Time Series Analysis & Forecasting

Class 8

Arnab Bose, Ph.D.

MSc Analytics

University of Chicago

ARIMA Transfer Function Model

ARIMA Process

$$\phi(B)Y_t = \theta(B)e_t$$

$$Y_t = \phi^{-1}(B)\theta(B)e_t = \frac{1 - \theta_1 B - \cdots - \theta_q B^q}{1 - \emptyset_1 B - \cdots - \emptyset_p B^p} e_t$$

- TS is represented as output of a dynamic system where the input is white noise
- Transfer function is parsimoniously represented as a ratio of 2 polynomials in *B*

SISO Transfer Function Model

- Assume X_t and Y_t are stationary TS – in a single input single output system they are related through a linear filter:

$$Y_t = \vartheta(B)X_t + \eta_t$$

where η_t is the noise

$\vartheta(B)$ is the transfer function and

$$\vartheta(B) = \sum_{j=0}^{\propto} v_j B^j, \sum_{j=0}^{\propto} |v_j| <\propto$$

- The TF $\vartheta(B)$ may contain an infinite number of coefficients, so represent in the rational form:

$$\vartheta(B) = \frac{\omega(B)B^b}{\delta(B)}$$

SISO Transfer Function Model

The TF $\vartheta(B)$ may contain an infinite number of coefficients, so represent in the rational form:

$$\vartheta(B) = \frac{\omega(B)B^b}{\delta(B)}$$

Where $\omega(B) = \omega_0 - \omega_1 B - \cdots - \omega_s B^s$

$\delta(B) = 1 - \delta_1 B - \cdots - \delta_r B^r$

b is a delay parameter representing actual time lag before impulse of the input variable produces effect on the output variable

Intervention Analysis

Framework to study effect of intervention on a TS

$$Y_t = m_t + N_t$$

where m_t : change in the mean function

N_t : ARIMA process without intervention

2 Types of Intervention at time *T*:

Step function $$S_t^{(T)} = \begin{cases} 1, & t \geq T \\ 0, & otherwise \end{cases}$$

Pulse function $$P_t^{(T)} = \begin{cases} 1, & t = T \\ 0, & otherwise \end{cases}$$

Intervention causes change in mean defined by Type equation here.

Intervention Analysis

Intervention causes change in mean defined by

$$m_t = \omega S_t^{(T)}$$

Where ω is the unknown permanent change due to the intervention

Intervention with delay of *d* time units is defined by

$$m_t = \omega S_{t-d}^{(T)}$$

Intervention may affect the mean function gradually and can be modeled as AR(1)

$$m_t = \delta m_{t-1} + \omega S_{t-1}^{(T)}$$

Intervention Analysis

Intervention may affect the mean function gradually and can be modeled as AR(1)

$$m_t = \delta m_{t-1} + \omega S_{t-1}^{(T)}$$

$$m_t = \begin{cases} \omega \dfrac{1-\delta^{t-T}}{1-\delta}, & t > T \\ 0, & otherwise \end{cases}$$

Usually $0 < \delta < 1$ so that the ultimate change in mean for large t is

$$m_t = \frac{\omega}{1-\delta}$$

Intervention Analysis

Likewise short lived intervention is specified as

$$m_t = \delta m_{t-1} + \omega P_{t-1}^{(T)}$$

$$\Rightarrow m_t = \delta B m_t + \omega B P_t^{(T)}$$

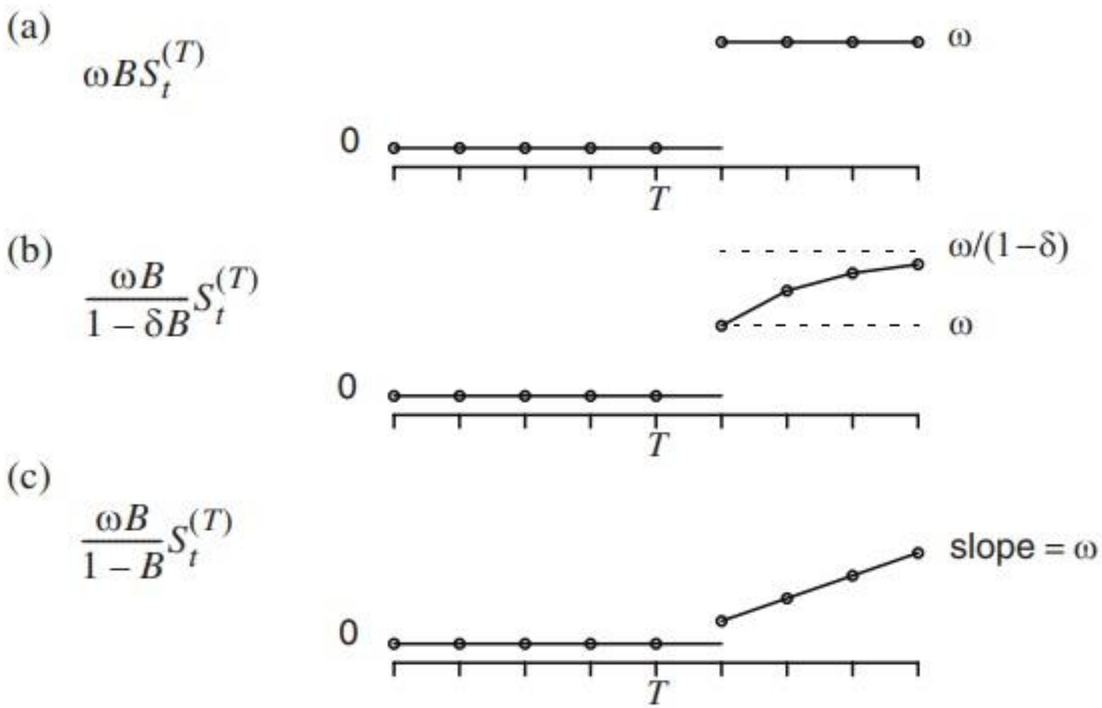
$$\Rightarrow m_t = \frac{\omega B}{1 - \delta B} P_t^{(T)}$$

Note that

$$S_t^{(T)} = \frac{1}{1 - B} P_t^{(T)}$$

Step Interventions

Exhibit 11.3 Some Common Models for Step Response Interventions (All are shown with a delay of 1 time unit)

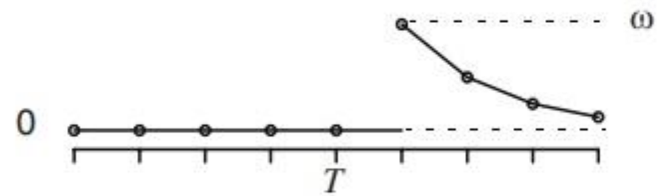


Source: TSA, page 253

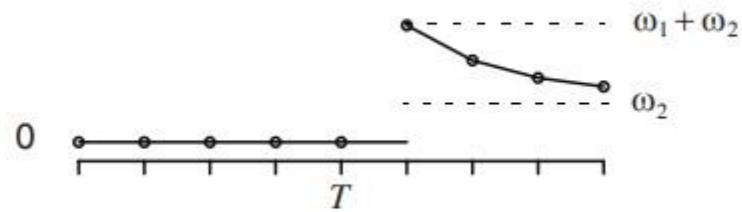
Pulse Interventions

Exhibit 11.4 Some Common Models for Pulse Response Interventions (All are shown with a delay of 1 time unit)

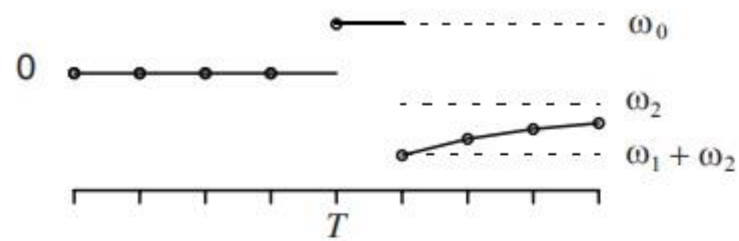
(a) $\dfrac{\omega B}{1-\delta B}P_t^{(T)}$



(b) $\left[\dfrac{\omega_1 B}{1-\delta B}+\dfrac{\omega_2 B}{1-B}\right]P_t^{(T)}$



(c) $\left[\omega_0+\dfrac{\omega_1 B}{1-\delta B}+\dfrac{\omega_2 B}{1-B}\right]P_t^{(T)}$



Source: TSA, page 254

R code – Intervention Analysis

```
library("TSA")

data(airmiles)

airmiles.pre.intervention <- window(airmiles, end=c(2001,8))

auto.arima(airmiles.pre.intervention)


P911 <- 1*(seq(airmiles)==69)

S911 <- 1*(seq(airmiles)>=69)


air.mPulse <-
arimax(log(airmiles),order=c(0,1,1),seasonal=list(order=c(0,1,1),
period=12), xtransf=data.frame(P911, P911), transfer=list(c(0,0),c(1,0)),
method='ML')

air.mPulse


plot(ts(filter(P911, filter=0.8901, method='recursive', side=1)*(-0.2419),
frequency = 12, start=1996), type='h',ylab='9/11 Pulse Effects')
```

R code – Intervention Analysis

```
library("TSA")

air.mStep <-
arimax(log(airmiles),order=c(0,1,1),seasonal=list(order=c(0,1,1),
period=12), xtransf=data.frame(S911, S911),
transfer=list(c(0,0),c(1,0)), method='ML')

air.mStep


plot(ts(S911*(2.2415)+filter(S911, filter=-0.0618, method='recursive',
side=1)*(-2.6169), frequency = 12, start=1996), type='h',ylab='9/11 Step
Effects')
```

Forecasting using arimax( )

1. arimax( ) does not implement a predict( ) function => cannot use forecast( )
2. Workaround from https://stats.stackexchange.com/questions/169564/arimax-prediction-using-forecast-package :
   1. Use the forecast::Arima( ) to determine pre-intervention noise series + add any outlier adjustment.
   2. Fit the same model in arimax but add the transfer function.
   3. Take the fitted values for the transfer function (coefficients from arimax) and add them as xreg in Arima.
   4. Forecast with Arima.

R code – arimax( ) forecast

```
steps.ahead = 5

air.m1<-arimax(log(airmiles), order=c(0,1,1), seasonal = c(0,1,1),
xtransf=data.frame(I911=1*(seq(airmiles)==69)), transfer=list(c(1,0)))

tf<-filter(1*(seq(1:(length(airmiles) + steps.ahead))==69), filter=0.6948,
method='recursive',side=1) * (-0.3459)

forecast.arima<-Arima(log(airmiles), order=c(0,1,1), seasonal = c(0,1,1),
xreg=tf[1:(length(tf) - steps.ahead)])

forecast.arima

predict(forecast.arima, n.ahead = 5, newxreg=tf[length(tf) - steps.ahead +
1:length(tf)])
```

Outlier Analysis – General

tsoutliers and tsclean: https://robjhyndman.com/hyndsight/forecast5/

```
y<-c(0.59, 0.61, 0.59, 1.55, 1.33, 3.50, 1.00, 1.22, 2.50, 3.00, 3.79,
3.98, 4.33, 4.45, 4.59, 4.72, 4.82, 4.90, 4.96, 7.92, 5.01, 5.01, 4.94,
5.05, 5.04, 5.03, 5.06, 5.10, 5.04, 5.06, 7.77, 5.07, 5.08, 5.08, 5.12,
5.12, 5.08, 5.17, 5.18)

ts.plot(y)

tsoutliers(y)


y_clean <- tsclean(y)

lines(y_clean, col='red')
```

Outliers are computed if the data lies outside of the following upper U and lower bounds L :

$$U = q_{0.9} + 2 * (q_{0.9} - q_{0.1})$$

$$L = q_{0.9} - 2 * (q_{0.9} - q_{0.1})$$

Where $q_{0.1}$ and $q_{0.9}$ are 10[th] and 90[th] percentiles of the data, respectively

Outlier Analysis – Detailed

Outliers happen due to measurement and/or copying errors or abrupt changes to the underlying process

2 Types of Outliers:

Additive Outliers

$$Y_t' = Y_t + \omega_A P_t^{(T)}$$

where Y_t is the unperturbed TS

Innovative Outliers occurs at time *t* if the error is perturbed

$$e_t' = e_t + \omega_I P_t^{(T)}$$

R code – Additive Outlier

```
library("TSA")
set.seed(12345)
y <- arima.sim(model=list(ar=0.8, ma=0.5), n.start=158, n=100)
y[10]
ts.plot(y)
y[10] <- 10
ts.plot(y)
acf(y)
pacf(y)
eacf(y)
m1 <- Arima(y, order=c(1,0,0))
m1
detectAO(m1)
```

R code – Innovation Outlier

```
# co2 example from TSA

data(co2)

m1.co2 <- Arima(co2, order=c(0,1,1), seasonal=list(order=c(0,1,1),
period=12))

m1.co2

detectAO(m1.co2)

detectIO(m1.co2)

m1.co2.io <- arimax(co2, order=c(0,1,1), seasonal=list(order=c(0,1,1),
period=12),io=c(57))

acf(m1.co2$residuals)

acf(m1.co2.io$residuals)
```

Twitter's Open Source AnomalyDetection Package

1. This package implements an elaborate on the Generalized ESD (Extreme Student Deviant) (refer to https://www.itl.nist.gov/div898/handbook/eda/section3/eda35h3.htm).

2. The Generalized ESD is built on a statistical test called the Grubbs test (https://www.itl.nist.gov/div898/handbook/eda/section3/eda35h1.htm).

3. The Generalized ESD tests for multiple outliers.

4. This package implements a Seasonal Hybrid ESD built on the Generalized ESD to account for seasonality via time series decomposition.

https://github.com/hrbrmstr/AnomalyDetection

R code – Twitter's AnomalyDetection

```
# follow instructions at https://github.com/hrbrmstr/AnomalyDetection

library(AnomalyDetection)
library(ggplot2)
data(raw_data)
res <- ad_ts(raw_data, max_anoms=0.02, direction='both')

ggplot() +
  geom_line(
    data=raw_data, aes(timestamp, count),
    size=0.125, color="lightslategray"
  )  +
  geom_point(
    data=res, aes(timestamp, anoms), color="#cb181d", alpha=1/3
  ) +
  scale_x_datetime(date_labels="%b\n%Y")
```

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

$f_k = \frac{k}{T}$ 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 – Fourier Transform

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 is $f_k = \frac{k}{T}$ (or not), then $I(f_k)$ will be large (or 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 the sum of the periodogram is the variance of the TS.

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

https://scitechdaily.com/fourier-transformations-reveal-how-ai-learns-complex-physics/

R code – Frequency Domain Representation

```
library(forecast)
library(xts)
library(TSA)
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
library(TSA)
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);periodogr
am(maNegLo)
```

Multiple Seasonality Modeling using auto.arima

1. Fit a dynamic harmonic regression model with ARMA errors
2. Pass Fourier terms for each seasonal period to xreg
3. **Advantage** – allows for covariates
4. **Disadvantage** – seasonality cannot change over time

R code – Multiple Seasonality Modeling using auto.arima

```
deaths.model <- auto.arima(USAccDeaths, xreg=fourier(USAccDeaths,K=6),
seasonal=FALSE)

deaths.fcast <- forecast(deaths.model, xreg=fourier(USAccDeaths, K=6,
h=36))

autoplot(deaths.fcast)
```

Note that arima works for (even for single seasonality) short seasonal periods such as 12 for monthly, 52 for weekly, 4 for quarterly, and 24 or 48 for hourly or half-hourly, respectively (https://robjhyndman.com/hyndsight/seasonal-periods/).

For seasonal periods > 350, arima runs out-of-memory so use the above Fourier series method. Details are at https://robjhyndman.com/hyndsight/longseasonality/.

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. It is a generalization of the BATS model with the trigonometric regressors.

"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 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.

Advantage – seasonality can change slowly over time

Disadvantage – no covariates

R code – Multiple Seasonality Modeling using TBATS

```
data(taylor)

# Taylor was defined taking the half hour electricity demand TS - msts i
s 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

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

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

Textbook Chapters

Materials covered available in book chapters:

FPP: 4

Thank You


Even the genius ask questions
- Tupac Shakur
www.brainquotes.org
Genius Ask Questions Quotes