Time Series Analysis & Forecasting

Class 3

Arnab Bose, Ph.D.

MSc Analytics

University of Chicago

Autoregressive (AR)

- Current value of a process is expressed as a finite, linear aggregate of the pervious values of the process and white noise $\{e_t\}$

- Mathematically AR(p): $Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p} + e_t$

- Define operator $\Phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \ldots . - \phi_p B^p$

- Mathematically the TS can be represented as

$$\Phi(B) Y_t = e_t$$

AR Stationarity

- Mathematically the TS can be represented as

$$\Phi(B)Y_t = e_t$$

$$Y_t = \Phi^{-1}(B)e_t = \Psi(B)e_t$$

- For stationarity, $\Psi(B)$ needs to be a convergent series

- In other words, all roots of $\Phi(B) = 0$ must be greater than 1 in absolute value

Moving Average (MA)

- Current value of a process depends on a finite number of white noise $\{e_t\}$
- Mathematically MA(q): $Y_t = e_t - \theta_1 e_{t-1} - \theta_2 e_{t-2} + \ldots + \theta_q e_{t-q}$
- Define operator $\Theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \ldots . - \theta_q B^q$
- Mathematically the TS can be represented as

$$Y_t = \Theta(B) e_t$$

MA Invertibility

- MA process has the characteristic polynomial

$$\Theta(x) = 1 - \theta_1 x - \theta_2 x^2 - \ \ldots - \theta_q x^q$$

- And the characteristic equation

$$1 - \theta_1 x - \theta_2 x^2 - \ \ldots - \theta_q x^q = 0$$

- MA(q) is invertible if and only if the roots of the characteristic equation > 1

The “Moving Average” name

It is a bit misleading because the weights $\theta_1, \ \theta_2, \ldots, \theta_q$

- May not total/sum to 1
- May not all be positive > 0

Note that infinite models (as in Wold representation) are fine in theory, in practice the application is a finite order model.

For example, a finite order MA model where except for the first θ_q weights, the rest of the weights are set to 0.

Mixed Autoregressive Moving Average Model

- Assume TS is partly AR(p) and partly MA(q)

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \ldots + \phi_p Y_{t-p} + e_t - \theta_1 e_{t-1} - \theta_2 e_{t-2} + \ldots + \theta_q e_{t-q}$$

- ARMA(p,q) mathematically represented as

$$\Phi(B) Y_t = \Theta(B) e_t$$

- ARMA(p,q) require both stationarity and invertibility

Models for Non-stationary TS

- Consider a Random Walk model $Y_t = Y_{t-1} + w_t$
- Take first difference $Y_t - Y_{t-1} = w_t$ => gives white noise that is stationary
- Represented as $\nabla Y_t = w_t$
- d^{th} difference to get process stationarity and then apply ARMA(p,q) process => ARIMA
- ARIMA(p,d,q) mathematically represented as

$$\Phi(B)(1-B)^d Y_t = \Theta(B)e_t$$

R code – auto.arima with Box-Cox

```
lambda <- BoxCox.lambda(lynx)

lynx.fit.arima.boxcox <- auto.arima(lynx, lambda=lambda) # Using Box-Cox
transformation

plot(forecast(lynx.fit.arima.boxcox, h=20))


# note if you do the Box-Cox separately, then forecast does not invert
it

lambda <- BoxCox.lambda(lynx)

lynx.fit.arima <- auto.arima(BoxCox(lynx, lambda))

plot(forecast(lynx.fit.arima, h=20))
```

auto.arima: Hyndman-Khandakar Algo + Manual Model Fitting

Refer to https://otexts.com/fpp2/arima-r.html

Note that **manual model fitting** can be done with forecast::Arima as follows.

1. Start with the automated model fitting (auto.arima) as a baseline.

2. Try different values for the orders ***p*** and ***q*** to improve model metrics.

Popular ARIMA models (with zero mean, i.e. no constant)

1. ARIMA(0, 0, 0) => White noise
2. ARIMA(0, 1, 0) => Random walk with no drift
3. ARIMA(0,1,1) => simple exponential smoothing
4. ARIMA(0, 2, 2) => linear exponential smoothing

R code

```
arma11 <- arima.sim(list(order=c(1,0,1), ar=0.75, ma=0.9), n=100)
est_mod <- auto.arima(arma11)
summary(est_mod)

mod1 <- Arima(arma11, order = c(2,0,1))
summary(mod1)
```

forecast::Arima vs stats::arima – Time Series with Drift

```
set.seed(1)            # so you can reproduce the results
v = rnorm(100,1,1)     # v contains 100 iid N(1,1) variates
x = cumsum(v)          # x is a random walk with drift = 1
plot(x)
model1 <- arima(x, order = c(1,1,0))
plot(forecast(model1, h=12))
kpss.test(x) # not stationary due to drift
kpss.test(x, null = "Trend") # cannot detect drift, only deterministic trend

# same effect as it does not consider drift term
model2 <- Arima(x, order = c(1,1,0))
plot(forecast(model2, h=12))

# correct results with drift included
model3 <- Arima(x, order = c(1,1,0), include.drift = T)
plot(forecast(model3, h=12))
# Refer to   https://www.stat.pitt.edu/stoffer/tsa4/Rissues.htm
```

Model Specification

- For MA(q) TS, autocorrelation $\rho_k = 0, k > q$

- For AR(p) TS, **partial autocorrelation** $\phi_{kk} = 0, k > q$

- For ARMA, neither acf nor pacf has a distrinct pattern, so use *eacf()*

Partial autocorrelation function *– defined as the correlation between* y_t *and* y_{t-k} *after removing the effects of the intervening variables* $y_{t-1,}\, y_{t-2}, \ldots, y_{t-k+1}$

Model Specification acf and pacf

Exhibit 6.3 General Behavior of the ACF and PACF for ARMA Models

	AR(p)	MA(q)	ARMA(p, q), $p>0$, and $q>0$
ACF	Tails off	Cuts off after lag q	Tails off
PACF	Cuts off after lag p	Tails off	Tails off

Cryer, Jonathan D. and Kung-Sik Chan. *Time Series Analysis with Applications in R*. 2nd Edition. Springer, 2008.

Model Specification eacf

Exhibit 6.4 Theoretical Extended ACF (EACF) for an ARMA(1,1) Model

AR/MA	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1	x	0*	0	0	0	0	0	0	0	0	0	0	0	0
2	x	x	0	0	0	0	0	0	0	0	0	0	0	0
3	x	x	x	0	0	0	0	0	0	0	0	0	0	0
4	x	x	x	x	0	0	0	0	0	0	0	0	0	0
5	x	x	x	x	x	0	0	0	0	0	0	0	0	0
6	x	x	x	x	x	x	0	0	0	0	0	0	0	0
7	x	x	x	x	x	x	x	0	0	0	0	0	0	0

Cryer, Jonathan D. and Kung-Sik Chan. *Time Series Analysis with Applications in R*. 2nd Edition. Springer, 2008.

R code – arima with acf and pacf

```
ar1 <- arima.sim(list(order=c(1,0,0), ar=0.75), n=100)
acf(ar1)
pacf(ar1)

ma1 <- arima.sim(list(order=c(0,0,1), ma=0.9), n=100)
acf(ma1)
pacf(ma1)
```

R code

```
arma11 <- arima.sim(list(order=c(1,0,1), ar=0.75, ma=0.9), n=100)

acf(arma11)

pacf(arma11)

eacf(arma11)


arma21 <- arima.sim(list(order=c(2,0,1), ar=c(0.9, -0.75), ma=0.9),
n=100)

acf(arma21)

pacf(arma21)

eacf(arma21)
```

R code – simulate arima

```
ar <- arima.sim(list(order=c(1,0,0), ar=0.75), n=100)

plot(ar)

mean(ar)


ar_m <- arima.sim(list(order=c(1,0,0), ar=0.75), n=100) + 10 # adding mean to AR process

plot(ar_m)

mean(ar_m)


ar_im <- arima.sim(list(order=c(1,0,0), ar=0.75), n=100, mean = 10) # adding mean to innovation (error)

plot(ar_im)

mean(ar_im)



ma_im <- arima.sim(list(order=c(0,0,1), ma=0.9), n=100, mean = 10) # adding mean to innovation (error)

plot(ma_im)

mean(ma_im)
```

Model Estimation

- TS $\{y_t\}$
- Maximum Likelihood (ML) is estimating the unknown parameters such that the probability of observing the given $\{y_t\}$ is as high as possible
- Use differential calculus to solve for the parameters and get the maximum value
- t-value given by

$$t = \frac{\hat{\beta} - \beta}{std\_error(\beta)} = \frac{\hat{\beta} - \beta}{\frac{\hat{\sigma}}{\sqrt{\sum x_i^2}}}$$

- ARIMA models include any parameter whose |t-value| > 2

Yule-Walker Equations

- Consider AR(2) process

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + e_t$$

- Multiply both sides with Y_{t-k} and take expectations to get autocovariance

$$\gamma_k = \phi_1 \gamma_{k-1} + \phi_2 \gamma_{k-2}, for\ k = 1, 2, 3, \ldots$$

And dividing by γ_0 we get autocorrelation defined by

$$\rho_k = \phi_1 \rho_{k-1} + \phi_2 \rho_{k-2}$$

- The above autocovariance and autocorrelation equations are called the Yule-Walker equations.
- Use sample moments to recursively solve for AR parameter estimates.

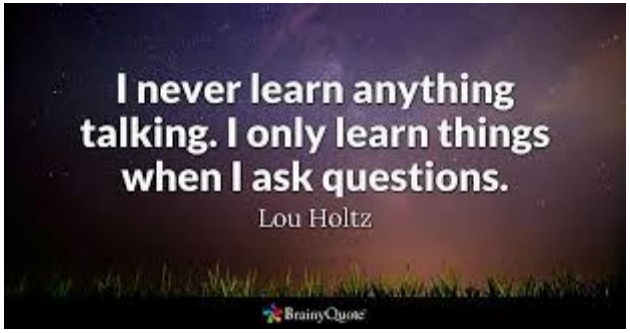
Textbook Chapters

Materials covered available in book chapters:

FPP: 9

PTS: 6

Thank You


I never learn anything talking. I only learn things when I ask questions.
Lou Holtz
BrainyQuote