Queen Mary University of London

School of Mathematical Sciences

MTH6139/MTH6139P Time Series Tutorial Week 6

Section 1: Seasonal Indicator Regression

In the previous tutorial we used the Brockwell Davies algorithm as implemented in the R function `decompose()` to decompose the Johnson & Johnson quarterly earnings per share (contained in the R `ts` object `jj`) into trend, seasonal component and noise. This algorithm relied on using a two-sided moving average of size a whole seasonal period to compute the trend.
One disadvantage of two-sided moving averages is that it is hard to use in forecasting as it uses data in the future. In this exercise we will explore doing something similar but using a linear trend instead of moving averages. In mathematical terms this can be written as follows:

$$x_t = \underbrace{\beta_0 + \beta_1 t}_{\text{trend}} + \underbrace{\alpha_1 Q_1(t) + \alpha_2 Q_2(t) + \alpha_3 Q_3(t) + \alpha_4 Q_4(t)}_{\text{seasonal}} + \underbrace{y_t}_{\text{noise}} \tag{1}$$

Where the function $Q_1(t)$ is equal to 1 if t is in quarter 1 and 0 otherwise, and likewise for the other $Q_i(t)$:

$$Q_i(t) = \begin{cases} 1 & \text{if } t \text{ is in quarter } i \\ 0 & \text{otherwise} \end{cases}$$

(sometimes functions like this are called *indicator* functions).
This model can be trivially employed in forecasting, just estimate a future x_t by $\beta_0 + \beta_1 t + \alpha_1 Q_1(t) + \alpha_2 Q_2(t) + \alpha_3 Q_3(t) + \alpha_4 Q_4(t)$, which just means add to the trend $\beta_0 + \beta_1 t$ the number α_1 if t is in quarter 1, or α_2 if t is in quarter 2,... In R a regression like (1) is accomplished by using a *categorical* variable as we will see below.

1.1 Basic linear model

Fit a linear model to the `jj` time series and plot the result in red overlaid on top of the original data.

```
# run a linear regression on jj
# the na.action=NULL makes the regression know this is a ts object and
    it
# records the time component
model1 = lm(jj~time(jj), na.action=NULL)
```

```
plot(jj, type="l") # plot jj
lines(fitted(model1), col="red") # overlay fitted(model) in red
```

You see that the fit is not very good as the time series displays exponential growth typical in monetary quantities.

1.2 Using the log

Repeat the analysis above but now with the log of the time series.

```
# run a linear regression on jj
# the na.action=NULL makes the regression know this is a ts object and
    it
# records the time component
model2 = lm(jj~time(jj), na.action=NULL)
plot(jj, type="l")
lines(fitted(model2), col="red")
```

The fit is much better. Now let us display the parameters of the linear regression using the function `msummary` from the package `mosaic` (which is less cluttered that the default `summary`).

```
# check the intercept and coefficient of the linear regression

```

Recall that `intercept` stands for β_0 and `time(jj)` stands for the coefficient affecting t, β_1.

1.3 Adding a categorical variable

We will now include a categorical variable that indicates what quarter we are in. We saw that the function `cycle()` applied to a `ts` object returns a new time series with a number indicating the position in the yearly cycle. In our case, with quarterly data this will be 1, 2, 3 or 4. Try it

```
# run the function cycle() on the object jj

```

In order to find the α_1, α_1, α_1 and α_1 in (1) we will need to convert this numerical data to categorical. Categorical data is called `factor` in R. This is accomplished with the function `factor` as follows:

```
SeasonalLabels = factor(cycle(jj),
    labels=c("Quarter 1", "Quarter 2", "Quarter 3", "Quarter 4"))
# Now visualize this by writing its name and executing

```

If you run a linear regression on these variables only, you will get a model like in (1) but with no trend. Try it

```
model3 = lm(log(jj)~0+SeasonalLabels, na.action=NULL)
# now visualise coefficients of regression with msummary

```

If you plot this you will see the seasonal effect is captured but we are missing the trend.

```r
# plot log(jj) and overlay it with fitted(model4) in red


```

1.4 Trend and seasonality

Let us add the trend.

```r
model4 = lm(log(jj)~0+time(jj)+SeasonalLabels, na.action=NULL)
# check the parameters of the regression using musmmary from the
   package mosaic

# now plot log(jj) and overlay it with model5


```

1.5 One remark

Note that we have added `0+` at the start of the formula for some linear regressions. This means that we want to set the intercept, β_0, to zero. We do this because the expression (1) has redundant parameters. You could add 1 to β_0 and then subtract 1 from each of the αs and the model would be identical. In Brockwell and Davies, to remove this redundancy, we demand that the average seasonal effect is zero which here means $(\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4)/4 = 0$. But in this exercise, for ease of coding, we remove β_0.

Section 2: Sample correlation

In studying the error terms in a time series y_t we will often compute correlations between y_t and y_{t-h} to see if the past values influence future values. If the correlation is small then we will conclude there is no such effect. However, the correlation number we will compute, the sample correlation, will also be subject to randomness and so sometimes by chance we might observe larger correlation numbers just by chance. This exercise demonstrates how this can happen by computing the sample correlation of two uncorrelated $N(0,1)$ random variables.
Recall that the correlation of two random variables is defined to be

$$\rho_{XY} = \mathbb{E}\left((X - \mathbb{E}(X))\,(Y - \mathbb{E}(Y))\right).$$

If we take a sample of our random variables $(X_1, Y_1), \ldots, (X_n, Y_n)$ then its sample correlation is defined to be

$$\widehat{\rho}_{XY} = \frac{\sum_{i=1}^{n}\left(X_i - \overline{X}\right)\left(Y_i - \overline{Y}\right)}{\sqrt{\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2}\sqrt{\sum_{i=1}^{n}\left(Y_i - \overline{Y}\right)^2}}$$

where $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$, and $\overline{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$, are the sample means.
Note that we are using capital letters which indicates that these numbers are random variables. If, say, X_i indicates height of a random individual, and Y_i their weight, every time we take a new sample we will get different values of X_i, Y_i, $\overline{X}$ and $\overline{Y}$ and $\widehat{\rho}_{XY}$.

2.1 Sample correlation

Let us see this in R, run the code below

```
# this creates 100 random numbers from the N(0,1) distribution
# run ?rnorm for more information
x = rnorm(100)
# 100 more random numbers, independent of the previous
y = rnorm(100)
# In the same way that when you flip a coin 100 times you are not
# guaranteed exactly 50 heads but just something close to it
# Both x, and y, are not guaranteed to have mean 0 and sd 1
# even though they are samples of N(0,1)
mean(x); mean(y)
sd(x); sd(y)
# And correlation is not guaranteed to be exactly zero
cor(x,y)
```

Run the code above a few times to check that for each sample you get different values.

2.2 Sampling the sample correlation

As usual if you are getting random values, for say `cor(x,y)`, it is of interest to see how they are distributed. To this effect we would repeat the experiment a few thousand times running the code above and see what values we get.
As manually re-running the code above, does not seem practical, we use the R function `replicate` which repeats an operation and writes the results to a vector. Run the code below:

```
# this creates two sets of 100 random numbers
# and calculates their correlation
cor(rnorm(100), rnorm(100))
# This does the same 20 times and stores the results in the vector
   called r
r = replicate(20, cor(rnorm(100), rnorm(100)))
# visualize this
r
```

2.3 A histogram for sample correlation

Next we run the calculation above 10,000 times and plot a histogram:

```
# calculate sample correlation of N(0,1) for n=100
# 10,000 times
r = replicate(10000, cor(rnorm(100), rnorm(100)))
# plot a histogram
hist(r, freq=TRUE)
```

This tells us that in general the sample correlation, ρ_{XY} of two $N(0,1)$ independent random variables, X and Y tends to be close to zero which you would expect. But sometimes it deviates. In fact the pdf of the sample standard correlation is related to the *beta distribution* which you might have encountered in other areas of statistics.

2.4 Confidence bars

It is useful to have a notion of how far a random quantity might be away from its mean. For instance, how often might be encounter large sample correlations of uncorrelated random variables.
By considering quantiles of the pdf of $\widetilde{\rho}_{XY}$ it can be seen that values of $\widetilde{\rho}_{XY}$ greater in absolute value than $1.96/\sqrt{n}$ will be seen only 5% of times.
This can be displayed as follows

```
hist(r, freq=FALSE,
     main="Distribution of sample correlation
of uncorrelated normal variables
with 95% confidence bands")
threshold = 1.96/sqrt(n)
abline(v=c(threshold, -threshold), col="blue", lty=2)
```

So in our case, with $n = 100$, even though X and Y are uncorrelated, by random chance, we expect to see correlations greater than $1.96/\sqrt{100} = 0.196$ in absolute value 5% of times.
For example, if you are going to a bunch of schools and calculating the sample correlation of 100 students' academic grades and time of the day of birth, you will get very small numbers most of the time as these quantities are obviously uncorrelated.
But 5% of times, one out of twenty, you will see a correlation greater than 0.196 in absolute value (i.e. greater than 0.196 or smaller than -0.196).
This will be of interest in the analysis of the error of time series.

Section 3: Theory Questions

The Box-Cox transformation has been defined as[1]:

$$f(x) = \begin{cases} \frac{x^\lambda - 1}{\lambda} & \text{if } \lambda > 0 \\ \log(x) & \text{if } \lambda = 0 \end{cases}$$

So we expect that if λ becomes very small the first equation should resemble the second. Let us prove this. But before, some graphs in R to confirm this:

```
x =  # use seq to create a vector of numbers from 0 to 10 in steps of
   0.1
logx = # calculate the log
 # plot (x,logx)
lambda = 0.5
BC0.5 = # replace by the Box-Cox formula
# overlay plot of BC0.5 in colour red
lambda = 0.1
BC0.1 = # replace by the Box-Cox formula
# add to chart above in colour darkgreen
lambda = 0.05
```

[1]Unless otherwise indicated, all logarithms after primary school are in base e, i.e. natural logarithms, also written ln(). Decimal logarithms are rarely used outside of specialized applications. In R `log()` stands for the natural logarithm. In Excel you must use `LN()` as `LOG()` stands for the decimal logarithm.

```
BC0.05 = # replace by the Box-Cox formula
# add to chart above in colour blue
# As a challenge, write an R function to perform the Box-Cox function
```

So we see that it seems like $\lim_{\lambda \to 0}(x^\lambda - 1)/\lambda = \log(x)$. Prove it! **Hint:** Use l'Hôpital's rule. To differentiate something like x^λ with respect to λ it is useful to write it as $e^{\log(x)\lambda}$.