시계열 예측

김태경

Chapter 01 시계열 분석 개요

시계열 자료(time series data)

- 연도별, 분기별, 월별, 일별, 시간별 등 시간의 흐름에 따라 순서대로 관측되는 자료

- • 예 : 국내총생산(GDP), 물가지수, 판매량, 종합주가지수(KOSPI), 강우량, 태양 흑점수, 실험 및 관측자료 등

- 시간 단위 외에도 사회적 변화나 환경적 변동요인을 기준으로 시계열자료를 구분하는 경우도 있다.

시계열 자료(time series data)

- 일반적으로 시계열 자료는 추세변동, 순환변동, 계절변동, 불규칙변동 요인으로 구성된다.

- 시계열들은 생성되는 특성에 따라 연속적으로 생성되는 연 속시계열(continuous time series)과 이산적 시점에서 생성 되는 이산시계열(discrete time series)로 구분할 수 있음.

- 실제로는 많은 시계열들이 연속적으로 생성되고 있지만 일정한 시차를 두고 관측되므로 이산시계열의 형태를 지니는 경우가 많음.

시계열 분석 개요(the nature of time series analysis)

- 시계열자료(time series data)들은 시간의 경과에 따라 관측된 자료이므로 시간에 영향을 받음.
  - 시계열자료를 분석할 때 관측시점들 간의 시차(time lag)가 중요한 역할을 함.
  - 예를 들어 오늘의 주가가 한달 전, 일주일 전의 주가보다는 어제의 주가에 더 많은 영향을 받는 것과 마찬가지로 가까운 관측시점일수록 관측자료들 간에 상관관 계가 커짐.
  - 시계열은 일반적으로 시간 t를 하첨자로 하여 다음과 같이 표현됨. {Zt : t=1, 2, 3, ×××} 또는 Z1 , Z2 , Z3 , ×××

시계열 분석 개요(the nature of time series analysis)

- 시계열분석(time series analysis)의 목적
  - 과거 시계열자료의 패턴(pattern)이 미래에도 지속적으로 유지된다는 가정하에서 현재까지 수집된 자료들을 분석하여 미래에 대한 예측(forecast)을 한다.
  - 시계열자료를 분석할 때 관측시점들 간의 시차(time lag)가 중요한 역할을 한다.
  - 시계열자료가 생성된 시스템 또는 확률과정을 모형화하여 시스템 또는 확률과정을 이해하고 제어(control)할 수 있도록 한다.

시계열 분석 개요(the nature of time series analysis)

- **예측, 계획 그리고 목표**
  - 예측은 경영분야에 있어서 생산 계획, 수송, 인사에 관한 결정을 알리거나 장기 전략 계획을 세우는데 도움을 줄 수 있는 흔한 통계적 업무이다.
  - **목표**는 예측 및 계획과 연관되어 있는 것이 좋지만, 이것이 항상 일어나는 것은 아니다. 목표를 어떻게 달성할지에 대한 계획과 목표가 실현 가능한지에 대한 예측 없이 목표를 세우는 경우가 너무나도 자주 있다.
  - **계획**은 예측과 목표에 대한 대응입니다. 계획은 여러분의 예측과 목표를 일치시키는데 필요한 적절한 행동을 결정하는 일을 포함한다.

시계열 분석 개요(the nature of time series analysis)

- **단기 예측**은 인사, 생산, 수송 계획 등에 필요하고, 계획 과정의 한 부분으로 수요 예측도 필요하다.

- **중기 예측**은 원자재 구입, 신규 채용, 장비나 기계 구입 등 미래 자원 공급을 결정하는데 필요하다.

- **장기 예측**은 전략적으로 계획을 세우는데 사용하고, 시장 기회, 환경 요인, 내부 자원을 반드시 고려하여 결정한다.

시계열 분석 개요(the nature of time series analysis)

시계열의 형태(the components of time series)

- 불규칙변동(irregular variation 또는 확률적 변동 : random variation)은 시계열자료에서 시간에 따른 규칙적인 움직임과는 달리 어떤 규칙성이 없이 예측이 불가능하게 우연적으로 발생하는 변동을 말함.

  예 : 전쟁, 홍수, 화재, 지진, 파업 등

- 체계적 변동에는 장기간에 걸쳐 어떤 추세로 나타나는 추세 변동(trend variation), 추세선을 따라 주기적으로 오르고 내림을 반복하는 순환변동(cyclical variation), 그리고 계절적 요인이 작용하여 1년 주기로 나타나는 계절변동(seasonal variation)이 있음.

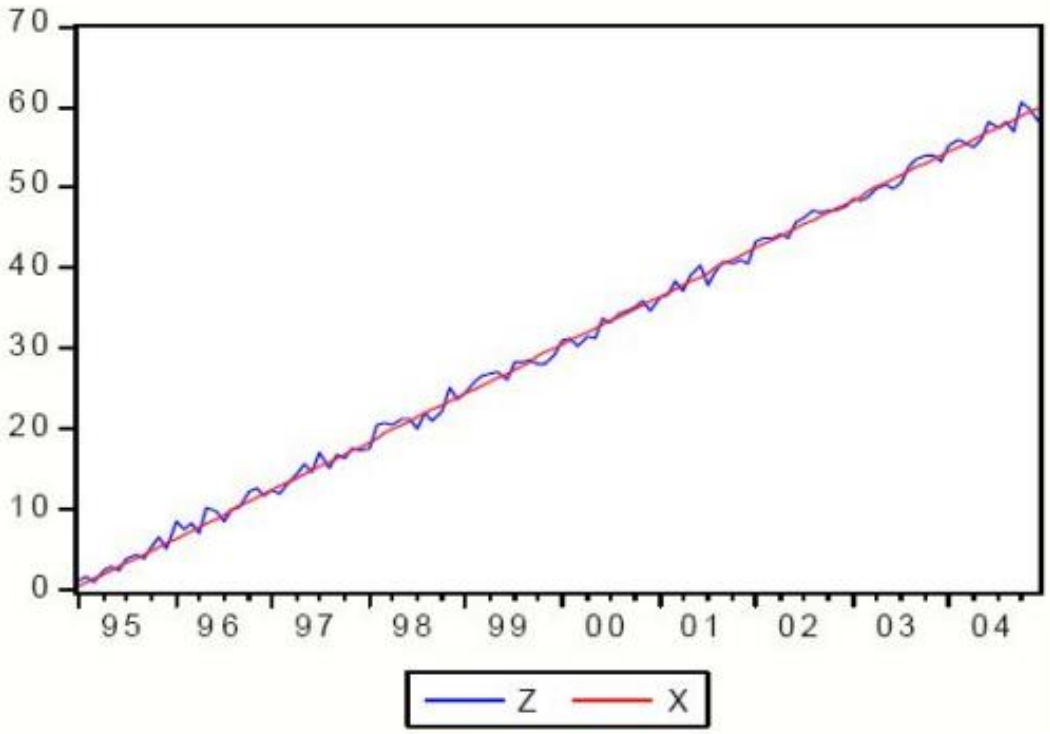
시계열 분석 개요(the nature of time series analysis)

추세변동(trend variation)

- 추세변동이란 시계열자료가 갖는 장기적인 변화추세임.
- 추세란 장기간에 걸쳐 지속적으로 증가 또는 감소하거나 또는 일정한 상태(stationary)를 유지하려는 성향을 의미함.
- 그러므로 시계열자료에서 짧은 기간 동안에는 추세변동을 찾기 어려움.
- 따라서 추세변동은 짧은 기간 동안 급격하게 변동하는 것이 아니라 장기적인 추세경향이 나타나는 것으로 직선이나 부드러운 곡선의 연장선으로 표시함. 이러한 추세는 직선뿐만아니라 곡선, S자 형태의 추세를 가질 수도 있음.
- 예 : 국내총생산(GDP), 인구증가율, 기술변화 등

시계열 분석 개요(the nature of time series analysis)

추세변동(trend variation)



시계열 분석 개요(the nature of time series analysis)

추세변동(trend variation)

```
import numpy as np
import pandas as pd


# DatetiemIndex
dates = pd.date_range('2020-01-01', periods=48, freq='M')

# additive model: trend + cycle + seasonality + irregular factor
timestamp = np.arange(len(dates))
trend_factor = timestamp*1.1
cycle_factor = 10*np.sin(np.linspace(0, 3.14*2, 48))
seasonal_factor = 7*np.sin(np.linspace(0, 3.14*8, 48))
np.random.seed(2004)
irregular_factor = 2*np.random.randn(len(dates))


df = pd.DataFrame({'timeseries': trend_factor + cycle_factor + seasonal_factor + irregular_factor,
                   'trend': trend_factor,
                   'cycle': cycle_factor,
                   'seasonal': seasonal_factor,
                   'irregular': irregular_factor},
                   index=dates)
```

시계열 분석 개요(the nature of time series analysis)

추세변동(trend variation)

```
# Time series plot
import matplotlib.pyplot as plt

plt.figure(figsize=[10, 6])
df.timeseries.plot()
plt.title('Time Series (Additive Model)', fontsize=16)
plt.ylim(-12, 55)
plt.show()
```

시계열 분석 개요(the nature of time series analysis)

추세변동(trend variation)

```
# -- Trend variation
#timestamp = np.arange(len(dates))
#trend_factor = timestamp*1.1


plt.figure(figsize=[10, 6])
df.trend.plot()
plt.title('Trend Factor', fontsize=16)
plt.ylim(-12, 55)
plt.show()
```

시계열 분석 개요(the nature of time series analysis)

순환변동(cyclical variation)

- 추세변동은 장기적으로(일반적으로 1년 초과) 나타나는 추세경향이지만, 순환변동은 대체로 2~3년 정도의 일정한 기간을 주기로 순환적으로 나타남.

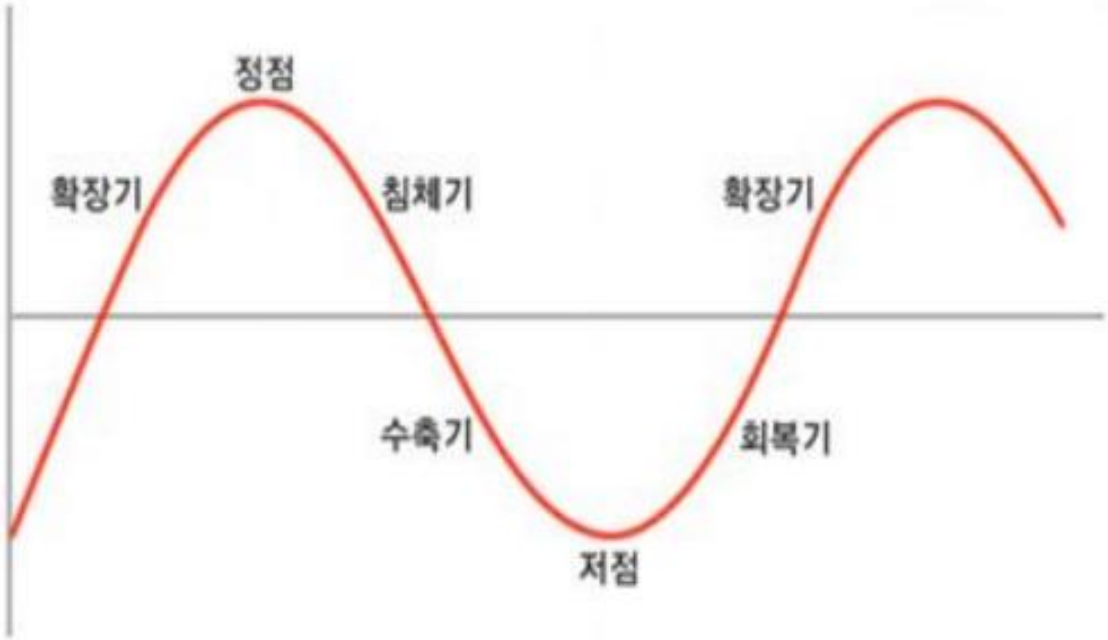
- 즉, 1년 이내의 주기로 곡선을 그리며 추세변동에 따라 변동하는 것을 말함.
- 시간의 경과(흐름)에 따라 상하로 반복되는 변동으로 추세선을 따라 변화하는 것이 순환변동임.

- 경기변동곡선(business cycle curve)은 불황과 경기회복, 호황과 경기후퇴로 인하여 수년을 주기로 나타나고 있는데 순환변동을 나타내는 좋은 예임.

  예 : 경기변동 등

시계열 분석 개요(the nature of time series analysis)

순환변동(cyclical variation)



시계열 분석 개요(the nature of time series analysis)

순환변동(cyclical variation)

```
# 4년 주기
# -- Cycle variation
#cycle_factor = 10*np.sin(np.linspace(0, 3.14*2, 48))


plt.figure(figsize=[10, 6])
df.cycle.plot()
plt.title('Cycle Factor', fontsize=16)
plt.ylim(-12, 55)
plt.show()
```

시계열 분석 개요(the nature of time series analysis)

계절변동(seasonal variation)

- 시계열자료에서 보통 계절적 영향과 사회적 관습에 따라 1년 주기로 발생하는 변동요인을 계절변동이라 하고, 보통계절에 따라 순환하며 변동하는 특성을 지님.

  예 : 설, 추석 등 명절요인 등

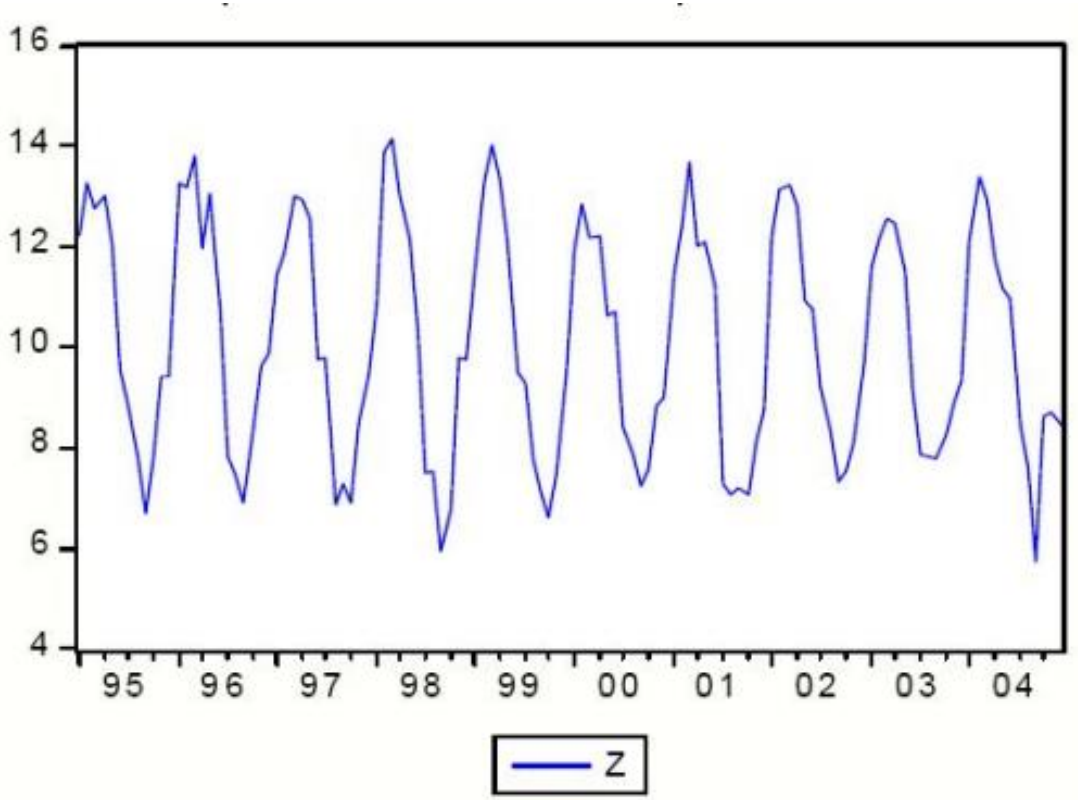
- 그런데 계절변동이 순환변동과 다른 점은 순환주기가 짧다는 점임.

- 그러나 대부분의 경제관련 시계열들은 추세와 계절요인을 동시에 포함함.

- 이는 경제성장에 따라 백화점의 판매액, 해외여행자수, 청량음료, 전력소비량 등과 같이 계절상품 판매량 자료들이 시간의 변화에 따라 증가하기 때문임.

시계열 분석 개요(the nature of time series analysis)

계절변동(seasonal variation)



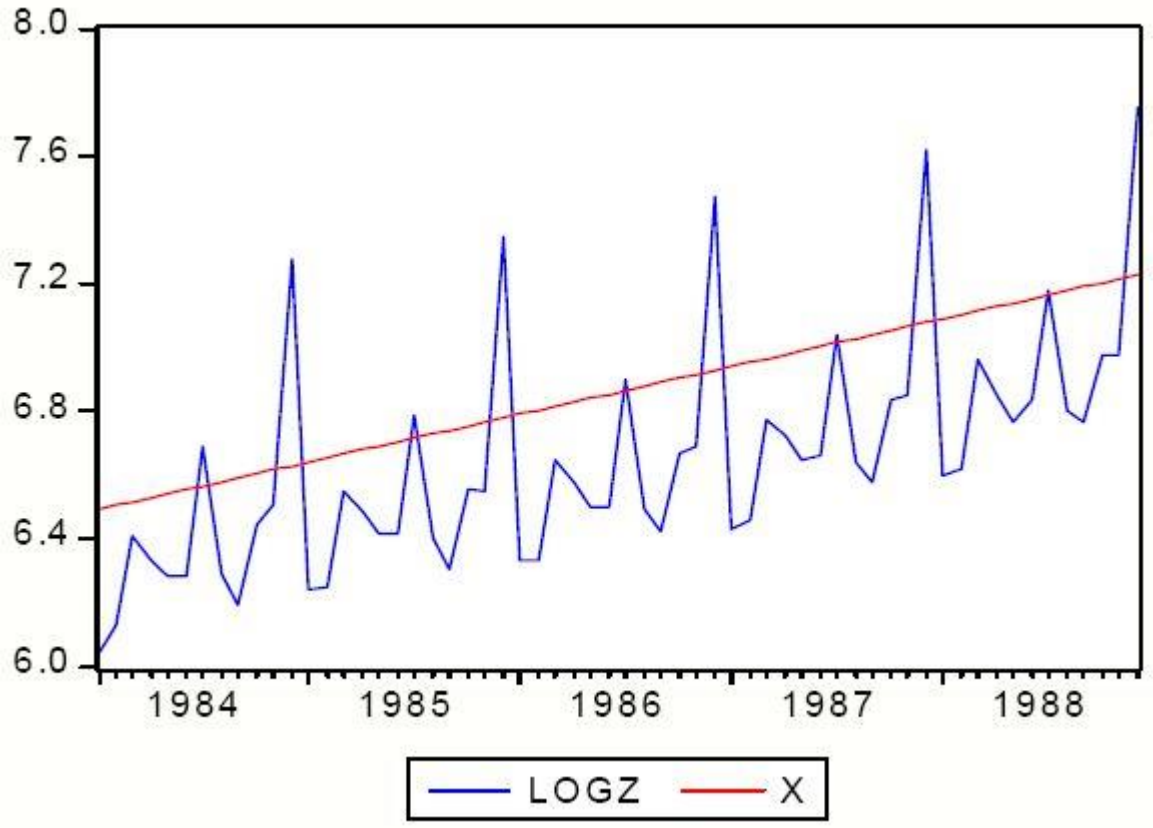
시계열 분석 개요(the nature of time series analysis)

순환변동(cyclical variation)

```
# -- Seasonal factor
#seasonal_factor = 7*np.sin(np.linspace(0, 3.14*8, 48))


plt.figure(figsize=[10, 6])
df.seasonal.plot()
plt.title('Seasonal Factor', fontsize=16)
plt.ylim(-12, 55)
plt.show()
```

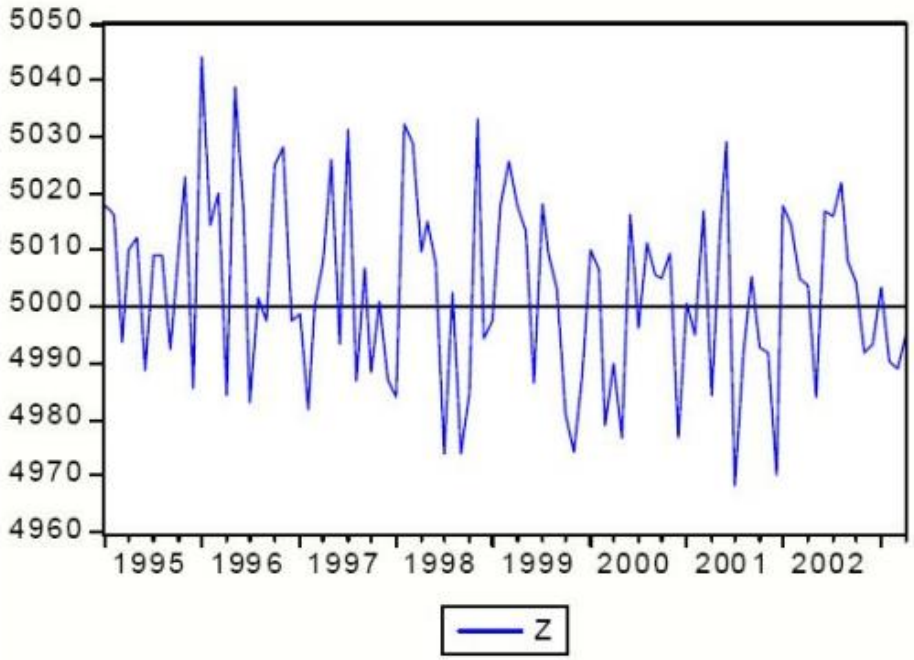
시계열 분석 개요(the nature of time series analysis)

추세와 계절변동요인을 갖는 시계열



시계열 분석 개요(the nature of time series analysis)

불규칙변동(irregular variation or random variation)



시계열자료에서 어떤 규칙성이 없어 예측 불가능하게 우연적으로 발생하는 변동을 말한다. 즉, 시계열 자료에서 위 세 가지 변동 요인을 조정한 후에 나타나는 변동이 불규칙 변동이다. 시계열 자료로 예측을 할 때, 불규칙 변동이 많이 존재하면 신뢰성이 있는 예측을 하기 어렵다.

평활화 기법(Smoothing Methods)

- 데이터 셋를 모델링 하기 전에 기술통계와 시각화로 데이터 셋을 탐색하는 과정이 있듯이, 시계열(time-series)에서도 복잡한 모델 구성에 앞서 수치나 시각화로 시계열을 기술하는 일이 분석작업의 출발점이다.

- 평활화(smoothing)는 분석작업 중 하나로, 일반적인 시계열의 복잡한 추세(trend)를 명확하게 파악하기 위한 방법이다.

- 시계열은 전형적으로, 명백한 불규칙(or 오차)성분을 포함한다.

- 시계열 자료의 특정 패턴을 파악하기 위해, 이같은 급격한 파동을 줄이는 평활화(smoothing) 곡선 플롯(curve-plot)으로 변환시키는 방법이 평활법이다. 대표적인 평활법은 이동평균법과 지수평활법이 있다.

평활화 기법(Smoothing Methods)

▪ 이동평균법(moving average method)

- 시계열을 평활화하는 가장 단순한 방법은 이동평균(moving average)을 사용하는 방법이다.

- 시계열 자료의 특정시점(a time point) 관측치와 이 관측치의 이전과 이후 관측치의 평균으로 대체하는 방법을 '중심이동평균'(centered moving average)라고 한다. 쉽게 말해, 한 시점 앞 뒤 관측치를 평균내는 방법이다. 따라서 이동평균법을 하면 전체 관측치의 개수가 줄어 든다.

- 예를 들어 n=3이면 3기간 단순이동평균(M3), n=5이면 5기간 단순이동평균(M5), n=10이면 10기간 단순이동 평균(M10)이다.

평활화 기법(Smoothing Methods)

▪ 이동평균법(moving average method)

- 이동평균법을 이용할 때 해결해야 하는 가장 중요한 문제는 이동평균을 계산하기 위해 사용하는 과거자료의 적정개수, 즉 n의 개수를 결정하는 것이다.

- 일반적으로 시계열자료에 뚜렷한 추세가 나타나 있거나 불규칙변동이 심하지 않은 경우에는 작은 n의 개수를 사용하고, 그렇지 않은 경우에는 n의 개수를 크게 한다.

$$M_t = \frac{Z_t + Z_{t-1} + \cdots + Z_{t-n+1}}{n}$$

최근 n개의 관측값 Zt , Zt-1 , ... , Zt-n+1을 이용하여 계산한 이동평균이다.

평활화 기법(Smoothing Methods)

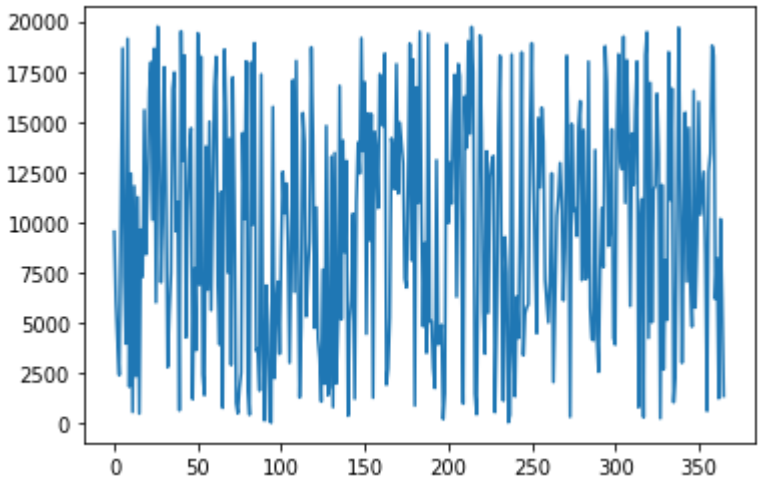
Simple Moving Average (SMA)

- 단순 이동 평균은 가장 일반적인 평균 유형이다.
- SMA에서는 최근 데이터 포인트의 합계를 수행하고 기간별로 나눈다.
- 슬라이딩 너비의 값이 클수록 데이터가 더 평활해지지만, 값이 크면 정확도가 떨어질 수 있다.
- SMA를 계산하기 위해 pandas의 Series.rolling() 메서드를 사용한다.

평활화 기법(Smoothing Methods)

```
import pandas as pd

df['brandA'].plot()
```

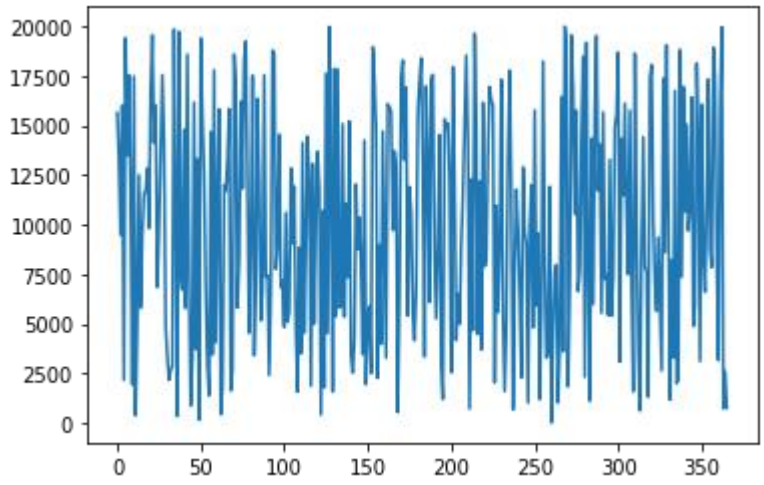
	Date	brandA	brandB
0	01-01-2020	9525	15644
1	02-01-2020	6021	13394
2	03-01-2020	4035	9492
3	04-01-2020	2388	15999
4	05-01-2020	10299	2170



평활화 기법(Smoothing Methods)

```
import pandas as pd

df['brandB'].plot()
```

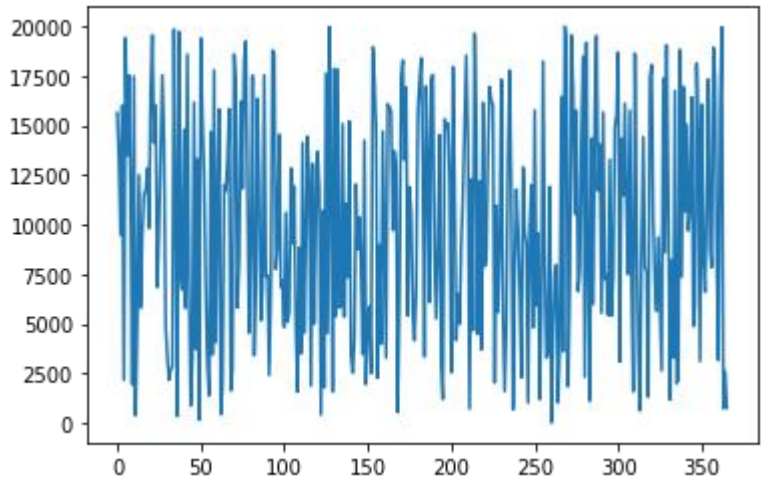
	Date	brandA	brandB
0	01-01-2020	9525	15644
1	02-01-2020	6021	13394
2	03-01-2020	4035	9492
3	04-01-2020	2388	15999
4	05-01-2020	10299	2170



평활화 기법(Smoothing Methods)

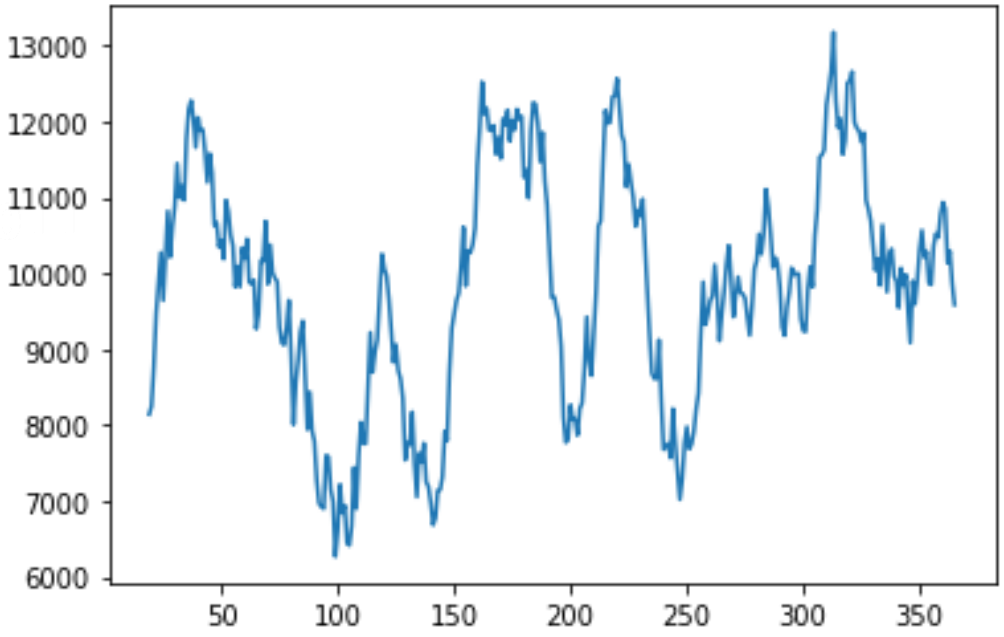
```
import pandas as pd

df['brandB'].plot()
```

	Date	brandA	brandB
0	01-01-2020	9525	15644
1	02-01-2020	6021	13394
2	03-01-2020	4035	9492
3	04-01-2020	2388	15999
4	05-01-2020	10299	2170



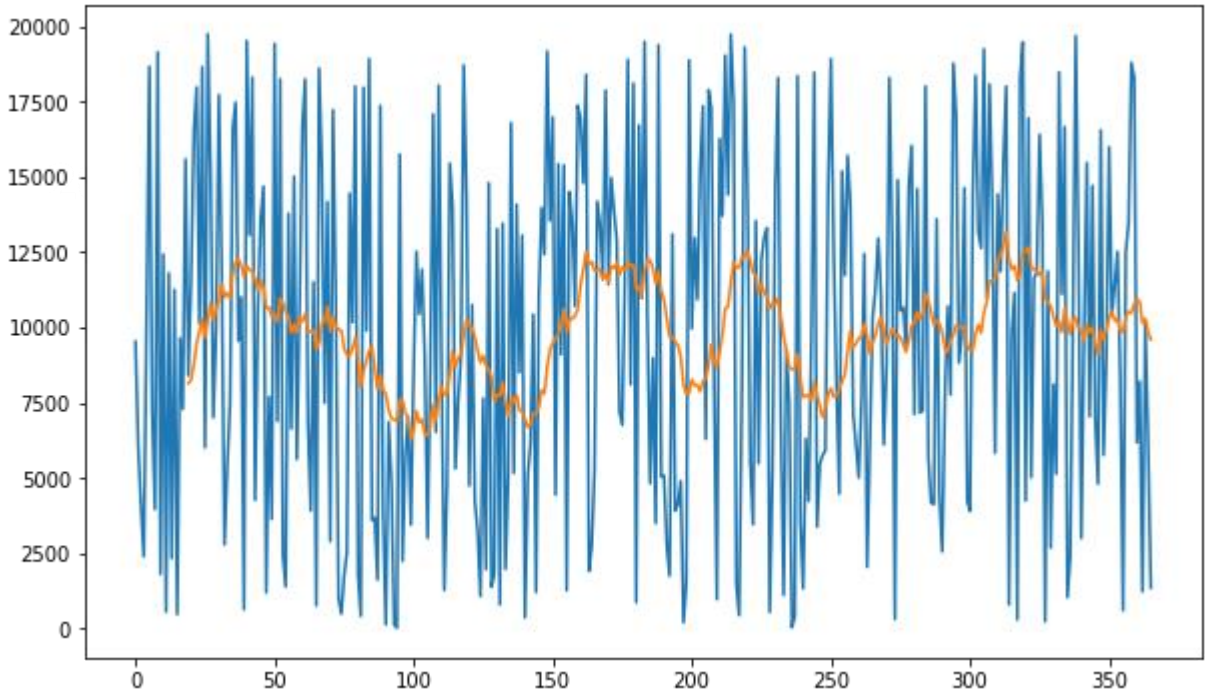
평활화 기법(Smoothing Methods)

```
import pandas as pd

df['brandA'].rolling(window =20).mean().plot()
```



평활화 기법(Smoothing Methods)

```
import pandas as pd

df['brandA'].plot(figsize=(10,6))

df['brandA'].rolling(window =20).mean().plot()
```



평활화 기법(Smoothing Methods)

애플 주가 분석

다음과 같이 pandas 메소드를 쓰면 일요일이 주단위로 8개 출력된다.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime
%matplotlib inline


# 2020년 8월 부터 일요일 8개를 조회
# start : 시작일, periods : 생성할 날짜의 개수, freq : 생성할 날짜의 주기
pd.date_range(start="2020-08", periods=8, freq="W")
```

평활화 기법(Smoothing Methods)

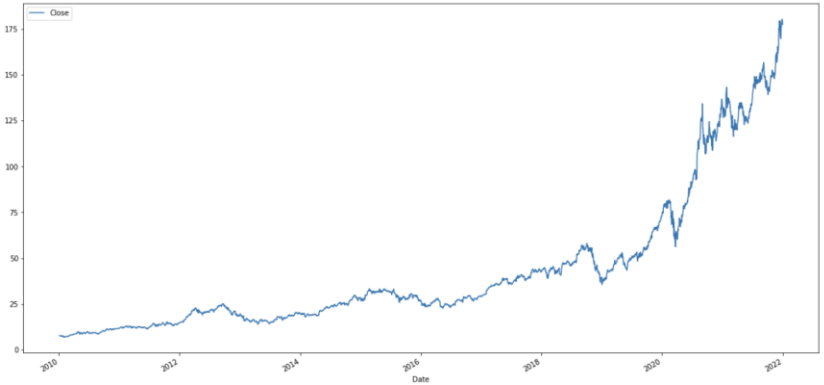
데이터 프레임 컬럼으로 사용했을 때와 3주치 평균 컬럼을 추가

```
df = pd.DataFrame({
    "week":pd.date_range(start="2020-08", periods=8, freq="W"),
    "sales":[39,44,40,45,38,43,39,np.nan],
    "3MA":[0]*8
})
# 0~2주차 평균을 3주차에 shift해서 적용
df["3MA"] = df[["sales"]].rolling(3).mean().shift(1)
df
```

평활화 기법(Smoothing Methods)

Simple Moving Average를 이용한 월별 애플사 주식가격 예측

```
import FinanceDataReader as fdr
# 2010년~현재까지의 애플 주가를 데이터 프레임으로 불러오기
df_apple = fdr.DataReader('AAPL', start = '2010')

# 가장 마지막(최신)의 10일치 주가 출력
df_apple.tail(10)
```

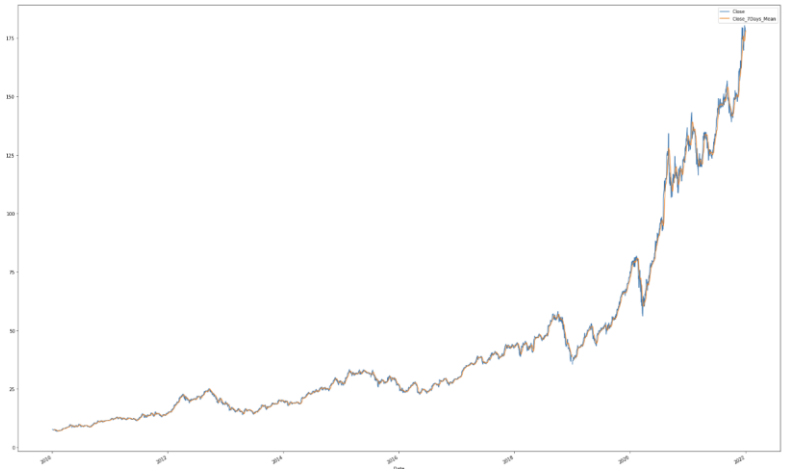



```
df_apple[['Close']].plot(figsize=(20,10))
df_apple['Close_7Days_Mean'] = df_apple['Close'].rolling(7).mean()
plt.title('Close Price for Apple')
```

평활화 기법(Smoothing Methods)

2010~20222년까지 애플의 종가 그래프이다. 7일전 평균값을 shift 하여 이동평균 그래프를 추가 7일평균값으로 부드러워진 곡선 그래프를 볼 수 있다.

```
last_day = datetime(2022, 1, 2)
df_apple.loc[last_day, "Close"] = np.nan
df_apple['Close_7Days_Mean'] = df_apple['Close'].rolling(7).mean().shift(1)
df_apple[['Close', 'Close_7Days_Mean']].plot(figsize=(30,20))
# 7일전 평균값 그래프 + 종가 그래프
```



평활화 기법(Smoothing Methods)

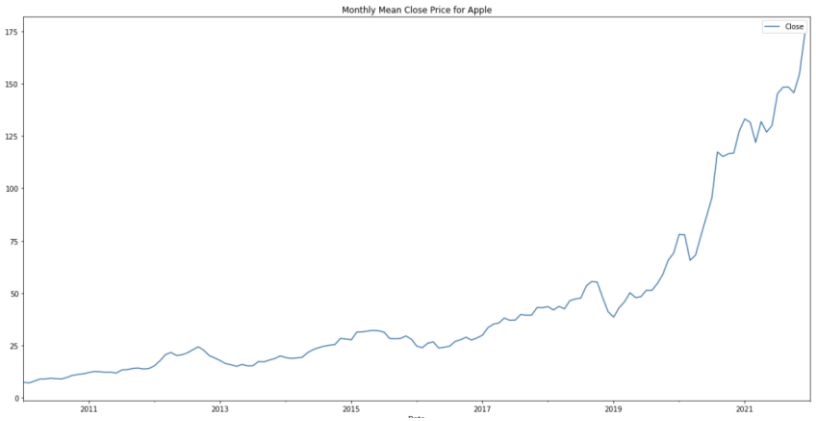
pandas dataframe에는 resample이라는 데이터프레임의 시계열 인덱스 기준으로 샘플링을 편하게 해주는 메소드가 있다. 아래와 같이 하면 월단위로 시계열 데이터를 다시 만들어 준다.

```
# 월단위로 주식 가격의 평균을 샘플링
df_apple_monthly = df_apple.resample(rule='M').mean()
# 마지막 컬럼(Close_7Days_Mean) 제외
df_apple_monthly = df_apple_monthly.iloc[:,:-1]
# 월별 주가(종가)를 시각화
df_apple_monthly[['Close']].plot(figsize=(20,10))
plt.title('Monthly Mean Close Price for Apple')
```

평활화 기법(Smoothing Methods)

pandas dataframe에는 resample이라는 데이터프레임의 시계열 인덱스 기준으로 샘플링을 편하게 해주는 메소드가 있다. 아래와 같이 하면 월단위로 시계열 데이터를 다시 만들어 준다.

```
# 월단위로 주식 가격의 평균을 샘플링
df_apple_monthly = df_apple.resample(rule='M').mean()
# 마지막 컬럼(Close_7Days_Mean) 제외
df_apple_monthly = df_apple_monthly.iloc[:,:-1]
# 월별 주가(종가)를 시각화
df_apple_monthly[['Close']].plot(figsize=(20,10))
plt.title('Monthly Mean Close Price for Apple')
```



평활화 기법(Smoothing Methods)

월단위 평균값을 또 3개월치씩 이동평균을 적용하는 코드

```
df_apple_monthly[['Close_3Month_Mean']] =
df_apple_monthly[['Close']].rolling(3).mean().shift(1)
df_apple_monthly[['Close', 'Close_3Month_Mean']].plot(figsize=(15,20))
```

평활화 기법(Smoothing Methods)

Exponential Moving Average (EMA)

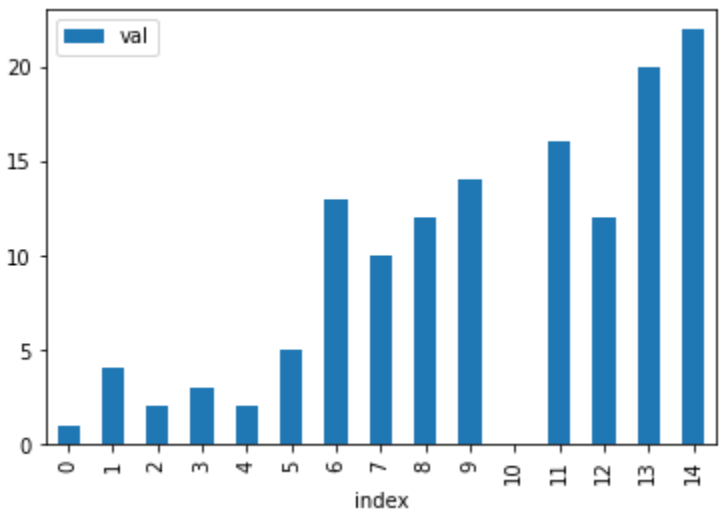
- EMA는 새로운 데이터에 더 많은 가중치를 부여하여 최근 데이터에 더욱 초점을 맞춘다.
- EMA의 주요 아이디어는 이전 데이터보다 최근 데이터를 더 선호하는 것이다.
- 데이터가 오래될수록 데이터에 할당된 가중치가 줄어듭니다. 이 때문에 EMA는 모든 값에 동일한 가중치가 주어지는 SMA에 비해 추세 변화에 더 민감하다.
- SMA를 계산하기 위해 pandas의 pandas.Series.ewm() 메서드를 사용한다.

평활화 기법(Smoothing Methods)

Exponential Moving Average (EMA)

df.ewm(com=None, span=None, halflife=None, alpha=None, min_periods=0, adjust=True, ignore_na=False, axis=0, times=None, method='single')

```
import pandas as pd

data = {'val':[1,4,2,3,2,5,13,10,12,14,np.NaN,16,12,20,22]}

df = pd.DataFrame(data).reset_index()

# df['val'].plot.bar(rot=0, subplots=True)

df.plot(kind='bar',x='index',y='val')
```



평활화 기법(Smoothing Methods)

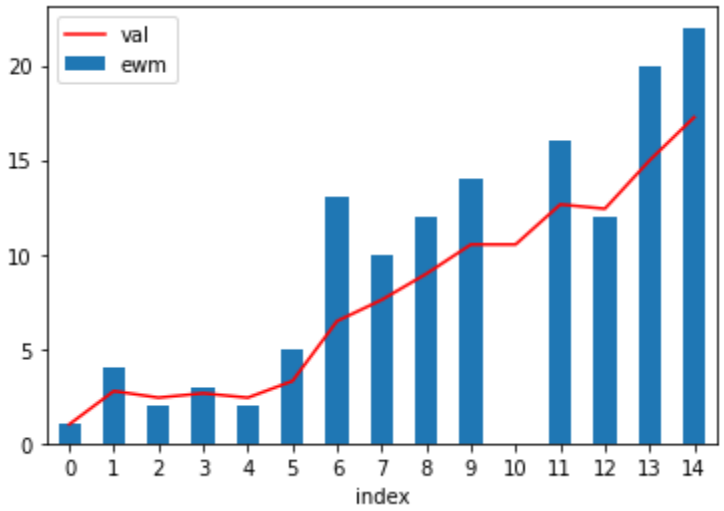
Exponential Moving Average (EMA)

df.ewm(com=None, span=None, halflife=None, alpha=None, min_periods=0, adjust=True, ignore_na=False, axis=0, times=None, method='single')

```
import matplotlib.pyplot as plt

df2 = df.assign(ewm=df['val'].ewm(alpha=0.3).mean()) # val열에 ewm 메서드적용 후 df에 추가
ax = df.plot(kind='bar',x='index',y='val') # ax에 df의 bar chart 생성
ax2= df2.plot(kind='line',x='index', y='ewm', color='red', ax=ax) # ax2에 df2의 line chart 생성후 ax에 추가
plt.show() # 그래프 출력
```

평활화 기법(Smoothing Methods)

Exponential Moving Average (EMA)



평활화 기법(Smoothing Methods)

•myEWMA는 지수이동평균값을 df.ewm(span=3).mean()과 같이 계산해주도록 정의한 메소드

```
import pandas as pd
import numpy as np

df = pd.DataFrame({
    "week":pd.date_range(start="2020-08", periods=8, freq="W"),
    "sales":[39,44,40,45,38,43,39,np.nan],
    "3EMA":[0]*8
})

# 지수 이동 평균을 계산할 함수
# data: 지수 이동 평균을 계산할 데이터
# span: 지수 이동 평균의 거리 (강의 자료에서는 3주마다의 지수 이동 평균이므로 3)
```

평활화 기법(Smoothing Methods)

```
def myEWMA(data, span):
  # 지수 이동 평균을 계산해서 저장할 리스트
  ewma=[0]*len(data)
  # 지수 이동 평균의 분자
  molecule=0
  # 지수 이동 평균의 분모
  denominator=0
  # 값에 곱해지는 가중치
  alpha = 2.0 / (1.0 + span)
  for i in range(len(data)):
    # 분자 계산 data+(1-alpha)앞의 데이터
    molecule = (data[i] + (1.0-alpha)*molecule)
    # 분모 계산 (1-alpha)의 i승
    denominator+=(1-alpha)**i
    print("index:",i)
    print("molecule:",molecule)
    print("denominator:",denominator)
    # 지수 이동 평균 계산
    ewma[i] = molecule/denominator
    print("ewma",ewma[i])
    print("="*100)
  return ewma
```

평활화 기법(Smoothing Methods)

Apple 주식을 SMA 방식과 같이 분석

- 전에 코드를 활용하여 본다.

평활화 기법(Smoothing Methods)

Cumulative Moving Average (CMA)

- 누적 이동 평균은 SMA와 같이 현재 시간 't'까지의 모든 데이터의 평균이며 가중치가 없는 평균이다.
- 모든 값에 동일한 가중치가 할당된다.
- 창 크기가 일정한 SMA와 달리 CMA에서는 창 폭이 지속 시간이 길수록 커진다.
- CMA를 계산하기 위해 pandas의 Series.expanding() 메서드를 사용한다.

평활화 기법(Smoothing Methods)

▪ 이동평균법(moving average method)

- 이동평균법을 이용할 때 해결해야 하는 가장 중요한 문제는 이동평균을 계산하기 위해 사용하는 과거자료의 적정개수, 즉 n의 개수를 결정하는 것이다.

- 일반적으로 시계열자료에 뚜렷한 추세가 나타나 있거나 불규칙변동이 심하지 않은 경우에는 작은 n의 개수를 사용하고, 그렇지 않은 경우에는 n의 개수를 크게 한다.

$$M_t = \frac{Z_t + Z_{t-1} + \cdots + Z_{t-n+1}}{n}$$

최근 n개의 관측값 Zt , Zt-1 , ... , Zt-n+1을 이용하여 계산한 이동평균이다.

자기상관과 부분 자기상관

시차를 적용한 시계열 데이터

시차를 적용한 나일강 유량 시계열 데이터. 주어진 시차만큼 관측값이 뒤의 연도로 이동한다.

시차	1868	1869	1870	1871	1872	1873	1874	1875	1876	…
0				1120	1160	963	1210	1160	1160	…
1			1120	1160	963	1210	1160	1160	813	…
2		1120	1160	963	1210	1160	1160	813	1230	…
3	1120	1160	963	1210	1160	1160	813	1230	1370	…

- 자기상관은 시차(lag)를 적용한 시계열 데이터를 이용하여 계산
- 시차를 적용한다는 것은 특정 시차만큼 관측값을 뒤로(즉 과거의 시점으로) 이동시키는 것을 의미

자기상관과 부분 자기상관

자기상관함수(ACF; AutoCorrelation Function)

자기상관함수

- 자기상관은 다른 시점의 관측값 간 상호 연관성을 나타내므로 이는 시차를 적용한 시계열 데이터 간의 상관관계를 의미

- 자기상관 AC_k는 원래의 시계열 데이터(y_t)와 k 시차가 고려된, 즉 k 기간 뒤로 이동한 시계열 데이터(y_{t-k}) 간의 상관관계로 정의
  - 예를 들어, AC_1은 시차0 시계열 데이터와 시차1 시계열 데이터 간의 상관관계
  - AC_0는 동일한 시계열 데이터 간의 상관관계이므로 항상 1

자기상관과 부분 자기상관

자기상관함수(계속)

- 시차에 따른 일련의 자기상관 $\{AC_1, AC_2, \cdots, AC_k\}$를 자기상관함수(autocorrelation function, ACF)라고 함
- ACF는 시차에 따른 관측값 간의 연관 정도를 보여주며, 시차가 커질수록 ACF는 점차 0에 가까워지게 됨
- ACF는 시계열의 정상성을 평가할 때 유용
  - 정상 시계열의 ACF는 상대적으로 빨리 0으로 접근함
  - 비정상 시계열의 ACF는 천천히 감소하며 종종 큰 양의 값을 가짐

자기상관과 부분 자기상관

자기상관함수(계속)

- 시차에 따른 일련의 자기상관 $\{AC_1, AC_2, \cdots, AC_k\}$를 자기상관함수(autocorrelation function, ACF)라고 함
- ACF는 시차에 따른 관측값 간의 연관 정도를 보여주며, 시차가 커질수록 ACF는 점차 0에 가까워지게 됨
- ACF는 시계열의 정상성을 평가할 때 유용
  - 정상 시계열의 ACF는 상대적으로 빨리 0으로 접근함
  - 비정상 시계열의 ACF는 천천히 감소하며 종종 큰 양의 값을 가짐

자기상관과 부분 자기상관

편자기상관함수

- 편자기상관(partial autocorrelation)은 시차가 다른 두 시계열 데이터 간의 순수한 상호 연관성을 나타냄
- 편자기상관 PAC_k는 원래의 시계열 데이터(y_t)와 시차k 시계열 데이터(y_{t-k}) 간의 순수한 상관관계로서 두 시점 사이에 포함된 모든 시계열 데이터($y_{t-1}, y_{t-2}, \cdots, y_{t-k+1}$)의 영향은 제거됨
- 시차에 따른 일련의 편자기상관 $\{PAC_1, PAC_2, \cdots, PAC_k\}$를 편자기상관함수(partial autocorrelation function, PACF)라고 함

자기상관과 부분 자기상관

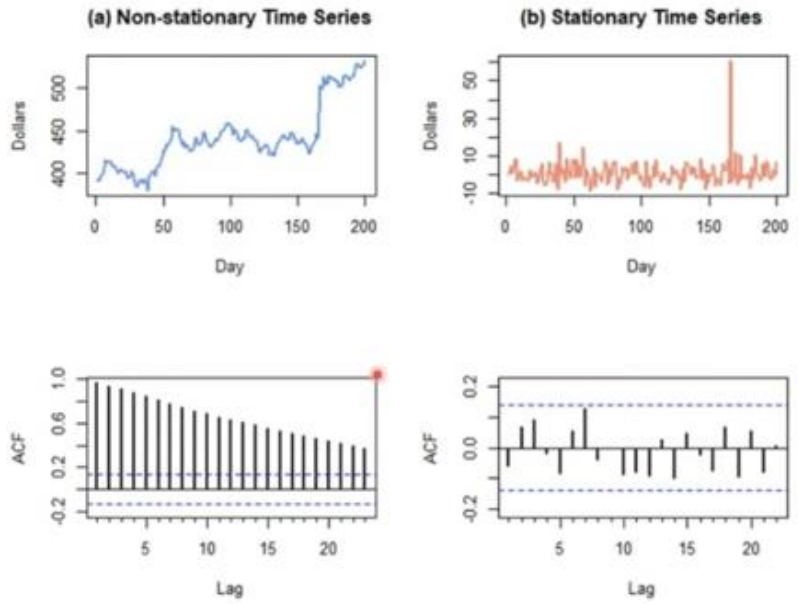
ACF도표와 PACF도표

- ACF도표(ACF plot)와 PACF도표(PACF plot)는 ACF와 PACF를 그래프로 시각화

- 활용
  - 시계열 데이터의 정상성 평가
  - ARIMA모델의 패러미터 결정 및 모델의 적합도 평가

자기상관과 부분 자기상관

ACF도표를 이용한 정상성 평가



- (a)의 비정상 시계열에 대응되는 ACF도표는 자기상관이 크고 양수이며 천천히 감소하는 패턴을 보여줌
- (b)의 정상 시계열에 대응되는 ACF도표는 모든 시차에서 0에 근접한 자기상관을 보여줌

자기상관과 부분 자기상관

정상 시계열로의 변환

- 변동폭이 일정하지 않으면 로그 변환을 통해 시간의 흐름에 따라 분산이 일정하게 유지되는 정상 시계열로 변환
- 추세나 계절적 요인이 관찰되면 차분(differencing, 시계열 y_t의 각 관측값을 $y_t - y_{t-1}$로 대체) 과정을 통해 전 기간에 걸쳐 평균이 일정한 정상 시계열로 변환
- 변동폭이 일정하지 않고 추세와 계절적 요인 또한 존재하면 로그 변환과 차분 과정을 모두 적용하여 정상 시계열로 변환

자기상관과 부분 자기상관

정상 시계열로의 변환

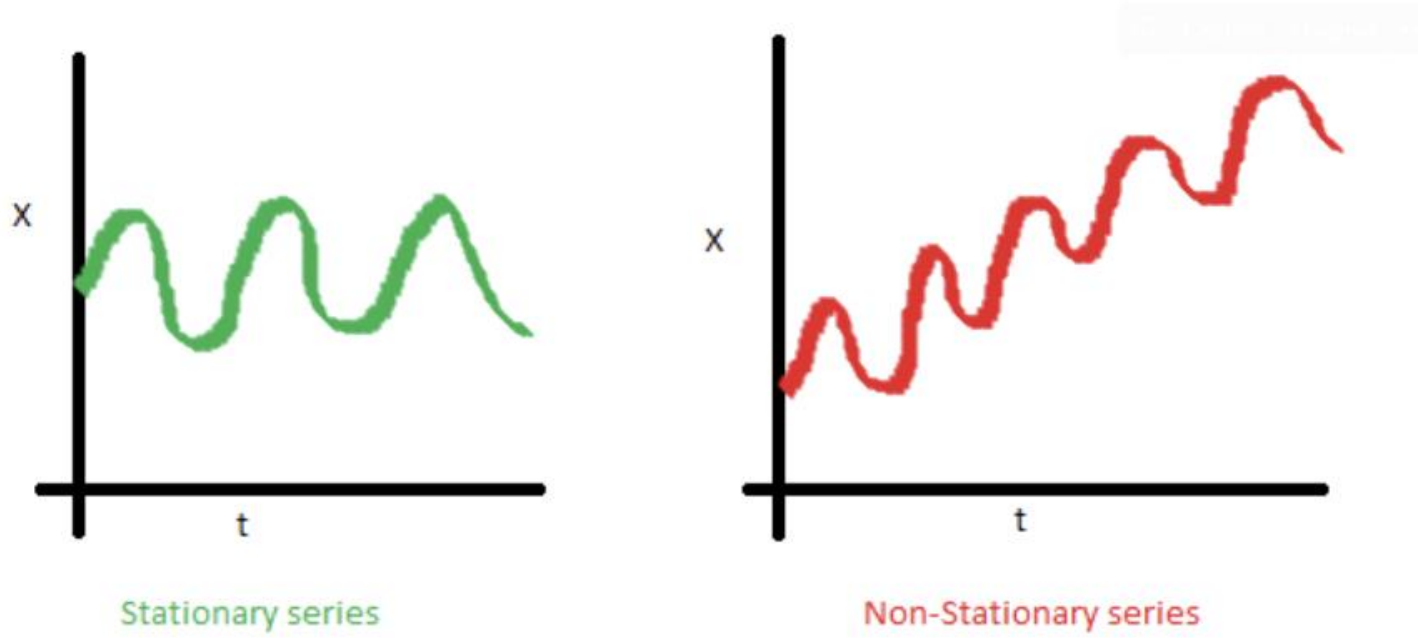
- 변동폭이 일정하지 않으면 로그 변환을 통해 시간의 흐름에 따라 분산이 일정하게 유지되는 정상 시계열로 변환
- 추세나 계절적 요인이 관찰되면 차분(differencing, 시계열 y_t의 각 관측값을 $y_t - y_{t-1}$로 대체) 과정을 통해 전 기간에 걸쳐 평균이 일정한 정상 시계열로 변환
- 변동폭이 일정하지 않고 추세와 계절적 요인 또한 존재하면 로그 변환과 차분 과정을 모두 적용하여 정상 시계열로 변환

정상성(stationarity)과 차분(differencing)

- **정상성**
  - 정태성이라고도 하며, 일정하여 늘 한결같은 성질을 뜻한다.
  - 시계열에서 정상성이 있다는 것은 추세나 동향이 없는 상태로, 시계열의 평균이 시간 축에 평행하다는 의미다.
  - **시계형 자료가 시계열 모형으로 적합시키기 위한 전제 조건에 해당한다. 즉, 추세와 동향이 있는 상태로는 모형을 만들 수 없다.(다루기가 어렵다)**

정상성(stationarity)과 차분(differencing)

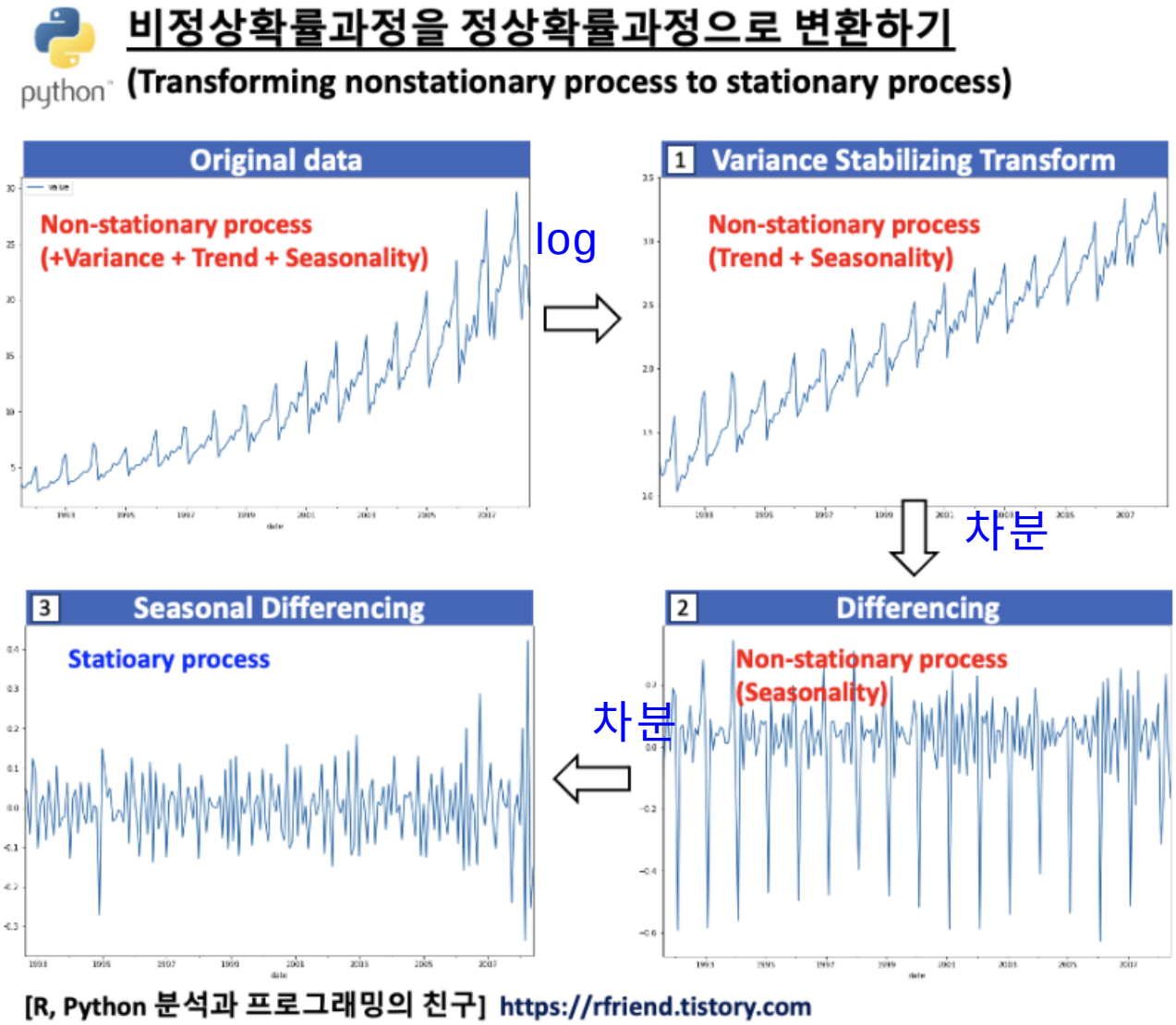
- **정상성**



비정상성 시계열은 정상시계열로 변환해야 하며, 차분(differencing)을 이용한다.

정상성(stationarity)과 차분(differencing)

- 정상성



정상성(stationarity)과 차분(differencing)

- **차분**
  - 차분은 현시점 자료에서 전 시점 자료를 빼는 것을 말한다.
  - 일반차분(regular difference)는 바로 전 시점의 자료를 빼는 것이고, 계절차분(seasonal difference)는 여러 시점 전의 자료를 빼는 것이다.

정상성(stationarity)과 차분(differencing)

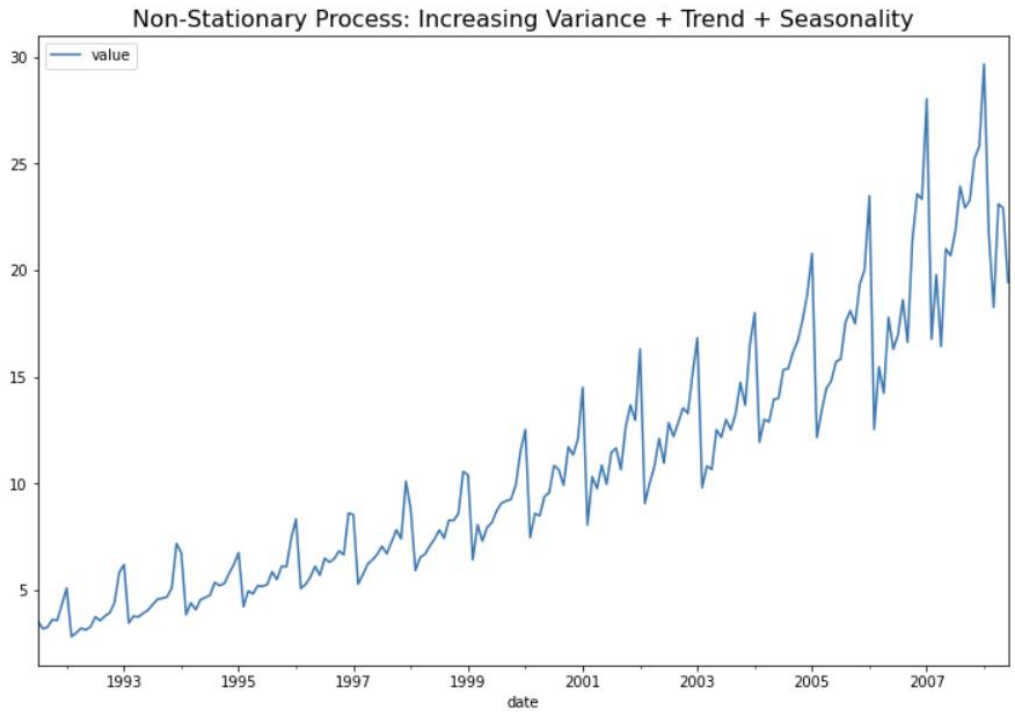
- **차분**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
## getting drug sales dataset
file_path
= 'https://raw.githubusercontent.com/selva86/datasets/master/a10.csv'
df = pd.read_csv(file_path, parse_dates=['date'], index_col='date')
df.head(12)
```

정상성(stationarity)과 차분(differencing)

- **차분**

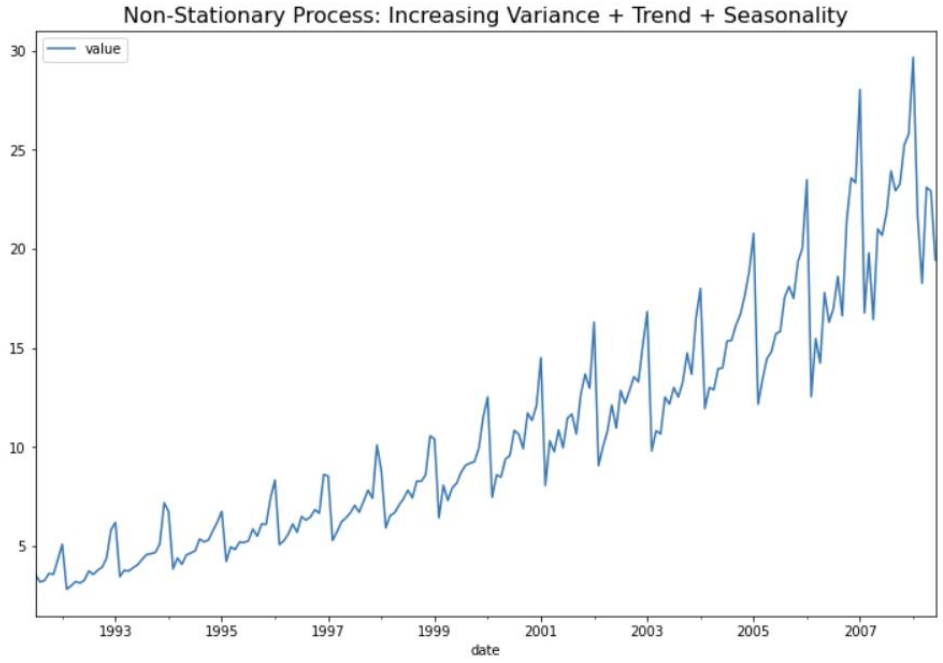
```
df.plot(figsize=[12, 8])
plt.title('Non-Stationary Process: Increasing Variance + Trend + Seasonality',
fontsize=16)
plt.show()
```



정상성(stationarity)과 차분(differencing)

- **차분**

위의 시계열 그래프에서 볼 수 있는 것처럼, (a) 분산이 시간의 흐름에 따라 증가하고 (분산이 고정이 아님), (b) 추세(trend)가 있으며, (c) 1년 주기의 계절성(seasonality)이 있으므로, 비정상확률과정(non-stationary process)이다.



정상성(stationarity)과 차분(differencing)

- **차분**

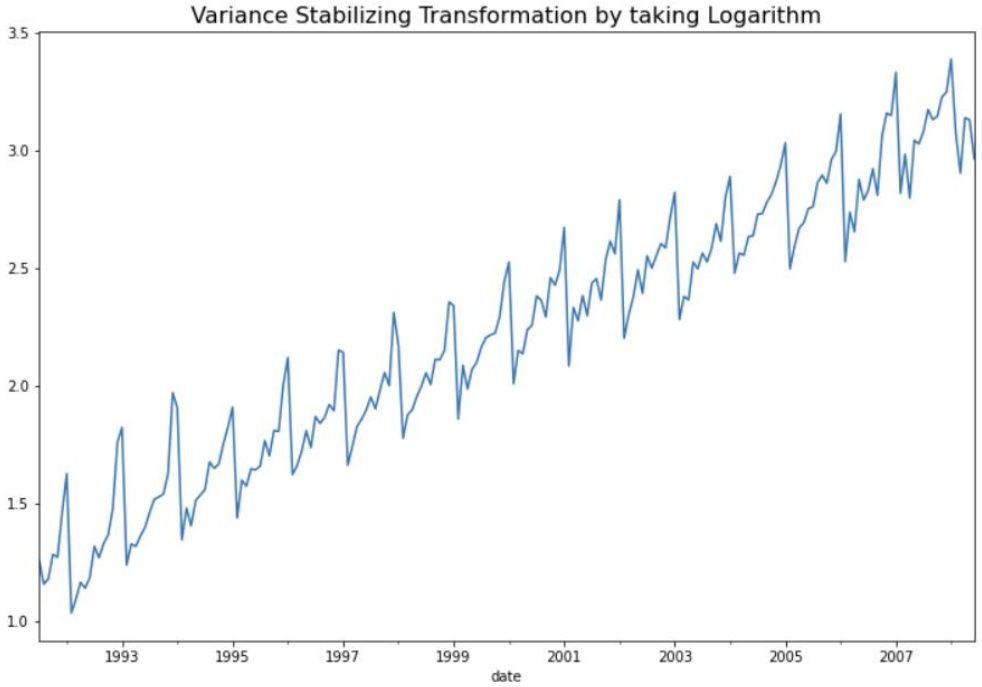
```
## Variance Stabilizing Transformation (VST) by Taking Logarithm
df_vst = np.log(df.value)
df_vst.head()
```

```
Out[24]: date
         1991-07-01    1.260332
         1991-08-01    1.157161
         1991-09-01    1.179338
         1991-10-01    1.283986
         1991-11-01    1.271408
         Name: value, dtype: float64
```

정상성(stationarity)과 차분(differencing)

- **차분**

```
## plotting
df_vst.plot(figsize=(12, 8))
plt.title("Variance Stabilizing Transformation by taking Logarithm", fontsize=16)
plt.show()
```



정상성(stationarity)과 차분(differencing)

- **차분**

추세가 있는 경우 차분을 통한 추세 제거 (de-trend by differencing)

차분(differencing)은 현재의 시계열 값에서 시차 t 만큼의 이전 값을 빼주는 것이다.

1차 차분 = Delta1_Z(t) = Z(t) - Z(t-1)
2차 차분 = Delta2_Z(t) = Z(t) - Z(t-1) - (Z(t-1) - Z(t-2)) = Z(t) - 2Z(t-1) + Z(t-2)

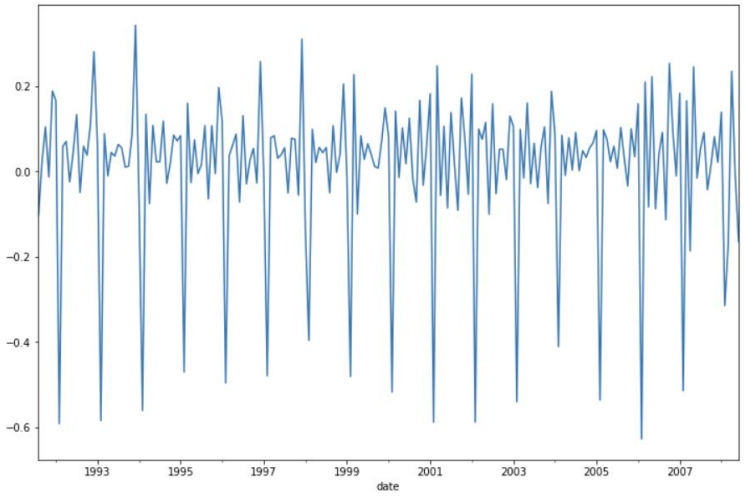
Python의 diff() 메소드를 사용해서 차분을 해줄 수 있다.
이때 차분의 차수 만큼 결측값이 생기는 데요, dropna() 메소드를 사용해서 결측값은 제거

정상성(stationarity)과 차분(differencing)

- **차분**

추세가 있는 경우 차분을 통한 추세 제거 (de-trend by differencing)

```
## De-trend by Differencing
df_vst_diff1 = df_vst.diff(1).dropna()
df_vst_diff1
plt.figure(figsize=(12,8))
df_vst_diff1.plot()
# plt.title("De-trend by 1st order Differencing", fontsize=16)
# plt.show()
```



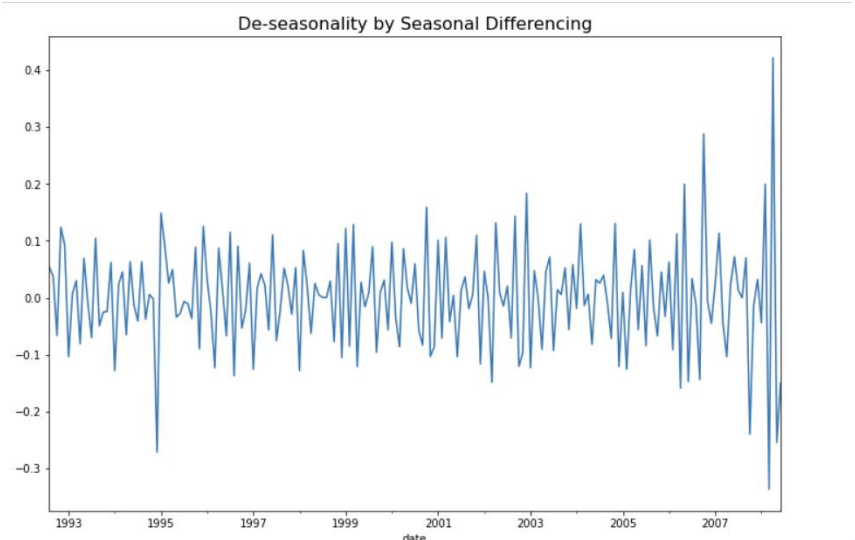
정상성(stationarity)과 차분(differencing)

- **차분**
  - 계절성이 있는 경우 계절 차분을 통한 계절성 제거 (de-seasonality by seaanl differencing)
  - 아직 남아있는 계절성(seasonality)을 계절 차분(seasonal differencing)을 사용해서 제거.
  - 1년 12개월 주기의 계절성을 띠고 있으므로 diff(12) 함수로 계절 차분을 실시하고, 12개의 결측값이 생기는데요 dropna() 로 결측값은 제거

정상성(stationarity)과 차분(differencing)

- **차분**

```
## Stationary Process: De-seasonality by Seasonal Differencing
df_vst_diff1_diff12 = df_vst_diff1.diff(12).dropna()
## plotting
plt.figure(figsize=(12,8))
df_vst_diff1_diff12.plot(
plt.title("De-seasonality by Seasonal Differencing", fontsize=16)
plt.show()
```



정상성(stationarity)과 차분(differencing)

- **차분**

비정상 시계열(non-stationary process)이었던 원래 데이터

(1) log transformation을 통한 분산 안정화

(2) (2) 차분(differencing)을 통한 추세 제거

(3) (3) 계절 차분(seasonal differencing)을 통한 계절성 제거를 모두 마쳐서 정상 시계열(stationary process) 로 변환

시계열 모형

- **시계열 모형 종류**

(1) AR(p) - 자기 회귀 모형

(2) MA(q) - 이동평균 모형

(3) ARMA(p,q)

(4)ARIMA(p,d,q) - 자기회귀누적이동평균 모형

: 차수의 개수(d)는 거의 2를 넘지 않는다.

(5)SARIMA(Seasonal ARIMA) - 계절 자기회귀이동평균 모형

시계열 모형

- **시계열 모형 종류**

(1) AR(p) - 자기 회귀 모형

(2) MA(q) - 이동평균 모형

(3) ARMA(p,q)

(4)ARIMA(p,d,q) - 자기회귀누적이동평균 모형

: 차수의 개수(d)는 거의 2를 넘지 않는다.

(5)SARIMA(Seasonal ARIMA) - 계절 자기회귀이동평균 모형

시계열 모형

▪ AR – 자기회기 모형

- **AR(Autoregressive) 모델**은 자기회귀(Autoregressive) 모델로 자기상관성을 시계열 모델로 구성한 것이다.

- 예측하고 싶은 특정 변수의 과거 자신의 데이터와 선형 결합을 통해 특정 시점 이후 미래값을 예측하는 모델이다.

- 이름 그대로 이전 자신의 데이터가 이후 자신의 미래 관측값에 영향을 끼친다는 것을 기반으로 나온 모델이다.

- **AR(1)**에 적용하기 위해선 **$-1<\varphi 1<1$조건**이 필요하다.

시계열 모형

▪ AR – 자기회기 모형

- **AR(Autoregressive) 모델**은 자기회귀(Autoregressive) 모델로 자기상관성을 시계열 모델로 구성한 것이다.

- 예측하고 싶은 특정 변수의 과거 자신의 데이터와 선형 결합을 통해 특정 시점 이후 미래값을 예측하는 모델이다.

- 이름 그대로 이전 자신의 데이터가 이후 자신의 미래 관측값에 영향을 끼친다는 것을 기반으로 나온 모델이다.

- **AR(1)**에 적용하기 위해선 **$-1<\varphi 1<1$조건**이 필요하다.

시계열 모형

- AR – 자기회기 모형

ArmaProcess(ar = [1,-phi_1, -phi_2, ..., -phi_p], ma = [1])로 생성

```
# ArmaProcess로 모형 생성하고 nobs 만큼 샘플 생성
def gen_arma_samples (ar,ma,nobs):
        arma_model = ArmaProcess(ar=ar, ma=ma) # 모형 정의
        arma_samples = arma_model.generate_sample(nobs) # 샘플 생성
        return arma_samples
# drift가 있는 모형은 ArmaProcess에서 처리가 안 되어서 수동으로 정의해줘야 함
def gen_random_walk_w_drift(nobs,drift):
        init = np.random.normal(size=1, loc = 0)
        e = np.random.normal(size=nobs, scale =1)
        y = np.zeros(nobs)
        y[0] = init
        for t in (1,nobs):
                y[t] = drift + 1 * y[t-1] + e[t]
                return y
```

시계열 모형

- AR – 자기회기 모형

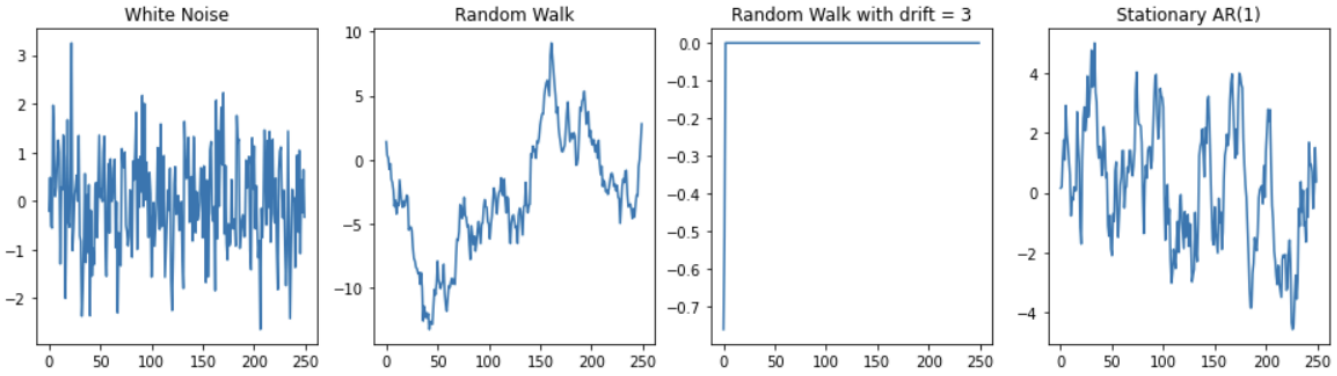
그백색 잡음 모형 (white_noise), 임의 보행 모형 (random_walk),
표류가 있는 임의 보행 모형 (random_walk_w_drift), 정상성을 만족하는 ϕ1=0.9ϕ1=0.9인
AR(1) 모형 (stationary_ar_1)을 각각 250개씩 샘플을 생성하여 그림

```
np.random.seed(12345)

white_noise= gen_arma_samples(ar = [1], ma = [1], nobs = 250)
# y_t = epsilon_t
random_walk  = gen_arma_samples(ar = [1,-1], ma = [1], nobs = 250)
# (1 - L)y_t = epsilon_t
random_walk_w_drift = gen_random_walk_w_drift(250, 2)
# y_t = 2 + y_{t-1} + epsilon_t
stationary_ar_1     = gen_arma_samples(ar = [1,-0.9], ma = [1],nobs=250)
# (1 - 0.9L) y_t = epsilon_t

fig,ax = plt.subplots(1,4)
ax[0].plot(white_noise)
ax[0].set_title("White Noise")
```

시계열 모형

- AR – 자기회기 모형

```
ax[1].plot(random_walk)
ax[1].set_title("Random Walk")

ax[2].plot(random_walk_w_drift)
ax[2].set_title("Random Walk with drift = 3")

ax[3].plot(stationary_ar_1)
ax[3].set_title("Stationary AR(1)")

fig.set_size_inches(16,4)
```



시계열 모형

MA (q) 모형 생성하기

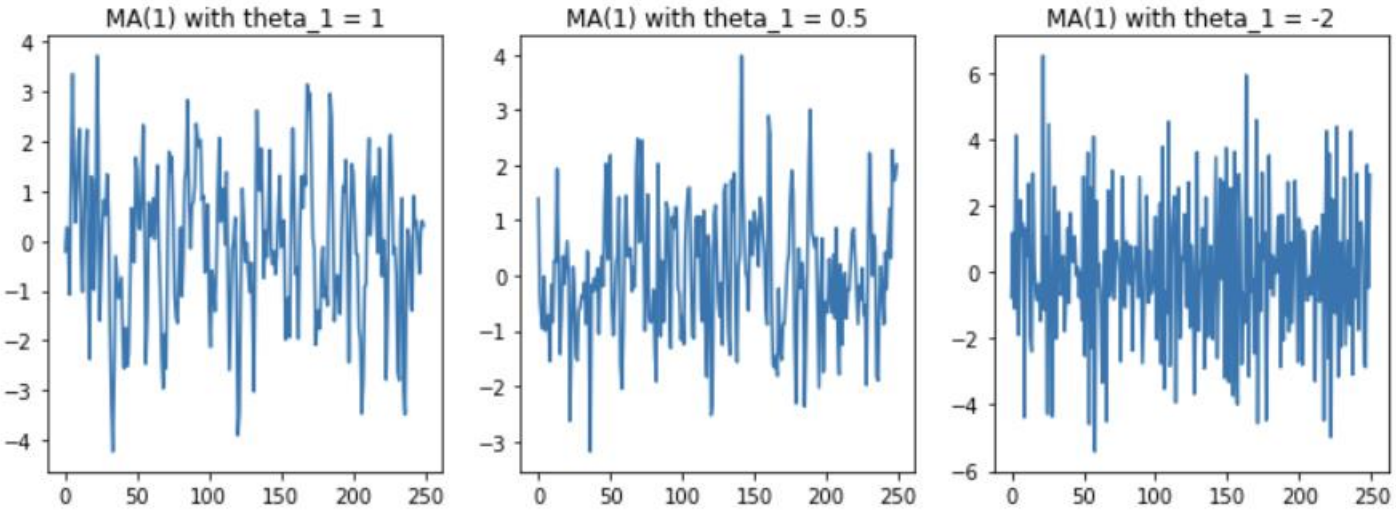
`ArmaProcess(ar = [1], ma = [1, theta_1, theta_2, ..., theta_q])`로 생성

```
np.random.seed(12345)

ma_1  = gen_arma_samples(ar = [1], ma = [1,1], nobs = 250)      # y_t = (1+L) epsilon_t
ma_2  = gen_arma_samples(ar = [1], ma = [1,0.5], nobs = 250)     # y_t = (1+0.5L)epsilon_t
ma_3  = gen_arma_samples(ar = [1], ma = [1,-2], nobs = 250)      # y_t = (1-2L) epsilon_t

fig,ax = plt.subplots(1,3, figsize = (12,4))

ax[0].plot(ma_1)
ax[0].set_title("MA(1) with theta_1 = 1")

ax[1].plot(ma_2)
ax[1].set_title("MA(1) with theta_1 = 0.5")

ax[2].plot(ma_3)
ax[2].set_title("MA(1) with theta_1 = -2")

plt.show()
```

시계열 모형

MA (q) 모형 생성하기

`ArmaProcess(ar = [1], ma = [1, theta_1, theta_2, ..., theta_q])`로 생성



위 그림에서 볼 수 있듯이 θ1의 값에 따라 모형 형태가 약간 다르긴 하지만 특정한 트렌드도 없고 평균과 분산이 일정하기 때문에 정상성을 만족함을 확인하실 수 있다.

시계열 모형

ARIMA (p,d,q) 모형 생성하기

`ArmaProcess(ar = [1,-phi_1, -phi_2, ..., -phi_p], ma = [1, theta_1,theta_2, ..., theta_q])`로 생성 후 `unintegrate(x, level)`

ARIMA (p,d,q)를 따르는 데이터를 생성하기 위해선

- ARMA (p,q) 과정을 따르는 데이터를 생성하고

- 이 데이터가 d 번 차분한 값이기 때문에 원상복귀해주는 `unintegrate` 함수를 사용해야 합니다. `unintegrate (x, level)` 에서 만약 1차 차분이라면 `level = [1]`, 2차 차분이라면 `level = [1,2]`과 같이 정의해주어야 합니다.

시계열 모형

ARIMA (p,d,q) 모형 생성하기

```
np.random.seed(12345)
from statsmodels.tsa.arima_model import unintegrate, unintegrate_levels

arma_1  = gen_arma_samples (ar = [1,-.5], ma = [1,1], nobs = 250) # 차분한 값이 ARMA (1,1)을 따름
arima_1 = unintegrate(arma_1, [1])                                # unintegrate: 차분한 값을 다시 원상 복귀
fig,ax = plt.subplots(1,2, figsize = (16,4))

ax[0].plot(arma_1)
ax[0].set_title("ARMA(1,1) with phi_1 = 0.5, theta_1 = 1")

ax[1].plot(arima_1)
ax[1].set_title("ARIMA(1,1,1) with phi_1 = 0.5, d = 1, theta_1 = 1")
plt.show()
```

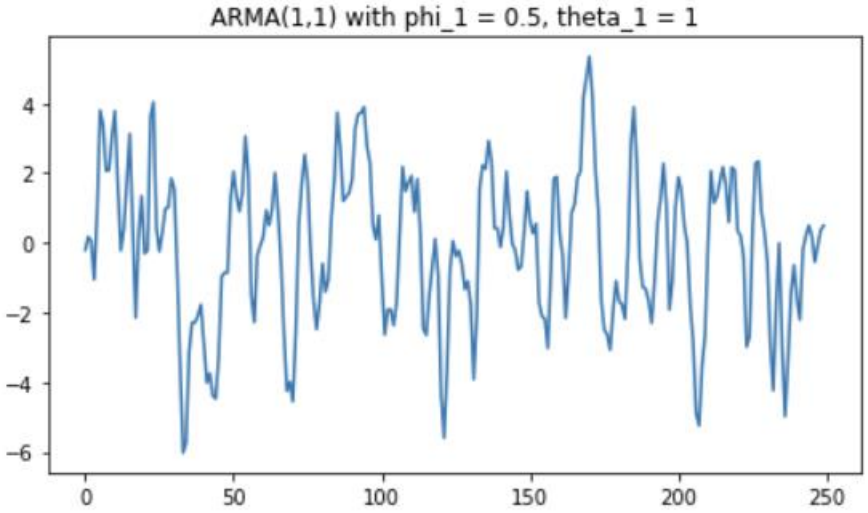
시계열 모형

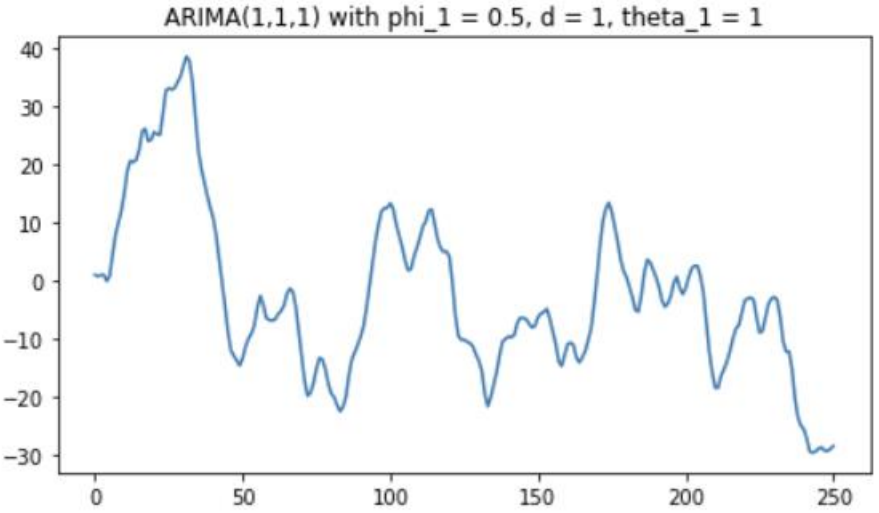
ARIMA (p,d,q) 모형 생성하기

```
np.random.seed(12345)
from statsmodels.tsa.arima_model import unintegrate, unintegrate_levels

arma_1  = gen_arma_samples (ar = [1,-.5], ma = [1,1], nobs = 250) # 차분한 값이 ARMA (1,1)을 따름
arima_1 = unintegrate(arma_1, [1])                                # unintegrate: 차분한 값을 다시 원상 복귀
fig,ax = plt.subplots(1,2, figsize = (16,4))

ax[0].plot(arma_1)
ax[0].set_title("ARMA(1,1) with phi_1 = 0.5, theta_1 = 1")

ax[1].plot(arima_1)
ax[1].set_title("ARIMA(1,1,1) with phi_1 = 0.5, d = 1, theta_1 = 1")
plt.show()
```

시계열 모형

ARIMA (p,d,q) 모형 생성하기





시계열 모형

ARIMA, Python으로 하는 시계열분석 (feat. 비트코인 가격예측)