

# ARIMA & SARIMA

단국대학교 산업공학과

노영후

ywnok999@dankook.ac.kr

# 01 | 시계열 기초

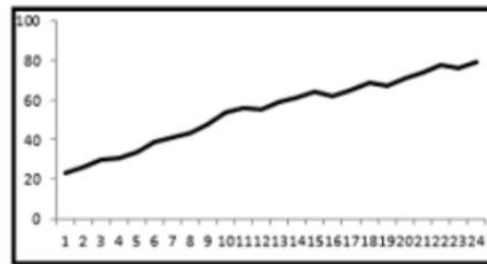


- 시계열 예측
  - ✓ 시계열 데이터를 분석하여 미래 값을 예측하는 분석 기술
- 시계열 데이터
  - ✓ 일련의 시간 순서에 따라 기록된 데이터

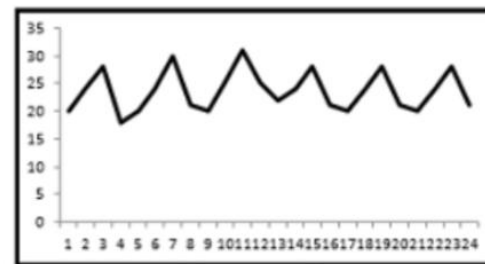
# 01 | 시계열 기초



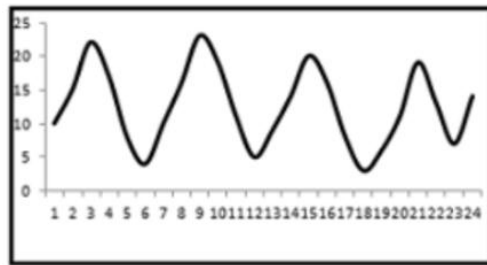
- 시계열 예측
  - ✓ 시계열 데이터를 분석하여 미래 값을 예측하는 분석 기술
- 시계열 데이터
  - ✓ 일련의 시간 순서에 따라 기록된 데이터



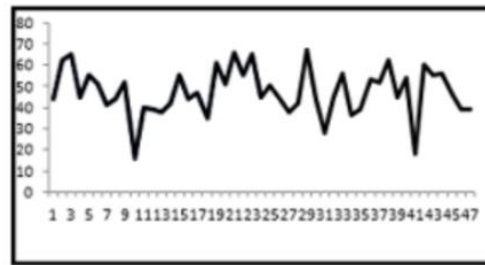
(a) Trend



(b) Seasonality



(c) Cyclicality

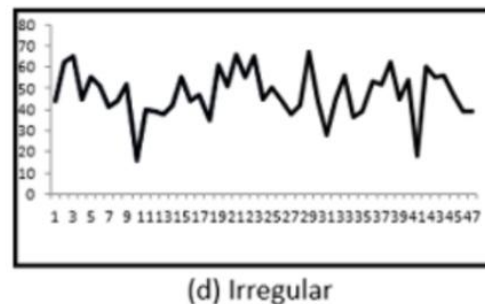
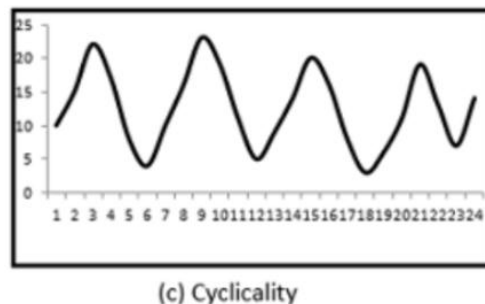
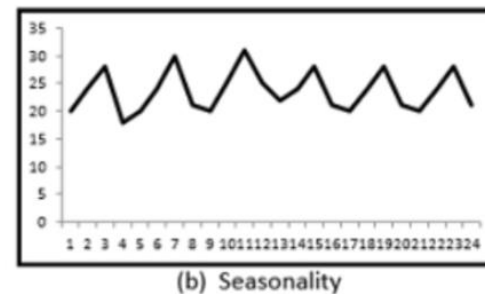
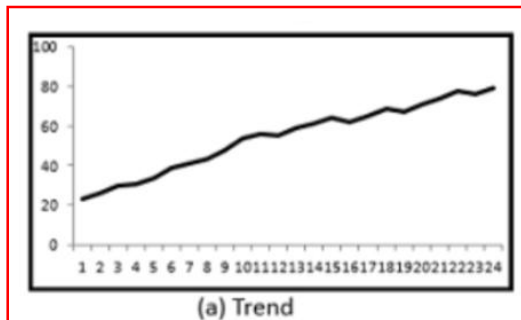


(d) Irregular

# 01 | 시계열 기초



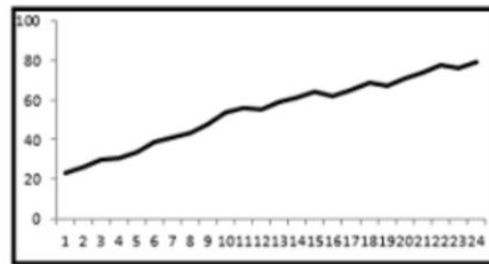
- 시계열 예측
  - ✓ 시계열 데이터를 분석하여 미래 값을 예측하는 분석 기술
- 시계열 데이터
  - ✓ 일련의 시간 순서에 따라 기록된 데이터



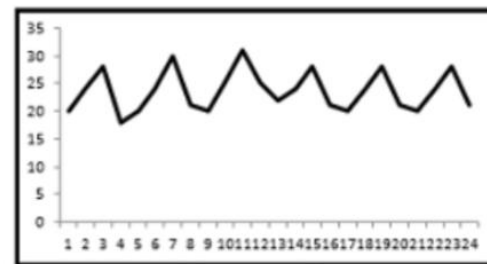
# 01 | 시계열 기초



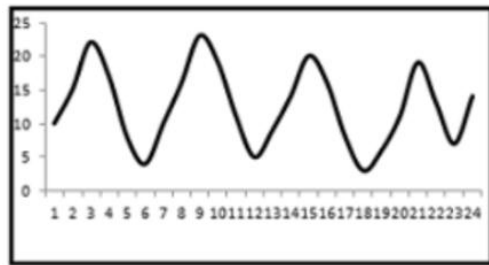
- 시계열 예측
  - ✓ 시계열 데이터를 분석하여 미래 값을 예측하는 분석 기술
- 시계열 데이터
  - ✓ 일련의 시간 순서에 따라 기록된 데이터



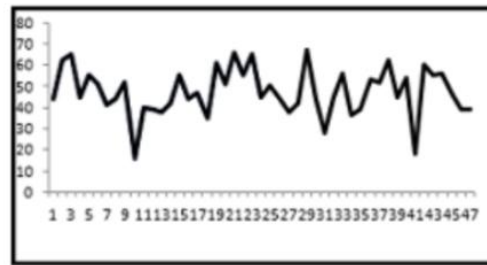
(a) Trend



(b) Seasonality



(c) Cyclicalality

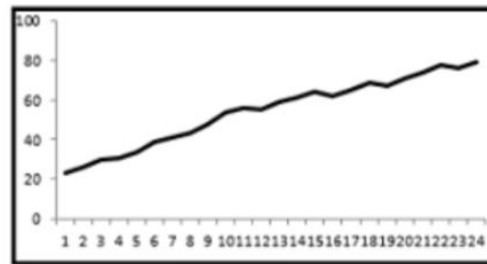


(d) Irregular

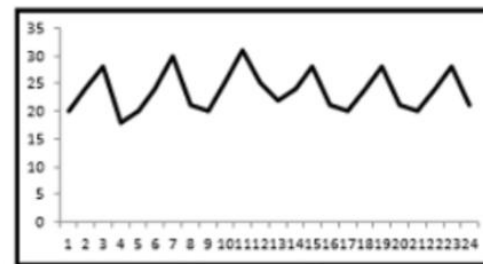
# 01 | 시계열 기초



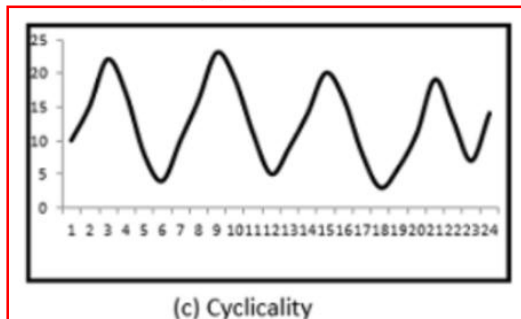
- 시계열 예측
  - ✓ 시계열 데이터를 분석하여 미래 값을 예측하는 분석 기술
- 시계열 데이터
  - ✓ 일련의 시간 순서에 따라 기록된 데이터



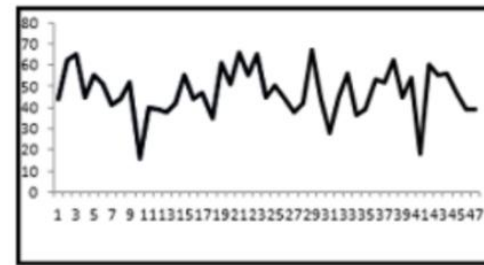
(a) Trend



(b) Seasonality



(c) Cyclical

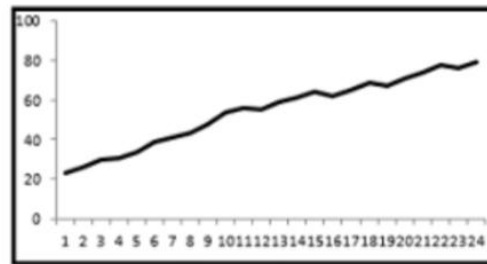


(d) Irregular

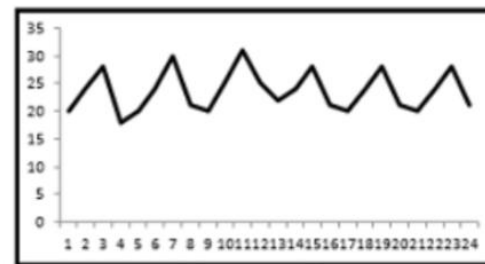
# 01 | 시계열 기초



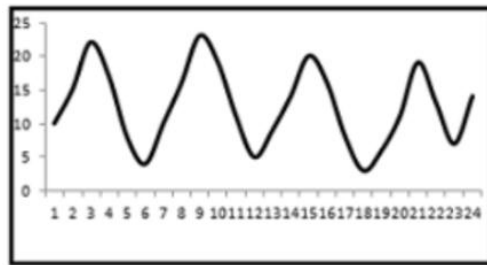
- 시계열 예측
  - ✓ 시계열 데이터를 분석하여 미래 값을 예측하는 분석 기술
- 시계열 데이터
  - ✓ 일련의 시간 순서에 따라 기록된 데이터



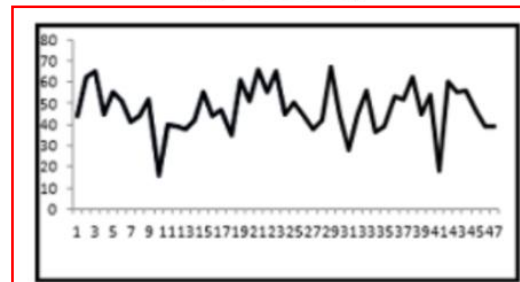
(a) Trend



(b) Seasonality



(c) Cyclical



(d) Irregular

# 01 | 시계열 기초



- 시계열 예측 단계

- ① 문제 정의 및 목표 설정
- ② 데이터 수집
- ③ 데이터 전처리
- ④ 탐색적 데이터 분석
- ⑤ 모델 선택 및 구축
- ⑥ 모델 평가
- ⑦ 예측

## • 시계열 예측 단계

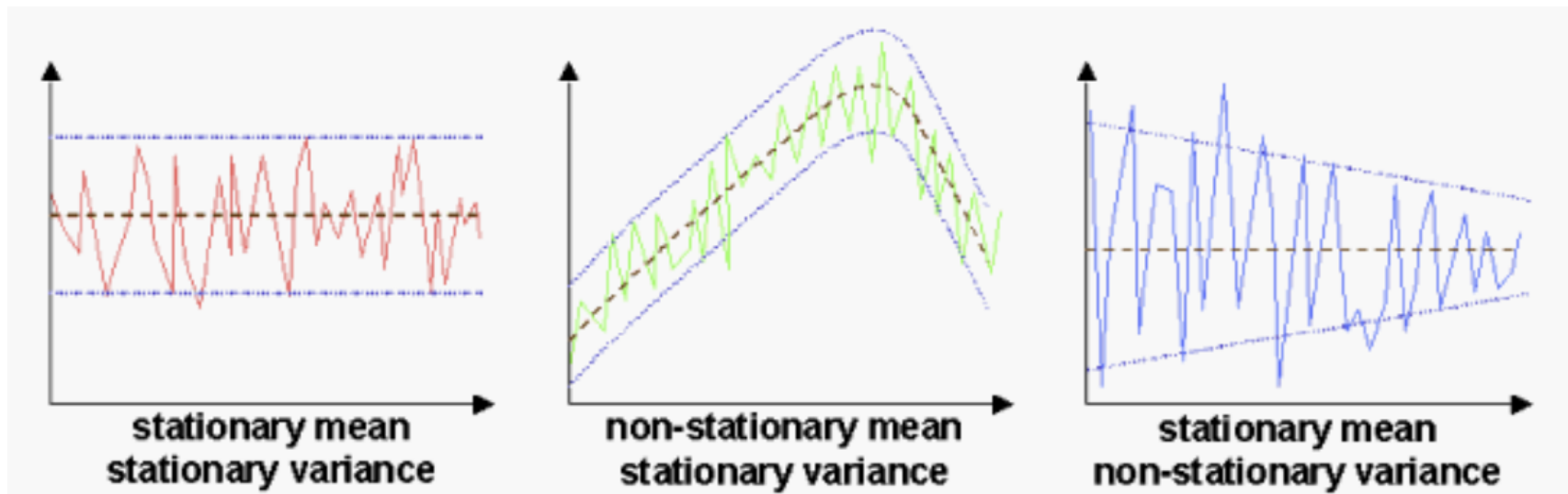
- ① 문제 정의 및 목표 설정
  - ② 데이터 수집
  - ③ 데이터 전처리
  - ④ 탐색적 데이터 분석
  - ⑤ 모델 선택 및 구축
  - ⑥ 모델 평가
  - ⑦ 예측
- ① 특정 지역의 월별 주택 가격 예측
  - ② 부동산 거래 데이터, 경제 지표 데이터, 인구 통계...
  - ③ 누락된 거래 정보 보완, 비정상적으로 높은 거래 가격 제거
  - ④ 주택 가격의 시간에 따른 변동 시각화 및 패턴 분석
  - ⑤ ARIMA 모델 구축 및 파라미터 조정
  - ⑥ 평가지표를 통한 예측 정확도 평가
  - ⑦ 월별 주택 가격 예측 및 투자 전략 수립

- **Stationary(정상) Data**

- ✓ 시간과 관계없이 평균과 분산이 일정한 시계열 데이터
  - 모델의 단순성 및 해석 가능성
  - 예측 정확성

- Stationary(정상) Data

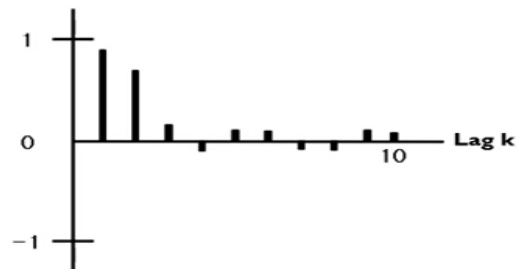
- ✓ 시간과 관계없이 평균과 분산이 일정한 시계열 데이터
  - 모델의 단순성 및 해석 가능성
  - 예측 정확성



- **Stationarity Test**
  - ✓ 시계열 데이터의 정상성 여부 확인
- **Autocorrelation(자기상관)**
  - ✓ 시계열 데이터의 특정 시점과 과거 시점 사이의 상관관계
    - 과거 시점의 데이터가 현재 시점의 데이터에 어떤 영향을 미치는지 평가

- **ACF (Autocorrelation Function)**
  - ✓ 시계열 데이터에서 특정 시점과 그 이전 시점 간의 상관관계 측정
- **PACF (Partial Autocorrelation Function)**
  - ✓ 시계열 데이터의 특정 시점과 그 이전 시점 간의 상관관계를 측정하되, 중간 시점의 영향은 제거한 상태에서 측정

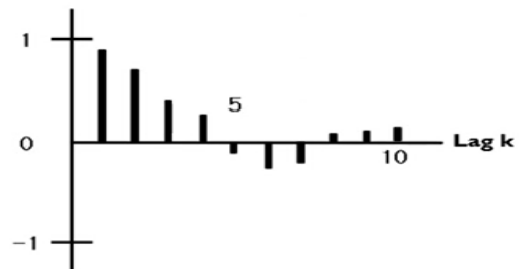
- Stationarity Test



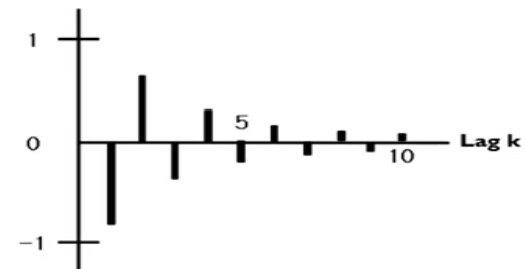
(a) Cuts off after lag 2



(b) Damped exponential dying down

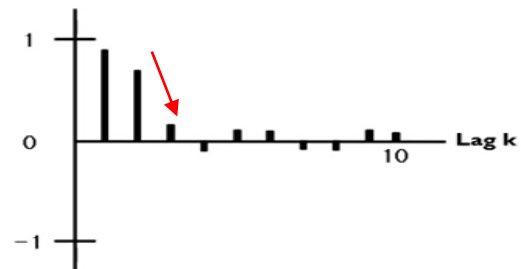


(c) Damped sine-wave dying down

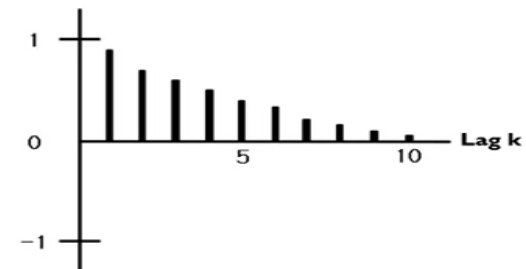


(d) Damped exponential dying down with oscillation

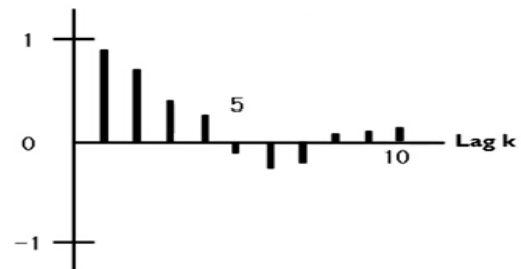
- Stationarity Test



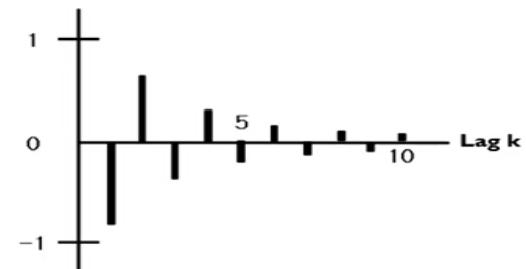
(a) Cuts off after lag 2



(b) Damped exponential dying down

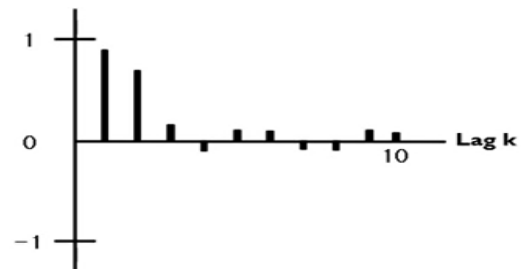


(c) Damped sine-wave dying down

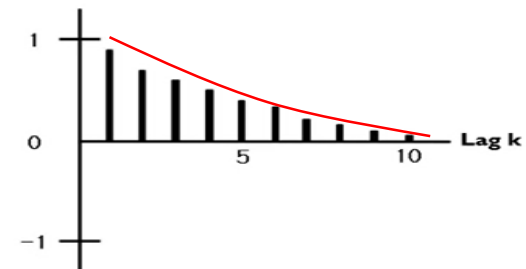


(d) Damped exponential dying down with oscillation

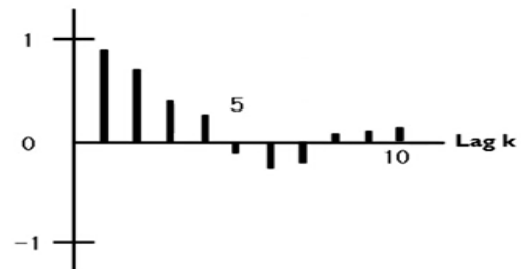
- Stationarity Test



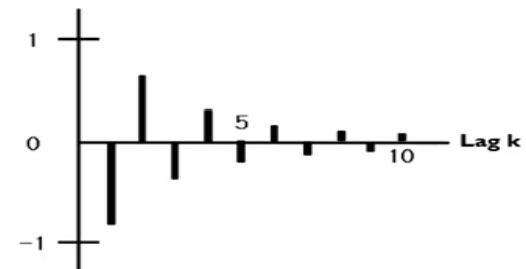
(a) Cuts off after lag 2



(b) Damped exponential dying down

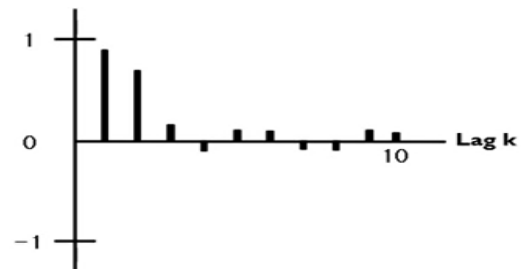


(c) Damped sine-wave dying down



(d) Damped exponential dying down with oscillation

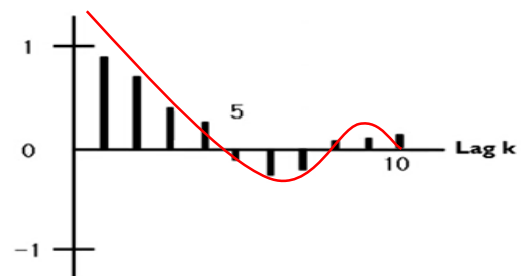
- Stationarity Test



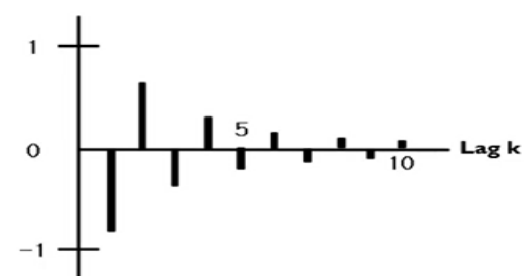
(a) Cuts off after lag 2



(b) Damped exponential dying down

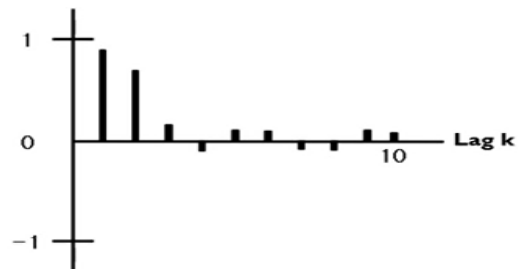


(c) Damped sine-wave dying down

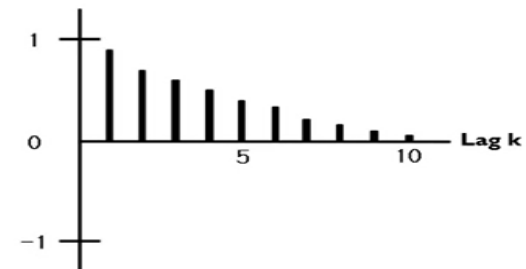


(d) Damped exponential dying down with oscillation

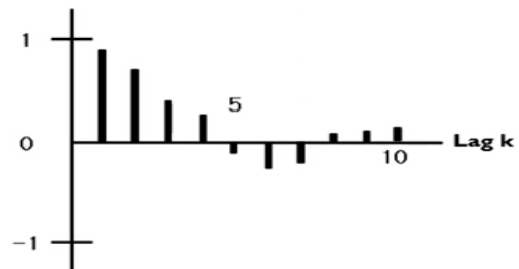
- Stationarity Test



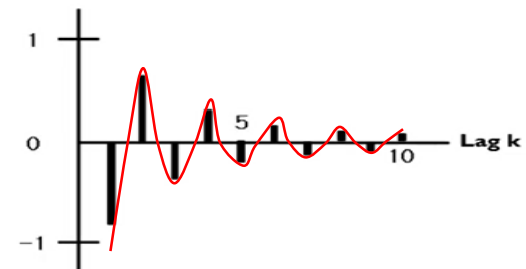
(a) Cuts off after lag 2



(b) Damped exponential dying down



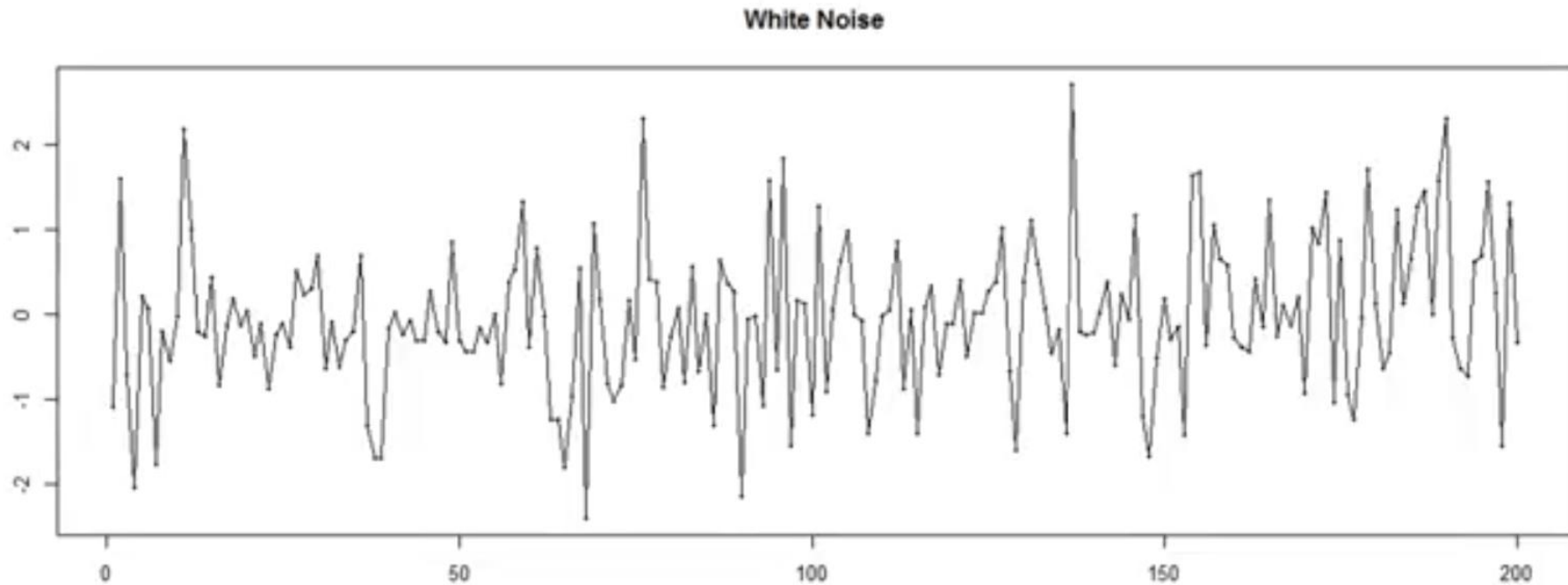
(c) Damped sine-wave dying down



(d) Damped exponential dying down with oscillation

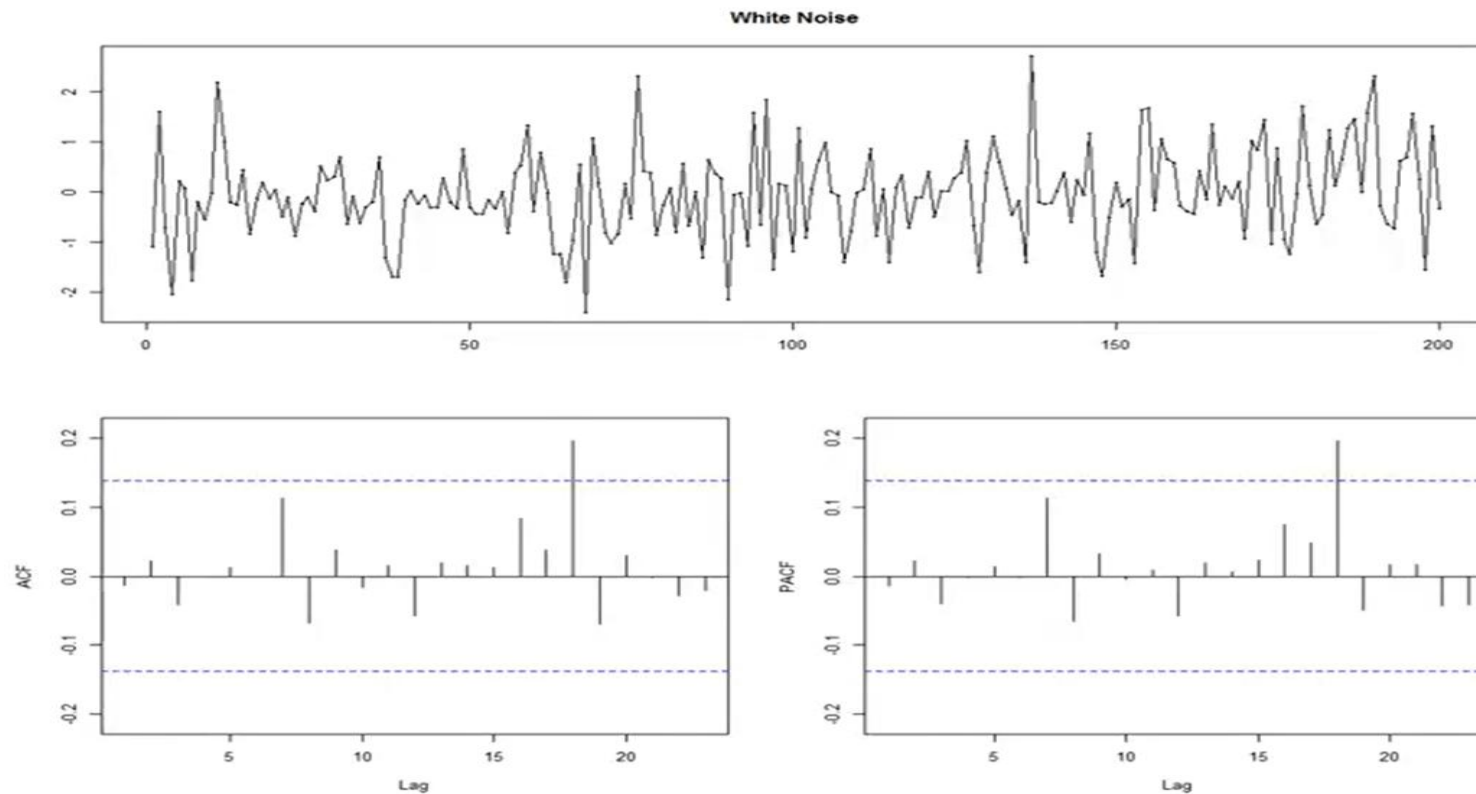
- **Stationarity Test**

- ✓ **Stationary Data vs Non-Stationary Data**



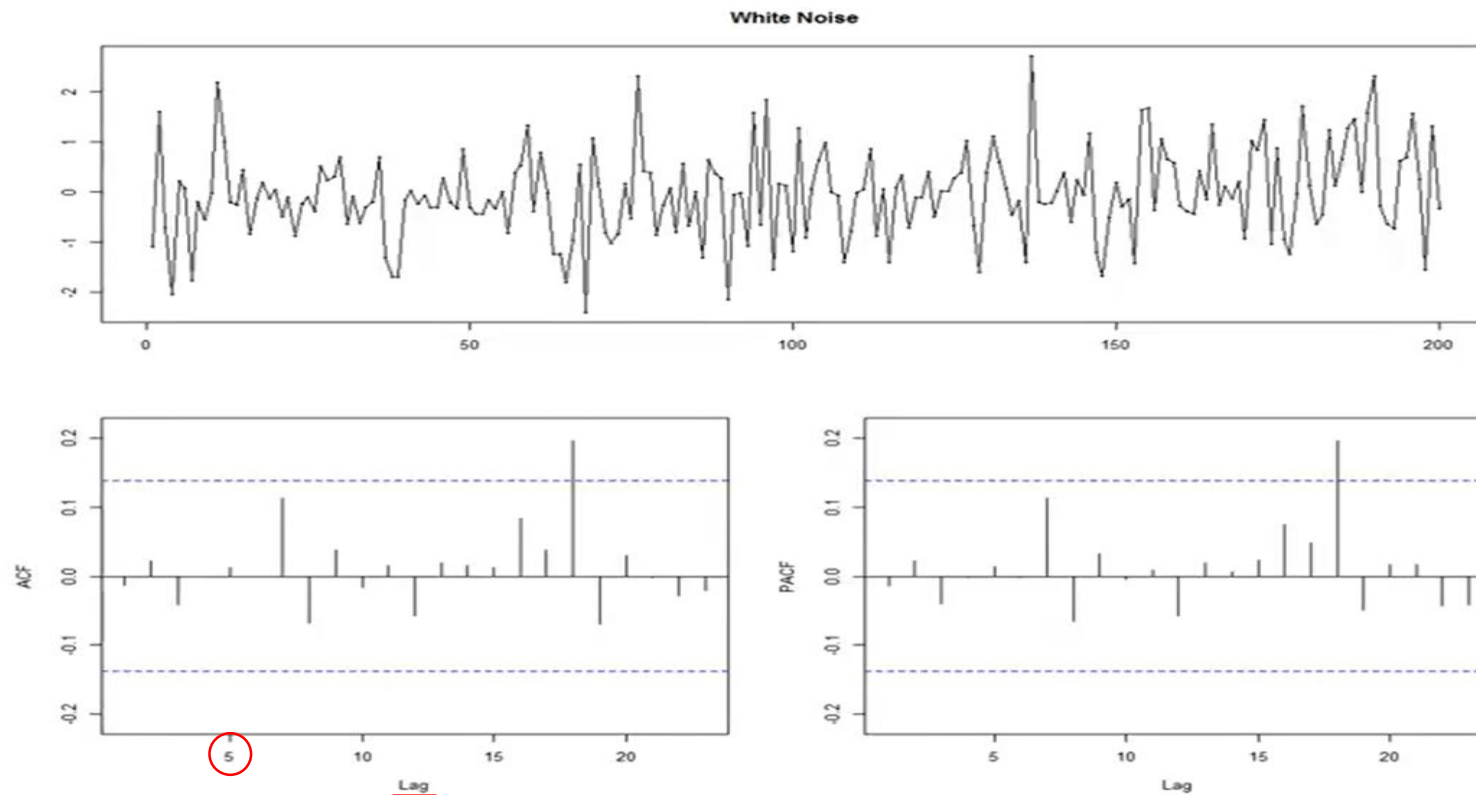
- Stationarity Test

- ✓ Stationary Data vs Non-Stationary Data



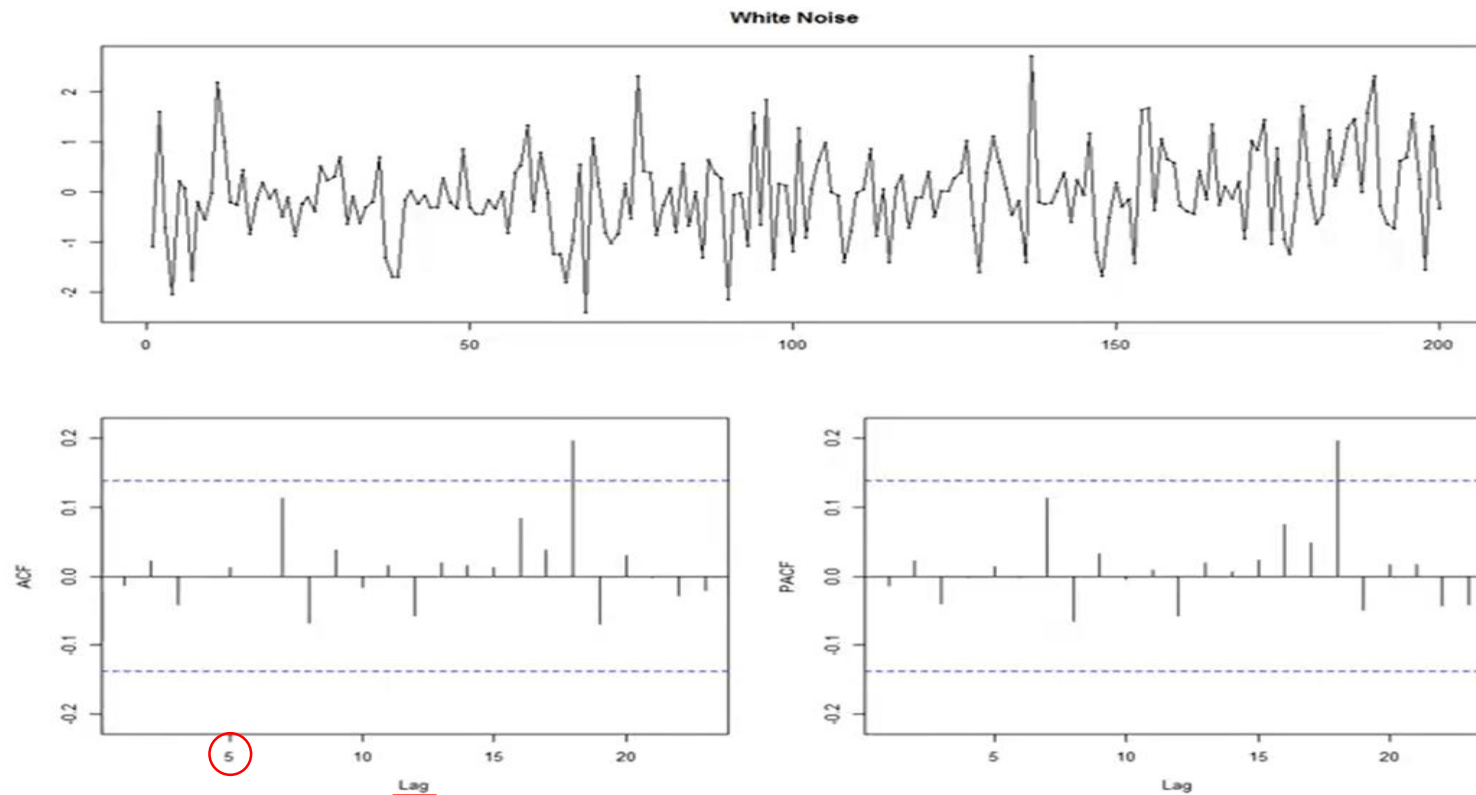
- Stationarity Test

- ✓ Stationary Data vs Non-Stationary Data



- Stationarity Test

✓ **Stationary Data** vs Non-Stationary Data



## 02 | 시계열 예측 모델



- **Stationary Data**
  - ✓ AR Model, MA Model, ARMA Model
- **Non-Stationary Data**
  - ✓ ARIMA Model, SARIMA Model

- AR Model (Auto Regressive Model)

- ✓ AR( $p$ )

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

$$Y_t = c + \varepsilon_t + \sum_{i=1}^p \phi_i Y_{t-i}$$

$Y_t$ : 현재 시점의 값

$\phi_i$ : AR 파라미터

$p$ : AR 차수

$\varepsilon_t$ : 오차항

$c$ : 초기항

## 02 | 시계열 예측 모델



- AR Model (Auto Regressive Model)

✓ AR(1),  $c=1$ ,  $\phi_i = 0.6$

$$Y_t = c + \varepsilon_t + \sum_{i=1}^p \phi_i Y_{t-i}$$

시점	실제값	수식	예측값
1	3	-	-
2	4	$1 + 0.6 \times 3$	2.8
3	8	$1 + 0.6 \times 2.8$	2.68
4	5	$1 + 0.6 \times 2.68$	2.608
5	6	$1 + 0.6 \times 2.608$	2.565
6	10	$1 + 0.6 \times 2.565$	2.539
7	8	$1 + 0.6 \times 2.539$	2.523
8	14	$1 + 0.6 \times 2.523$	2.514
9	15	$1 + 0.6 \times 2.514$	2.508
10	12	$1 + 0.6 \times 2.508$	2.505

- MA Model (Moving Average Model)

- ✓ MA(q)

$$Y_t = c + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_p \varepsilon_{t-p} + \varepsilon_t$$

$$Y_t = c + \varepsilon_t + \sum_{i=1}^q \theta_i \varepsilon_{t-i}$$

$Y_t$ : 현재 시점의 값

$\theta_i$ : MA 파라미터(MA 계수)

$q$ : MA 차수

$\varepsilon_{t-j}$ : 예측오차

$c$ : 초기항

## 02 | 시계열 예측 모델



- MA Model (Moving Average Model)

✓ MA(1),  $c=1$ ,  $\theta_j = 0.6$

$$Y_t = c + \varepsilon_t + \sum_{i=1}^q \theta_j \varepsilon_{t-j}$$

시점	실제값	$\varepsilon_t$	수식	예측값
1	3	-	-	-
2	4	$4 - 1 = 3$	-	-
3	8	$8 - 2.8 = 5.2$	$1 + 0.6 \times 3$	2.8
4	5	$5 - 4.12 = 0.88$	$1 + 0.6 \times 5.2$	4.12
5	6	$6 - 1.53 = 4.47$	$1 + 0.6 \times 0.88$	1.53
6	10	$10 - 3.68 = 6.32$	$1 + 0.6 \times 4.47$	3.68
7	8	$8 - 4.79 = 3.21$	$1 + 0.6 \times 6.32$	4.79
8	14	$14 - 2.93 = 11.07$	$1 + 0.6 \times 3.21$	2.93
9	15	$15 - 7.64 = 7.36$	$1 + 0.6 \times 11.07$	7.64
10	12	$12 - 5.42 = 6.58$	$1 + 0.6 \times 7.36$	5.42

- **ARMA Model (Auto Regressive Moving Average Model)**

- ✓ **ARMA(p,q)**

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} \\ + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_p \varepsilon_{t-p} + \varepsilon_t$$

$$Y_t = c + \varepsilon_t + \sum_{j=1}^q \theta_j \varepsilon_{t-j} + \sum_{i=1}^p \phi_i Y_{t-i}$$

$Y_t$ : 현재 시점의 값

$\phi_i$ : AR 파라미터

$p$ : AR 차수

$\varepsilon_t$ : 오차항

$\theta_i$ : MA 파라미터

$q$ : MA 차수

$c$ : 초기항

- ARMA Model

✓ ARMA(1,1) ,  $c=1$  ,  $\phi_i = 0.6$  ,  $\theta_j = 0.6$  ,  $\varepsilon_{t-j}$  = 예측 오차

$$Y_t = c + \varepsilon_t + \sum_{j=1}^q \theta_j \varepsilon_{t-j} + \sum_{i=1}^p \phi_i Y_{t-i}$$

시점	실제값	수식	예측값
1	3	-	-
2	4	$1 + 0.6 \times 3 + 0.6 \times 0$	2.8
3	8	$1 + 0.6 \times 2.8 + 0.6 \times 1.2$	3.4
4	5	$1 + 0.6 \times 3.4 + 0.6 \times 4.6$	5.8
5	6	$1 + 0.6 \times 5.8 + 0.6 \times (-0.8)$	4.0
6	10	$1 + 0.6 \times 4.0 + 0.6 \times 2.0$	4.6
7	8	$1 + 0.6 \times 4.6 + 0.6 \times 5.4$	7.0
8	14	$1 + 0.6 \times 7.0 + 0.6 \times 1.0$	5.8
9	15	$1 + 0.6 \times 5.8 + 0.6 \times 8.2$	9.4
10	12	$1 + 0.6 \times 9.4 + 0.6 \times 5.6$	10.0

## 02 | 시계열 예측 모델



- AR Model, MA Model, ARMA Model
  - ✓ Parameter :  $c = 1$  ,  $\phi_i = 0.6$  ,  $\theta_j = 0.6$

Model	MSE
AR(1)	11.51
MA(1)	14.00
ARMA(1,1)	11.53

## 02 | 시계열 예측 모델



- Non-Stationary Data

- ✓ ARIMA Model, SARIMA Model

"Non-Stationary Data → Stationary Data"

- Non-Stationary Data

- ✓ ARIMA Model, SARIMA Model

"Non-Stationary Data → Stationary Data"

- ✓ Differencing(차분) : 현 시점 데이터에서 d시점 이전 데이터를 뺀 것

- Non-Stationary Data

- ✓ ARIMA Model, SARIMA Model

"Non-Stationary Data → Stationary Data"

- ✓ Differencing(차분)

X
2
7
10
5
8

- Non-Stationary Data

- ✓ ARIMA Model, SARIMA Model

“Non-Stationary Data → Stationary Data”

- ✓ Differencing(차분)

X	
2	
7	
10	
5	
8	

X
2
7
10
5
8

- Non-Stationary Data

- ✓ ARIMA Model, SARIMA Model

“Non-Stationary Data → Stationary Data”

- ✓ Differencing(차분)

X		
2		X
7	—	2
10	—	7
5	—	10
8	—	5
		8

Y
5
3
-5
3
-

1차 차분:  $Y_t = X_t - X_{t-1} = \nabla X_t$

2차 차분:  $Y_t^{(2)} = X_t - X_{t-2} = \nabla^{(2)} X_t$

d차 차분:  $Y_t^{(d)} = X_t - X_{t-d} = \nabla^{(d)} X_t$

## 02 | 시계열 예측 모델

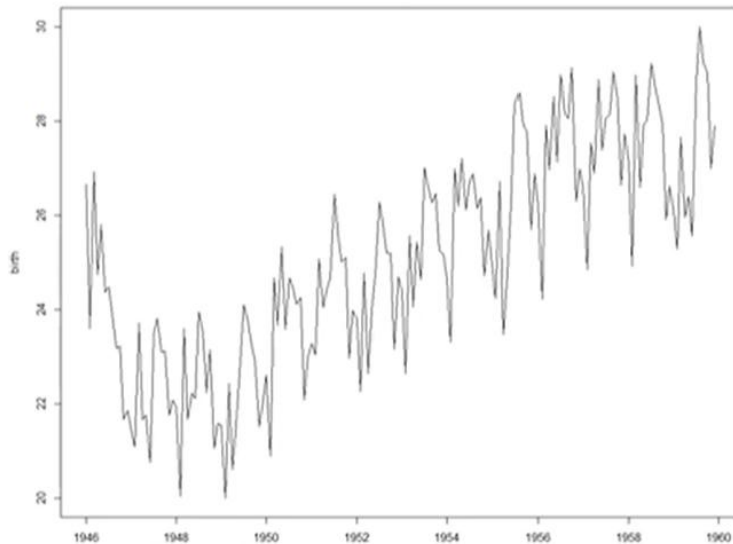


- Non-Stationary Data

- ✓ ARIMA Model, SARIMA Model

“Non-Stationary Data → Stationary Data”

- ✓ Differencing(차분)



## 02 | 시계열 예측 모델

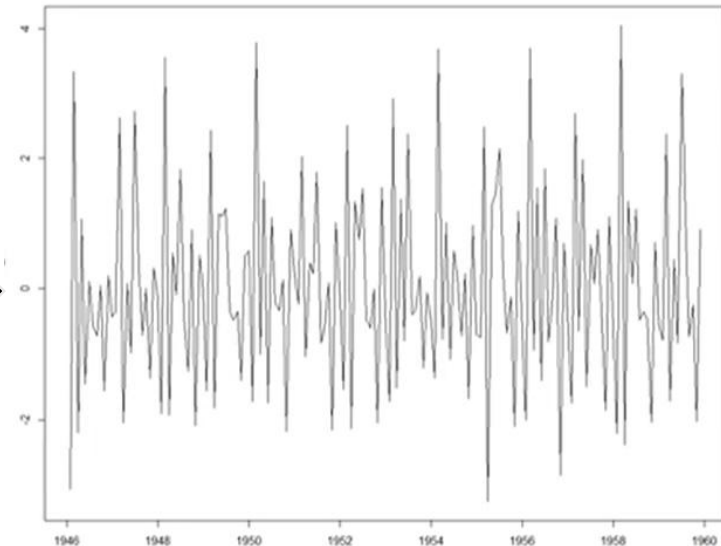
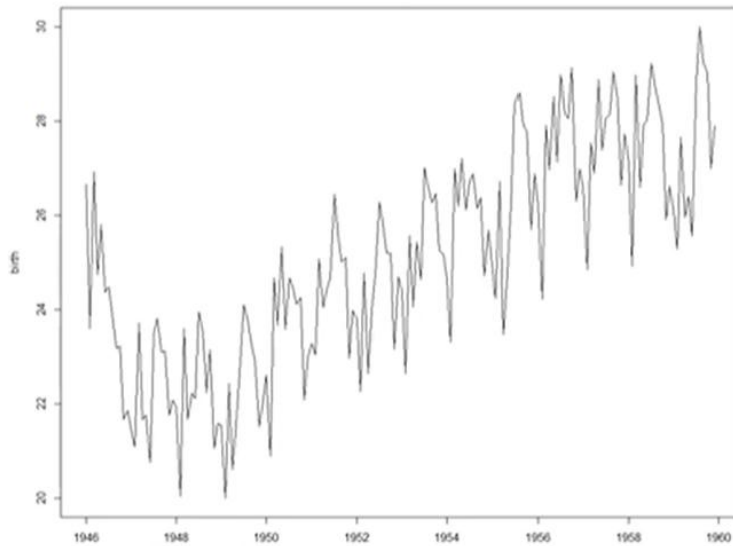


- Non-Stationary Data

- ✓ ARIMA Model, SARIMA Model

"Non-Stationary Data → Stationary Data"

- ✓ Differencing(차분)

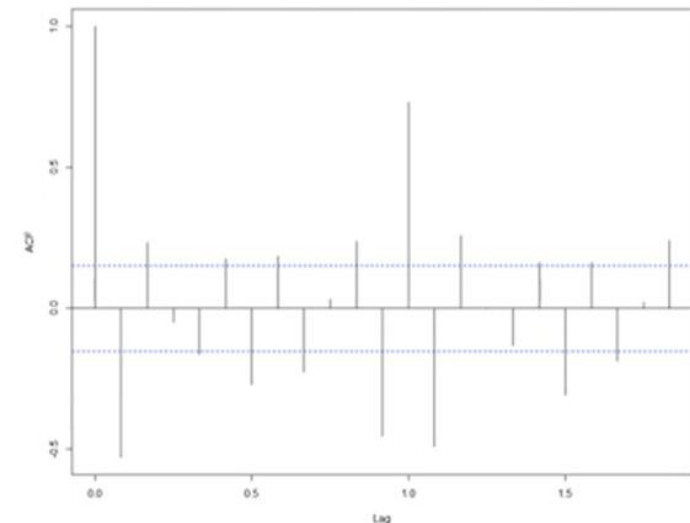
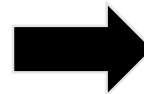
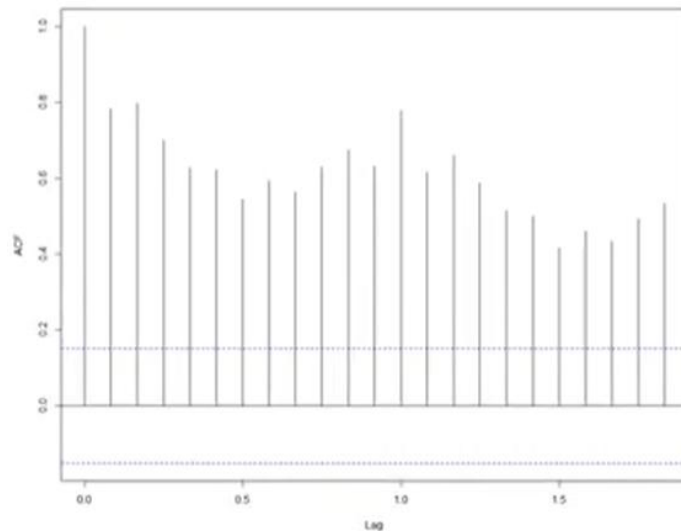


- Non-Stationary Data

- ✓ ARIMA Model, SARIMA Model

“Non-Stationary Data → Stationary Data”

- ✓ Differencing(차분)



- **ARIMA Model** (Auto Regressive Integrated Moving Average Model)
  - ✓ ARIMA(p,d,q)

$$(1 - B)^d Y_t = c + \varepsilon_t + \sum_{i=1}^p \phi_i Y_{t-i} + \sum_{j=1}^q \theta_j \varepsilon_{t-j}$$

$\sum_{i=1}^p \phi_i Y_{t-i}$  : AR 다항식

$\sum_{j=1}^q \theta_j (B) \varepsilon_{t-j}$  : MA 다항식

$(1 - B)^d$ : 차분 연산자 ( $BY_t = Y_{t-1}$ )

$B$ : 지연 연산자 ( $BY_t = Y_{t-1}$ )

- SARIMA Model

**Seasonal** Auto Regressive Integrated Moving Average Model

✓ SARIMA(p,d,q)(P,D,Q)s

“ARIMA Model에 **계절 변동**을 반영”

- ✓ 계절성 요인을 포함하여 더 정확한 예측 가능
- ✓ 유연성 있는 모델
- ✓ 장기적인 예측에 효과적

- SARIMA Model

Seasonal Auto Regressive Integrated Moving Average Model

- ✓ SARIMA(p,d,q)(P,D,Q)s

$$(1 - B)^d(1 - B^S)^D Y^t = c + \sum_{i=1}^p \phi_i Y_{t-i} + \sum_{j=1}^q \theta_j \varepsilon_{t-j} + \sum_{l=1}^P \Phi_l Y_{t-l} + \sum_{J=1}^Q \Theta_J \varepsilon_{t-J} + \varepsilon_t$$

$\sum_{i=1}^p \phi_i Y_{t-i}$  : 비계절 AR 다항식

$\sum_{l=1}^P \Phi_l Y_{t-l}$  : 계절 AR 다항식

$\sum_{j=1}^q \theta_j \varepsilon_{t-j}$  : 비계절 MA 다항식

$\sum_{J=1}^Q \Theta_J \varepsilon_{t-J}$  : 계절 MA 다항식

$(1 - B)^d$  : 비계절 차분 연산자

$(1 - B^S)^D$  : 계절 차분 연산자

B: 지연 연산자 ( $BY_t = Y_{t-1}$ )

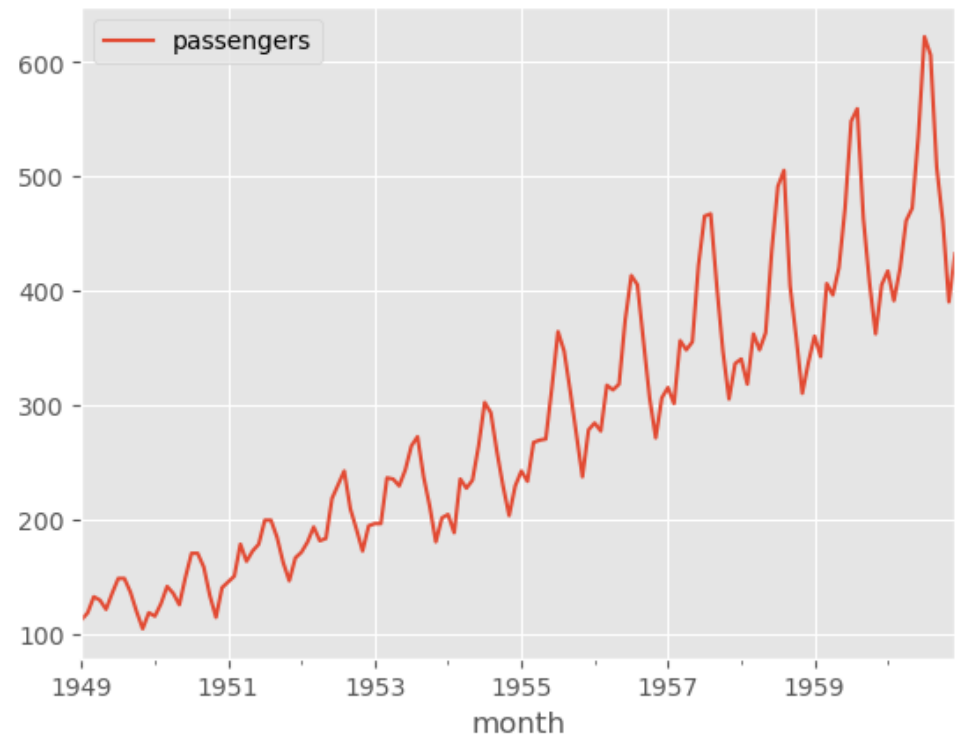
s: 계절 주기 (월=1, 분기=4, 년=12)

# 03 | Example



- **Model**
  - ✓ ARIMA Model, SARIMA Model

- **Data**
  - ✓ AirPassengers Data.csv



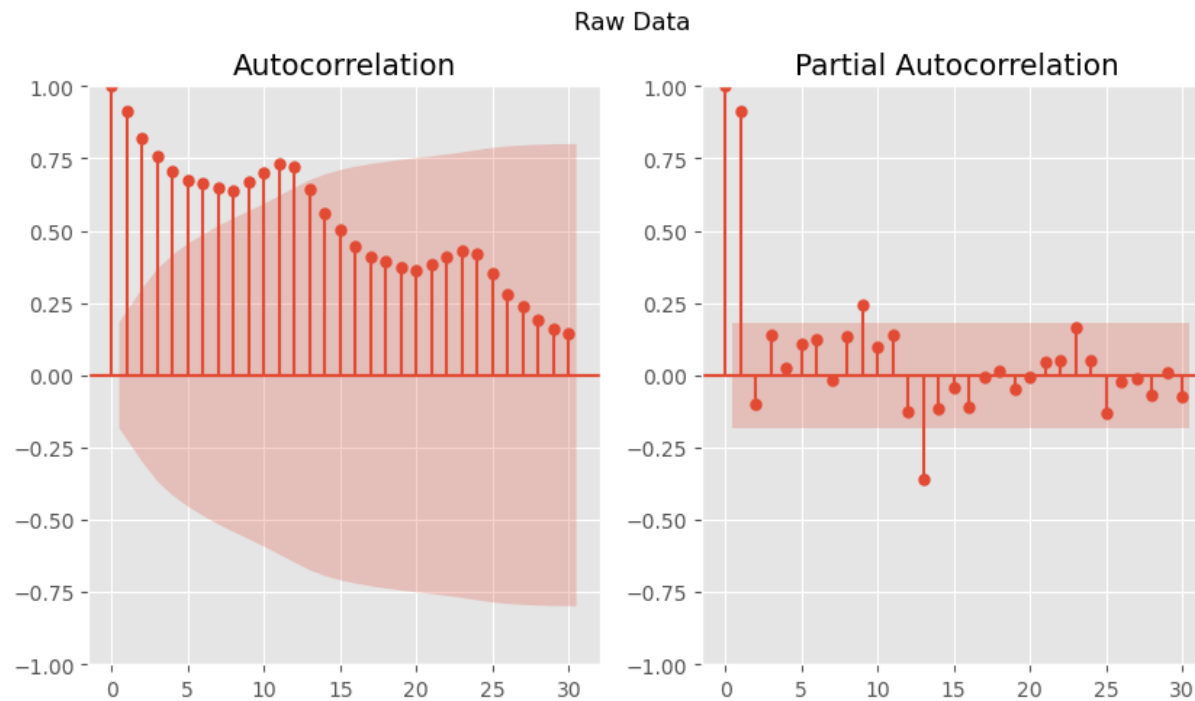
# 03 | Example



- Stationarity Test (Raw Data)

```
train_data, test_data = train_test_split(data, test_size=0.2, shuffle=False)

fig, ax = plt.subplots(1, 2, figsize=(10, 5))
fig.suptitle('Raw Data')
sm.graphics.tsa.plot_acf(train_data.values.squeeze(), lags=30, ax=ax[0])
sm.graphics.tsa.plot_pacf(train_data.values.squeeze(), lags=30, ax=ax[1]);
```



# 03 | Example



- Differencing

```
diff_train_data = train_data.copy()
diff_train_data = diff_train_data['passengers'].diff()
diff_train_data = diff_train_data.dropna() # 차분으로 인한 결측치 제거
print('##### RAW DATA #####')
print(train_data)
print('### Differenced Data ###')
print(diff_train_data)
```

##### RAW DATA #####

month	passengers
-------	------------

1949-01-01	112
1949-02-01	118
1949-03-01	132
1949-04-01	129
1949-05-01	121
...	...
1958-03-01	362
1958-04-01	348
1958-05-01	363
1958-06-01	435
1958-07-01	491

[115 rows x 1 columns]  
### Differenced Data ###

month	
-------	--

1949-02-01	6.0
1949-03-01	14.0
1949-04-01	-3.0
1949-05-01	-8.0
1949-06-01	14.0
...	...
1958-03-01	44.0
...	...
1958-05-01	15.0
1958-06-01	72.0
1958-07-01	56.0

Name: passengers, Length: 114, dtype: float64

# 03 | Example



- Differencing

```
diff_train_data = train_data.copy()
diff_train_data = diff_train_data['passengers'].diff()
diff_train_data = diff_train_data.dropna() # 차분으로 인한 결측치 제거
print('##### RAW DATA #####')
print(train_data)
print('### Differenced Data ###')
print(diff_train_data)
```

##### RAW DATA #####  
passengers

month	passengers
1949-01-01	112
1949-02-01	118
1949-03-01	132
1949-04-01	129
1949-05-01	121
...	...
1958-03-01	362
1958-04-01	348
1958-05-01	363
1958-06-01	435
1958-07-01	491

115 rows x 1 columns]  
### Differenced Data ###  
month

1949-02-01	6.0
1949-03-01	14.0
1949-04-01	-3.0
1949-05-01	-8.0
1949-06-01	14.0
...	...
1958-03-01	44.0
...	...
1958-05-01	15.0
1958-06-01	72.0
1958-07-01	56.0

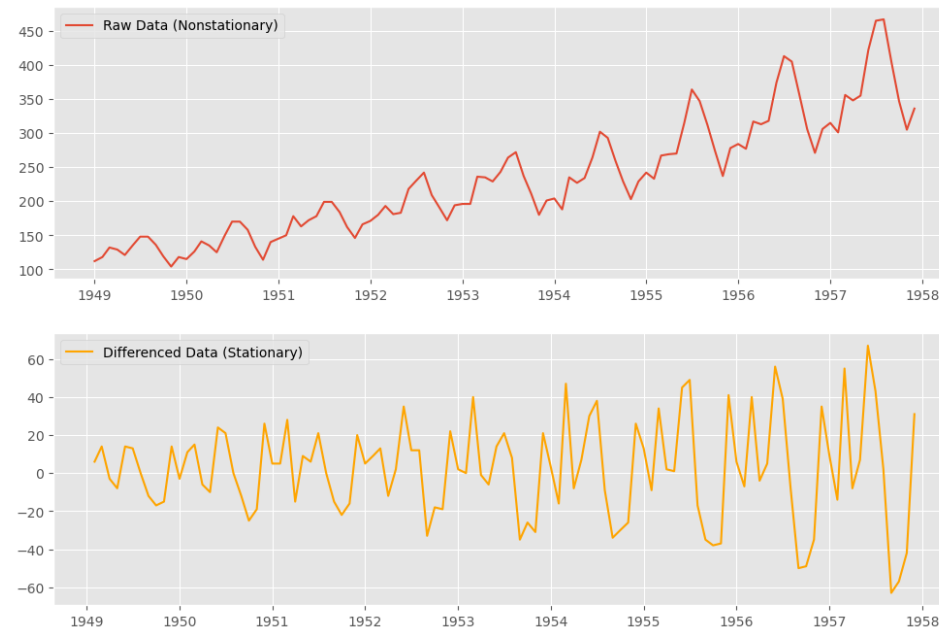
Name: passengers, Length: 114, dtype: float64

# 03 | Example



- Differencing

```
plt.figure(figsize=(12,8))
plt.subplot(211)
plt.plot(train_data['passengers'])
plt.legend(['Raw Data (Nonstationary)'])
plt.subplot(212)
plt.plot(diff_train_data,'orange')
plt.legend(['Differenced Data (Stationary)'])
plt.show()
```

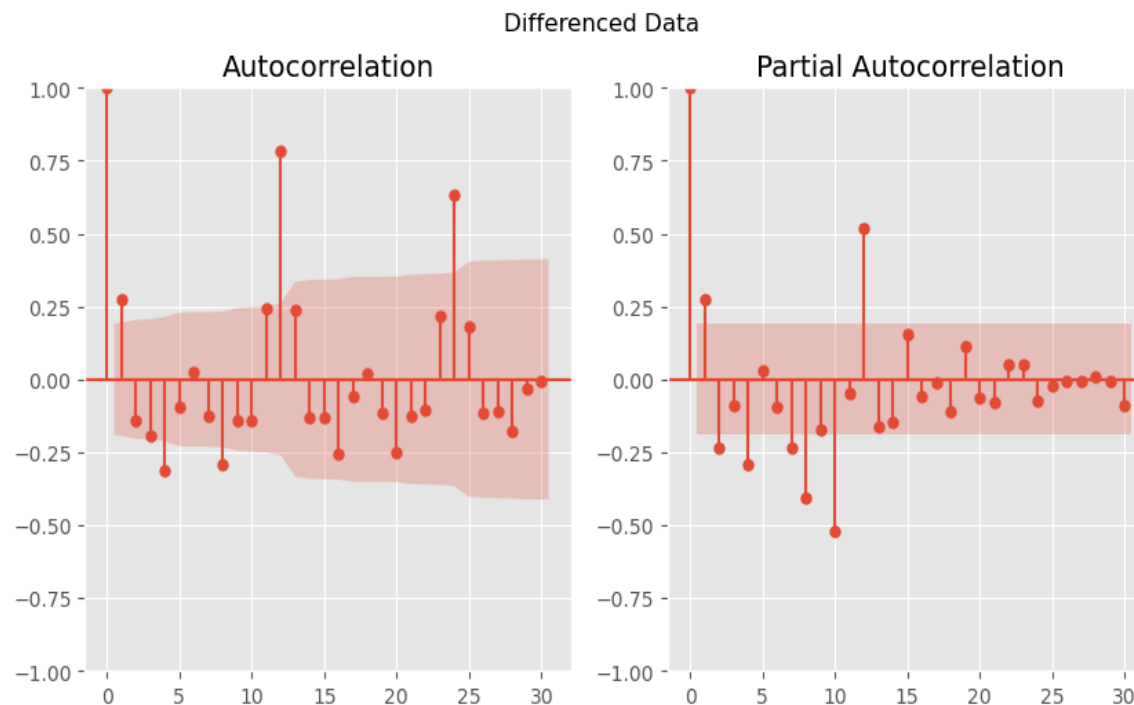


# 03 | Example



- **Stationarity Test (Differenced Data)**

```
fig, ax = plt.subplots(1,2,figsize=(10,5))  
fig.suptitle('Differenced Data')  
sm.graphics.tsa.plot_acf(diff_train_data.values.squeeze(), lags=30, ax=ax[0])  
sm.graphics.tsa.plot_pacf(diff_train_data.values.squeeze(), lags=30, ax=ax[1]);
```



# 03 | Example



- Parameter Estimation

- ✓ ARIMA

```
model_opt = ARIMA(train_data.values, order=(2,1,1))
model_opt_fit = model_opt.fit()
model_opt_fit.summary()
```

✓ 0.1s

## SARIMAX Results

Dep. Variable:	y	No. Observations:	115			
Model:	ARIMA(2, 1, 1)	Log Likelihood	-525.324			
Date:	Sun, 29 Sep 2024	AIC	1058.648			
Time:	17:59:18	BIC	1069.592			
Sample:	0	HQIC	1063.089			
	- 115					
Covariance Type:	opg					
	coef	std err	z	P> z	[0.025	0.975]
ar.L1	1.0717	0.111	9.615	0.000	0.853	1.290
ar.L2	-0.4397	0.105	-4.177	0.000	-0.646	-0.233
ma.L1	-0.8309	0.110	-7.556	0.000	-1.046	-0.615
sigma2	585.6414	79.145	7.400	0.000	430.519	740.764
Ljung-Box (L1) (Q):	0.35	Jarque-Bera (JB):	7.55			
Prob(Q):	0.56	Prob(JB):	0.02			
Heteroskedasticity (H):	6.24	Skew:	0.61			
Prob(H) (two-sided):	0.00	Kurtosis:	2.69			

# 03 | Example



- Parameter Estimation

- ✓ ARIMA(2,1,1)

```
model_opt = ARIMA(train_data.values, order=(2,1,1))  
model_opt_fit = model_opt.fit()  
model_opt_fit.summary()
```

✓ 0.1s

## SARIMAX Results

Dep. Variable:	y	No. Observations:	115			
Model:	ARIMA(2, 1, 1)	Log Likelihood	-525.324			
Date:	Sun, 29 Sep 2024	AIC	1058.648			
Time:	17:59:18	BIC	1069.592			
Sample:	0	HQIC	1063.089			
	- 115					
Covariance Type:	opg					
	coef	std err	z	P> z	[0.025	0.975]
ar.L1	1.0717	0.111	9.615	0.000	0.853	1.290
ar.L2	-0.4397	0.105	-4.177	0.000	-0.646	-0.233
ma.L1	-0.8309	0.110	-7.556	0.000	-1.046	-0.615
sigma2	585.6414	79.145	7.400	0.000	430.519	740.764
Ljung-Box (L1) (Q):	0.35	Jarque-Bera (JB):	7.55			
Prob(Q):	0.56	Prob(JB):	0.02			
Heteroskedasticity (H):	6.24	Skew:	0.61			
Prob(H) (two-sided):	0.00	Kurtosis:	2.69			

# 03 | Example



- Parameter Estimation

- ✓ ARIMA

```
p = range(0, 3)
d = range(1, 2)
q = range(0, 3)
pdq = list(itertools.product(p,d,q))

aic=[]
best_model_info = None

for i in pdq:
    try:
        model = ARIMA(train_data.values, order=i)
        model_fit = model.fit()
        aic_value = round(model_fit.aic, 2)
        print(f'ARIMA: {i} ==> AIC : {aic_value}')
        aic.append((i, aic_value))

        # Update the best model information if the current model has a lower AIC
        if best_model_info is None or aic_value < best_model_info[1]:
            best_model_info = (i, aic_value)
    except Exception as e:
        print(f'ARIMA: {i} 모델 적합 중 오류 발생: {e}')

# Print the best model and its AIC value
if best_model_info:
    print(f'\n최적의 ARIMA 모델: {best_model_info[0]} ==> 최저 AIC: {best_model_info[1]}')
else:
    print('모든 모델 적합 중 오류 발생.')
```

✓ 0.2s

```
ARIMA: (0, 1, 0) ==> AIC : 1076.27
ARIMA: (0, 1, 1) ==> AIC : 1063.65
ARIMA: (0, 1, 2) ==> AIC : 1060.69
ARIMA: (1, 1, 0) ==> AIC : 1068.54
ARIMA: (1, 1, 1) ==> AIC : 1058.25
ARIMA: (1, 1, 2) ==> AIC : 1057.33
ARIMA: (2, 1, 0) ==> AIC : 1065.64
ARIMA: (2, 1, 1) ==> AIC : 1058.65
ARIMA: (2, 1, 2) ==> AIC : 1057.52
```

최적의 ARIMA 모델: (1, 1, 2) ==> 최저 AIC: 1057.33

# 03 | Example



- Parameter Estimation

- ✓ ARIMA

```
p = range(0, 3)
d = range(1, 2)
q = range(0, 3)
pdq = list(itertools.product(p,d,q))

aic=[]
best_model_info = None

for i in pdq:
    try:
        model = ARIMA(train_data.values, order=i)
        model_fit = model.fit()
        aic_value = round(model_fit.aic, 2)
        print(f'ARIMA: {i} ==> AIC : {aic_value}')
        aic.append((i, aic_value))

        # Update the best model information if the current model has a lower AIC
        if best_model_info is None or aic_value < best_model_info[1]:
            best_model_info = (i, aic_value)
    except Exception as e:
        print(f'ARIMA: {i} 모델 적합 중 오류 발생: {e}')

# Print the best model and its AIC value
if best_model_info:
    print(f'\n최적의 ARIMA 모델: {best_model_info[0]} ==> 최저 AIC: {best_model_info[1]}')
else:
    print('모든 모델 적합 중 오류 발생.')
```

✓ 0.2s

ARIMA: (0, 1, 0) ==> AIC : 1076.27  
ARIMA: (0, 1, 1) ==> AIC : 1063.65  
ARIMA: (0, 1, 2) ==> AIC : 1060.69  
ARIMA: (1, 1, 0) ==> AIC : 1068.54  
ARIMA: (1, 1, 1) ==> AIC : 1058.25  
ARIMA: (1, 1, 2) ==> AIC : 1057.33  
ARIMA: (2, 1, 0) ==> AIC : 1065.64  
ARIMA: (2, 1, 1) ==> AIC : 1058.65  
ARIMA: (2, 1, 2) ==> AIC : 1057.52

최적의 ARIMA 모델: (1, 1, 2) ==> 최저 AIC: 1057.33

# 03 | Example



- Parameter Estimation

✓ **ARIMA(1,1,2) → AIC = 1057.33**

```
# 1. 데이터 분할 (train/test)
train_data, test_data = train_test_split(data, test_size=0.2, shuffle=False)

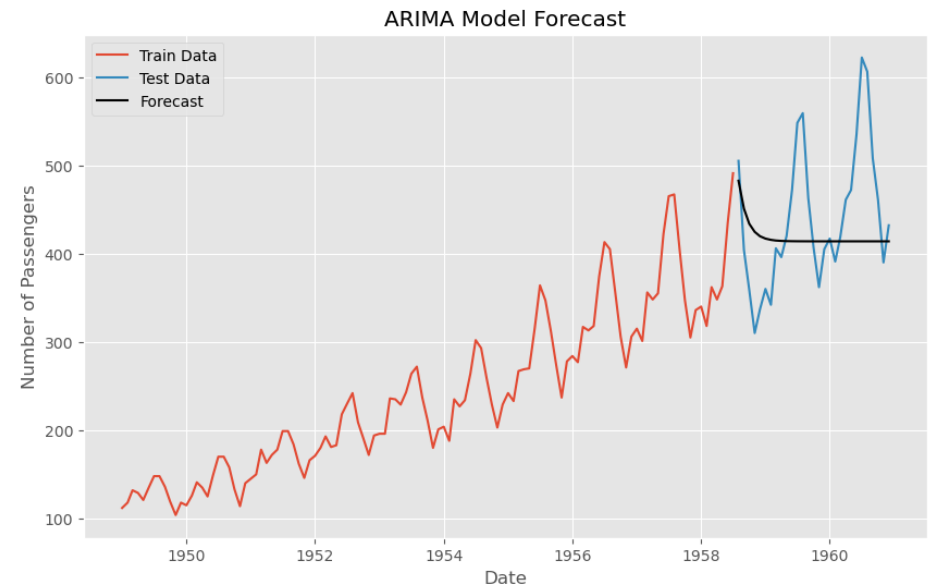
# 2. ARIMA 모델 훈련 및 예측 수행
model = ARIMA(train_data, order=(1,1,2))
model_fit = model.fit()

# 3. 예측 (테스트 데이터 기간에 대한 예측 수행)
forecast = model_fit.forecast(steps=len(test_data))

# 4. 예측 값에 대한 날짜 인덱스 생성 (기존 데이터 마지막 이후부터)
forecast_index = pd.date_range(start=train_data.index[-1], periods=len(test_data)+1, freq='MS')[1:]

# 5. 예측 값에 인덱스 할당
forecast = pd.Series(forecast, index=forecast_index)

# 6. 예측 결과와 실제 데이터를 시각화
plt.figure(figsize=(10, 6))
plt.plot(train_data.index, train_data, label='Train Data')
plt.plot(test_data.index, test_data, label='Test Data')
plt.plot(forecast.index, forecast, label='Forecast', color='black')
plt.title('ARIMA Model Forecast')
plt.xlabel('Date')
plt.ylabel('Number of Passengers')
plt.legend()
plt.grid(True)
plt.show()
```



# 03 | Example



- Parameter Estimation

- ✓ SARIMA

```
# 파라미터 범위 설정
p = range(0, 3) # 비계절성 p 값
d = range(1, 2) # 비계절성 d 값
q = range(0, 3) # 비계절성 q 값
P = range(0, 3) # 계절성 P 값
D = range(0, 2) # 계절성 D 값
Q = range(0, 3) # 계절성 Q 값
s = [12] # 계절성 주기 (예: 월별 데이터라면 12)

# 모든 조합 생성
pdq = list(itertools.product(p, d, q))
seasonal_pdq = list(itertools.product(P, D, Q, s))

# AIC 값을 저장할 리스트와 최적의 모델 정보
aic = []
best_model_info = None

# SARIMA 모델 학습 및 AIC 계산
for param in pdq:
    for seasonal_param in seasonal_pdq:
        try:
            # SARIMA 모델 생성 및 학습
            model = SARIMAX(train_data.values,
                           order=param,
                           seasonal_order=seasonal_param,
                           enforce_stationarity=False,
                           enforce_invertibility=False)
            model_fit = model.fit(dispatch=False)

            # AIC 계산
            aic_value = round(model_fit.aic, 2)
            print(f'SARIMA: {param} x {seasonal_param} ==> AIC : {aic_value}')
            aic.append((param, seasonal_param, aic_value))

            # 최적의 모델 정보 업데이트
            if best_model_info is None or aic_value < best_model_info[2]:
                best_model_info = (param, seasonal_param, aic_value)
        except Exception as e:
            print(f'SARIMA: {param} x {seasonal_param} 모델 적합 중 오류 발생: {e}')

# 최적의 SARIMA 모델 출력
if best_model_info:
    print(f'\n최적의 SARIMA 모델: {best_model_info[0]} x {best_model_info[1]} ==> 최저 AIC: {best_model_info[2]}')
else:
    print('모든 모델 적합 중 오류 발생.')
```

최적의 SARIMA 모델: (0, 1, 2) x (1, 1, 2, 12) ==> 최저 AIC: 566.38

- Parameter Estimation

✓ SARIMA(0,1,2)(1,1,2)12 → AIC = 566.38

```
# 1. 데이터 분할 (train/test)
train_data, test_data = train_test_split(data, test_size=0.2, shuffle=False)

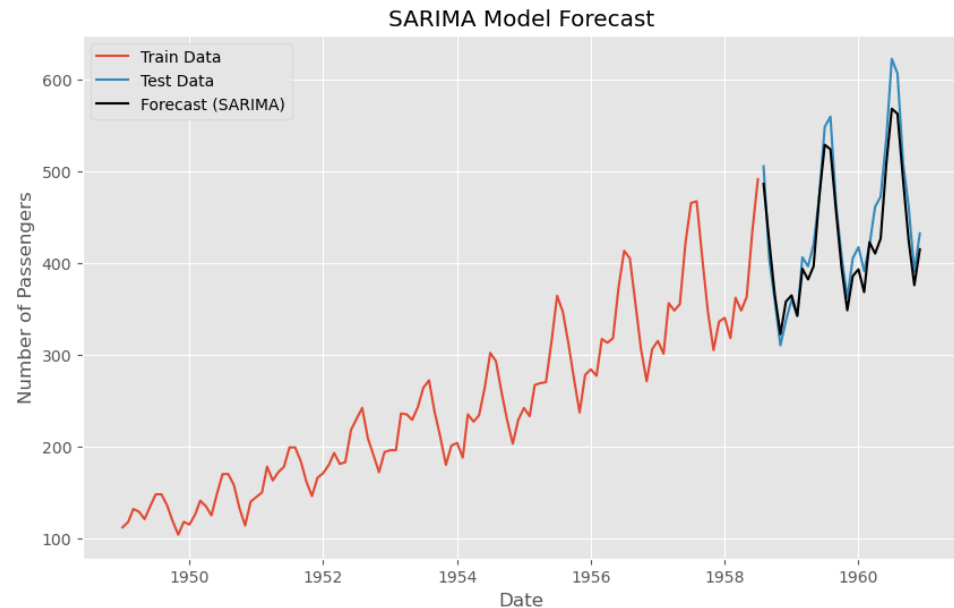
# 2. SARIMA 모델 훈련 및 예측 수행
# order=(p, d, q), seasonal_order=(P, D, Q, s)
model = SARIMAX(train_data, order=(0, 1, 2), seasonal_order=(1, 1, 2, 12))
model_fit = model.fit()

# 3. 예측 (테스트 데이터 기간에 대한 예측 수행)
forecast = model_fit.forecast(steps=len(test_data))

# 4. 예측 값에 대한 날짜 인덱스 생성 (기존 데이터 마지막 이후부터)
forecast_index = pd.date_range(start=train_data.index[-1], periods=len(test_data)+1, freq='MS')[1:]

# 5. 예측 값에 인덱스 할당
forecast = pd.Series(forecast, index=forecast_index)

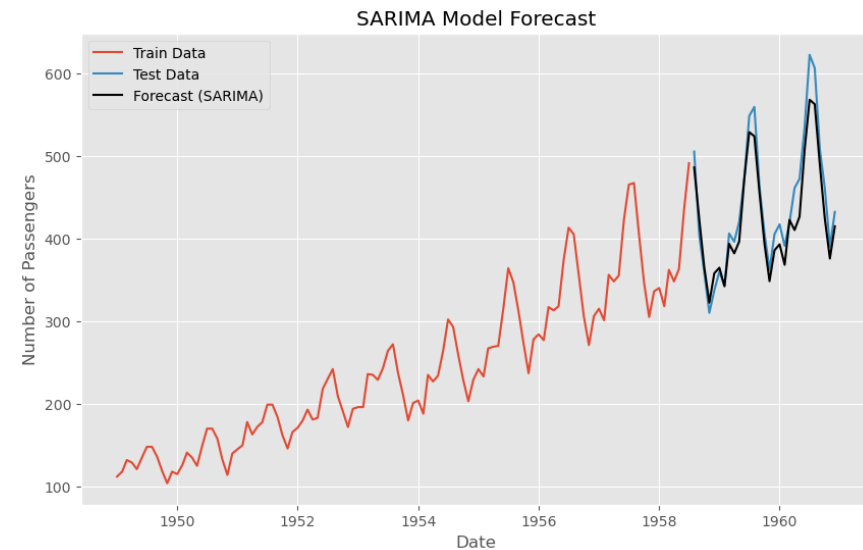
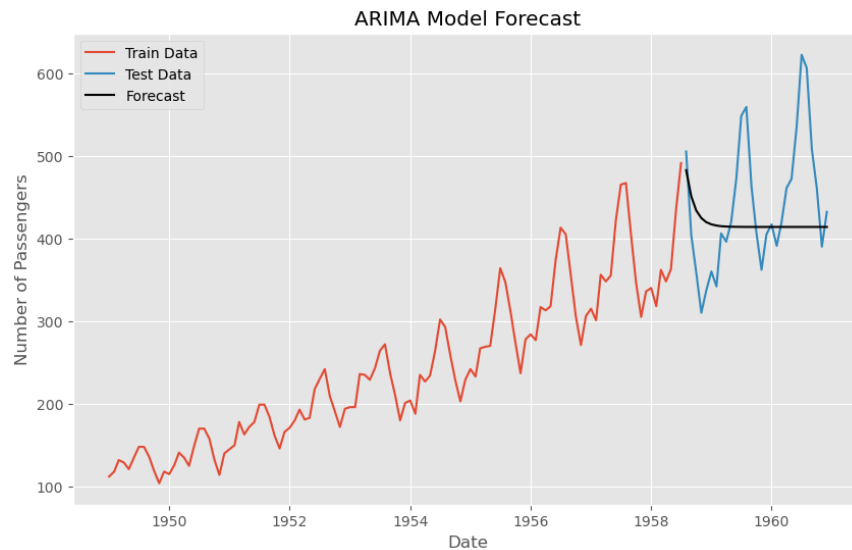
# 6. 예측 결과와 실제 데이터를 시각화
plt.figure(figsize=(10, 6))
plt.plot(train_data.index, train_data, label='Train Data')
plt.plot(test_data.index, test_data, label='Test Data')
plt.plot(forecast.index, forecast, label='Forecast (SARIMA)', color='black')
plt.title('SARIMA Model Forecast')
plt.xlabel('Date')
plt.ylabel('Number of Passengers')
plt.legend()
plt.grid(True)
plt.show()
```



# 03 | Example



- **Model Performance Evaluation**



감사합니다.