Time Series Analysis & Forecasting

Class 9

Arnab Bose, Ph.D.

MSc Analytics

University of Chicago

Cointegration

Regression of a non-stationary TS on another non-stationary TS

$$Y_t = \beta_0 + \beta_1 X_t + u_t$$

i.e. $u_t = Y_t - \beta_0 - \beta_1 X_t$

where Y_t, X_t ~ *I(d)*

u_t ~ *I(0)*

The linear combination cancels out the stochastic trends in X_t and Y_t.

X_t and Y_t are said to be cointegrated

To check for cointegration, verify that the residuals u_t are *I(0)* or stationary

Error Correction Model

From cointegration, X_t and Y_t have a long term equilibrium relationship.

But that may get off-balance in the short term

$$\Delta Y_t = \alpha_0 + \alpha_1 \Delta X_t + \alpha_2 u_{t-1} + \varepsilon_t$$

where Δ is the first difference operator

ε_t is the random error

u_{t-1} one-period lagged error from cointegration regression

Note that $\alpha_2 < 0$ to drive short term disequilibrium to equilibrium

R code - Cointegration

```
library("quantmod", lib.loc="~/R/win-library/3.3")
getSymbols(“SPY", from="2013-01-01", to="2013-12-31")
getSymbols(“IVV", from="2013-01-01", to="2013-12-31")
spyAdj <- unclass(SPY$SPY.Adjusted)
ivvAdj <- unclass(IVV$IVV.Adjusted)
adf.test(spyAdj)
adf.test(ivvAdj)
kpss.test(spyAdj, null="T")
kpss.test(ivvAdj, null="T")
ivv_spy <-lm(ivvAdj ~ spyAdj + 0)
regB <-lm(ewcAdj ~ ewaAdj + 0)
adf.test(ivv_spy$residuals)
adf.test(spy_ivv$residuals)
kpss.test(ivv_spy$residuals, null="T")
kpss.test(spy_ivv$residuals, null="T")
coef(ivv_spy)[1] * coef(spy_ivv)[1]#should be close to 1
```

Granger Causality Test

Two TS X_t and Y_t - determine which one is useful in forecasting the other.

Use this test to determine G-causality

H_0: X_t does not Granger-cause Y_t

2 regressions

$$Y_t = a_0 + a_1 Y_{t-1} + .. + a_m Y_{t-m} + residual$$

$$Y_t = a_0 + a_1 Y_{t-1} + .. + a_m Y_{t-m} + b_p X_{t-p} + .. + b_q X_{t-q}$$

Use F-test whose null hypotheses is no explanatory power is added by X_t

Note if TS are cointegrated, then there must be a G-causality between them – either one-way or in both directions

R code – Granger Causality Test

```
library("lmtest", lib.loc="~/R/win-library/3.3")

grangertest(ChickEgg[, 1], ChickEgg[, 2], order = 3)
```

Frequency Domain Representation

Frequency domain of a stationary TS representation

$$Y_t = \sum_{k=1}^{T}[a_k \sin(2\pi f_k t) + b_k \cos(2\pi f_k t)]$$

where

$$f_k = \frac{k}{T} \text{ k is the \# of harmonics (Fourier frequencies)}$$

$$a_k = \frac{2}{T}\sum_{k=1}^{T}[\cos(2\pi f_k t)]$$

$$b_k = \frac{2}{T}\sum_{k=1}^{T}[\sin(2\pi f_k t)]$$

The auto-covariance is given by

$$\gamma_k = \sum_{j=1}^{T} \sigma_j^2 \cos(2\pi f_j t)$$

And the periodogram is given by

$$I(f_k) = \frac{T}{2}\left(a_k^2 + b_k^2\right)$$

Frequency Domain Representation

The periodogram is given by

$$I(f_k) = \frac{T}{2}\left(a_k^2 + b_k^2\right)$$

- The periodogram is quickly computed using Fourier transform and is a "rough" estimate of the spectral density. Conversely, the periodogram is smoothed and scaled to produce the spectrum of the spectral density function.
- Generally if the frequency $f_k = \frac{k}{T}$ is important (not), then $I(f_k)$will be large (small). The height of the periodogram shows the relative strength of sine-cosine pairs at various frequencies in the overall behavior of the TS.
- It can be shown that

$$\sum_{k=1}^{T} [I(f_k)] = \sigma^2$$

R code – Frequency Domain Representation

```
library(“forecast", lib.loc="~/R/win-library/3.3")
library("xts", lib.loc="~/R/win-library/3.3")
library("TSA", lib.loc="~/R/win-library/3.3")
data("USAccDeaths")
plot(as.xts(USAccDeaths), major.format = "%y-%m")
p <- periodogram(USAccDeaths)
P
max_freq <- p$freq[which.max(p$spec)]
seasonality <- 1/max_freq
seasonality

# white noise
periodogram(rnorm(1000))
```

R code – Frequency Domain Representation

```
# AR  & MA models with positive and negative coefficients
par(mfrow=c(2,2))

arPosHi <- arima.sim(list(order=c(1,0,0), ar=0.9), n=100)
arPosLo <- arima.sim(list(order=c(1,0,0), ar=0.09), n=100)
arNegHi <- arima.sim(list(order=c(1,0,0), ar=-0.9), n=100)
arNegLo <- arima.sim(list(order=c(1,0,0), ar=-0.09), n=100)
periodogram(arPosHi); periodogram(arPosLo); periodogram(arNegHi);periodogram(arNegLo)

maPosHi <- arima.sim(list(order=c(0,0,1), ma=0.9), n=100)
maPosLo <- arima.sim(list(order=c(0,0,1), ma=0.09), n=100)
maNegHi <- arima.sim(list(order=c(0,0,1), ma=-0.9), n=100)
maNegLo <- arima.sim(list(order=c(0,0,1), ma=-0.09), n=100)
periodogram(maPosHi);periodogram(maPosLo);periodogram(maNegHi);periodogram(maNegLo)
```

Multiple Seasonality Modeling using TBATS

The TBATS model was introduced by De Livera, Hyndman & Snyder (2011, JASA). It is a generalization of the Holt-Winters model.

"TBATS" is an acronym denoting its salient features:

T for trigonometric regressors to model multiple-seasonalities

B for Box-Cox transformations

A for ARMA errors

T for trend

S for seasonality

The TBATS model is a generalization of the BATS model, which is similar except for lacking the trigonometric regressors.

The trigonometric output includes the periodicity and the number of harmonics/pairs for the time series.

The TBATS model can be fitted using the tbats() command in the forecast package for R.

R code – Multiple Seasonality Modeling using TBATS

data(taylor)

 # Taylor was defined taking the half hour electricity demand TS – msts is part of forecast pkg

taylor <- msts(x, seasonal.periods=c(24 * 2, 24 * 2 * 7))

plot(taylor)

model <- tbats(taylor)

comp <- tbats.components(model)

plot(comp)

plot(forecast(model, h=100))

Interpret TBATS Output in R

TBATS(0.999, {2,2}, 1, {<52.18,8>})*

Box-Cox transformation of 0.999 (essentially doing nothing)

ARMA(2,2) errors,

Box-Cox damping parameter of 1 (doing nothing)

Seasonality modeled using 8 Fourier harmonics/pairs with period m=52.18

$$
\begin{aligned}
y_t &= \ell_{t-1} + b_{t-1} + s_{t-1} + \alpha d_t \\
b_t &= b_{t-1} + \beta d_t \\
s_t &= \sum_{j=1}^{8} s_{j,t} \\
s_{j,t} &= s_{j,t-1} \cos\left(\frac{2\pi jt}{52.18}\right) + s^*_{j,t-1} \sin\left(\frac{2\pi jt}{52.18}\right) + \gamma_1 d_t \\
s^*_{j,t} &= -s_{j,t-1} \sin\left(\frac{2\pi jt}{52.18}\right) + s^*_{j,t-1} \cos\left(\frac{2\pi jt}{52.18}\right) + \gamma_2 d_t,
\end{aligned}
$$

where d_t is an ARMA(2,2) process and α, β, γ_1 and γ_2 are smoothing parameters. Here the seasonality has been handled with 18 parameters (the sixteen initial values for $s_{j,0}$ and $s^*_{j,0}$ and the two smoothing parameters γ_1 and γ_2). The total number of degrees of freedom is 26 (the other 8 coming from the two smoothing parameters α and β, the four ARMA parameters, and the initial level and slope values ℓ_0 and b_0).

* https://robjhyndman.com/hyndsight/forecasting-weekly-data/, Accessed May 2018

Polynomial and Non-linear Regression

Polynomial regression

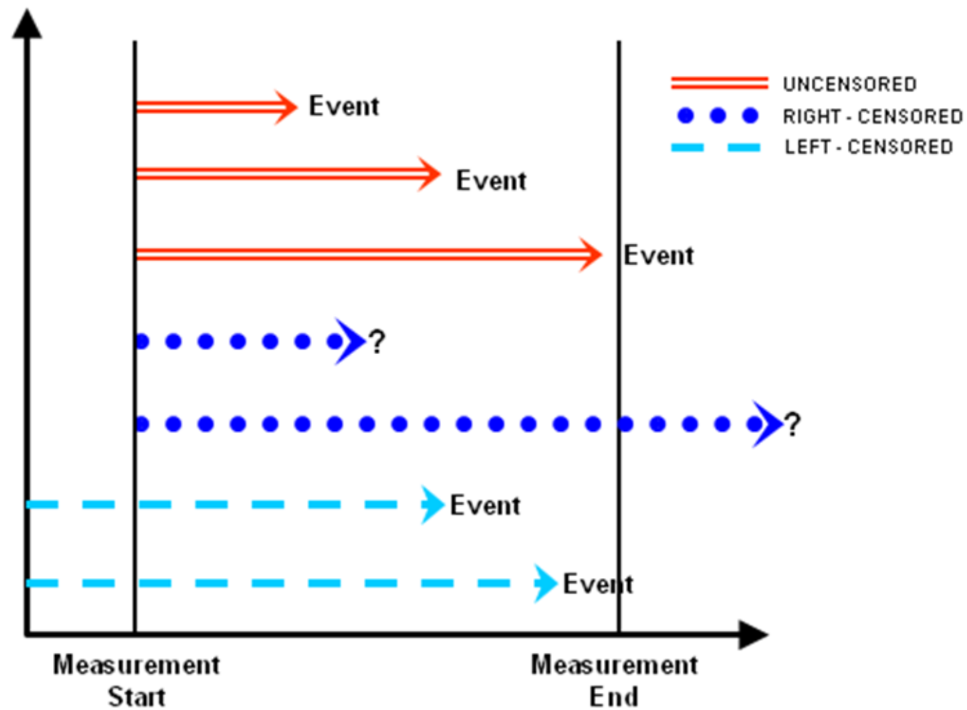
$$Y_t = \beta_0 + \beta_1 X_t + \beta_2 X_t^2 + u_t$$

The squared is represented directly in R using $I(.)$

Non-linear regression

$$Y_t = a - be^{-cX_t}$$

Survival Regression – Data censoring for time-to-event



Survival Regression – Cox Regression

The Cox model is expressed by the hazard function denoted by h(t) that can be interpreted as the risk of dying at time t and estimated as follows:

$$h(t) = h_0(t) * \exp(b_1 x_1 + b_2 x_2 + \cdots + b_p x_p)$$

where,

t represents the survival time

$h(t)$ is the hazard function determined by a set of p covariates the coefficients $(x_1, x_2, \ldots, x_p)$

measure the impact (i.e., the effect size) of covariates

$h_0(t)$ is the baseline hazard. It corresponds to the value of the hazard if all the covariates are equal to zero (the quantity exp(0) equals 1).

The ‘t’ in $h(t)$ reminds us that the hazard may vary over time.

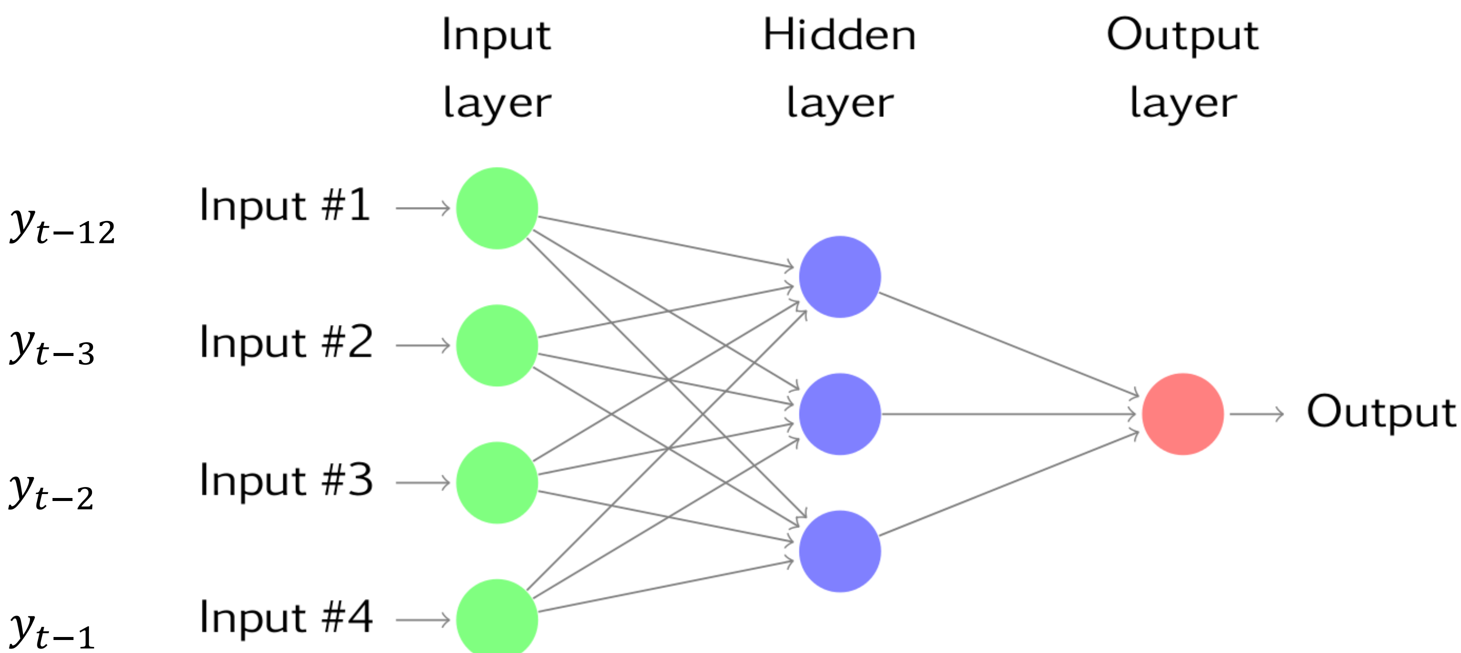
http://www.sthda.com/english/wiki/cox-proportional-hazards-model

R code – Survival Regression

```r
library("survival", lib.loc="~/R/win-library/3.6")
data("lung")
uni.cox <- coxph(Surv(time, status) ~ sex, data = lung)
mv.cox <- coxph(Surv(time, status) ~ age + sex + ph.ecog, data =  lung)
summary(mv.cox)
```

Neural Networks for Time Series

- Fully connected Neural Network with 1 hidden layer
- NNAR $(3,1,2)_{12}$ packages: forecast (nnetar) and nnfor (elm and mlp)



https://otexts.org/fpp2/nnetar.html#fig:nnet2

R code – Neural Network Forecasting (nnfor)

```
library("nnfor", lib.loc="~/R/win-library/3.3")

?elm

 fit <- elm(AirPassengers, hd=10)

 print(fit)

 plot(fit)

 frc <- forecast(fit,h=36)

 plot(frc)


?mlp

fit2 <- mlp(AirPassengers, hd = c(10,5))
```

two hidden layers, one with 10 and one with 5

https://kourentzes.com/forecasting/2019/01/16/tutorial-for-the-nnfor-r-package/

Recurrent Neural Network (RNN)

- RNNs address the temporal relationship of their inputs by maintaining an internal state.
- RNNs are biased towards learning patterns which occur in temporal order – i.e. they are less prone to learning random correlations which do not occur in temporal order.
- In theory, RNNs are absolutely capable of handling “long-term dependencies.” But in practice, RNNs suffer from vanishing gradient problem.
- Long Short Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies*.

* http://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed Nov, 2016.

Textbook Chapters

- Materials covered available in book:
  - MJK: Chapter 7, TSA: Chapter 13

- R package nnfor: https://github.com/trnnick/nnfor/blob/master/R/