# Lecture 16 – A few examples of using ARIMA

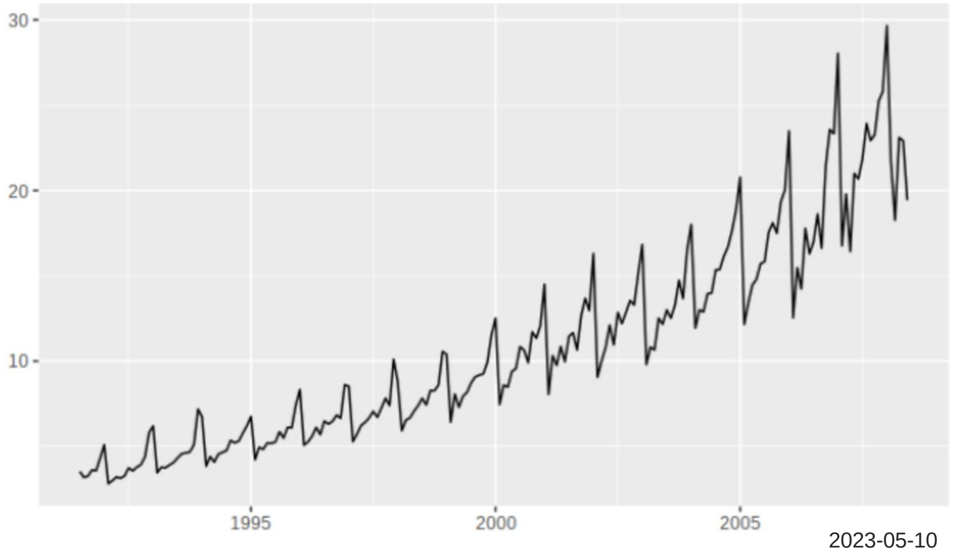
Wengang Mao (Marine Technology)
Department of Mechanics and Maritime Sciences,
Chalmers University of Technology,
Goteborg, Sweden

# Outline

- Basic concept of time series analysis example
- Data pre-processing, visualization and analysis (trend, periodic, seasonal, error analysis)
- Stationary and nonstationary time series
- Missing data in the time series
- ACF, PACF and their usage for **ARIMA**(*p*, *d*, *q*)
- Qualify and quantity the predictability of time series
- Improve forecast by using other series (Granger causality test)
- ARIMA and its application example

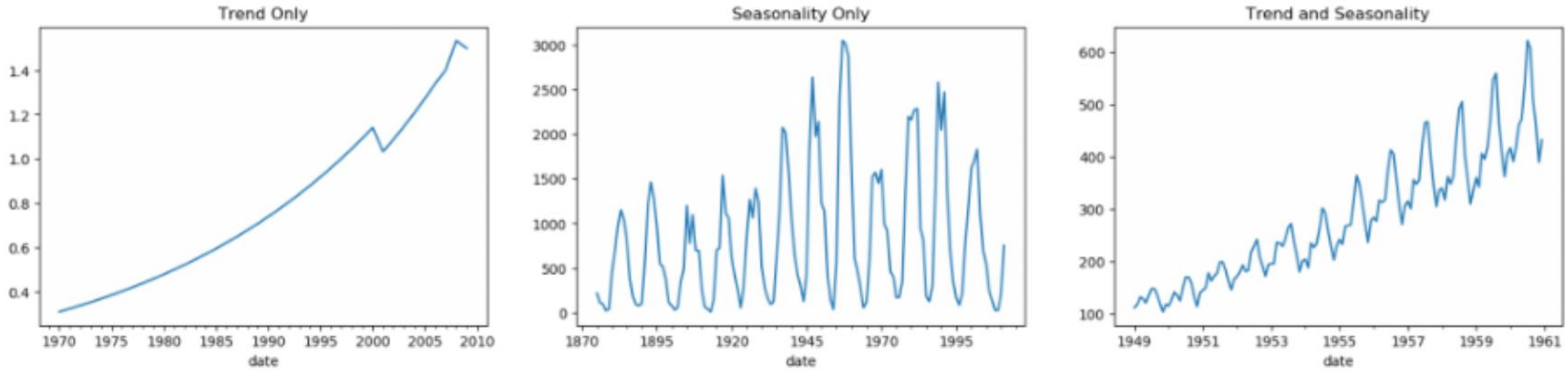
# Basic concept of time series example

- **Normally, there are some common features in observed time series**
- **From the above example time series, we can see:**
  - ✓Trend of increase in *Y* (drug sales)
  - ✓Seasonal
  - ✓Cyclic
  - ✓Variations around 0 (Random noise)



# Basic concept of time series example

- The characteristics of the series (Trend, Seasonal, Cyclic, etc.) can be see by decomposing the time series through careful data analysis



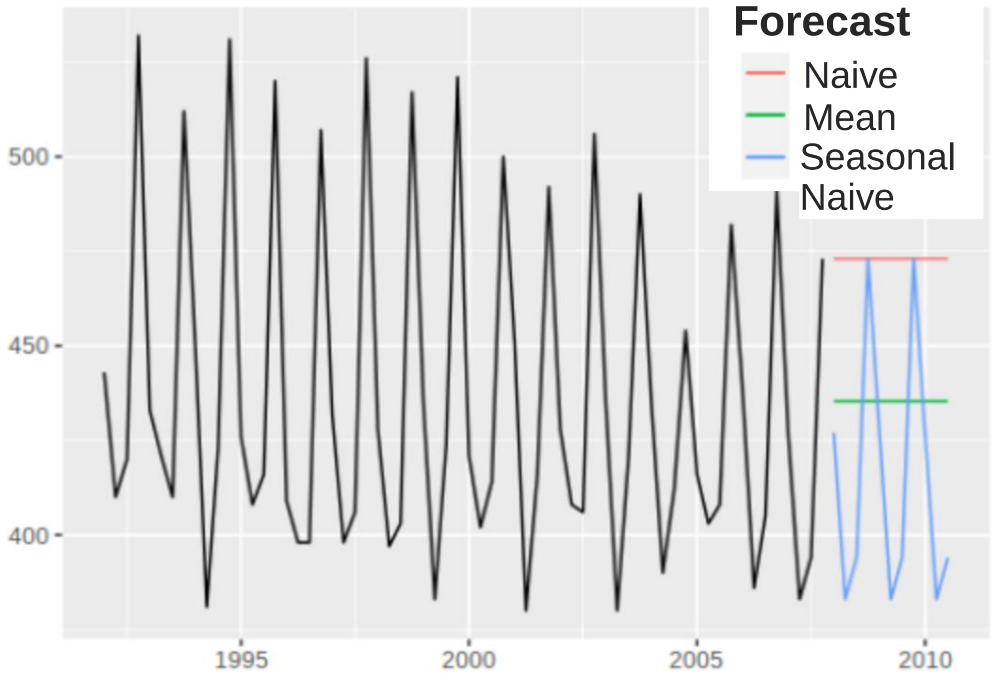
# Basic concept of time series example

Basic forecast/estimation method **rather than** ARIMA

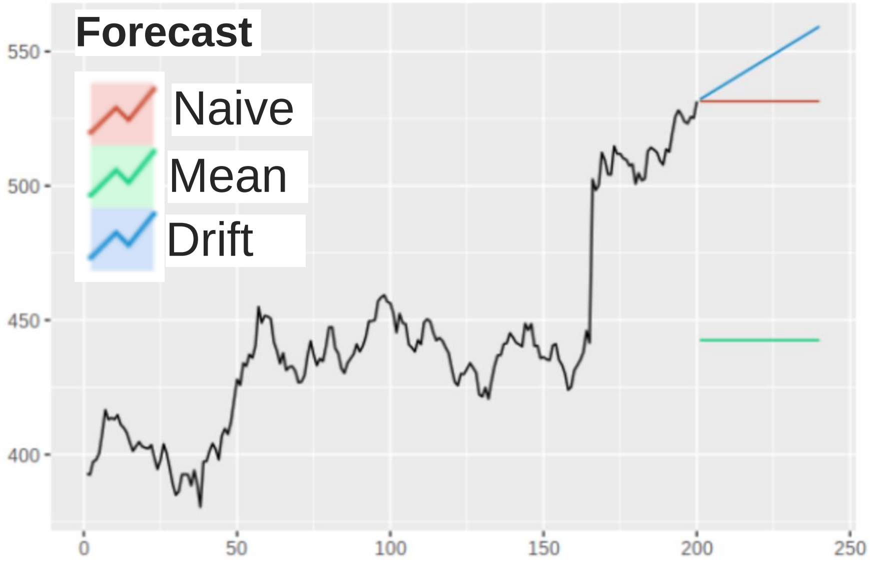
- ✓Mean estimation: all future forecast = mean of history data
- ✓Naive Approach: the last data as the future forecast value
- ✓Seasonal naïve approach: the data of the date of last month/week as the forecast of the same date of future month/week
- ✓Drift method:

$$\hat{y}_{T+h|T} = y_T + \frac{h}{T-1}\sum_{t=2}^{T}(y_t - y_{t-1}) = y_T + h\left(\frac{y_T - y_1}{T-1}\right)$$

# Basic concept of time series example



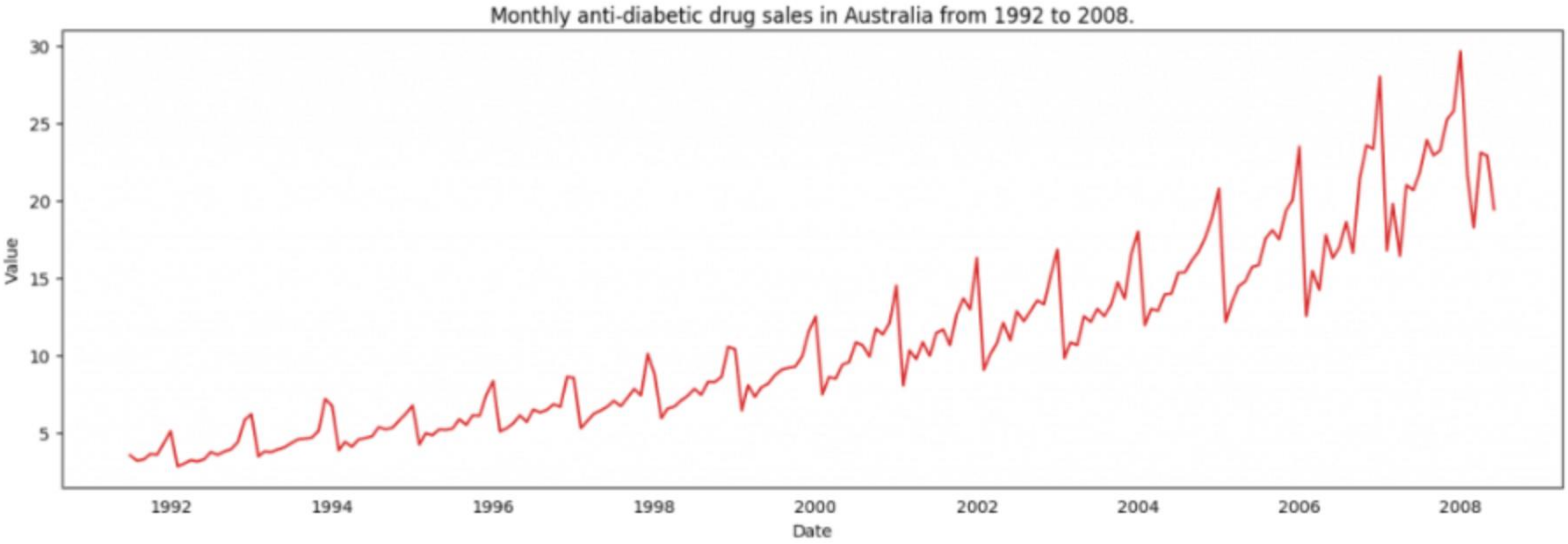

**Beer sale in terms of year**




**Stock index in terms of day**

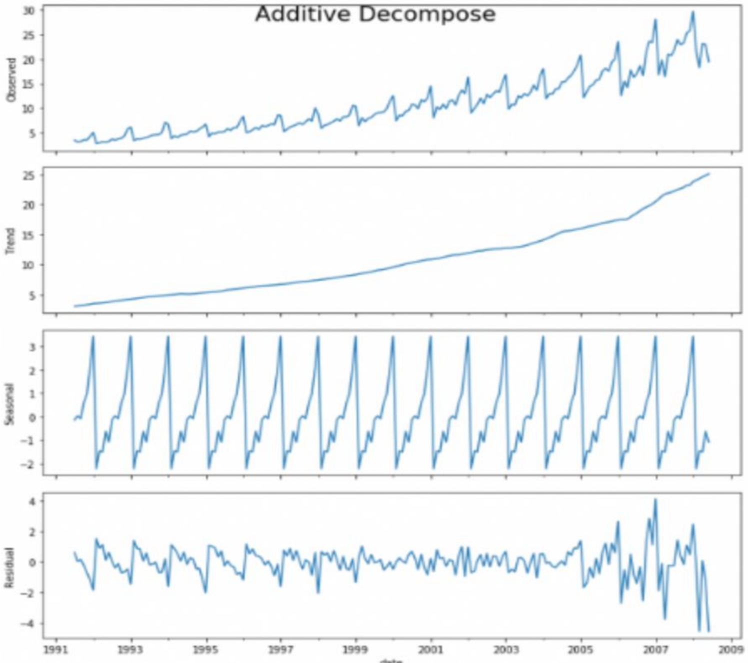
# Data pre-process and visualization

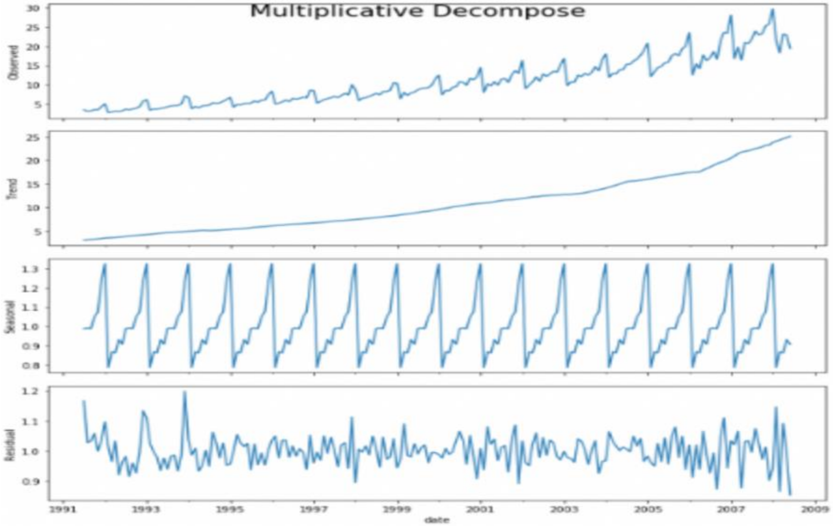
- Plot the time series to quantify such as trend, seasonal, and errors



# Data pre-process and visualization

- One could also analyse the time series to quantity such as trend, seasonal, and errors
  - Value = Trend + Seasonality + **Error**
  - Value = Trend * Seasonality * **Error**




For the above time series, it seems that the multiplicative decompose behaves better since the residuals look more random while the other still contains some seasonal effect.

# Stationary and nonstationary series

- **Why stationary process for forecast**
  - ✓It means that characteristics of historical data can be used to describe future data.
  - ✓For example, the mean, variance, ACF can be used in statistical models (AR model, MA model, ARMA model, etc.) for the prediction of future time series
  - ✓NB1: the prediction is not deterministic prediction. The mean is often corresponding to the “conventional” prediction, but there is also associated with uncertainties (variances/confidence interval)
  - ✓NB2: the white noise is a stationary but purely random process without internal correlation, which is not useful for the prediction.
- **How to check if a time series of a stationary process**
  - ✓**Qualitative check** the characteristics (by plot)
  - ✓**Quantitative check** the characteristics (by statistical test and estimation)

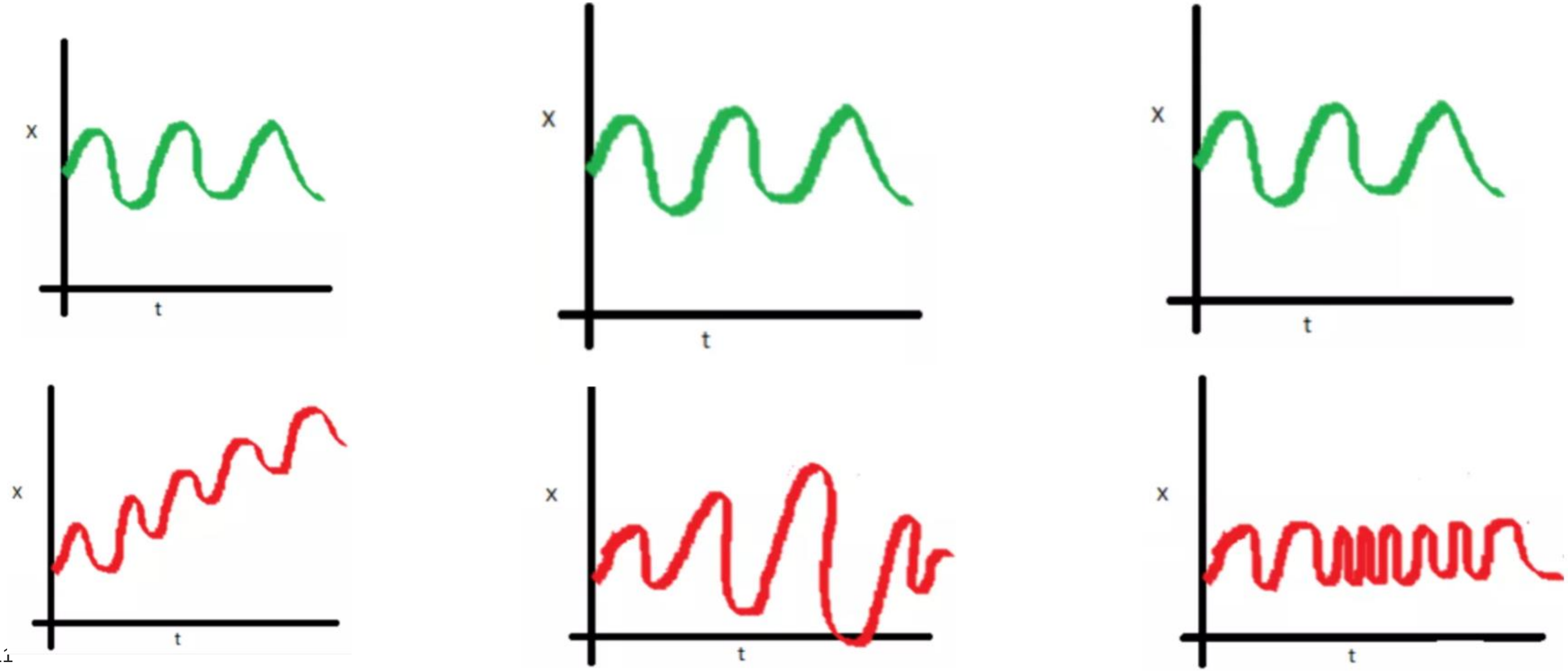
# Stationary and nonstationary series

- **How to transform a non-stationary process into a stationary process**
  - ✓Difference (with various orders)
  - ✓Take the logarithm (especially for skewed data)
  - ✓Take different orders of exponential transformation
  - ✓Decomposition (trend itself, trend with other parameters—mixed effect model,…)
  - ✓Combination of above methods

# Stationary and nonstationary series

## Qualitative check by plotting:

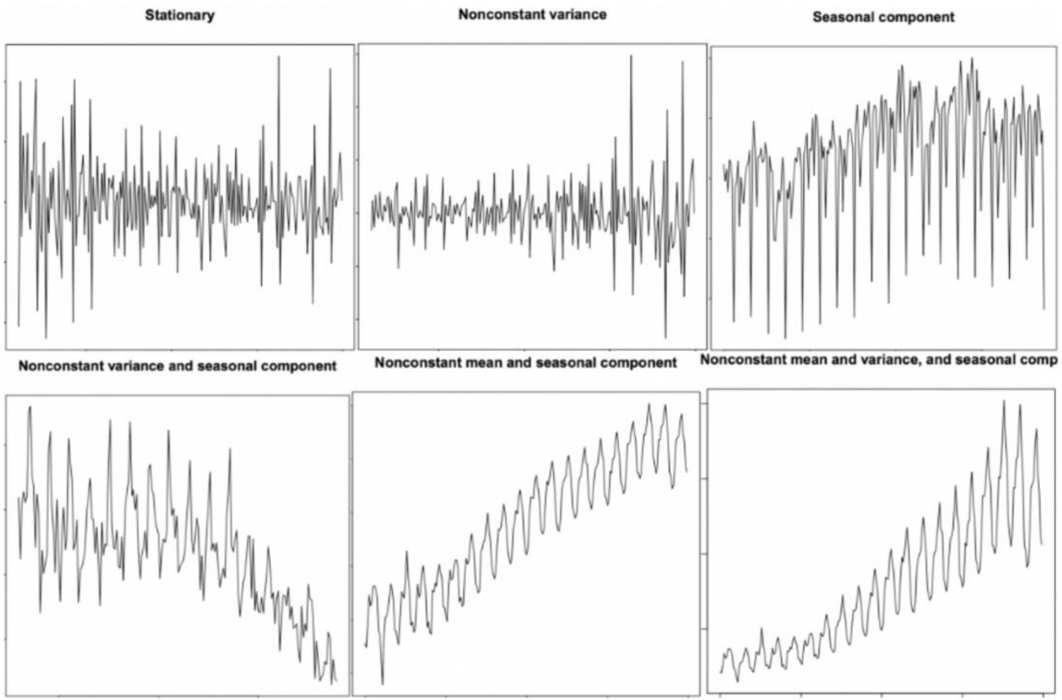
**Characteristics of stationarity: (1) Constant mean, (2) Variance not dependent on time, (3) Covariance is only dependent on time interval $Cov(X_{t+h}, X_{t+h}) = cov(h)$**



# Stationary and nonstationary series

## Qualitative check by plotting:

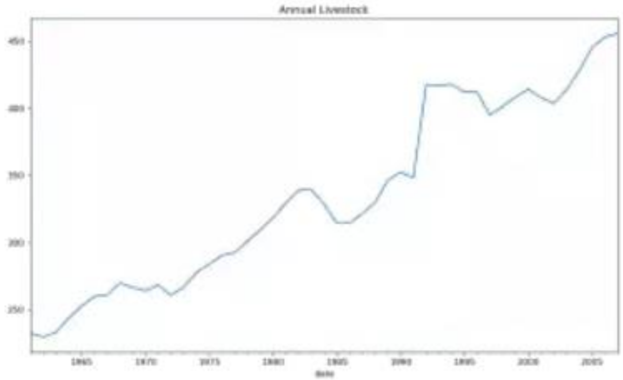
- **The time series of trend, seasonal components are NOT stationary process**



# Stationary and nonstationary series

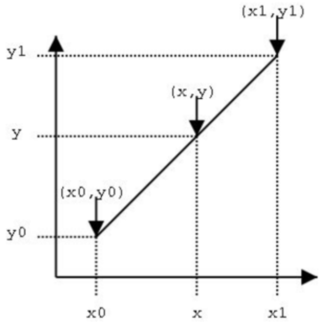
**Quantitative check by plotting:**

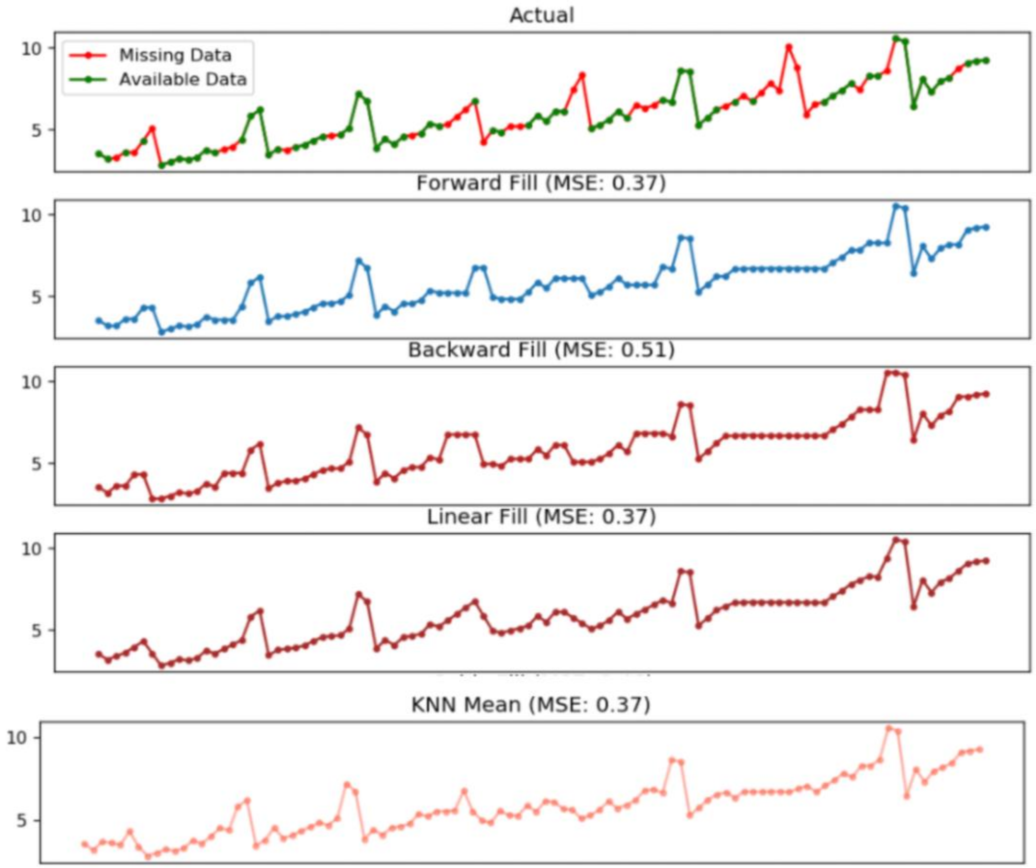
- **Augmented Dickey Fuller test (ADF Test)**
  - The null-hypothesis is: There is a unit root of the time series, i.e., the series is non-stationary. If the P-value is less than the significant level say 5%, then one can reject the null hypothesis.
- **Kwiatkowski-Phillips-Schmidt-Shin – KPSS test (trend stationary)**
  - ✓The null-hypothesis is: the series is stationary. One can tell python code to assign "regression='ct'" to treat "the time series of deterministic trend" be of stationary process.
- **Python implementation for the test**
  - ✓from statsmodels.tsa.stattools import adfuller, kpss
  - ✓result = adfuller(df.value.values, autolag='AIC')
  - ✓result = kpss(df.value.values, regression='ct')



# Missing data in the time series

- It is quite a normal task to deal with time series with missing data
- Different ways can be used to fill the missing data to form a “complete” time series
  - ✓All with zeros
  - ✓Forward fill
  - ✓Backward fill
  - ✓K-nearest neighbour mean。
  - ✓Linear interpolation

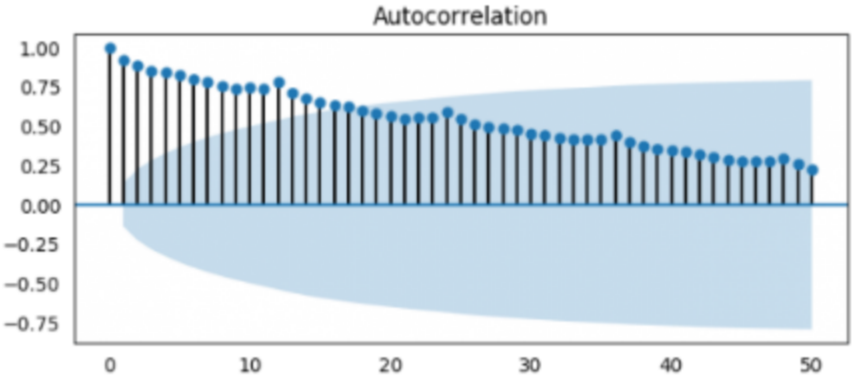


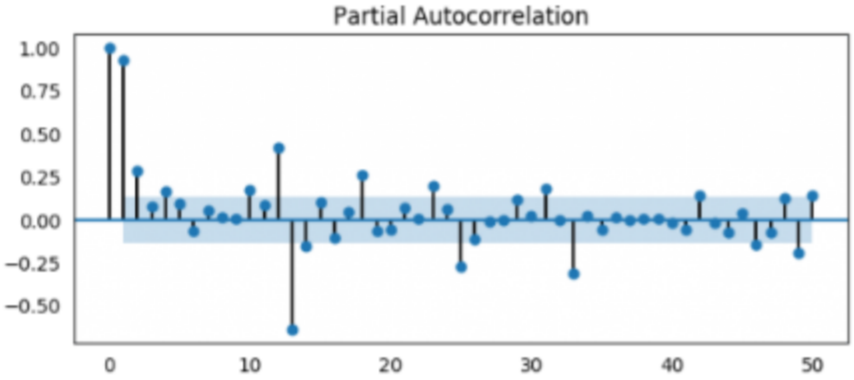



# ACF, PACF and ARIMA(*p*, *d*, *q*)

- The estimation of ACF and PACF is often included in a standard package/library
  - ✓ From the ACF and PACF, we can check when the correlation of the "new series" become white noise, i.e., their ACF&PACF become zero. However, due to random disturbance, their values will not be zero. But we need to check if they are located within the 95% CI of a white noise process

```
from statsmodels.tsa.stattools import acf, pacf
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
```

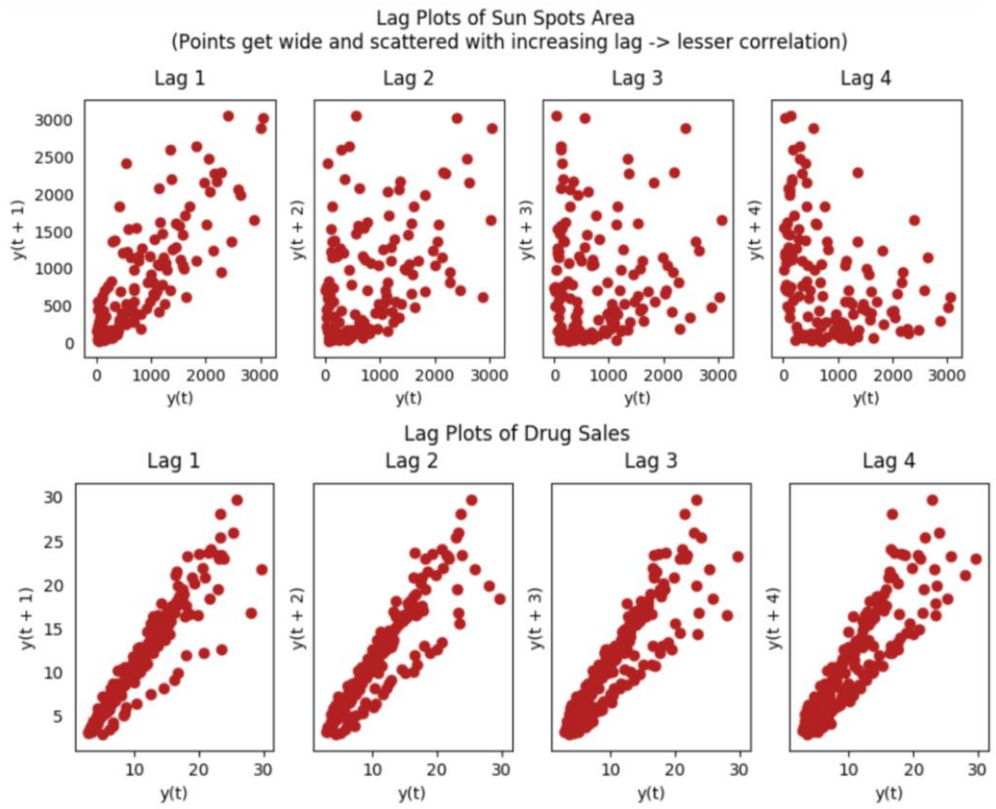



NB: if there is a long tail of the ACF (as left plot), it means the process contain trend, and difference is needed. If ACF approaches zero after one lag, it means over differenced. For the AR model, the order can be decided by PACF, while the MA model is by ACF.

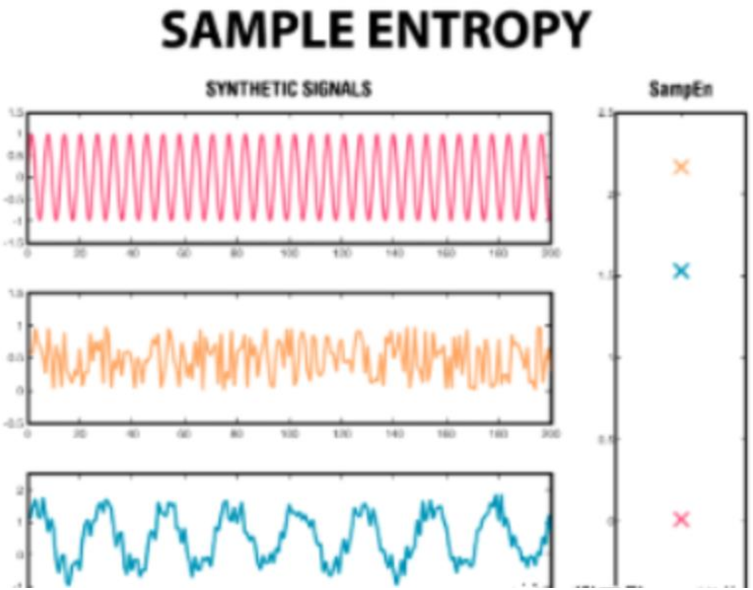
# ACF, PACF and ARIMA(*p*, *d*, *q*)

- The ACF can be also visualized by plot the data with various lags $(X_t, X_{t-n})$, or $(X_t, X_{t+n})$



# Qualify and quantity the predictability of time series

- **Sample Entropy:** It can be used to judge the degree of disturbance/error uncertainty of a series. The larger the Sample Entropy,  the harder for reliable prediction due to large fluctuation.




***Let a time series of $x_i, i = 0, 1, \ldots$,N. Sample Entropy = -log(A/B):***

✓ r:  distance tolerance, often taken as 0.15-0.3 std(X)

✓ X{M}: $[x_0, x_1, \ldots, x_M], [x_1, x_2, \ldots, x_{M+1}], \ldots, [x_{N-M-1}, x_{N-M}, \ldots, x_N]$

✓ X{M+1}: $[x_0, x_1, \ldots, x_{M+1}], [x_1, x_2, \ldots, x_{M+2}], \ldots,$ $[x_{N-M-2}, x_{N-M-1}, \ldots, x_N]$

✓ Maximum distance of two vectors in X{M}: For example, the first two vectors, their maximum distance is defined as:

✓ max(X{M,1}, X{M,2})=max(abs($x_0$- $x_1$),abs($x_1 - x_2$),…, abs($x_M - x_{M+1}$))

**A**: number of vectors within X{M} that their maximum distance less than r

**B**: number of vectors within X{M+1} that their maximum distance less than *r*

# Improve forecast by using other series

**Granger causality test:** one time series is related to other time series (very normal)

- The Granger causality test is used to test if by introducing ***Y***, the prediction of ***X*** in the future can be more accurate, i.e., the errors have been significantly reduced.
- In the method, there are two series as inputs. The first input is ***X***, i.e., the one that needed for the prediction, and the second is ***Y***, which may be useful to improve the prediction of ***X***.
- The second parameter in the test is the max lag time

```
from statsmodels.tsa.stattools import grangercausalitytests
```

```
grangercausalitytests(df[['value', 'month']], maxlag=2)
```

# ARIMA and its application

## Brief summary of ARIMA models

- A p-order autoregressive model AR(*p*)

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t$$

*For the AR(1) model:* $y_t = c + \phi_1 y_{t-1} + \varepsilon_t$

✓when $\phi_1 = 0$, $y_t$ is a white noise
✓when $\phi_1 = 1$ and *c*=0, $y_t$ is a random walk
✓when $\phi_1 = 0$ and *c*≠0, $y_t$ is a random walk with drift
✓when $\phi_1 < 0$, $y_t$ is fluctuated around zero
✓when $|\phi_1| < 1$, $y_t$ is a causal process

- A *q*-order moving average model MA(q)

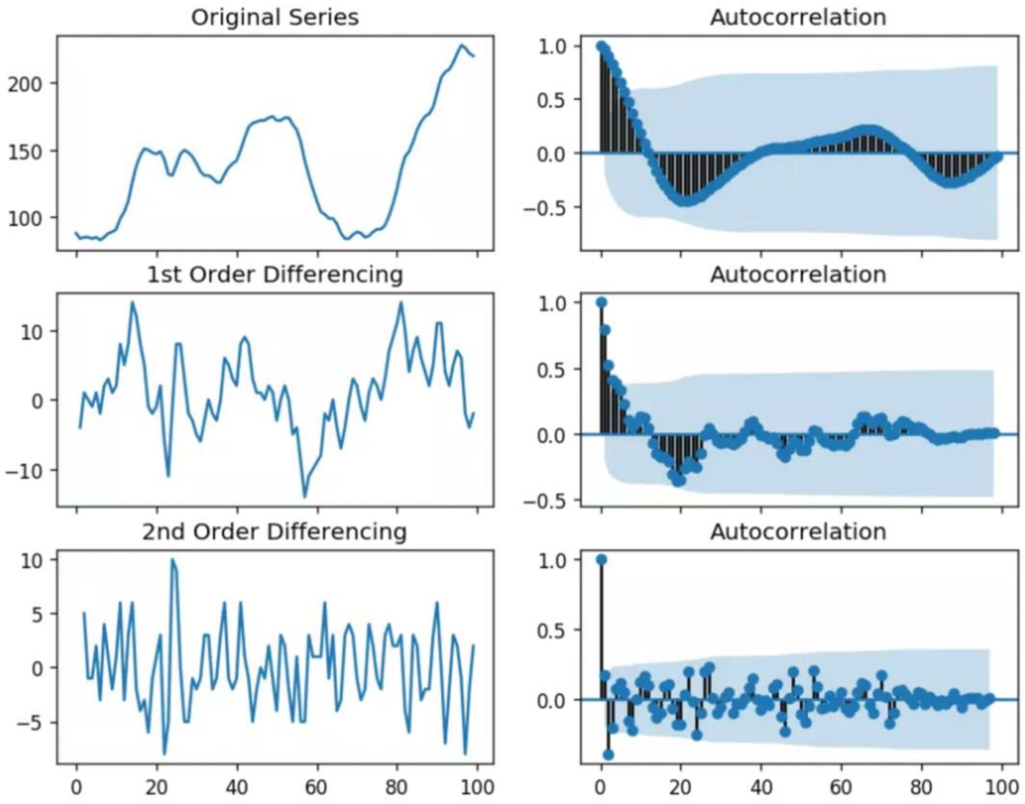
$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q}$$

✓$y_t$ is a weighted moving average to historical data, the order q indicates the expected error from historical data

# ARIMA and its application

**How to determine the order of the *d-order* of model ARIMA(*p, d, q*)**

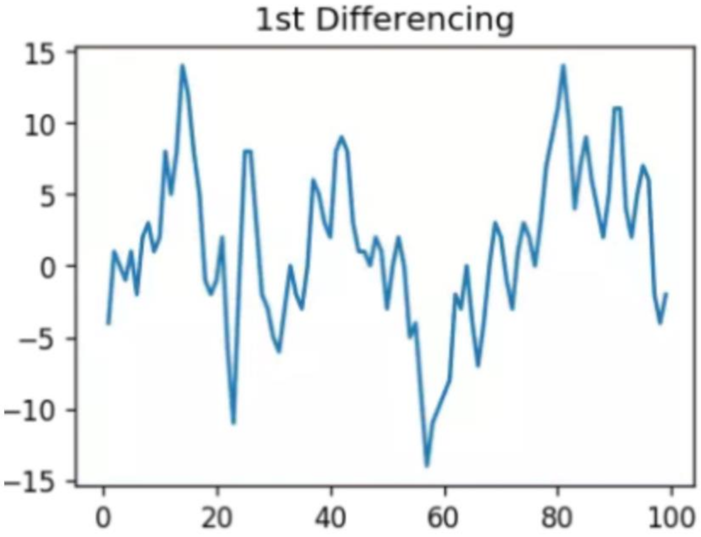
- The order of difference *d:* by cut value of ACF as Rule of thumb:
  - ✓If ACF is cut (approaches 0) with lag>10, further difference
  - ✓If ACF is cut (approaches 0) with lag=1, reduce difference
  - ✓If ACF is cut with similar lag values, pick the low difference order

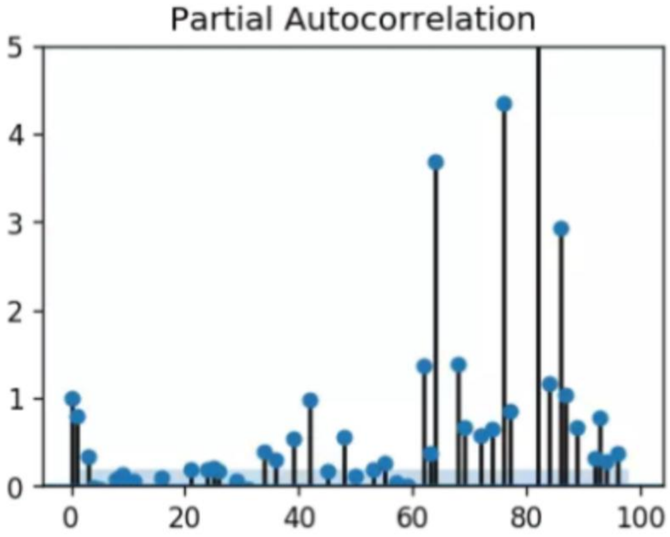


# ARIMA and its application

**How to determine the AR(*p*) order of the ARIMA model ARIMA(*p, d, q*)**

- The order of AR(*p*) of *p:* by cut value of PACF by the Rule:
  - ✓It equals to the value that the lag at which its PACF **starts to** locate within its CI

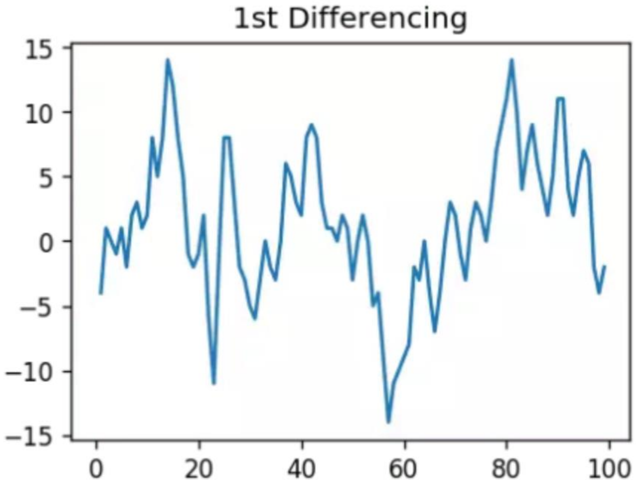


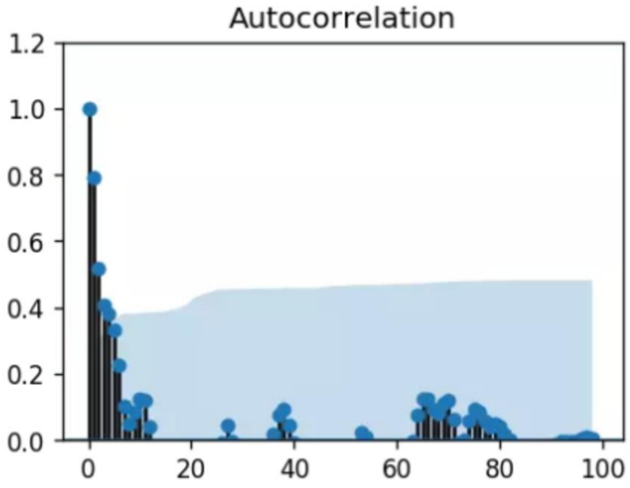



# ARIMA and its application

**How to determine the MA(*q*) order of the ARIMA model ARIMA(*p, d, q*)**

- The order of MA(*q*) of *q:* by cut value of ACF by the Rule:
  - ✓It equals to the value that the lag at which its ACF starts to located within its CI



# ARIMA and its application

**How to determine which ARIMA(*p, d, q*) is the most accurate. Criteria:**

- AIC(Akaike Information Criterion)

$$\text{AIC} = -2\log(L) + 2(p+q+k+1)$$

- Revised AIC

$$\text{AICc} = \text{AIC} + \frac{2(p+q+k+1)(p+q+k+2)}{T-p-q-k-2}$$

- BIC(Bayesian Information Criterion)

$$\text{BIC} = \text{AIC} + [\log(T) - 2](p+q+k+1)$$

L: likelihood function of the data

*k*=1: the model constant *c* is considered;

k=0: otherwise

Lower values of AIC or BIC mean better models, while often AICc and BIC are used for the evaluation.

# ARIMA and its application

**How to estimate all the coefficients within a ARIMA(*p*, *d*, *q*) model**

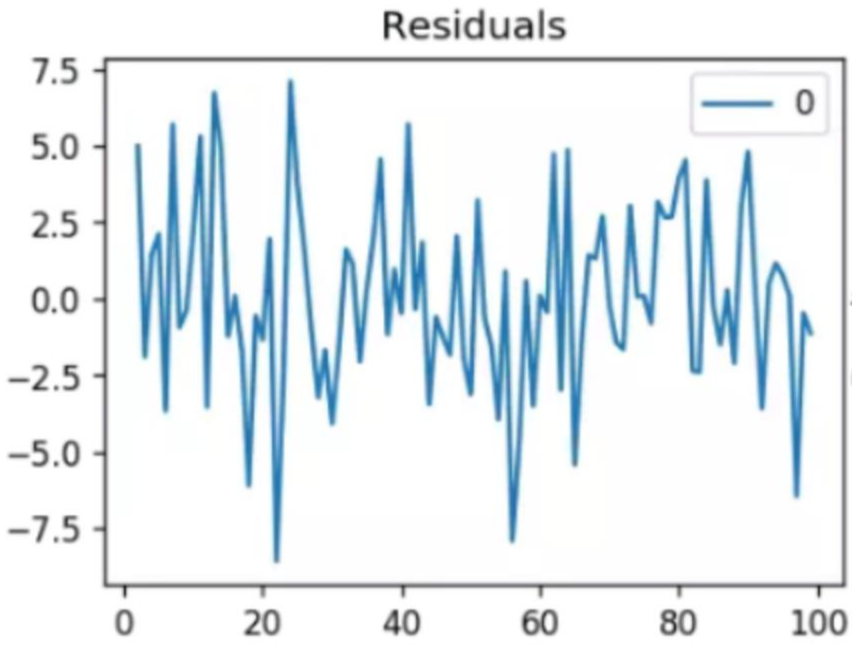
```
from statsmodels.tsa.arima_model import ARIMA

# 1,1,2 ARIMA Model
model = ARIMA(df.value, order=(1,1,2))
model_fit = model.fit(disp=0)
print(model_fit.summary())
```

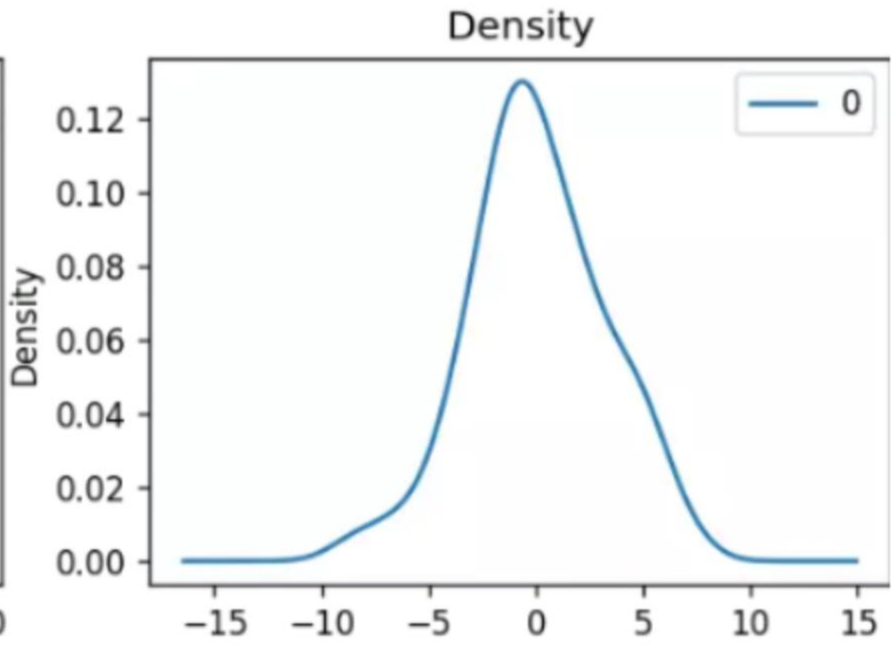
```
                              ARIMA Model Results
==============================================================================
Dep. Variable:                D.value   No. Observations:                   99
Model:                 ARIMA(1, 1, 1)   Log Likelihood                -253.790
Method:                       css-mle   S.D. of innovations              3.119
Date:                Sat, 09 Feb 2019   AIC                            515.579
Time:                        12:16:06   BIC                            525.960
Sample:                             1   HQIC                           519.779
```

```
                              ARIMA Model Results
==============================================================================
Dep. Variable:                D.value   No. Observations:                   99
Model:                 ARIMA(1, 1, 2)   Log Likelihood                -253.790
Method:                       css-mle   S.D. of innovations              3.119
Date:                Wed, 06 Feb 2019   AIC                            517.579
Time:                        23:32:56   BIC                            530.555
Sample:                             1   HQIC                           522.829

=================================================================================
                    coef    std err          z      P>|z|      [0.025      0.975]
---------------------------------------------------------------------------------
const             1.1202      1.290      0.868      0.387      -1.409       3.649
ar.L1.D.value     0.6351      0.257      2.469      0.015       0.131       1.139
ma.L1.D.value     0.5287      0.355      1.489      0.140      -0.167       1.224
ma.L2.D.value    -0.0010      0.321     -0.003      0.998      -0.631       0.629
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            1.5746           +0.0000j            1.5746            0.0000
MA.1           -1.8850           +0.0000j            1.8850            0.5000
MA.2          545.3515           +0.0000j          545.3515            0.0000
-----------------------------------------------------------------------------
```

# ARIMA and its application

## Model diagnostics: check the residual

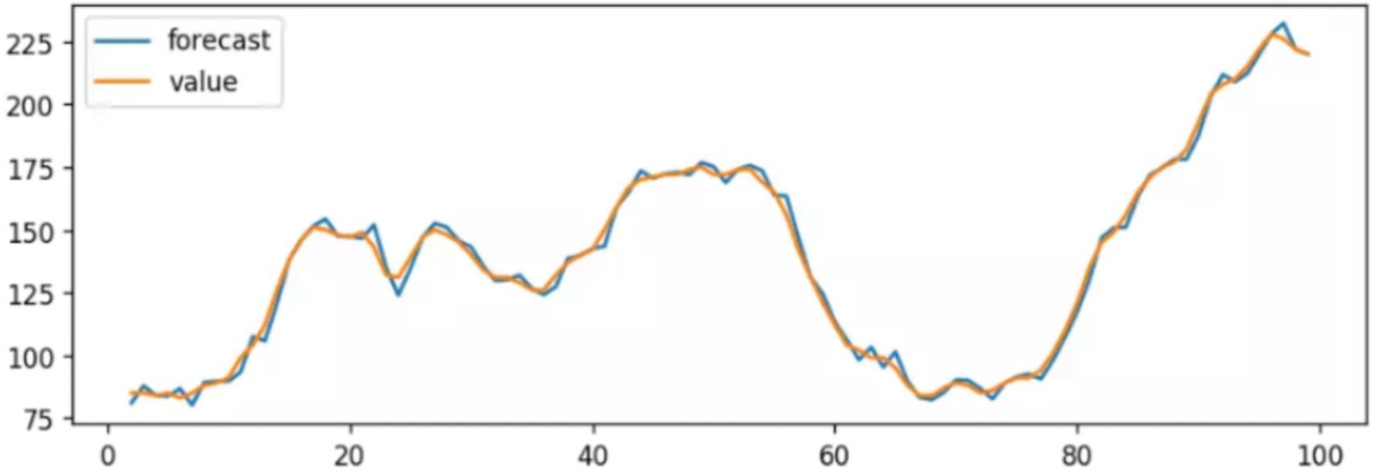


# ARIMA and its application

## Model fitting check

```python
# Actual vs Fitted
model_fit.plot_predict(dynamic=False)
plt.show()
```



# ARIMA and it application

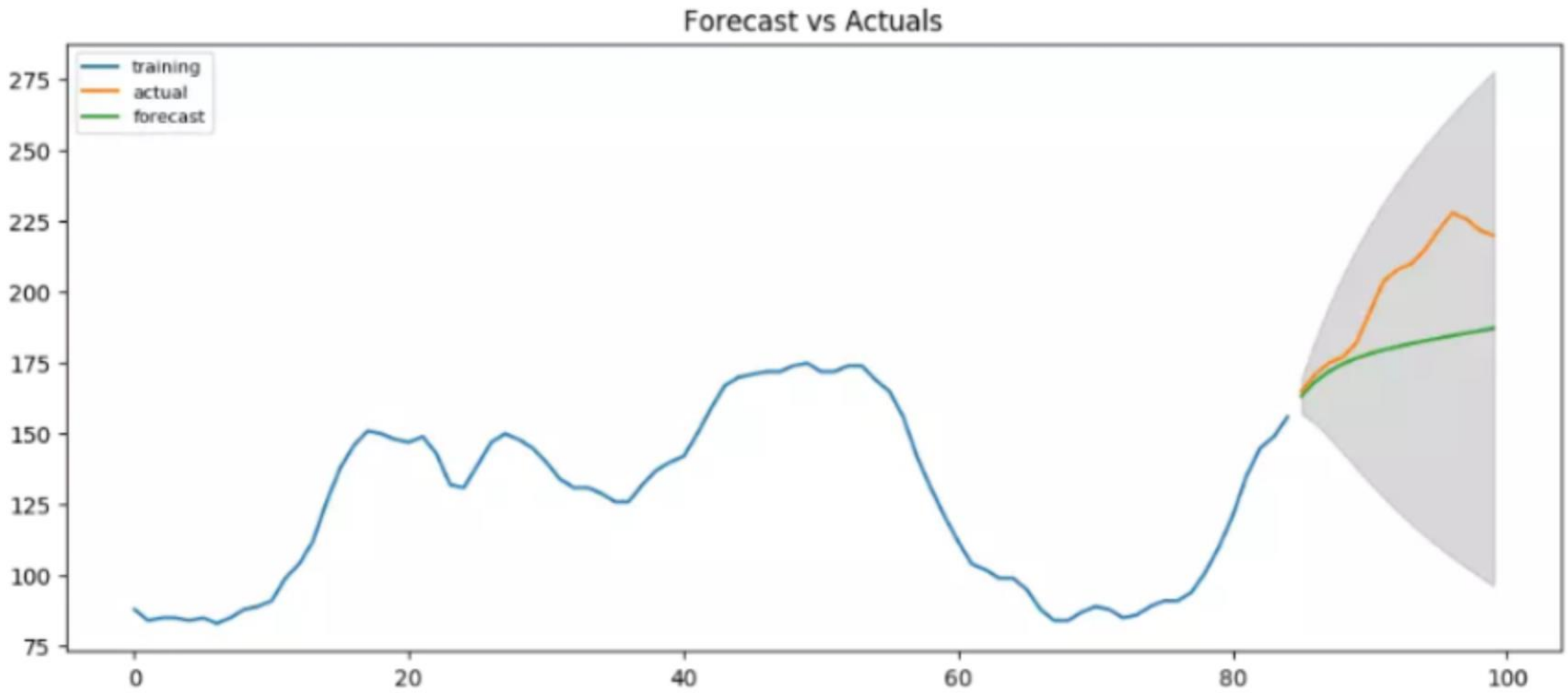
## Model for forecasting by fitting ARIMA(3,2,1) and ARIMA(1,1,1)

```
from statsmodels.tsa.stattools import acf

# Create Training and Test
train = df.value[:85]
test = df.value[85:]

# Build Model
# model = ARIMA(train, order=(3,2,1))
model = ARIMA(train, order=(1, 1, 1))
fitted = model.fit(disp=-1)

# Forecast
fc, se, conf = fitted.forecast(15, alpha=0.05)  # 95% conf

# Make as pandas series
fc_series = pd.Series(fc, index=test.index)
lower_series = pd.Series(conf[:, 0], index=test.index)
upper_series = pd.Series(conf[:, 1], index=test.index)
```

```
# Plot
plt.figure(figsize=(12,5), dpi=100)
plt.plot(train, label='training')
plt.plot(test, label='actual')
plt.plot(fc_series, label='forecast')
plt.fill_between(lower_series.index, lower_series, upper_series,
                 color='k', alpha=.15)
plt.title('Forecast vs Actuals')
plt.legend(loc='upper left', fontsize=8)
plt.show()
```

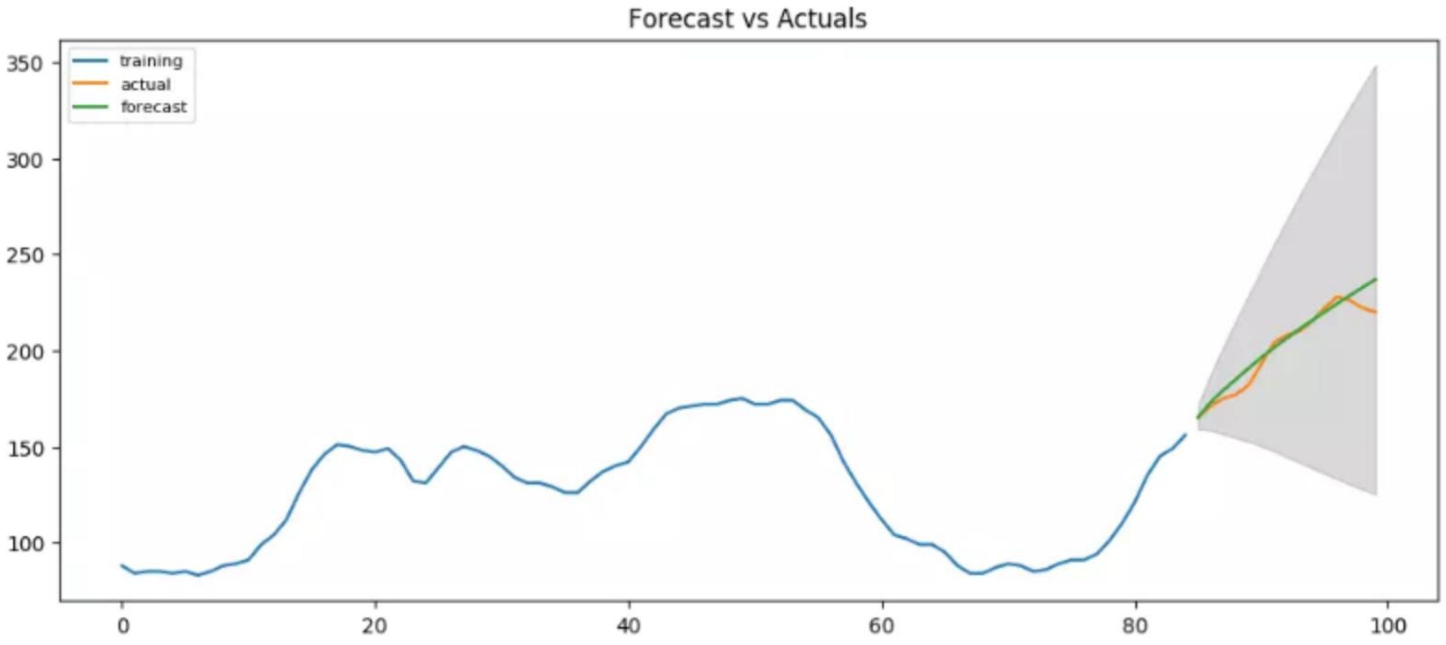
# ARIMA and it application

**Model for forecasting ARIMA(3,2,1)**



# ARIMA and it application

**Model for forecasting ARIMA(1,1,1)**



# ARIMA and it application

## Which ARIMA(*p, d, q*) provides us the best model and forecasting?

✓By checking the ACF and PACF for different order of differencing sometimes may be too time consuming. There is a package that can search the best ARIMA model based on the data, i.e., the *auto_arima* in the package *pmdarima*.

```
from statsmodels.tsa.arima_model import ARIMA
import pmdarima as pm
```

```
model = pm.auto_arima(df.value, start_p=1, start_q=1,
                      information_criterion='aic',
                      test='adf',       # use adftest to find optimal 'd'
                      max_p=3, max_q=3, # maximum p and q
                      m=1,              # frequency of series
                      d=None,           # let model determine 'd'
                      seasonal=False,   # No Seasonality
                      start_P=0,
                      D=0,
                      trace=True,
                      error_action='ignore',
                      suppress_warnings=True,
                      stepwise=True)

print(model.summary())
```

```
# Forecast
n_periods = 24
fc, confint = model.predict(n_periods=n_periods, return_conf_int=True)
index_of_fc = np.arange(len(df.value), len(df.value)+n_periods)

# make series for plotting purpose
fc_series = pd.Series(fc, index=index_of_fc)
lower_series = pd.Series(confint[:, 0], index=index_of_fc)
upper_series = pd.Series(confint[:, 1], index=index_of_fc)

# Plot
plt.plot(df.value)
plt.plot(fc_series, color='darkgreen')
plt.fill_between(lower_series.index,
                 lower_series,
                 upper_series,
                 color='k', alpha=.15)

plt.title("Final Forecast of WWW Usage")
plt.show()
```

# ARIMA and it application

**Which ARIMA(*p*,*d*,*q*) provides us the best model and forecasting?**

✓By checking the ACF and PACF for different order of differencing sometimes may be too time consuming. There is a package that can ***automatically*** search the best ARIMA model based on the data, i.e., the *auto_arima* in the package *pmdarima*.

```python
from statsmodels.tsa.arima_model import ARIMA
import pmdarima as pm
```

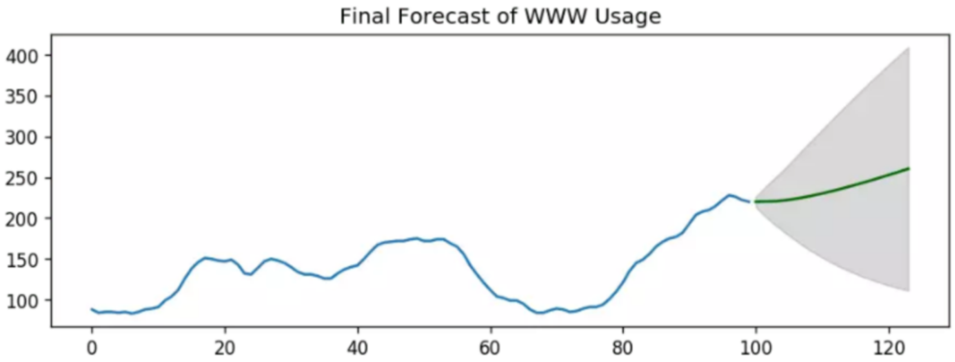
```python
model = pm.auto_arima(df.value, start_p=1, start_q=1,
                      information_criterion='aic',
                      test='adf',       # use adftest to find optimal 'd'
                      max_p=3, max_q=3, # maximum p and q
                      m=1,              # frequency of series
                      d=None,           # let model determine 'd'
                      seasonal=False,   # No Seasonality
                      start_P=0,
                      D=0,
                      trace=True,
                      error_action='ignore',
                      suppress_warnings=True,
                      stepwise=True)

print(model.summary())
```

```python
# Forecast
n_periods = 24
fc, confint = model.predict(n_periods=n_periods, return_conf_int=True)
index_of_fc = np.arange(len(df.value), len(df.value)+n_periods)

# make series for plotting purpose
fc_series = pd.Series(fc, index=index_of_fc)
lower_series = pd.Series(confint[:, 0], index=index_of_fc)
upper_series = pd.Series(confint[:, 1], index=index_of_fc)

# Plot
plt.plot(df.value)
plt.plot(fc_series, color='darkgreen')
plt.fill_between(lower_series.index,
                 lower_series,
                 upper_series,
                 color='k', alpha=.15)

plt.title("Final Forecast of WWW Usage")
plt.show()
```

# ARIMA and it application

## ***Automatically*** **search the best ARIMA(*p*,*d*,*q*) model**

```
#> Fit ARIMA: order=(1, 2, 1); AIC=525.586, BIC=535.926, Fit time=0.060 seconds
#> Fit ARIMA: order=(0, 2, 0); AIC=533.474, BIC=538.644, Fit time=0.005 seconds
#> Fit ARIMA: order=(1, 2, 0); AIC=532.437, BIC=540.192, Fit time=0.035 seconds
#> Fit ARIMA: order=(0, 2, 1); AIC=525.893, BIC=533.648, Fit time=0.040 seconds
#> Fit ARIMA: order=(2, 2, 1); AIC=515.248, BIC=528.173, Fit time=0.105 seconds
#> Fit ARIMA: order=(2, 2, 0); AIC=513.459, BIC=523.798, Fit time=0.063 seconds
#> Fit ARIMA: order=(3, 2, 1); AIC=512.552, BIC=528.062, Fit time=0.272 seconds
#> Fit ARIMA: order=(3, 2, 0); AIC=515.284, BIC=528.209, Fit time=0.042 seconds
#> Fit ARIMA: order=(3, 2, 2); AIC=514.514, BIC=532.609, Fit time=0.234 seconds
#> Total fit time: 0.865 seconds
#>                          ARIMA Model Results
#> ==============================================================================
#> Dep. Variable:                   D2.y   No. Observations:                   98
#> Model:                 ARIMA(3, 2, 1)   Log Likelihood                -250.276
#> Method:                       css-mle   S.D. of innovations              3.069
#> Date:                Sat, 09 Feb 2019   AIC                            512.552
#> Time:                        12:57:22   BIC                            528.062
#> Sample:                             2   HQIC                           518.825
#>
```



# ARIMA extension: seasonal ARIMA

CHALMERS
UNIVERSITY OF TECHNOLOGY

A seasonal ARIMA is composed of seasonal component and stationary components

$Y_t = ARIMA_{non-seasonal}(p, d, q) + ARIMA_{seasonal}(P, D, Q)_m$: ***m*** is the seasonal period

```python
# !pip3 install pyramid-arima
import pmdarima as pm
# Seasonal - fit stepwise auto-ARIMA
smodel = pm.auto_arima(data, start_p=1, start_q=1,
                         test='adf',
                         max_p=3, max_q=3, m=12,
                         start_P=0, seasonal=True,
                         d=None, D=1, trace=True,
                         error_action='ignore',
                         suppress_warnings=True,
                         stepwise=True)
smodel.summary()
```

```
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                                     y   No. Observations:              100
Model:             SARIMAX(2, 2, 0)x(0, 0, [1, 2], 12)   Log Likelihood             -249.542
Date:                               Tue, 15 Dec 2020   AIC                         509.083
Time:                                       09:01:05   BIC                         522.008
Sample:                                            0   HQIC                        514.311
                                               - 100
Covariance Type:                                 opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.2012      0.101      1.984      0.047       0.002       0.400
ar.L2         -0.4009      0.091     -4.428      0.000      -0.578      -0.223
ma.S.L12       0.2213      0.137      1.619      0.106      -0.047       0.489
ma.S.L24      -0.2499      0.138     -1.813      0.070      -0.520       0.020
sigma2         9.2452      1.472      6.282      0.000       6.361      12.129
===================================================================================
```

```
 ARIMA(0,2,1)(0,0,2)[12]             : AIC=519.484, Time=0.16 sec
 ARIMA(1,2,0)(0,0,2)[12]             : AIC=523.350, Time=0.16 sec
 ARIMA(2,2,1)(0,0,2)[12]             : AIC=511.081, Time=0.20 sec
 ARIMA(2,2,1)(0,0,1)[12]             : AIC=513.259, Time=0.08 sec
 ARIMA(2,2,1)(1,0,2)[12]             : AIC=512.869, Time=0.34 sec
 ARIMA(2,2,1)(1,0,1)[12]             : AIC=inf, Time=0.16 sec
 ARIMA(2,2,0)(0,0,2)[12]             : AIC=509.083, Time=0.16 sec
 ARIMA(2,2,0)(0,0,1)[12]             : AIC=511.322, Time=0.05 sec
 ARIMA(2,2,0)(1,0,2)[12]             : AIC=510.874, Time=0.24 sec
 ARIMA(2,2,0)(1,0,1)[12]             : AIC=inf, Time=0.14 sec
 ARIMA(3,2,0)(0,0,2)[12]             : AIC=511.082, Time=0.17 sec
 ARIMA(3,2,1)(0,0,2)[12]             : AIC=513.060, Time=0.37 sec
 ARIMA(2,2,0)(0,0,2)[12] intercept   : AIC=511.078, Time=0.21 sec

Best model:  ARIMA(2,2,0)(0,0,2)[12]
Total fit time: 3.423 seconds
```

# ARIMA extension: exogenous variable

**Often, the predictor is dependent on other time series, i.e., exogenous variable**

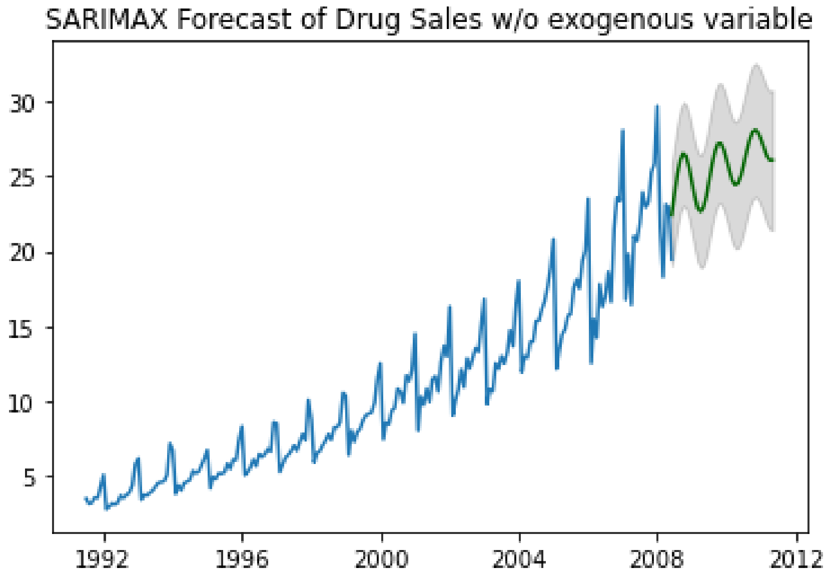
- ✓But if the exogenous variable is also a random time series, it may contain extra uncertainties if it is acting as the exogenous variable in the prediction.
- ✓Therefore, it is often practical to introduce deterministic variables as the exogenous variable
- ✓Of course, one of the most straightforward variable for time series is the time as variable

```
Best model:  ARIMA(2,1,3)(0,0,0)[0] intercept
Total fit time: 2.317 seconds

In [251]: sxmodel.summary()
Out[251]:
<class 'statsmodels.iolib.summary.Summary'>
"""
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  204
Model:               SARIMAX(2, 1, 3)   Log Likelihood                -392.943
Date:                Tue, 15 Dec 2020   AIC                            799.887
Time:                        12:20:34   BIC                            823.079
Sample:                             0   HQIC                           809.269
                                - 204
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      0.0258      0.008      3.408      0.001       0.011       0.041
ar.L1          1.7083      0.039     44.111      0.000       1.632       1.784
ar.L2         -0.9583      0.033    -28.909      0.000      -1.023      -0.893
ma.L1         -2.5524      0.084    -30.396      0.000      -2.717      -2.388
ma.L2          2.3292      0.157     14.859      0.000       2.022       2.636
ma.L3         -0.7397      0.081     -9.083      0.000      -0.899      -0.580
sigma2         2.9810      0.211     14.150      0.000       2.568       3.394
===================================================================================
```

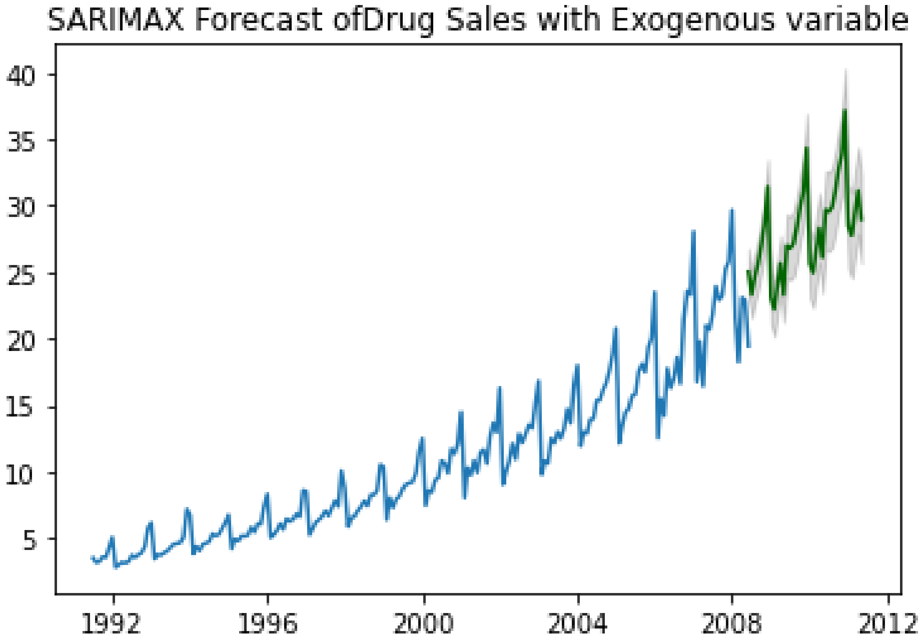
```
Best model:  ARIMA(3,1,2)(0,1,1)[12] intercept
Total fit time: 48.640 seconds
Out[243]:
<class 'statsmodels.iolib.summary.Summary'>
"""
                                      SARIMAX Results
==========================================================================================
Dep. Variable:                                  y   No. Observations:                  204
Model:             SARIMAX(3, 1, 2)x(0, 1, [1], 12)   Log Likelihood                -251.489
Date:                            Tue, 15 Dec 2020   AIC                            520.978
Time:                                    12:09:39   BIC                            550.249
Sample:                                         0   HQIC                           532.834
                                            - 204
Covariance Type:                              opg
==================================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------
intercept          0.0098      0.005      1.992      0.046       0.000       0.019
seasonal_index  9.401e-07      1e-09    938.191      0.000    9.38e-07    9.42e-07
ar.L1             -0.6178      0.127     -4.875      0.000      -0.866      -0.369
ar.L2              0.2981      0.101      2.965      0.003       0.101       0.495
ar.L3              0.4197      0.063      6.702      0.000       0.297       0.542
ma.L1             -0.3208      0.139     -2.309      0.021      -0.593      -0.049
ma.L2             -0.5836      0.124     -4.688      0.000      -0.828      -0.340
ma.S.L12          -0.4688      0.066     -7.125      0.000      -0.598      -0.340
sigma2             0.7935      0.056     14.240      0.000       0.684       0.903
==================================================================================
```

# ARIMA extension: exogenous variable

## Results comparison with & w/o the exogenous variable




No exogenous variable and seasonal impact



AVANCEZ

1829

CHALMERS

UNIVERSITY OF TECHNOLOGY