Homework 1 – Due Tuesday, Feb. 14

STAT-GB.2302, STAT-UB.0018: Forecasting Time Series Data

The problems in this assignment require data files, which are at

`http://ptrckprry.com/course/forecasting/`.

Electronic submissions are not accepted. Print out and turn in your plots and your R code.

The easiest way to complete this assignment is to download and edit the template files

`http://ptrckprry.com/course/forecasting/hw/01/prob1.Rmd`

and

`http://ptrckprry.com/course/forecasting/hw/01/prob2.Rmd`.

These files can be opened in RStudio. Once opened, press the "Knit HTML" button or choose "File ⇒ Knit" to run the R code and generate the plots. To print the source code, choose "File ⇒ Print". To print the generated text and plots, Knit the file, then click "Open in Browser", and print the document from your browser.

Problem 1

The file `gdp.csv` contains the real United Stats Gross Domestic Product (GDP), quarterly, from 1946 (Q4) to 2016 (Q3), a total of $n = 280$ observations. The values of the series are reported in billions of 2009 dollars (seasonally adjusted and inflation adjusted).

(a) Read the GDP series into R and extract the `date` and `gdp` variables by running the following commands:

```
data <- read.csv("http://ptrckprry.com/course/forecasting/data/gdp.csv")
date <- as.Date(data$date)
gdp <- data$gdp
```

Plot the GDP series versus date. (When talking about a plot of y versus x, it is traditional to give the response variable first, then the predictor variable. So, I am asking for GDP on the y-axis and date on the x-axis.) Does GDP seem to grow linearly over time?

(b) Plot the log of GDP, in a time series plot. Does log GDP appear to grow linearly over time? If so, then what does this imply about the growth of GDP itself?

(c) Fit an ordinary linear regression model for Log(GDP), using time as the predictor variable. Based on the output, forecast the Log(GDP) for the fourth quarter of the year 2016. Based on the linear regression model, construct a 95% prediction interval to go with the above point forecast. Do you think this interval forecast (prediction interval) is valid? Does it seem too wide, or too precise? Explain.

(d) On a single plot, superimpose the Log(GDP) series and the fitted line. Does the line fit well?

(e) Plot the residuals from the fitted line versus time. What potential problems with the linear model are indicated by this plot? Do you think these problems could spoil the validity of the forecast interval?

Problem 2

In this problem we will consider the daily U.S. Dollar-Euro exchange rate, daily, 4 Jan 1999 to 6 Feb 2017 ($n = 4636$).

(a) Read the data into R by running the commands

```
data <- read.csv("http://ptrckprry.com/course/forecasting/data/euro.csv")
date <- as.Date(data$date)
euro <- data$euro
```

Create a time-series plot of Euro. Does a straight-line model seem appropriate?

(b) Get point and interval predictions for 7 Feb 2017 (time = 4637) using two different methods: first, based on fitting a straight line to observations 1 to 700; second, based on fitting a straight line to observations 701 to 4636. Did both of the forecast intervals succeed in containing the actual value for 7 Feb 2017? (You can get the value for 7 Feb 2017 from `https://www.ecb.europa.eu/stats/exchange/eurofxref/html/eurofxref-graph-usd.en.html` .) If the forecast intervals do not contain the value, then use what you learned in Problem 1 to give a statistical explanation of what went wrong.

(c) Create a time series plot of Euro with both fitted lines superimposed. Use different line types for both lines.

R commands used in this assignment

To complete this assignment, you will need to use the following R commands.

- `abline`. Add a line to an existing plot. Examples:

  ```
  abline(1.2, 3) # y-intercept is 1.2, slope is 3

  abline(3, 2, lty=2, col="red") # specify line type ("lty") and color ("col")

  abline(v=2) # vertical line with x-coordinate 2

  abline(h=-1) # horizontal line with y-coordinate -1
  ```

- `as.Date`. Convert from a character string to a Date object. The character string must be in the format "YYYY-MM-DD". This is useful if you want to make a plot with dates on the x-axis. However, you should never use a Date variable as a predictor in a regression model.

- `coef`. Extract the fitted coefficients from a model. Examples:

  ```
  # fit a model with response variable "height" and predictor variable "weight"
  model <- lm(height ~ weight)

  # extract the coefficients
  beta <- coef(model)

  # get the fitted intercept
  beta[["(Intercept)"]]

  # get the coefficient the weight variable
  beta[["weight"]]

  # get the fitted value of height when weight = 150:
  beta[["(Intercept)"]] + beta[["weight"]] * 150
  ```

- `data.frame`. Construct a "data frame", i.e. a collection of variables measured on the same set of individuals.

- `legend`. Add a legend to a plot.

- `length`. Get the number of elements in a vector.

- `lm`. Fit a linear regression model. After running this command, use the `summary` command to see details of the fitted model.

```
# simple linear regression: response y, predictor x
model <- lm(y ~ x)
summary(model)

# multiple linear regression:
model <- lm(y ~ x1 + x2 + x3)
summary(model)
```

To see regression diagnostics, run the command `plot` command on the result, e.g. `plot(model)`.

- `log`. Compute the natural logarithm of each element in a numeric vector.
- `plot`. Make a scatter plot. Examples:

```
plot(x, y)              # plot y versus x

plot(x, y, type="l")    # line plot (Note: "l" is ell, not one)

plot(x, y, type="l", xlab="Foo", ylab="Bar", col="steelblue")
```

- `predict`. Use a fitted model to make a forecast. Example:

```
# fit a model with response variable "height" and predictor variable "weight"
model <- lm(height ~ weight)

# get a 95% prediction interval when weight = 150
newdata <- data.frame(weight = 150)
predict(model, newdata, interval = "prediction", level = 0.95)

# get a 99% confidence interval when weight = 200
newdata <- data.frame(weight = 200)
predict(model, newdata, interval = "confidence", level = 0.99)
```

- `read.csv`. Read a comma separated value (CSV) file. Examples:

```
# read from a URL:
data <- read.csv("http://ptrckprry.com/course/forecasting/data/gdp.csv")

# read from a file stored on your hard drive in the current directory:
data <- read.csv("gdb.csv")
```

- `residuals`. Get the residuals from a fitted model. Example:

```
model <- lm(y ~ x)
r <- residuals(model)
plot(x, r) # plot residuals versus x
```