train + 1]/valid.ts

}

mean(abs(error))

sqrt(mean(error^2))

mean(abs(percent.error))

|  |  |  |  |
| --- | --- | --- | --- |
| Method | MAE | RMSE | MAPE |
| Roll-forward one-month-ahead | 119.4289 | 169.1733 | 0.06206 |
| Roll-forward two-month-ahead | 159.5817 | 198.1042 | 0.08274239 |
| … | … | … | … |
| Roll-forward 36-month-ahead | 124.518 | 124.518 | 0.0583921 |
| Fixed Partitioning overall | 115.9234 | 142.7551 | 6.021396 |

R-Code on page 66. Naïve forecasting and evaluation measures

comparing naïve and seasonal naïve: Fixed partitioning

library(forecast)

library(knitr)

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

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

fixed.Valid <- 36

fixed.Train <- length(Amtrak.ts) - fixed.Valid

train.ts <- window(Amtrak.ts, start = c(1991, 1), end = c(1991, fixed.Train))

valid.ts <- window(Amtrak.ts, start = c(1991, fixed.Train + 1), end = c(1991, fixed.Train + fixed.Valid))

naive.pred <- naive(train.ts, h=fixed.Valid)#naive

snaive.pred <- snaive(train.ts, h=fixed.Valid)#seasonal naive

accuracy(naive.pred, valid.ts)

ME RMSE MAE MPE MAPE MASE ACF1 Theil's U

Training set 2.45091 168.1470 125.2975 -0.3460027 7.271393 1.518906 -0.2472621 NA

Test set -14.71772 142.7551 115.9234 -1.2749992 6.021396 1.405269 0.2764480 0.8346967

accuracy(snaive.pred, valid.ts)

ME RMSE MAE MPE MAPE MASE ACF1 Theil's U

Training set 13.93991 99.26557 82.49196 0.5850656 4.715251 1.000000 0.6400044 NA

Test set 54.72961 95.62433 84.09406 2.6527928 4.247656 1.019421 0.6373346 0.5532435

#A better display of errors

acc.naive.pred<-accuracy(naive.pred, valid.ts)

acc.snaive.pred <- accuracy(snaive.pred, valid.ts)

indexes <- c(2,3,5) # select only column 2 (RMSE), column 3 (MAE), and column 5 (MAPE)

kable(acc.naive.pred[,indexes])

| | RMSE| MAE| MAPE|

|:------------|--------:|--------:|--------:|

|Training set | 168.1470| 125.2975| 7.271393|

|Test set | 142.7551| 115.9234| 6.021396|

kable(acc.snaive.pred[,indexes])

| | RMSE| MAE| MAPE|

|:------------|--------:|--------:|--------:|

|Training set | 99.26557| 82.49196| 4.715251|

|Test set | 95.62433| 84.09406| 4.247656|