Time Series and Forecasting Using R

Lab Sessions 25 May 2016

Before doing any exercises in R,
load the **fpp** package using `library(fpp)`
and the **ggplot2** package using `library(ggplot2)`.

Lab Session: Time series visualization

Use the functions `ggtsdisplay(data, plot.type="scatter")`, `ggseasonplot`, and `ggmonthplot` to explore the trend, seasonality, and cyclicity patterns in the following time series:

- `USAccDeaths`
- `UKDriverDeaths`
- `mdeaths`
- `fdeaths`
- `hsales`
- `vehicles`
- `utility`
- `visitors`
- `enplanements`
- `austourists`
- `cafe`
- `a10`
- `h02`
- `cangas`
- `usmelec`
- `elec`

1. Can you spot the effects of seasonality, cyclicity and trend?
2. What can you say about the seasonal patterns?
3. Can you identify any unusual years/days?

Lab Session: Benchmark forecasting

Choose one of the series (other than USAccDeaths) used in the first lab session. (I'll use USAccDeaths in the examples below, but you should use a different series.)

1. Make a graph of the data with forecasts using the most appropriate of the four benchmark methods: mean, naive, seasonal naive or drift.

   For example:

   ```
   fc <- snaive(USAccDeaths)
   autoplot(fc)
   ```

2. Compute the residuals and plot their ACF. Do the residuals appear to be white noise? What did your forecasting method miss?

   For example:

   ```
   res <- residuals(fc)
   ggtsdisplay(res, plot.type="scatter")
   ```

3. Split the data into two parts using the window function to create a training set and a test set.

   For example

   ```
   training <- window(USAccDeaths, end=c(1976,12))
   test <- window(USAccDeaths, start=1977)
   ```

   Check that your data have been split appropriately by producing the following plot.

   ```
   tmp <- cbind(Data=USAccDeaths, Training=training, Test=test)
   autoplot(tmp)
   ```

4. Calculate forecasts using each of the four benchmark methods applied to the training data.

5. Compare the accuracy of your forecasts against the actual values stored in the test data.

   For example:

   ```
   f1 <- meanf(training)
   accuracy(f1,test)
   ```

6. Which method does best in terms of RMSE? Why?

7. For the best method, compute the residuals and plot them. Do the residuals appear to be uncorrelated and normally distributed?

Lab Session: Time series decomposition

In this session, please continue to use the same series you used in the previous session.

1. Does the series need a Box-Cox transformation to stabilize the variance? If so, find a suitable value of λ. For example:

   ```
   lambda <- BoxCox.lambda(USAccDeaths)
   autoplot(BoxCox(USAccDeaths,lambda))
   ```

2. Do an STL decomposition (on the transformed series if necessary) and plot the results. For example:

   ```
   fitstl <- stl(USAccDeaths, s.window="periodic")
   autoplot(fitstl)
   ```

3. Compute and plot the seasonally adjusted data. For example:

   ```
   x.sa <- seasadj(fitstl)
   autoplot(x.sa)
   ```

4. Use a random walk to produce forecasts of the seasonally adjusted data.

   ```
   fc <- rwf(x.sa)
   autoplot(fc)
   ```

5. Reseasonalize the results to give forecasts on the original scale.

   ```
   fcstlnaive <- forecast(fitstl, method="naive")
   autoplot(fcstlnaive)
   ```

6. Use the same technique to forecast on the training/test set obtained earlier, and compare your result with the benchmark method in the previous lab session.

Lab session: Exponential smoothing

1. Find a good ETS model for the seasonally adjusted data obtained in the previous lab session. For example:

   ```
   fitetssa <- ets(x.sa)
   autoplot(forecast(fitetssa))
   ```

2. Now obtain forecasts using this method on the original data as follows.

   ```
   fcstlets <- forecast(fitstl, method="ets")
   autoplot(fcstlets)
   ```

3. Does this give better results on the test data than the STL+Naive approach from the last lab session?

4. Compare the STL+ETS approach to applying ETS on the data directly (with no Box-Cox transformation).

   ```
   fitets <- ets(USAccDeaths)
   fcets <- forecast(fitets)
   autoplot(fcets)
   ```

5. Repeat on the training/test data.

Lab session: ARIMA models

We will now try to find a suitable ARIMA model for the time series you have been studying.

1. If necessary, transform the data using a suitable Box-Cox transformation.
2. Fit a suitable ARIMA model using `auto.arima()` with a `lambda` argument if you need a Box-Cox transformation.
3. Try some other plausible models by experimenting with the orders chosen;
4. Choose what you think is the best model and check the residual diagnostics.
5. Produce forecasts of your fitted model. Do the forecasts look reasonable?
6. Compare the results with what you obtained previously using `ets()`.
7. Compare the ARIMA model on the training/test data to the best model found previously.

Lab session: Time series cross-validation

We will compare some of the models found so far using time series cross-validation. The following code gives an example of how to compare a specific ETS model with a specific ARIMA model on the `visitors` time series, using four years of data in the initial training set.

```
k <- 48 # minimum size for training set
n <- length(visitors) # Total number of observations
e1 <- e2 <- visitors*NA # Vectors to record one-step forecast errors
for(i in 48:(n-1))
{
  train <- ts(visitors[1:i],freq=12)
  fit1 <- ets(train, "MAM", damped=FALSE)
  fit2 <- Arima(train, order=c(1,0,1), seasonal=c(0,1,2))
  fc1 <- forecast(fit1,h=1)$mean
  fc2 <- forecast(fit2,h=1)$mean
  e1[i] <- visitors[i+1]-fc1
  e2[i] <- visitors[i+1]-fc2
}
sqrt(mean(e1^2,na.rm=TRUE))
sqrt(mean(e2^2,na.rm=TRUE))
```

Do something similar using your data, with the best STL, ETS and ARIMA models you've found.