QBUS6840: Tutorial 10 – SARIMA models

Objectives

- Analyze MA and ARMA models
- Interpret ACF/PACF plots
- Fit ARMA models to observed data
- Develop Python skills

This tutorial follows the procedure outlined at https://www.otexts.org/fpp/8/7
For more detailed mathematical explanation, please also refer to https://otexts.com/fpp2/seasonal-arima.html

Continue with the previous tutorial document, we will discuss the Seasonal ARIMA model. SARIMA models are denoted SARIMA(p,d,q)(P,D,Q)$_m$, where m refers to the number of periods in each season.

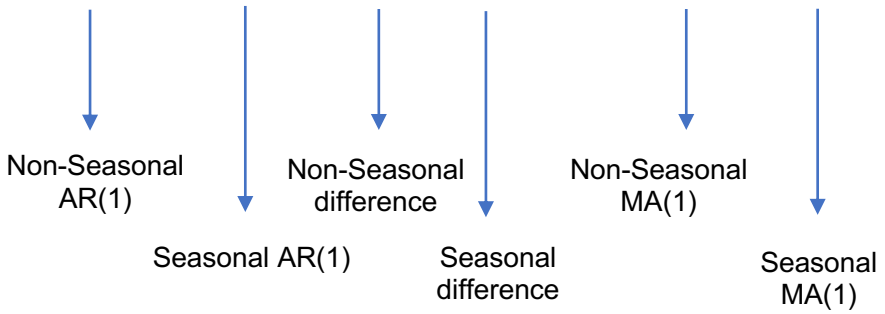
Since SARIMA model is complex to write out directly, we normally use a backshift operator to describe the entire process. For example SARIMA(1,1,1)(1,1,1)$_4$ without a constant is written as:

$$(1-\phi_1 B)\left(1-\boldsymbol{\Phi}_1 B^4\right)(1-B)(1-B^4)y_t = (1+\theta_1 B)(1+\Theta_1 B^4)\varepsilon_t$$

Can you identify what is the meaning of each factor in the above equation?

For

$$(1-\phi_1 B)\left(1-\boldsymbol{\Phi}_1 B^4\right)(1-B)(1-B^4)y_t = (1+\theta_1 B)(1+\Theta_1 B^4)\varepsilon_t$$



Why this matter?

If m=12, can you rewrite the backshift representation for SARIMA(0,0,1)(0,0,1)$_{12}$?

The non-seasonal MA(1) is $\theta(B) = (1+\theta_1 B)$
The seasonal MA(1) is $\Theta(B^{12}) = (1+\Theta_1 B^{12})$
Therefore, the model is $y_t = \theta(B)\,\Theta(B^{12})\varepsilon_t = (1+\theta_1 B)\,(1+\Theta_1 B^{12})\varepsilon_t$

When we multiply the 2 polynomials on the right side, we get:

$$y_t = (1+\theta_1 B + \Theta_1 B^{12} + \theta_1\Theta_1 B^{13})\varepsilon_t$$

$$y_t = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \Theta_1 \varepsilon_{t-12} + \theta_1 \Theta_1 \varepsilon_{t-13}$$

Conclusion1: the model has MA terms at lags 1, 12, and 13.

Task1: Observe the ACF curve for SARIMA(0,0,1)(0,0,1)$_{12}$

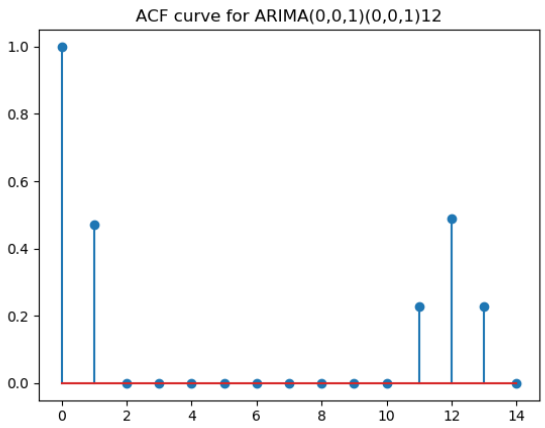
Step1. Set parameters for MA(1) and Seasonal MA(1)

For example, we set $\theta_1 = 0.7$ for MA(1) and $\Theta_1 = 0.8$ for Seasonal MA(1). Therefore, the weight for ε_{t-13} is equal to 0.7*0.8.

```
import matplotlib.pyplot as plt
import statsmodels as sm
import numpy as np

maparams = np.array([0.7, 0,0,0,0,0,0,0,0,0,0, 0.8,
0.8*0.7])
ma = np.r_[1, maparams]   # add zero-lag
zero_lag = np.array([1])
```

Step 2. Define the model and plot the ACF curve

```
ma_model = sm.tsa.arima_process.ArmaProcess(ar =
zero_lag, ma = ma)

# Plot ACF
plt.figure()
plt.stem(ma_model.acf()[:15])
plt.title("ACF curve for SARIMA(0,0,1)(0,0,1)12")
plt.show()
```



Observe the ACF plot, how many peaks we have? What are they?

Recall that SARIMA(0,0,1)(0,0,1)$_{12}$ model is equal to MA terms at lags 1,12, and 13. This leads many to think that the identifying ACF for the model will have non-zero autocorrelations only at lags 1, 12, and 13.

(Optional:)
However, from the above ACF plot, we can see $\rho_{11} \neq 0$. Why is this?

The correlation is defined as Covariance/ product of standard deviations.

$$Corr(y_t, y_{t-11}) = Cov(y_t, y_{t-11})/E(y_t - \mu)(y_t - \mu)$$

Where

$$Cov(y_t, y_{t-11}) = E(y_t - \mu)(y_{t-11} - \mu)$$

In **Conclusion1**, we have

$$y_t = \varepsilon_t + \theta_1\varepsilon_{t-1} + \Theta_1\varepsilon_{t-12} + \theta_1\Theta_1\varepsilon_{t-13}$$
$$y_{t-11} = \varepsilon_{t-11} + \theta_1\varepsilon_{t-1-11} + \Theta_1\varepsilon_{t-12-11} + \theta_1\Theta_1\varepsilon_{t-13-11}$$

The covariance between y_t and y_{t-11} is

$$Cov(y_t, y_{t-11}) = E(\varepsilon_t + \theta_1\varepsilon_{t-1} + \Theta_1\varepsilon_{t-12} + \theta_1\Theta_1\varepsilon_{t-13} - \mu)(\varepsilon_{t-11} + \theta_1\varepsilon_{t-1-11} + \Theta_1\varepsilon_{t-12-11} + \theta_1\Theta_1\varepsilon_{t-13-11} - \mu)$$

Here, ε are independent errors. The expected value of any product involving ε's with different subscripts will be 0. A covariance between ε's with the same subscripts will be the variance of ε.

If you inspect all possible products in the above equation, there will be one product with matching subscripts. They have lag t-12. Thus this expected value (covariance) will be different from 0.

This shows that the lag 11 autocorrelation will be different from 0. If you look at the more general problem, you can find that only lags 1, 11, 12, and 13 have non-zero autocorrelations for the ARIMA (0, 0, 1)(0, 0, 1)$_{12}$.

For more details, please refer to
https://newonlinecourses.science.psu.edu/stat510/node/67/

Can you also rewrite the backshift representation for SARIMA(1,0,0)(1,0,0)$_{12}$?

The non-seasonal AR(1) is $\phi(B) = (1 - \phi_1 B)$
The seasonal AR(1) is $\boldsymbol{\Phi}(B^{12}) = (1 - \boldsymbol{\Phi}_1 B^{12})$
Therefore, the model is

$$\phi(B)\boldsymbol{\Phi}(B^{12})y_t = (1 - \phi_1 B)(1 - \boldsymbol{\Phi}_1 B^{12})y_t = \varepsilon_t$$

Thus, we will get:

$$(1 - \phi_1 B)(1 - \boldsymbol{\Phi}_1 B^{12})y_t = \varepsilon_t$$
$$(1 - \phi_1 B - \boldsymbol{\Phi}_1 B^{12} + \phi_1\boldsymbol{\Phi}_1 B^{13})y_t = \varepsilon_t$$
$$y_t - \phi_1 y_{t-1} - \boldsymbol{\Phi}_1 y_{t-12} + \phi_1\boldsymbol{\Phi}_1 y_{t-13} = \varepsilon_t$$
$$y_t = \phi_1 y_{t-1} + \boldsymbol{\Phi}_1 y_{t-12} - \phi_1\boldsymbol{\Phi}_1 y_{t-13} + \varepsilon_t$$

Conclusion2: the model has AR terms at lag 1, 12, and 13.

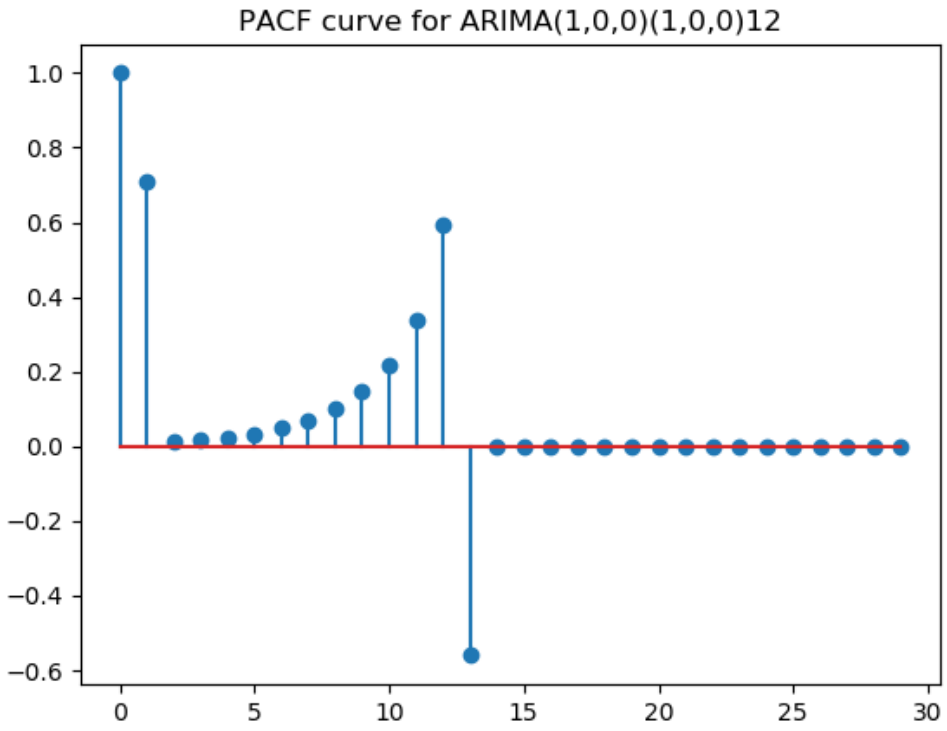
Task2: Observe the PACF curve for SARIMA(1,0,0)(1,0,0)$_{12}$
Step1. Set parameters for AR(1) and Seasonal AR(1)

For example, we set $\phi_1 = 0.7$ for AR(1) and $\boldsymbol{\Phi}_1 = 0.8$ for Seasonal AR(1). Therefore, the weight for y_{t-13} is equal to 0.7*0.8.

```
arparams = np.array([0.7, 0,0,0,0,0,0,0,0,0, 0.8,
-0.8*0.7])
ar = np.r_[1, -arparams]   # add zero-lag
```

Step 2. Define the model and plot the ACF curve

```
ar_model = sm.tsa.arima_process.ArmaProcess(ar = ar, ma =
zero_lag)

plt.figure()
plt.stem(ar_model.pacf()[:30])
plt.title("PACF curve for SARIMA(1,0,0)(1,0,0)12")
plt.show()
```



Observe the PACF plot, how many peaks we have? What are they?

There are distinct spikes at lags 1, 12, and 13 with a bit of action coming before lag 12. Then, it cuts off after lag 13.

Rules for SARIMA model selection from ACF/PACF plots

These are all rule of thumbs, not an exact science for picking the number of each parameters in SARIMA(p,d,q)(P,D,Q)$_m$. The following rules also apply to ARMA and ARIMA models.

1. Identifying the order of differencing:

d=0 if the series has no visible trend or ACF at all lags is low.
d$\geqslant$1 if the series has visible trend or positive ACF values out to a high number of lags.

Note 1: if after applying differencing to the series and the ACF at lag 1 is -0.5 or more negative the series may be over-differenced.

Note 2: If you find the best d to be d=1 then the original series has a constant trend. A model with d=2 assumes that the original series has a time-varying trend.

2. Identifying the number of AR and MA terms

p is equal to the last lag where the PACF value is above the significance level.
q is equal to the last lag where the ACF value is above the significance level.

3. Identifying the seasonal part of the model:

m is equal to the ACF lag with the highest value (typically at a high lag).
D=1 if the series has a stable seasonal pattern over time.
D=0 if the series has an unstable seasonal pattern over time.
P$\geqslant$1 if the ACF is positive at lag m, else P=0.
Q$\geqslant$1 if the ACF is negative at lag m, else Q=0.

Note 3: In most cases, d+D$\leqslant$2

Note 4: **In most cases, P+Q$\leqslant$2**

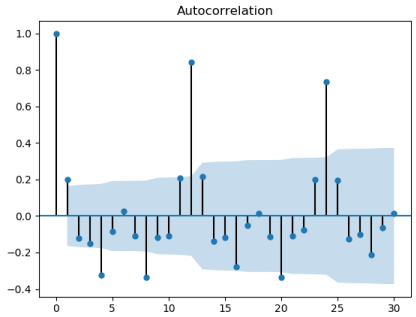
Task3: Forecasting with SARIMA
Continue with the task in previous tutorial, we will use SARIMA model to forecast the AirPassengers dataset.

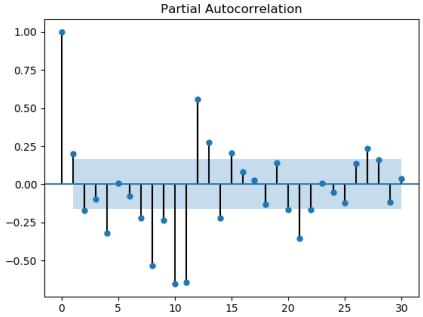
Repeat the steps in Step1-4 in Task2 of Tutorial09:

```
dateparse = lambda dates: pd.datetime.strptime(dates,
'%Y-%m')
airdata = pd.read_csv('AirPassengers.csv',
parse_dates=['Month'],
index_col='Month',date_parser=dateparse)
ts = airdata['Passengers']
plt.figure()
plt.plot(ts)
plt.title("Air passenger data")
```

```
#%% log the data
ts_log = np.log(ts)
plt.figure()
plt.plot(ts_log)
plt.title("Air passenger data (log)")

#%% take the 1st order diff
ts_log_diff = ts_log - ts_log.shift()
ts_log_diff.dropna(inplace=True)
plt.figure()
plt.plot(ts_log_diff)
plt.title("Air passenger data (log-1st diff)")

#%%
import statsmodels as sm
import statsmodels.api as smt
from statsmodels.tsa.statespace.sarimax import SARIMAX

smt.graphics.tsa.plot_acf(ts_log_diff, lags=30, alpha =
0.05)
smt.graphics.tsa.plot_pacf(ts_log_diff, lags=30, alpha =
0.05)
```

Replot the ACF and PACF curve:





Looking at the ACF and PACF plots of the differenced series we see our first significant cut off at lag 4 for ACF and at lag 2 for the PACF which suggest to use p = 2 and q = 4.

We also have a big value at lag 12 in the ACF plot which suggests our season is m = 12

Since this lag is positive it suggests P = 1 and Q = 0.

Since this is a differenced series for SARIMA we set d = 1, and since the seasonal pattern is not stable over time we set D = 0.

All together this gives us a SARIMA(2,1,4)$(1,0,0)_{12}$ model. Next we run SARIMA with these values to fit a model on our training data.

```
train_ratio = 0.7
split_point = round(len(ts_log)*train_ratio)
training, testing = ts_log[0:split_point],
ts_log[split_point:]

model = SARIMAX(training, \
                order=(2,1,4),\
                seasonal_order=(1,0,0,12),\
                enforce_stationarity=False,\
                enforce_invertibility=False)

model_fit = model.fit(disp=-1)
forecast = model_fit.forecast(len(testing))
```

Finally, we plot the forecasting results

```
plt.figure()
plt.plot(np.exp(forecast),'r')
plt.plot(np.exp(ts_log),'b')
plt.title('SARIMA(2,1,4)(1,0,0) RSS: %.4f'%
sum((model_fit.resid.values)**2))
plt.xlabel("Years")
plt.ylabel("Passengers")
plt.axvline(x=ts_log.index[split_point],color='black')
```