09. R 정형데이터 분석 04
시계열 예측

성현곤

충북대학교 도시공학과
Dept.of URBAN ENGINEERING

목차

R의 날짜와 시간

- R에서 날짜와 시간 클래스
  - Date : 날짜
    - 1970년 1월 1일 이후 경과된 일 수를 저장
  - POSIX(Portable Operating System Interface)
    - UNIX간 소통 가능한 프로그램 인터페이스 규약
    - POSIXct(continuous) POSIXt(POSIXlt)(list time)
    - R은 날짜, 시간 데이터를 처리 할 수 있도록 POSIXct, POSIXt(POSIXlt)클래스를 이용한다.
  - POSIXct : 날짜-시간
    - 1970년 1월 1일에서 경과된 초 수와 타임존을 저장
  - POSIXlt : 날짜-시간
    - 1900년에서 경과된 년수, 월, 일, 시간, 분, 초, 타임존 등을 리스트의 형태로 저장

```
> d <- Sys.Date()
> print(d)
[1] "2019-11-17"
> class(d)
[1] "Date"
>
> t <- Sys.time()
> print(t)
[1] "2019-11-17 14:06:26 KST"
> class(t)
[1] "POSIXct" "POSIXt"
>
> unclass(d); print.default(d)
[1] 18217
[1] 18217
attr(,"class")
[1] "Date"
> unclass(t); print.default(t)
[1] 1573967187
[1] 1573967187
attr(,"class")
[1] "POSIXct" "POSIXt"
```

날짜와 시간

날짜 형식(포맷)

```
> ## 날짜 포맷
> as.Date('1/15/2020',format='%m/%d/%Y')
[1] "2020-01-15"
> as.Date('April 26, 2020',format='%B %d, %Y')
[1] NA
> Sys.setlocale("LC_ALL", "English")
[1] "LC_COLLATE=English_United States.1252;LC_CTYPE=English_Unite
LC_NUMERIC=C;LC_TIME=English_United States.1252"
> as.Date('April 26, 2020',format='%B %d, %Y')
[1] "2020-04-26"
```

Conversion specification	Description	Example
%a	Abbreviated weekday	Sun, Thu
%A	Full weekday	Sunday, Thursday
%b or %h	Abbreviated month	May, Jul
%B	Full month	May, July
%d	Day of the month 01-31	27, 07
%j	Day of the year 001-366	148, 188
%m	Month 01-12	05, 07
%U	Week 01-53 with Sunday as first day of the week	22, 27
%w	Weekday 0-6 Sunday is 0	0, 4
%W	Week 00-53 with Monday as first day of the week	21, 27
%x	Date, locale-specific	
%y	Year without century 00-99	84, 05
%Y	Year with century on input: 00 to 68 prefixed by 20 69 to 99 prefixed by 19	1984, 2005
%C	Century	19, 20
%D	Date formatted %m/%d/%y	05/27/84, 07/07/05
%u	Weekday 1-7 Monday is 1	7, 4
%n	Newline on output or Arbitrary whitespace on input	
%t	Tab on output or Arbitrary whitespace on input	

출처: https://www.r-bloggers.com/date-formats-in-r/

날짜와 시간 표기

- 날짜 표기방식
  - 우리나라: '2020년 1월 3일'
  - 미국: 'Jan. 3. 2020'
  - 유럽: '3. Jan. 2020'
- ISO 8601 날짜
- ISO 8601 시간
- ISO 8601 날짜-시간
  - "2020년 1월 3일 16시 14분 15초 99"

스타일	표기	의미	예
기본형	(+-)YYYYMMDD	년월일	20200103
확장형	(+-)YYYY-MM-DD	년-월-일	2020-01-03
기본형	(+-)YYYYDDD	년일(1년의 몇번째 일)	2020003
확장형	(+-)YYYY-DDD	년-일(1년의 몇번째 일)	2020-003
기본형	(+-)YYYYWwwD	년주일	2020W013
확장형	(+-)YYYY-Www-D	년-W주-일	2020-W01-3

스타일	표기	의미
기본형	hhmmss(,ss)(Z)(+-hh(:)mm)	시분초(,100분의 1초)(Z)(타임존)
확장형	hh:mm:ss(,ss)(Z)(+-hh(:)mm)	년-월-일(,100분의 1초)(Z)(타임존)

```
20200103161415,99
20200103T161415,99
20200103 161415,99
```

출처: http://ds.sumeun.org/?p=1111

날짜와 시간 표기

- 날짜 표기 변환

```
> ## 날짜 표기 변환
> x <- Sys.time()
> format(x, '%Y-%m-%d %H:%M:%S')
[1] "2019-11-17 14:13:58"
> format(x, '%Y-%jT%H:%M:%S')
[1] "2019-321T14:13:58"
> format(x, '%G-W%V-%u %H:%M:%S')
[1] "2019-W46-7 14:13:58"
```

- 일상적 날짜-시간 표기 인식 및 변환
  - 지역설정으로 변경
    - 한국 -> 미국 -> 독일 ???

```
> ## 날짜 인식 및 변환
> library(magrittr) # 미설치시 install.packages("magrittr")
> ?Sys.setlocale() # Query or Set Aspects of the Locale
> d <- c("May 27 1984", "July 7 2005")
> d
[1] "May 27 1984" "July 7 2005"
> Sys.setlocale("LC_ALL", "English")
[1] "LC_COLLATE=English_United States.1252;LC_CTYPE=English_United States.1252;LC_MO
LC_NUMERIC=C;LC_TIME=English_United States.1252"
> Sys.setlocale("LC_ALL", "Korean")
[1] "LC_COLLATE=Korean_Korea.949;LC_CTYPE=Korean_Korea.949;LC_MONETARY=Korean_Korea.
rea.949"
> d <- c("May 27 1984", "July 7 2005")
> d
[1] "May 27 1984" "July 7 2005"
> Sys.setlocale("LC_ALL", "English")
[1] "LC_COLLATE=English_United States.1252;LC_CTYPE=English_United States.1252;LC_MO
LC_NUMERIC=C;LC_TIME=English_United States.1252"
> d <- as.Date('Jan 01 2020', format='%b %d %Y')
> d
[1] "2020-01-01"
> d <- as.Date('January 03 2020', format='%B %d %Y')
> d
[1] "2020-01-03"
> Sys.getlocale("LC_ALL") %>% strsplit(";")
[[1]]
[1] "LC_COLLATE=English_United States.1252"  "LC_CTYPE=English_United States.1252"
[3] "LC_MONETARY=English_United States.1252" "LC_NUMERIC=C"
[5] "LC_TIME=English_United States.1252"

> Sys.setlocale("LC_ALL", "Korean")
[1] "LC_COLLATE=Korean_Korea.949;LC_CTYPE=Korean_Korea.949;LC_MONETARY=Korean_Korea.
rea.949"
> d <- as.Date('10월 20 2020', format='%b %d %Y')
> d
[1] "2020-10-20"
```

출처: http://ds.sumeun.org/?p=1111
출처: https://www.stat.berkeley.edu/~s133/dates.html

날짜와 시간연산

```
> ## 날짜
> ?Sys.Date() # the current day in the current time zone.
> d <- Sys.Date() # 현재 날짜 확인
> d
[1] "2019-11-17"
> class(d) # 속성 확인
[1] "Date"
> ?format() # 날짜-> 문자열 포맷 변환 : https://stat.ethz.ch/R-manual/R-dev
> tt <- format(d, format = "%Y:%m:%d") # 날짜를 문자열로 변환
> tt
[1] "2019:11:17"
> class(tt)
[1] "character"
> dd <- as.Date(tt, format = "%Y:%m:%d") # 문자열을 날짜로 변환
> dd
[1] "2019-11-17"
> class(dd)
[1] "Date"
> ## 날짜 연산: 사칙연산 중 덧셈과 뺄셈만 가능
> d0 <- as.Date("09/28/2020", format = "%m / %d / %Y")
> d0
[1] "2020-09-28"
> d1 <- as.Date("2019-11-17")
> d2 <- as.Date("2019-12-11")
> d1>d2 # 크기 비교 연산1
[1] FALSE
> d2>d1 # 크기 비교 연산2
[1] TRUE
> d1 - d2 # 뺄셈
Time difference of -24 days
> d1 + d2 # 덧셈? 날자끼리 덧셈이 안되는 이유는?
Error in `+.Date`(d1, d2) :
  이항 연산자 + 는 "Date" 객체에 정의되어 있지 않습니다
> d1 * d2 # 곱셈
Error in Ops.Date(d1, d2) : "Date" 객체에 정의되지 않은 *입니다
> d1 / d2 # 나눗셈
Error in Ops.Date(d1, d2) : "Date" 객체에 정의되지 않은 /입니다
> d1 - 7 # 뺄셈
[1] "2019-11-10"
> d1 + 7 # 덧셈
[1] "2019-11-24"
> d1 * 7 # 곱셈
Error in Ops.Date(d1, 7) : "Date" 객체에 정의되지 않은 *입니다
> d1 / 7 # 나눗셈
Error in Ops.Date(d1, 7) : "Date" 객체에 정의되지 않은 /입니다
```

날짜와 시간

날짜와 시간연산

```
> ## 시간
> ?Sys.time() # an absolute date-time value which can be converted to various time zones
>             # Details, 내용 확인
> t <- Sys.time() # 현재 날짜 뿐 아니라 시간, 그리고 시간대(timezone) 정보까지 포함
> t
[1] "2019-11-17 14:36:11 KST"
> class(t) # the classes "POSIXlt" and "POSIXct" representing calendar dates and times.
[1] "POSIXct" "POSIXt"
> format(t, format="%H시 %M분") # 문자열 변환: 구체적인 내용 설정 필요
[1] "14시 36분"
> format(t, format="%Y년 %m월 %d일 %H시 %M분")
[1] "2019년 11월 17일 14시 36분"
> ?as.numeric() # 월, 일, 시, 분, 초 등의 정보를 숫자로 추출
> as.numeric(format(t, "%Y")) # 년
[1] 2019
> as.numeric(format(t, "%m")) # 월
[1] 11
> as.numeric(format(t, "%d")) # 일
[1] 17
> as.numeric(format(t, "%H")) # 시
[1] 14
> as.numeric(format(t, "%M")) # 분
[1] 36
> Sys.time()
[1] "2019-11-17 14:36:18 KST"
> (t <- Sys.time())              # the current date, as class "POSIXct"
[1] "2019-11-17 14:36:19 KST"
> Sys.time() - 3600 # an hour ago
[1] "2019-11-17 13:36:20 KST"
> as.POSIXlt(Sys.time(), "GMT") # the current time in GMT 그린위치 천문대의 시간(tz='GMT')
[1] "2019-11-17 05:36:20 GMT"
> t
[1] "2019-11-17 14:36:19 KST"
> attributes(t)$tzone <- "Asia/Seoul" # 시간대(zone) 속성을 변경
> t
[1] "2019-11-17 14:36:19 KST"
> t + 3600
[1] "2019-11-17 15:36:19 KST"
> t - 3600
[1] "2019-11-17 13:36:19 KST"
> t * 3600
Error in Ops.POSIXt(t, 3600) :
  "POSIXt" 객체에 정의되어 있지 않은 '*'입니다
> t / 3600
Error in Ops.POSIXt(t, 3600) :
  "POSIXt" 객체에 정의되어 있지 않은 '/'입니다
```

연습문제 00

- 아래와 같이 다양하게 표기되어 있는 문자형 날짜를 날짜형식으로 불러들이고, 그 속성을 확인하시오.
  - "08,30,20“
  - "Aug 30,2020“
  - "30aug2020“
  - "January 31, 2020"

시계열 자료와 모델

- 시계열 자료(time series data)
  - 시간의 흐름에 따라 관찰된 데이터(초, 분, 시간, 일, 주, 월, 분기, 년 등)
    - 시계열 자료의 관측 시점간의 거리는 동일하다고 가정
      - 결측치 존재할 경우 유사시점의 값으로 대체 또는 평균 등의 값으로 보정
    - 자기상관성: 시점간의 상관관계
      - 일정 기간 증가와 감소하는 경우 양의 상관성
      - 시점마다 증감이 반복되는 경우 음의 상관성
- 시계열 분석
  - 시간의 흐름에 따른 특성을 토대로 모형을 설정
    - 자료 자체의 과거값의 정보를 이용하여 미래 예측
  - 지표를 리뷰하며 트렌드에 이상치는 없는 지 모니터링 하는 데 사용되는 분석 기법

- 분석 목적
  - 수요 예측(주가 전망, 인구변동 등) 에 사용
  - 시계열 데이터의 특성 파악(경향, 주기, 계절성, 불규칙성 등)
    - 시계열 데이터의 불규칙성 원인 파악
- 시계열 분석 종류
  - 일변량 시계열 분석: 하나의 변수에 관심을 갖는 시계열 분석 (예: ARIMA모델)

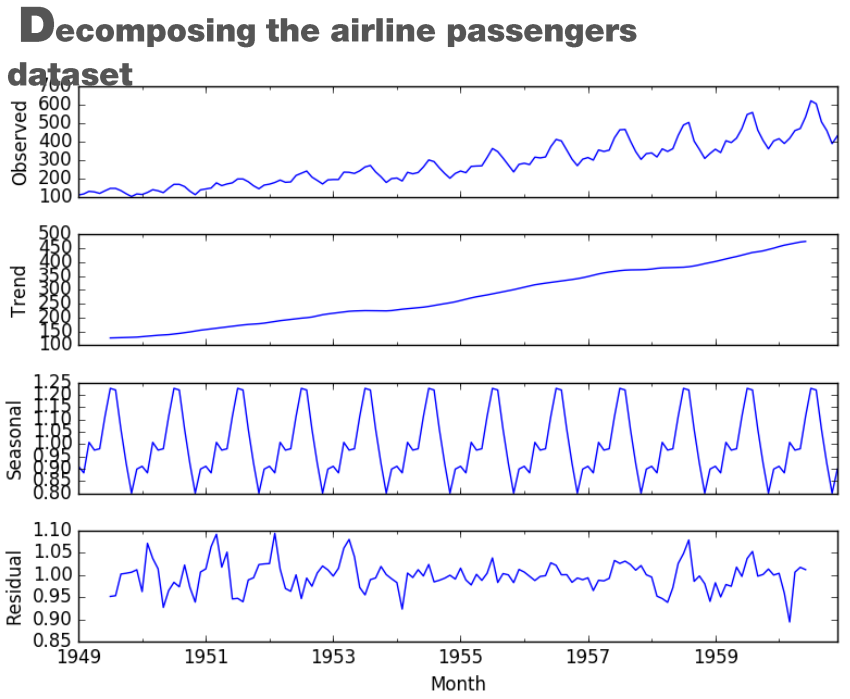
$$Z_t = \Phi_1 Z_{t-1} + \Phi_2 Z_{t-2} + \cdots + \Phi_p Z_{t-p} + \alpha_t$$

  - 다중시계열 분석: 일변량 종속 시계열 변수 이외에 이에 영향을 주는 원인 변수 시계열 분석
    - 계량경제 모델, 전이함수 모델, 개입모델, 상태공간 모델, 다변량 ARIMA모델 등

시계열 자료와 모델링 개요

시계열 자료 분해

- 시계열 분석의 장단점
  - 장점: 예측값의 추정에 유용
  - 단점: 시간의 흐름에 따른 특성만을 토대로 모형을 설정하므로, 이론적 관계를 고려하지 못한다는 한계
- 시계열 자료
  - 비정상성 시계열 자료
    - 시계열 분석에 사용되기 다소 어려운 자료(대부분의 자료)
  - 정상성 시계열 자료
    - 비정상 시계열 자료를 다루기 쉬운 변환한 자료
    - 변수의 변환(로그 등), 차분, 분해하여 정상성 자료로 변환
- 시계열 분해(Decomposing)
  - 체계적 vs. 비체계적 요소(components)
    - **Systematic: 일관성 도는 재현(발)성 요소**
    - **Non-Systematic: 모델링되어지는 않는 요소**
  - 요소
    - **Level**: The average value in the series.
    - **Trend**: The increasing or decreasing value in the series.
    - **Seasonality**: The repeating short-term cycle in the series.
    - **Noise**: The random variation in the series.
  - 가법과 승법 분해
    - **Additive Model: y(t) = Level + Trend + Seasonality + Noise**
    - **Multiplicative Model: : y(t) = Level * Trend * Seasonality * Noise**

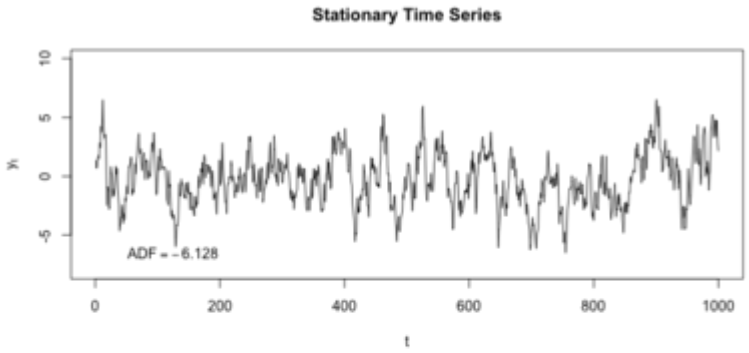


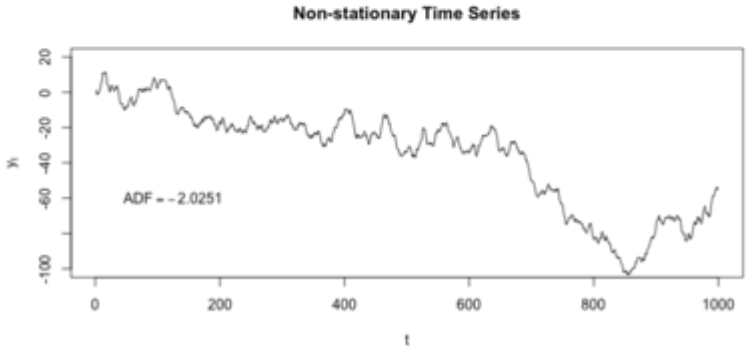
- Seasonality S_t: seasonal trends (e.g. gym memberships rise at the start of the new year)
- Trend T_t: overall trends (e.g. the global temperature is increasing)
- Error ϵ_t: unexplained noise (the less the better for a time-series model to be valid)

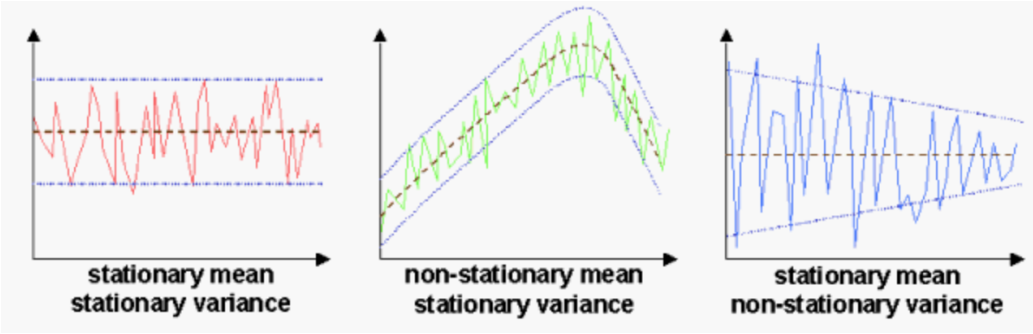
- Additive: amplitutdes of seasonal effects are similar in each period. $y_t = S_t + T_t + \epsilon_t$
- Multiplicative: seasonal trend changes with the progression of the time series. $y_t = S_t\ T_t\ \epsilon_t$

시계열 자료와 정상성

- 정상성(stationarity)
  - 시계열 자료의 평균, 분산이 일정하며, 공분산이 시차에만 의존할 경우
  - 대부분의 시계열 자료는 다루기 어려운 비정상성 시계열 자료이기 때문에 분석하기 쉬운 정상성 시계열 자료로 변환
    - The statistical properties of a process generating a time series do not change over time.
- 정상(stationarity) 시계열 요구
  - 안정 시계열이 되기 이전까지는 시계열 모델을 구축할 수 없기 때문
    - 검정방법: Dickey Fuller Test of Stationarity
- 정상화 방법: detrending, differencing, Seasonality 등 조정
  - Detrending(탈추세) : simply remove the trend component from the time series.
  - Differencing(차분): the Integration part in AR(I)MA
  - Seasonality(계절성): easily be incorporated in the ARIMA model directly.



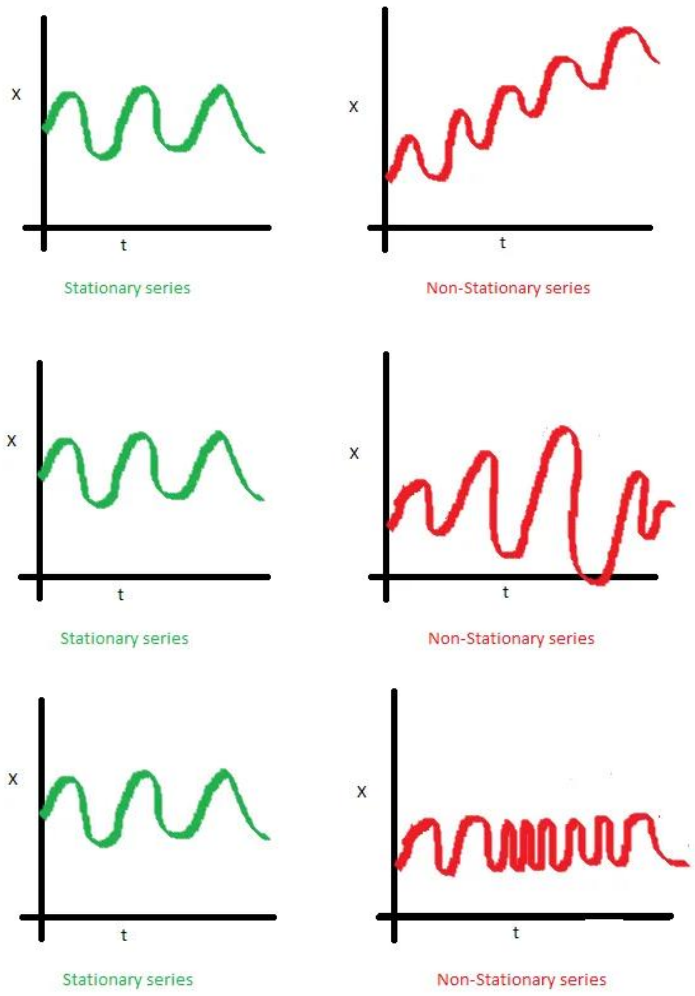




출처: https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322

시계열 자료와 정상성

- 정상(안정) 시계열의 3가지 특징
  - 정상시계열 = 평균, 분산, 공분산 기준 모두 만족
  - 평균 : 시간의 추이와 관계 없이 평균이 일정
    - 평균이 일정하지 않은 시계열은 차분(difference)을 통해 정상화
    - 차분은 현시점 자료에서 이전 시점 자료를 빼는 것
  - 분산 : 시간의 추이와 관계 없이 분산이 일정
    - 분산이 일정하지 않은 시계열은 변환(transformation)을 통해 정상화
  - 공분산: 두 시점 간의 공분산이 기준시점과 무관
    - 공분산도 시차에만 의존할 뿐, 특정 시점에는 의존하지 않음



출처: https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/

불안정 시계열과 모델링

- 불안정 시계열과 Random Walk(무작위 행보)
  - Random Walk(무작위 행보, 취보: drunkard walking)
    - 임의 방향으로 향하는 연속적인 걸음을 나타내는 수학적 개념
    - 예측 불가능한 경로로 변동이 이루어지는 것을 뜻함
  - 시간에 따른 편차의 평균이 0이지만 분산은 시간에 비례하여 증가
    - 따라서, 앞뒤로 움직일 확률이 동일하다고 해도 시간이 흐름에 따라 평균에서 점차 벗어나는 경향을 보임
- 불안정 시계열 모델
  - 확률보행 모델(random walk model)
    - 비정상 시계열에 대하여 가장 최근의 과거값만 현재에 영향을 미친다고 가정
    - 모델 적합 = 가장 최근 과거 값(t-1)
  - 표류가 있는 확률보행 모델(Random Walks with drift)
    - 확률보행모형에 상수항과 시간적 추세가 추가된 경우(시간적 추세 고려)
  - 허구적(가성적) 회귀(Spurious Regression) 모델
    - 아무런 관련이 없으나 안정적이지 않아 유의미한 회귀모형이 구축되는 경우

2. 확률보행(Random Walks) 모형

$Y_t=Y_{t-1}+e_t$와 같이 t기의 Y 값이 t-1기의 Y값과 t기의 오차항에 의해 결정되는 시계열 모형을 확률보행(or 랜덤워크)라고 부른다.

확률보행의 경우 Wold 표현법으로 나타낸다면 다음과 같이 나타낼 수 있다.

$$Y_t=Y_0+\sum_{s=1}^{t}e_s$$

3. 표류가 있는 확률보행(Random Walks with drift) 모형

$Y_t=\alpha+Y_{t-1}+e_t$와 같이 확률보행 모형에 상수항이 추가된 경우이다.

표류가 있는 확률보행모형도 Wold 표현법으로 나타낸다면 다음과 같이 나타낼 수 있다.

$$Y_t=\alpha t+Y_0+\sum_{s=1}^{t}e_s$$

그냥 확률보행과 달리 표류가 있는 확률보행은 αt라는 시간적 추세가 있다.

4. 허구적 회귀(=가성적 회귀)(Spurious Regression)

불안정적인 자료가 회귀분석에 사용될 경우 서로 연관이 없는 자료들이 서로 유의한(의미가 있는) 회귀분석 결과를 얻을 수 있는 위험이 있다. 이를 허구적 회귀라고 한다.

X와 Y 모두 단위근이 있다고 하자.(X~I(1)) 즉 X, Y 둘다 안정적이지 않은 확률변수이다. 그리고 X와 Y는 아무런 관련이 없는 변수들이라고 하자. 예를 들어보자.

$Y_t=\beta_0+\beta_1X_t+e_t$로 회귀분석을 돌렸을 때 $H0:\beta_1=0$, $H1:\beta_1\neq0$로 가설검정한 결과 X의 계수 β_1이 유의하다고 판정되면 허구적 회귀라고 말한다.

출처: http://blog.naver.com/PostView.nhn?blogId=jahyone20&logNo=220934340547

시계열 자료와 모델링 개요

시계열 모델: AR과 MA

- 자기회귀(Autoregressive, AR) 모델
  - p 시점 전의 자료가 현재 자료에 영향을 주는 모델
    - Generated as a linear function of its past values, plus a random noise/error
    - P 시점 이전(시차)의 자료가 현재 자료에 영향을 줌
  - *vector autoregressive (VAR)* model
    - AR model to the multivariate case; now each element of the vector *x[t]* of length k can be modeled as a linear function of all the elements of the past *p* vectors

$$x_t = c + \phi_1 x_{t-1} + \ldots + \phi_p x_{t-p} + \varepsilon_t$$

- ϕ_i(coefficients) = weights measuring the influence of these preceding values on the value x*[t]*
- c = constant intercept
- ε_i = a univariate *white noise process* (commonly assumed to be Gaussian)

$$x_t = c + A_1 x_{t-1} + \ldots + A_p x_{t-p} + e_t$$

- 이동평균 (Moving average, MA) 모델
  - **충격에 의해 데이터가 일시적으로 정상값과 떨어져서 그것이 다시 정상값으로 향하는 움직임을 표현하는 모델**
    - **과거 잘 알 수 없는 원인이 겹쳐서 현재의 데이터가 표현되어짐**
  - *MA(q)* = a linear function of the last *q+1* random shocks generated by ε_i, a univariate *white noise process*

$$x_t = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q}$$

- **Autoregressive moving average (ARMA) model(1)**
  - *ARMA(p,q)* model = a linear function of the last *p* values and the last *q+1* random shocks generated by ε_i, a univariate *white noise process*

$$x_t = \phi_1 x_{t-1} + \ldots + \phi_p x_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q}$$

출처1: https://www.datascienceblog.net/post/machine-learning/forecasting-an-introduction/
출처2: https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322

시계열 모델: AR과 MA

- **Autoregressive moving average (ARMA) model(2)**

ARMA stands for *autoregressive moving average*. ARMA models are only appropriate for stationary processes and have two parameters:

- **p:** the order of the autoregressive (AR) model
- **q**: the order of the moving-average (MA) model

The ARMA model can be specified as

$$\hat{y}_t = c + \epsilon_t + \sum_{i=1}^{p} \phi_i y_{t-i} - \sum_{j=1}^{q} \theta_j \epsilon_{t-j} \,.$$

with the following variables:

- c: the intercept of the model (e.g. the mean)
- ϵ_t: random error (white noise, residual) associated with measurement t with $\epsilon_t \sim N(0, \sigma^2)$.
- $\phi \in \mathbb{R}^p$: a vector of coefficients for the AR terms. In R, these parameters are called *AR1*, *AR2*, and so forth.
- y_t: outcome measured at time t
- $\theta \in \mathbb{R}^q$: a vector of coefficients for the MA terms. In R, these parameters are called *MA1*, *MA2*, and so forth.
- ϵ_t: noise associated with measurement t

출처: https://www.datascienceblog.net/post/machine-learning/forecasting-an-introduction/

시계열 자료와 모델링 개요

시계열 모델: AR과 MA

- 자기상관함수
  - 자기상관함수(Autocorrelation Function, ACF) : k 기간 떨어진 값들의 상관계수
    - 자기상관계수(ACF): 두 시계열 확률변수(시점)간의 상관관계만 계산
  - 부분자기상관함수(partial ACF) : 서로 다른 두 시점의 중간에 있는 값들의 영향을 제외시킨 상관계수
    - 부분자기상관계수: 두 시계열 확률변수(시점)간에 다른 시점의 확률변수의 영향력은 통제하고 상관관계만 계산
- p와 q의 차수 결정
  - AR의 p 차수 결정
    - ACF 빠르게 감소, PACF는 어느 시점에서 절단점을 갖는다
    - PACF가 2시점에서 절단점 가지면 AR(1) 모형
  - MA의 q 차수 결정
    - 유한한 갯수의 백색잡음 결합이므로 항상 정상성 만족
    - ACF가 절단점을 갖고, PACF는 빠르게 감소

출처: https://www.datascienceblog.net/post/machine-learning/forecasting-an-introduction/

-자기상관계수 (AutoCorrelation Function, ACF)

: 자기상관계수 (AutoCorrelation) : k 기간 떨어진 값들의 상관계수

$$p_k = \frac{\lambda_k}{\lambda_0} = \frac{Cov(Y_t, Y_{t-k})}{\sqrt{Var(Y_t)Var(Y_{t-k})}}$$

$where\ \lambda = Cov(Y_t, Y_{t-k})$: 자기공분산$(Autocovariance)$

-자기상관계수 함수(Autocorrelation Function, ACF) : 상관계수k를 함수 형태로 표시 Φ

만일 $-1 < \Phi < 1$ 이면 두 지점 간의 거리가 멀어질수록(k가 커질수록) ACF는 0으로 수렴

-부분(편) 자기상관계수(Partial ACF)

: 서로 다른 두 지점 사이의 관계를 분석할 때 중간에 있는 값들의 영향을 제외시킨 상관관계 개념

$$Corr(Y_t, Y_{t-k} | Y_{t-1}, \cdots, Y_{t-k+1})$$

자기상관함수(ACF)

1)자기회귀(AR) 프로세스 경우 : 지수함수 또는 사인곡선 형태로 서서히 '0'으로 감소

2)이동평균(MA) 프로세스 경우 : 이동평균 차수에 해당하는 시차에 두드러진 스파이크, 이 시차 이후에는 모두 '0'으로 절단

3)두 프로세스가 혼합되어 있는 경우 : '0'을 향해 서서히 감소

편자기상관함수(PACF)

1)자기회귀(AR) 프로세스 경우 : 자기회귀 차수에 해당하는 시차에 두드러진 스파이크, 이 시차 이후에는 모두 '0'으로 절단

2)이동평균(MA) 프로세스 경우 : 지수함수 또는 사인곡선 형태를 서서히 '0'으로 감소

3)두 프로세스가 혼합되어 있는 경우 : '0'을 향해 서서히 감소

확률과정	ACF	PACF
AR(p)	지수적으로 감소하거나 소멸하는 싸인함수 형태	시차 p이후에는 0으로의 절단형태
MA(q)	시차 q이후에는 0으로의 절단형태	지수적으로 감소하거나 소멸하는 싸인함수 형태
ARMA(p,q)	시차(q-p)이후에는 지수적으로 감소하거나 소멸하는 사인함수형태	시차(p-q)이후에는 지수적으로 감소하거나 소멸하는 싸인함수 형태

시계열 자료와 모델링 개요

시계열 모델: AR과 MA

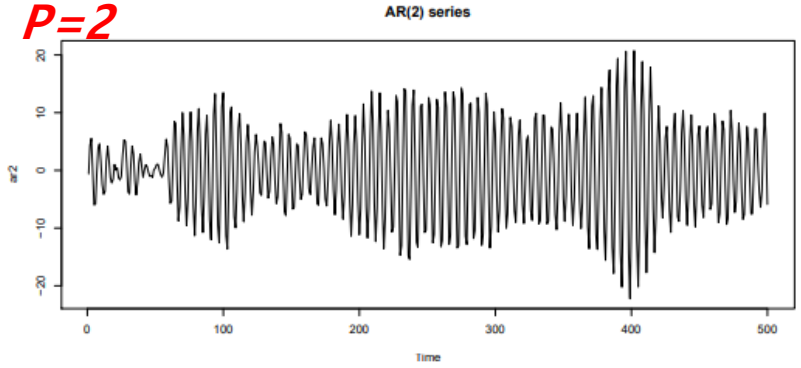
요약: AR/MA 모형과 ACF/PACF 관계

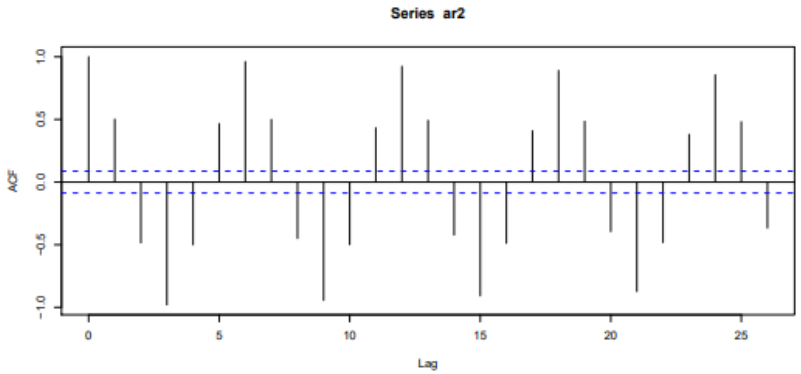
	AR(p)	MA(q)
ACF	점차적으로 감소	시차 q 이후에 0
PACF	시차 p 이후에 0	점차적으로 감소

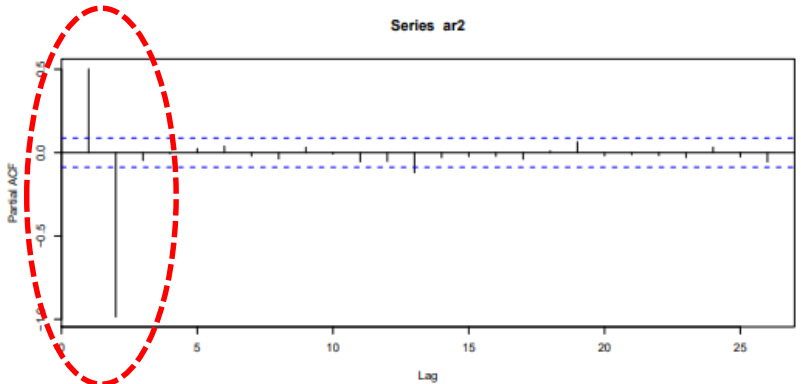
지수, 사인함수형태로

지수, 사인함수형태로

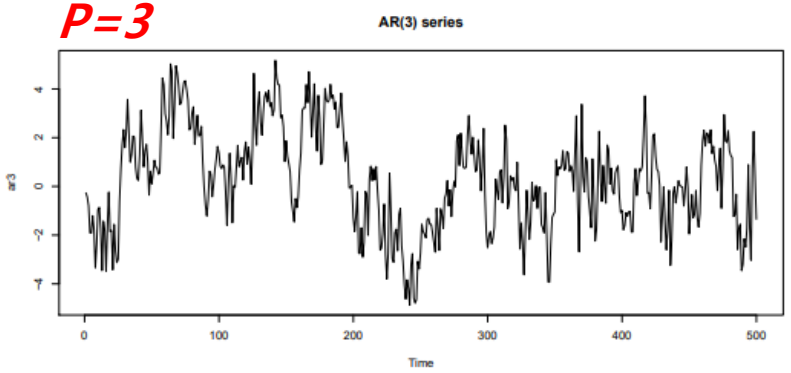
P=2

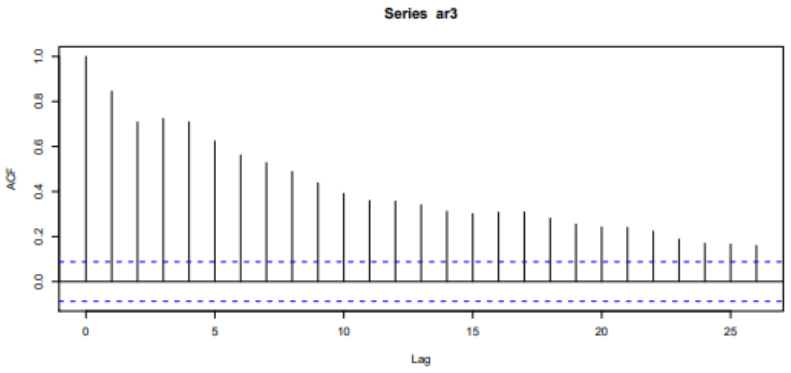


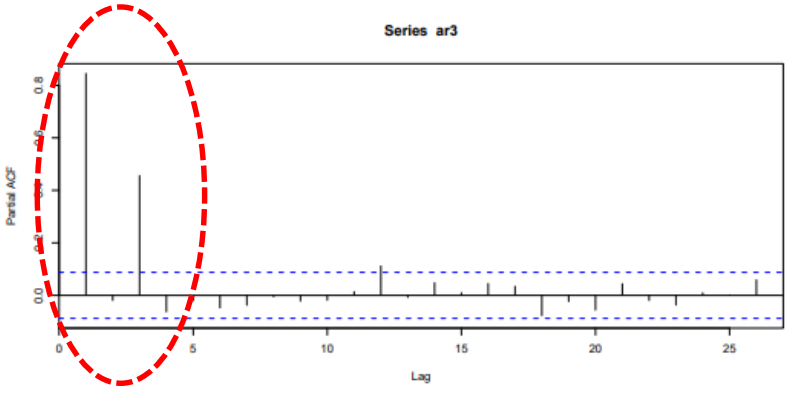




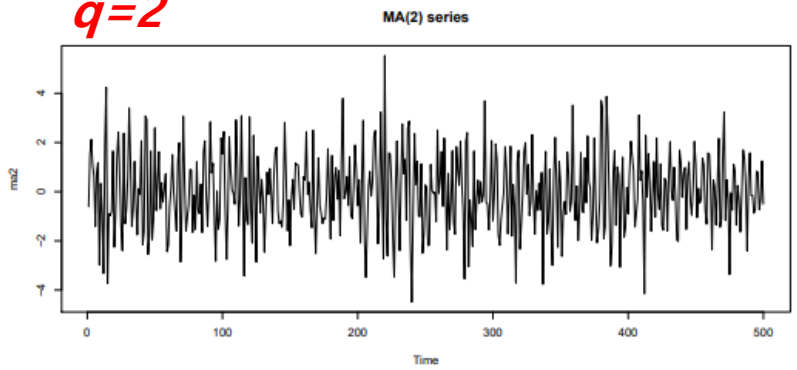
P=3

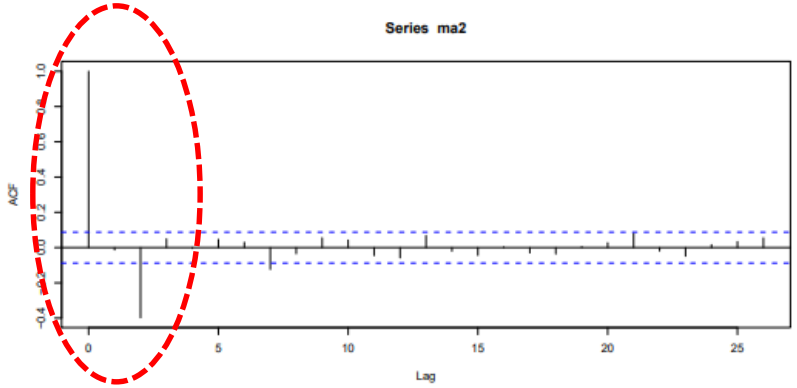


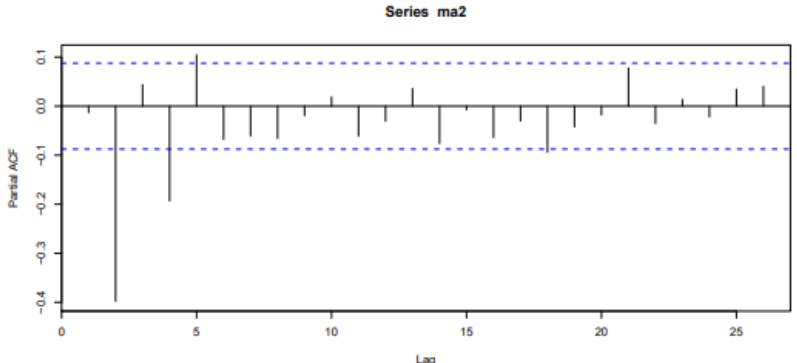




q=2







시계열 모델: ARIMA와 분해 시계열

- 자기회귀 누적 이동평균(Autoregressive integrated moving average, ARIMA) 모델
  - 비정상 시계열 모형
  - 차분이나 변환을 통해 AR, MA, 또는 이 둘을 합한 ARMA 모형으로 정상화
  - ARIMA(p, d, q)
    - d : 차분 차수
    - p : AR 모형 차수
    - q : MA 모형 차수
- 분해 시계열 (회귀)모델
  - 시계열에 영향을 주는 일반적인 요인을 시계열에서 분리해 분석하는 방법
  - 회귀분석적인 방법
    - 계절 요인(seasonal factor), 순환 요인(cyclical), 추세 요인(trend), 불규칙 요인(random)

The ARIMA model

ARIMA stands for *autoregressive integrated moving average* and is a generalization of the ARMA model. In contrast to ARMA models, ARIMA models are capable of dealing with non-stationary data, that is, time-series where the mean or the variance changes over time. This feature is indicated by the *I* (integrated) of ARIMA: an initial differencing step can eliminate the non-stationarity. For this purpose, ARIMA models require an additional parameter, d, which defines the degree of differencing.

Taken together, an ARIMA model has the following three parameters:

- **p:** the order of the autoregressive (AR) model
- **d:** the degree of differencing
- **q**: the order of the moving-average (MA) model

In the ARIMA model, outcomes are transformed to differences by replacing y_t with differences of the form

$$(1-B)^d y_t\,.$$

The model is then specified by

$$\phi_p(B)(1-B)^d y_t = \theta_q(B)\epsilon_t\,.$$

For $d=0$, the model simplifies to the ARMA model since $(1-B)^0 y_t = y_t$. For other choices of d we obtain backshift polynomials, for example:

$$(1-B)^1 y_t = y_t - y_{t-1}$$
$$(1-B)^2 y_t = (1-2B+B^2)y_t = y_t - 2y_{t-1} + y_{t-2}$$

In the following, let us consider the interpretation of the three parameters of ARIMA models.

$$Z_t = f(T_t, S_t, C_t, I_t)$$

T_t : 경향 요인
S_t : 계절 요인
C_t : 순환 요인
I_t : 불규칙 요인
Z_t : 시계열 값,
f : 미지의 함수

출처: https://www.datascienceblog.net/post/machine-learning/forecasting-an-introduction/

시계열 자료와 모델링 개요

정상성 검정

- 확률적 추세와 비정상성
  - 확률적 추세가 존재하는 경우 차분에 의해 시계열자료를 정상화한 후 분석을 수행
    - 결정적 추세가 있는 경우는 추세항을 모형에 포함시켜야 함
- 정상성 검정 = 단위근 검정(Unit root test)
  - 확률변수가 안정적인지 불안정적인지 여부를 확인하여 주는 방법
    - 원자료, Log 변환, 차분 변환 등 자료에 대하여 단위근 검정 실시
  - Dicky-Fuller Test: 정상성을 알아보기 위한 단위근 검정의 대표적인 방법 중 하나
    - The Augmented Dickey-Fuller test allows for higher-order autoregressive processes $\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \delta_1 \Delta y_{t-1} + \delta_2 \Delta y_{t-2} + \dots$
    - 3가지 종류
      - 상수항도 없고, 추세도 없는 경우
      - 상수항은 있고, 추세가 없는 경우
      - 상수항도 있고, 추세가 있는 경우
    - 가설
      - 귀무가설(H0): 자료에 단위근이 존재한다.(random walk)
      - 대립가설(H1): 시계열 자료가 정상성을 만족한다(또는 추세 정상성을 만족한다).

The Dickey-Fuller test is testing if $\phi = 0$ in this model of the data:

$$y_t = \alpha + \beta t + \phi y_{t-1} + e_t$$

$$\Delta y_t = y_t - y_{t-1} = \alpha + \beta t + \gamma y_{t-1} + e_t$$

where y_t is your data. It is written this way so we can do a linear regression of Δy_t against t and y_{t-1} and test if γ is different from 0. If $\gamma = 0$, then we have a random walk process. If not and $-1 < 1 + \gamma < 1$, then we have a stationary process.

1. Test for a unit root:

$$\Delta y_t = \delta y_{t-1} + u_t$$

2. Test for a unit root with drift:

$$\Delta y_t = a_0 + \delta y_{t-1} + u_t$$

3. Test for a unit root with drift and deterministic time trend:

$$\Delta y_t = a_0 + a_1 t + \delta y_{t-1} + u_t$$

백색 잡음과 자기상관성 검정

- 백색잡음(white noise)와 정규분포
  - 백색잡음: 자기상관(autocorrelation)이 없는 시계열
    - 잔차들 간의 선형 의존성이 없음
    - 백색잡음과정(white noise process): 시계열 분석에서의 오차항이 identically and independently distributed...(iid)임

      -평균이 0, 분산이 σ^2, 자기공분산이 0인 경우.

      -시계열간 확률적 독립인 경우 강(Strictly) 백색잡음과정이라 한다.

      -백색잡음과정이 정규분포를 따를 경우 이를 가우시안(Gaussian) 백색잡음과정이라고 한다. $\epsilon_t \sim \text{i.i.d. } N(\mu, \sigma^2)$
- 시계열 자기상관성 검정: 륭(Ljung)–박스(Box) 검정
  - Q 통계량
    - n = 표본수
    - ρ^2=자기상관계수
    - K와 h = 차수
    - Q는 카이스퀘어 통계량을 근사적으로 따름

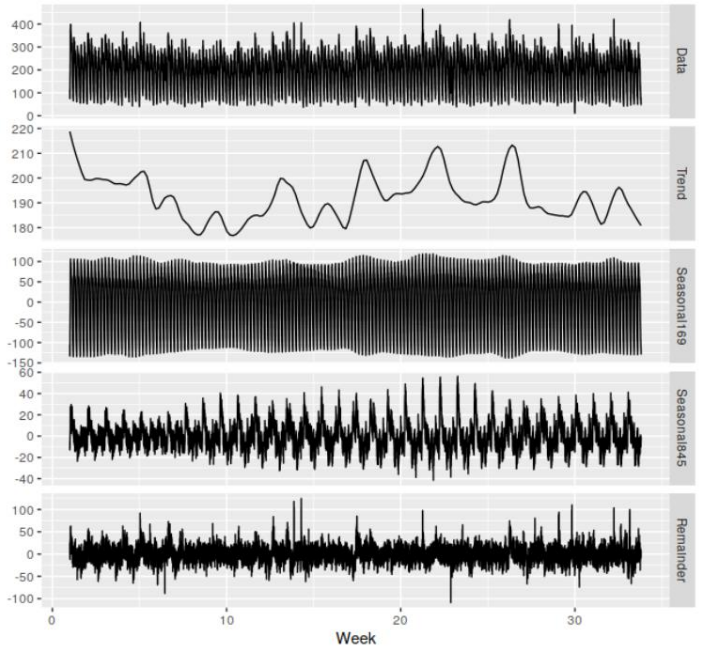
$$Q = n(n+2)\sum_{k=1}^{h}\frac{\hat{\rho}_k^2}{n-k}$$

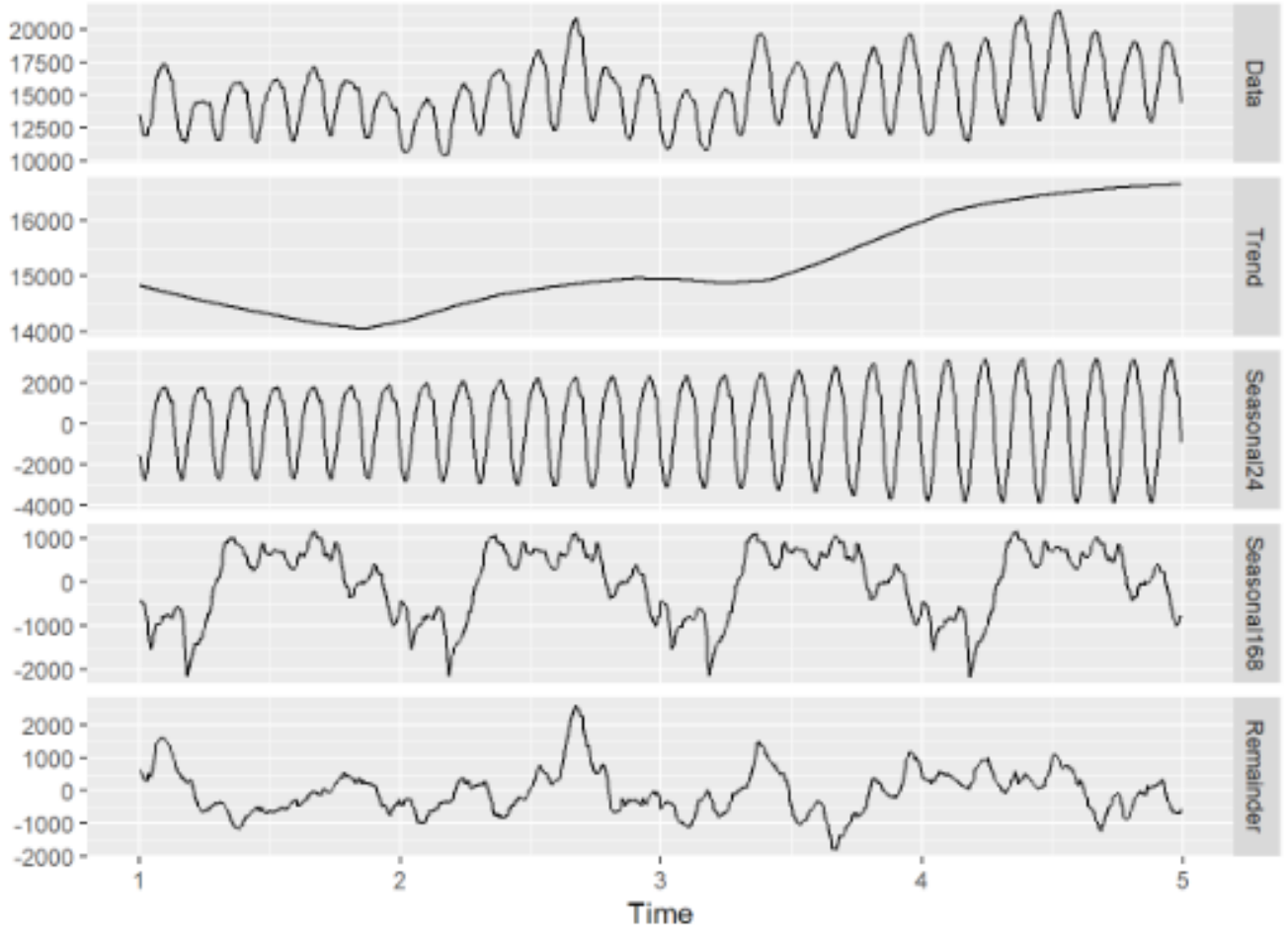
  - 시계열에 m차까지의 자기상관관계가 존재하지 않는다는 귀무가설을 검정하는 것이다.
    - 귀무가설: H0 : p(1) = p(2) = … = p(m) = 0, 잔차는 랜덤이다.
    - 대립가설: H_a: The data are not independently distributed; they exhibit serial correlation.

Box-Pierce test 도 자기상관성 검정 수행

시계열자료와 다중 계절성

- **Complex seasonality** = 다중계절성
  - 논문 빈도의 시계열에서 2개 이상의 시계열성 패턴이 나타남
    - 월별, 분기별 데이터와 달리 일별 시계열 데이터
      - 일별, 주별, 연간 계절성 내포



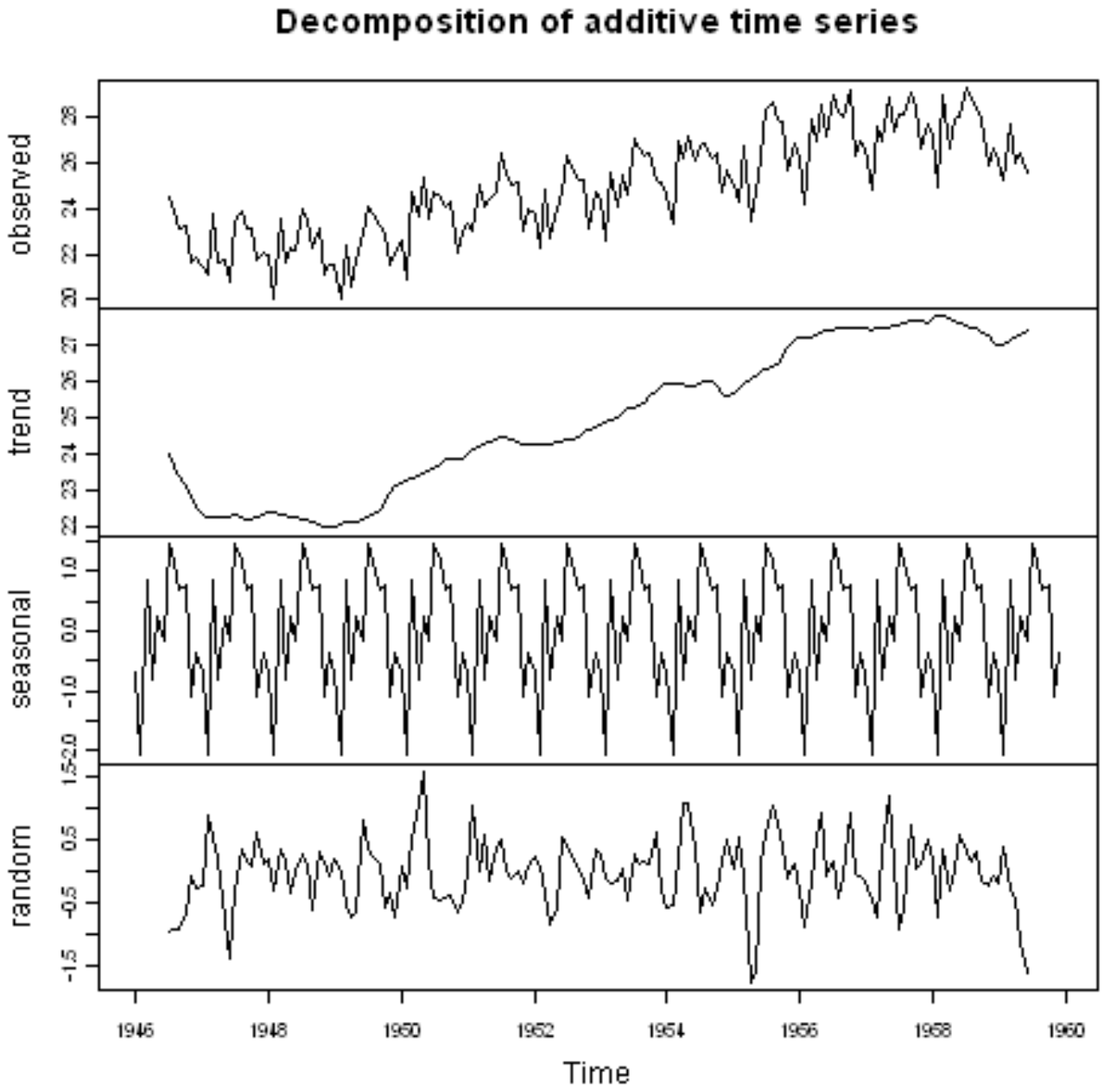


시계열자료와 다중 계절성

- 다중계절성을 고려하는 동적 조화회귀
  - (동적) 조화회귀(dynamic regression)
    - ARMA 오차로 다루는 단기 시계열 동역학을 고려하는 푸리에 항을 이용하는 조화회귀(harmonic regression) 접근 방식
    - 계절성 가변수와 푸리에 항
      - 더미변수로 처리
    - 푸리에 급수: 계절성 가변수 대신 긴 계절성 주기에 푸리에 항 활용
      - 사인과 코사인 항의 급수로 임의의 주기적 함수의 근사치를 낼 수 있다
        - 어떠한 길이의 계절성도 가능
        - 계절성 패턴의 매끄러운 정도(평활도; smoothness): 푸리에 사인과 코사인 쌍의 개수인 K 조절
          - 계절성 패턴은 KK가 작을수록 더 평활
        - 단기 동역학을 단순한 ARMA 오차로 다룸
      - 단점: 계절성 ARIMA 모델과 비교할 때, 계절성이 고정되어있다고 가정
- 다중계절성 ARIMA모델 패키지
  - R의 msts 패키지 활용: mstl() 함수
    - auto.arima()에서의 fourier() 함수 적용
      - fourier()의 첫 번째 입력은 계절성 주기 m과 돌려줄 예측변수(predictor variable)의 길이를 음
      - 두 번째 입력 K으로 사인과 코사인 항을 몇 개 넣을지 지정
        - 최대 K=m/2까지 지정
      - m은 계절성 주기: 최대값을 사용했기 때문에, 얻은 결과가 계절성 가변수를 사용했을 때와 같음

연습문제 01

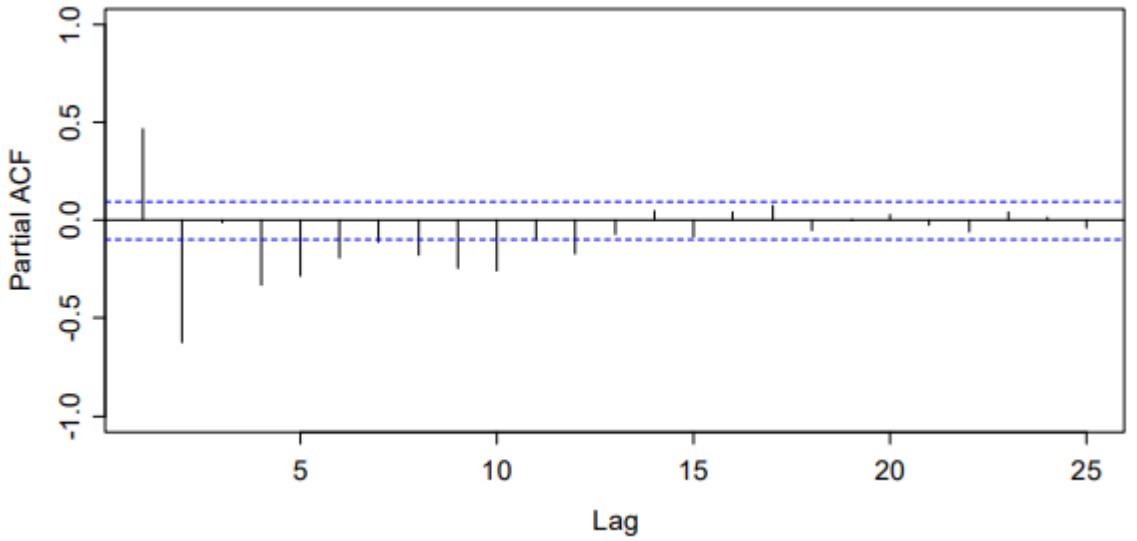
- 시계열 자료를 오른 쪽과 같이 trend, seasonal, random 으로 분해하였다. 이 자료의 특징에 대하여 설명하시오.



- Trend:
  - the estimated trend component shows a small decrease from about 24 in 1947 to about 22 in 1948, followed by a steady increase from then on to about 27 in 1959.
- Seasonality:
  - 12개월 주기
- Random walk:
  - 평균과 분산이 일정한 형태이면서 특정 시점에서의 충격파 존재

연습문제 02

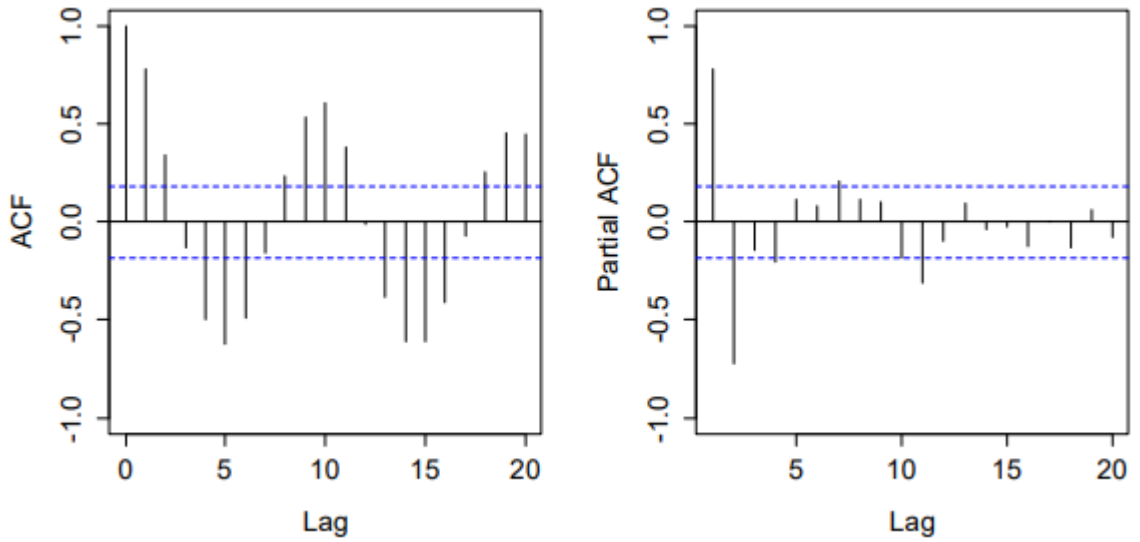
- 부분 자기상관계수(pacf)의 도표가 아래와 같이 나타났다. AR모델에서의 p 차수(order)는 얼마로 하는 것이 좋을까?



Some further PACF coefficients up to lag 10 seem significantly different from zero, but are smaller. From what we see here, we could try to describe the wave tank data with an AR(2) model.

연습문제 03

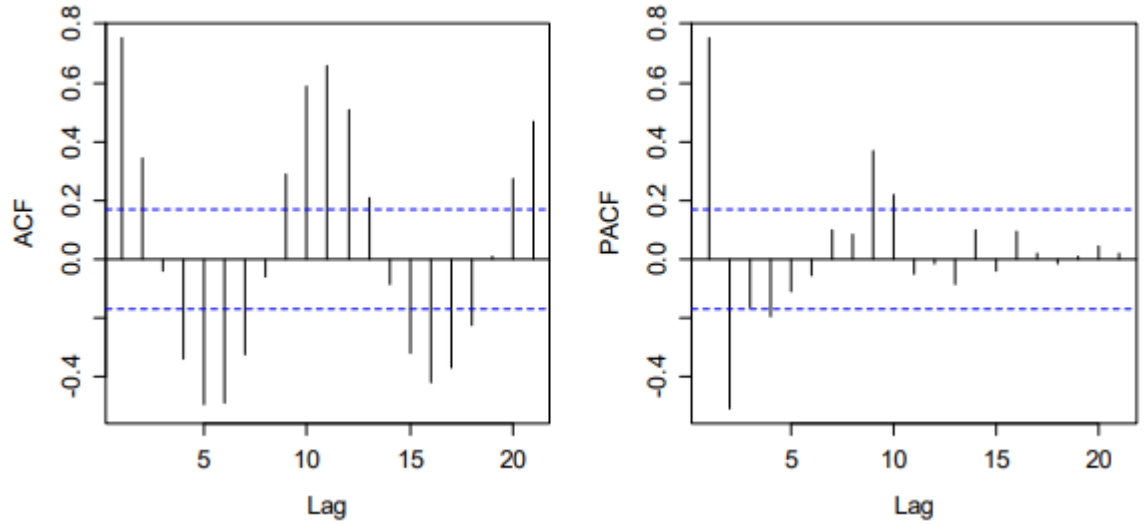
- ACF와 PACF의 도표가 아래와 같이 나타났다. AR모델에서의 p 차수(order)는 얼마로 하는 것이 좋을까?



That is a bit less clear, and several orders p (2,4,7,11) come into question. However in summary, we conjecture that there are no strong objections against fitting an AR p( ) . The choice of the order is debatable, but the parsimony paradigm tells us that we should try with the smallest candidate first, and that is p = 2 .

연습문제 04

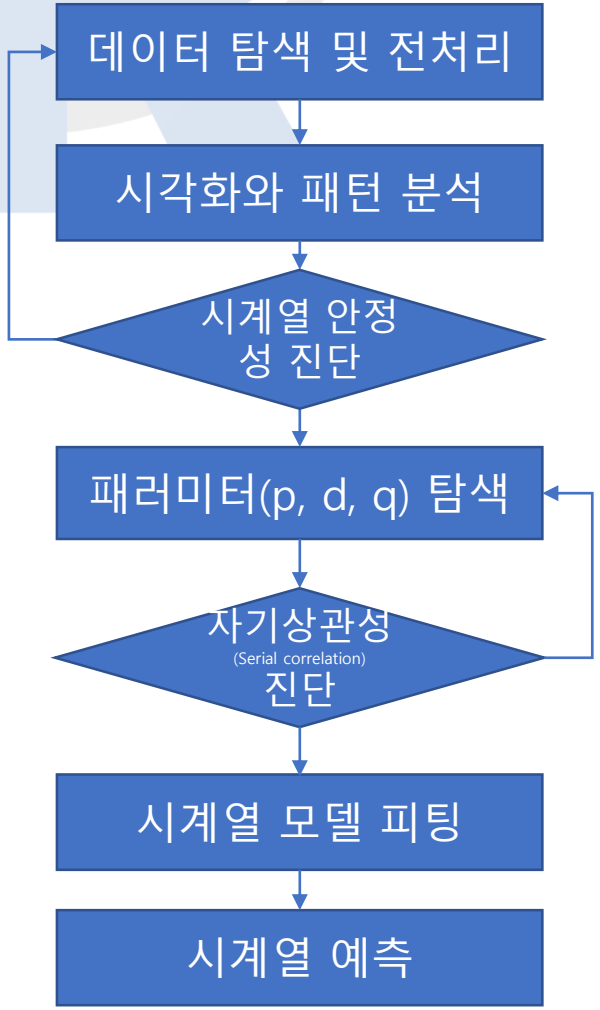
- ACF와 PACF의 도표가 아래와 같이 나타났다. ARMA모델에서의 p와 q의 차수(order)는 얼마로 하는 것이 좋을까?



The ACF has a slow decay and the PACF cuts-off after lag 10, suggesting an AR(10) . We complement with an exhaustive AIC-based search over all ARMA(p, q) up to p q <= 10 .
정답: ARMA(2,1)

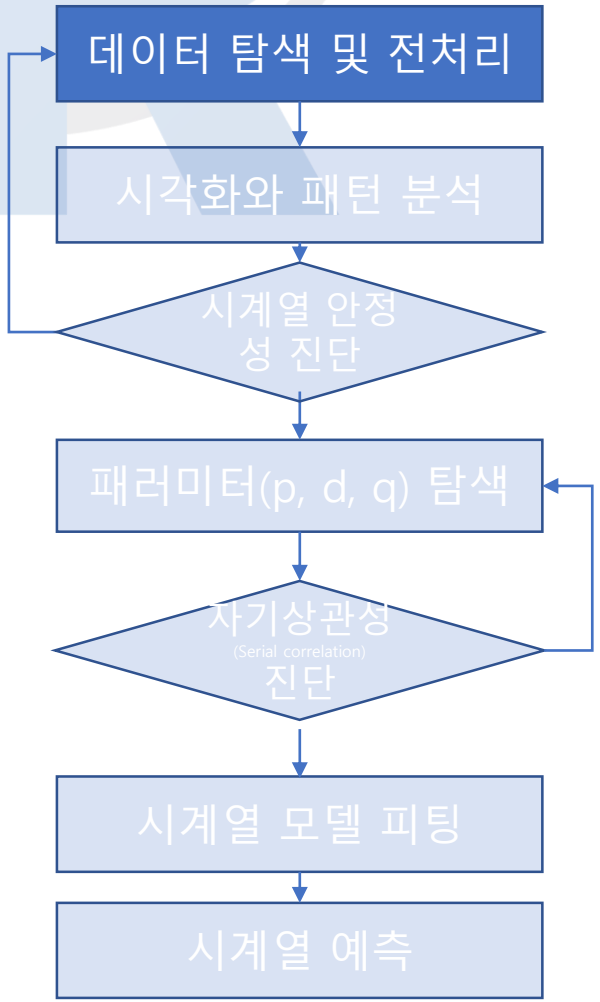
시계열 모델링 분석절차

- ARIMA 시계열 모델링의 분석절차



- 시계열 데이터로 변환과 탐색
  - ts(data, frequency = n, start = c(시작년도, 월))
- 시각화와 trend, seasonal, random 값 등으로 분해
  - decompose(data)
- 단위근 검정: Dickey Fuller Test of Stationarity: adf.test()
- 시계열 데이터를 이동평균한 값 생성
  - SMA(data, n = 이동평균수)
- 시계열 데이터를 차분
  - diff(data, differences = 차분횟수)
- 데이터를 활용하여 최적의 ARIMA 모형을 선택
  - auto.arima(data)
- 시계열 독립성 검정(Box-Pierce 또는 Ljung-Box test): Box.test()
  - ARIMA모형 진단: 잔차가 독립적인가?
- 선정된 ARIMA 모형으로 데이터를 보정(fitting): 모형 적합도 진단=AIC
- ARIMA 모형에 의해 보정된 데이터를 통해 미래값을 예측: forecast.Arima(fittedData, h = 미래예측수)
- 시계열 데이터를 그래프로 표현: plot.ts(시계열데이터)
- 예측된 시계열 데이터를 그래프로 표현: plot.forecast(예측된시계열데이터)

실습1: 월별 국제항공노선 이용자수(1949-1960)




```
> # 실습 1: AirPassengers
> ?AirPassengers # 내장 데이터 설명: 단위 천명
> ## 1. Loading the Data Set
> data(AirPassengers) # 데이터 불러오기
> class(AirPassengers) # 데이터 속성 확인하기
[1] "ts"
> start(AirPassengers) # 시작시점 확인
[1] 1949    1
> end(AirPassengers) # 종료시점 확인
[1] 1960   12
> frequency(AirPassengers) # 주기 확인
[1] 12
> summary(AirPassengers) # 요약통계
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  104.0   180.0   265.5   280.3   360.5   622.0
> str(AirPassengers) # 데이터 구조 확인하기
 Time-Series [1:144] from 1949 to 1961: 112 118 132 129 121 135 148 148 136 119 ...
```

AirPassengers {datasets} R Documentatior

Monthly Airline Passenger Numbers 1949-1960

Description

The classic Box & Jenkins airline data. Monthly totals of international airline passengers, 1949 to 1960.

Usage

```
AirPassengers
```

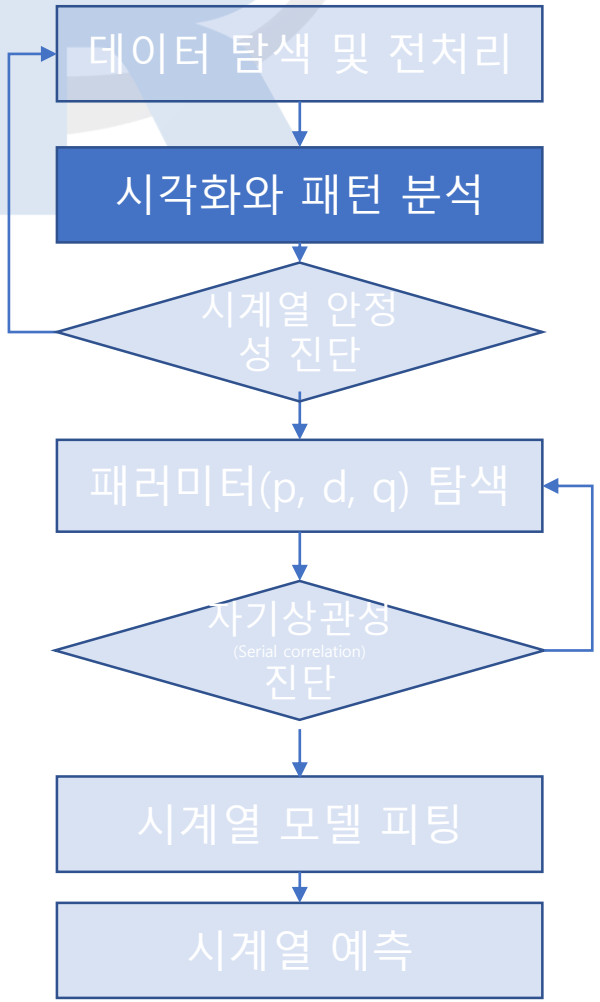
Format

A monthly time series, in thousands.

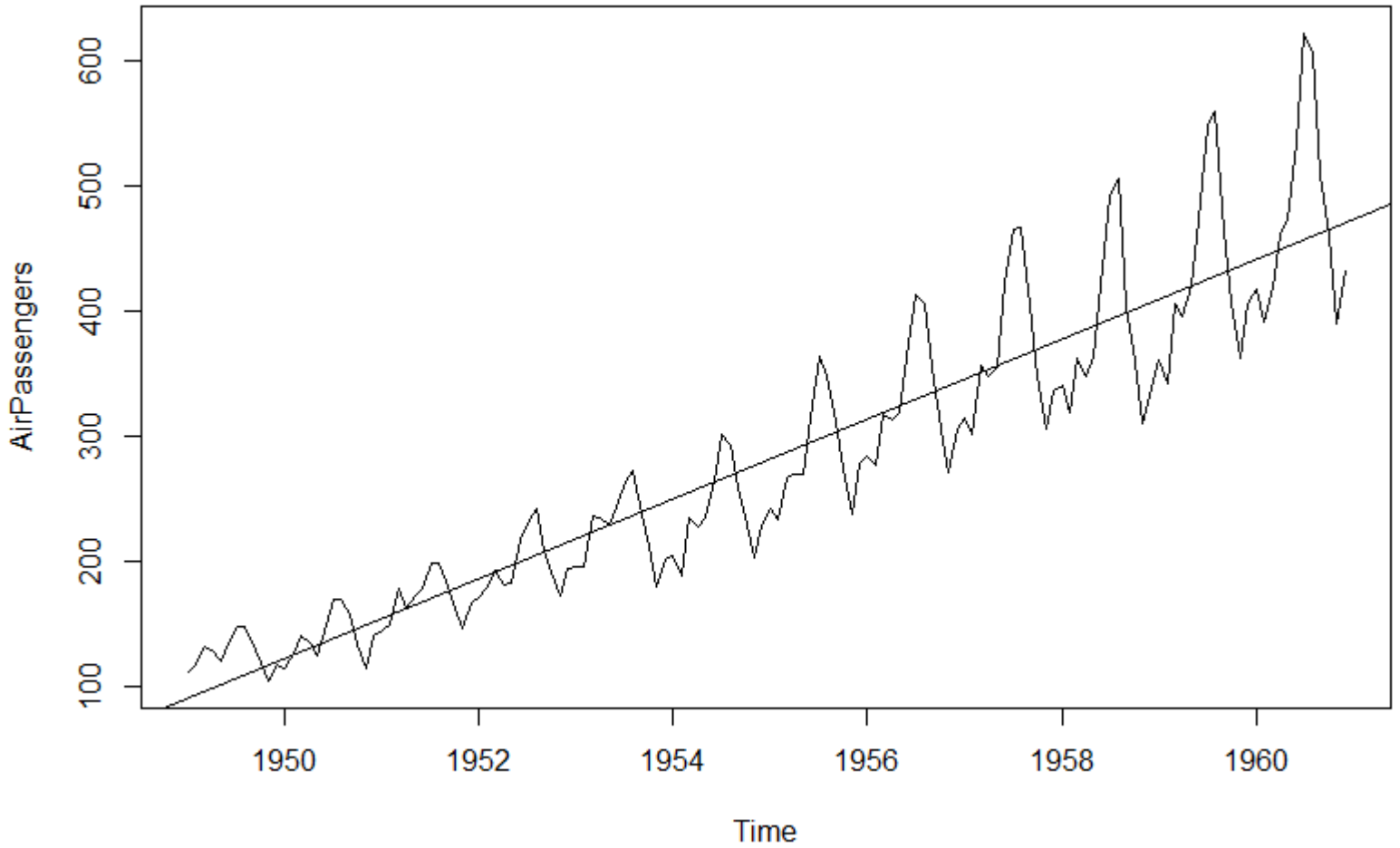
Source

Box, G. E. P., Jenkins, G. M. and Reinsel, G. C. (1976) *Time Series Analysis, Forecasting and Control*. Third Edition. Holden-Day. Series G.

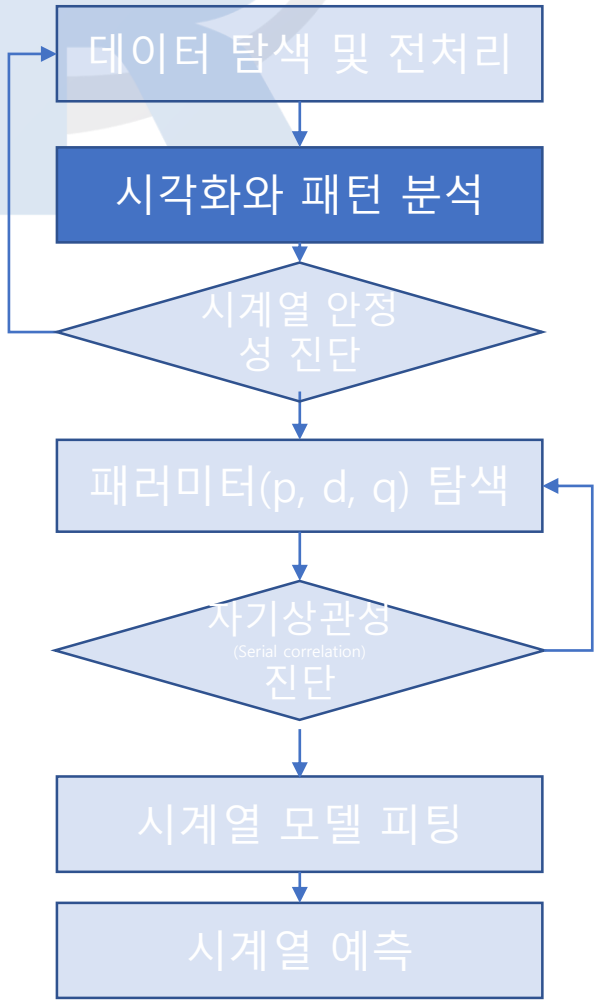
실습1: 월별 국제항공노선 이용자수(1949-1960)



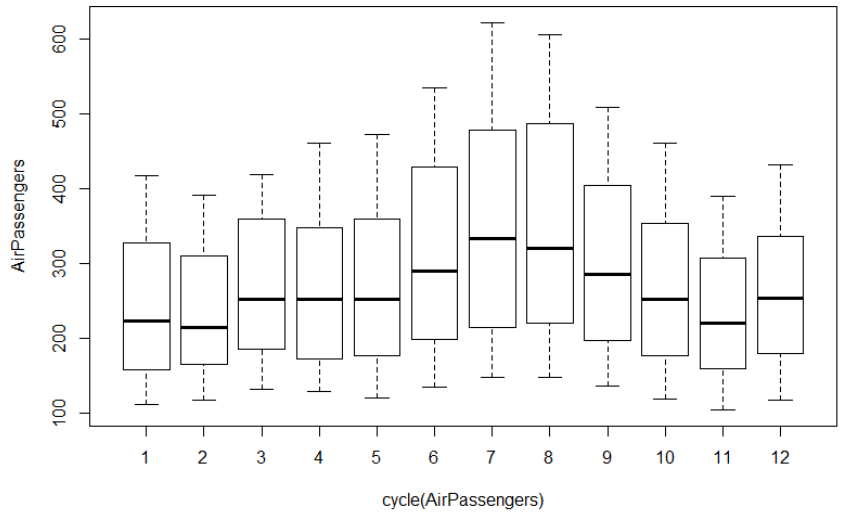

```
> ## 2. 시계열 시각화와 패턴 분석
> plot(AirPassengers) # a trend component which grows the passenger year by year. -> address the trend c
> abline(reg=lm(AirPassengers~time(AirPassengers))) # 회귀직선 추가하기
```



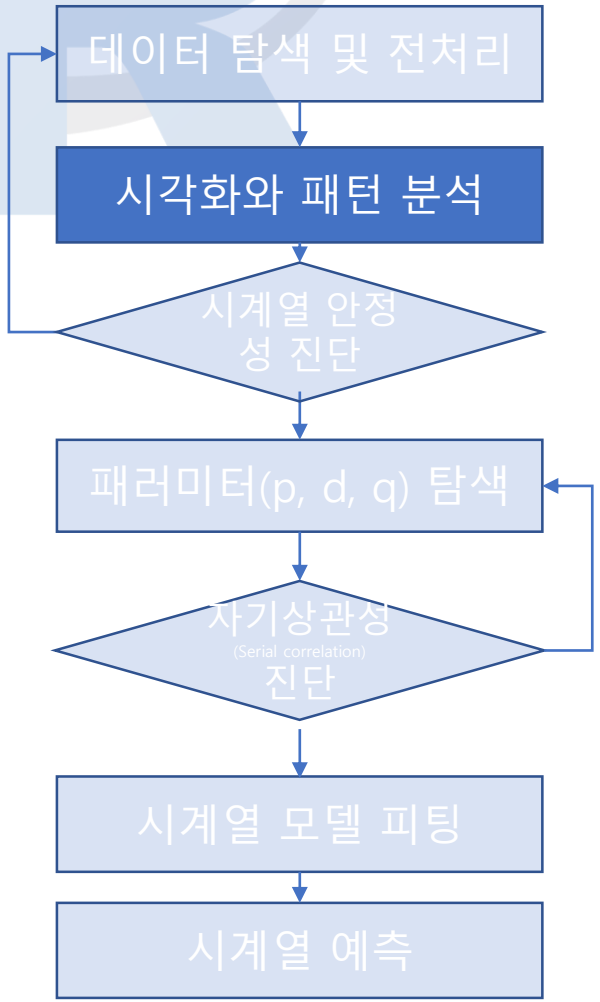
실습1: 월별 국제항공노선 이용자수(1949-1960)



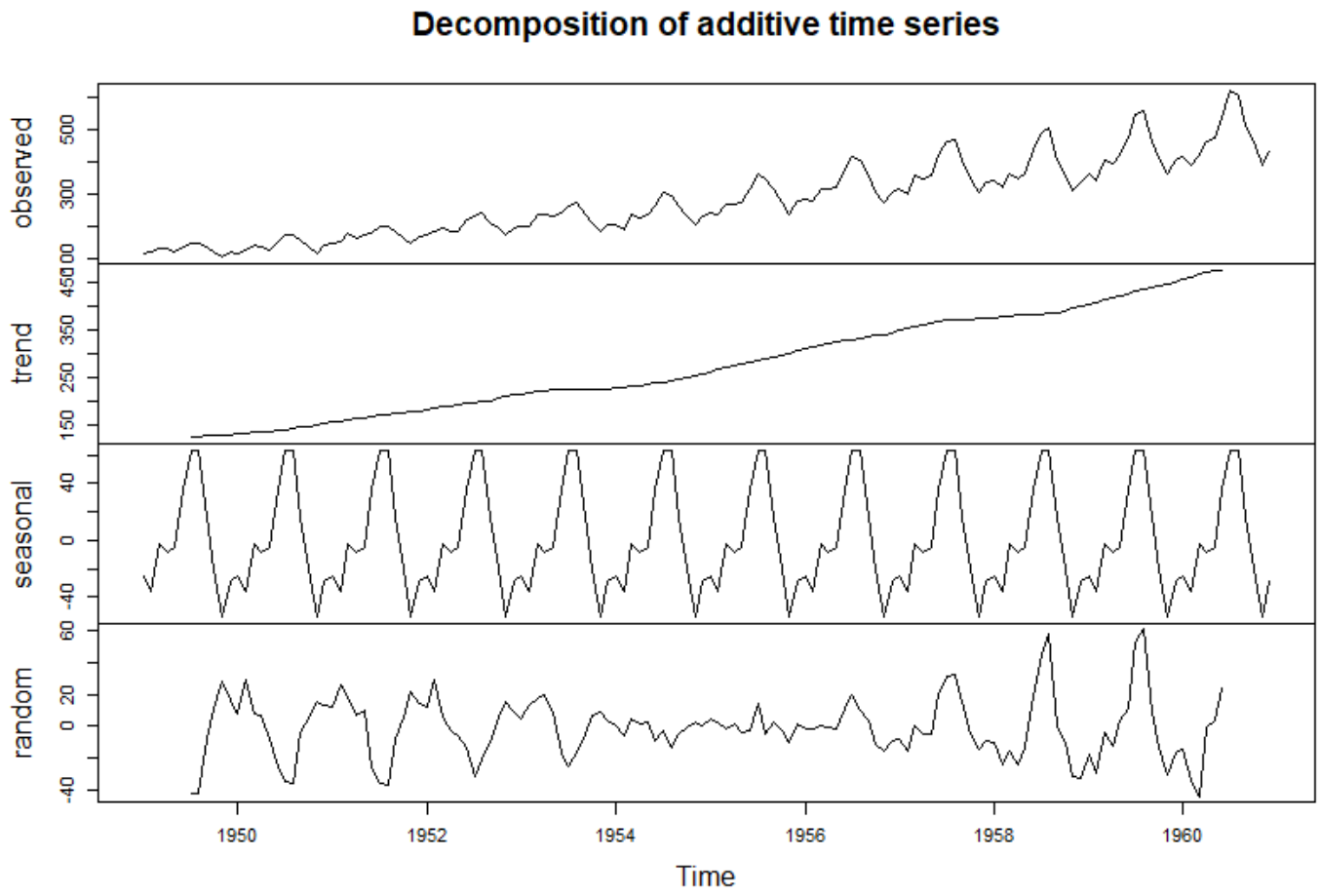

```
> ?aggregate() # Compute Summary Statistics of Data Subsets
>              # a function to compute the summary statistics which can be applied to all data subsets.
> plot(aggregate(AirPassengers, FUN=mean)) # 평균값으로 집계화하여 그래프 작성
> ?cycle() # the positions in the cycle of each observation.
> cycle(AirPassengers) # 주기 확인: 주기에서의 결측치 확인
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949   1   2   3   4   5   6   7   8   9  10  11  12
1950   1   2   3   4   5   6   7   8   9  10  11  12
1951   1   2   3   4   5   6   7   8   9  10  11  12
1952   1   2   3   4   5   6   7   8   9  10  11  12
1953   1   2   3   4   5   6   7   8   9  10  11  12
1954   1   2   3   4   5   6   7   8   9  10  11  12
1955   1   2   3   4   5   6   7   8   9  10  11  12
1956   1   2   3   4   5   6   7   8   9  10  11  12
1957   1   2   3   4   5   6   7   8   9  10  11  12
1958   1   2   3   4   5   6   7   8   9  10  11  12
1959   1   2   3   4   5   6   7   8   9  10  11  12
1960   1   2   3   4   5   6   7   8   9  10  11  12
> boxplot(AirPassengers~cycle(AirPassengers)) # 주기(월)별 박스그래프
```



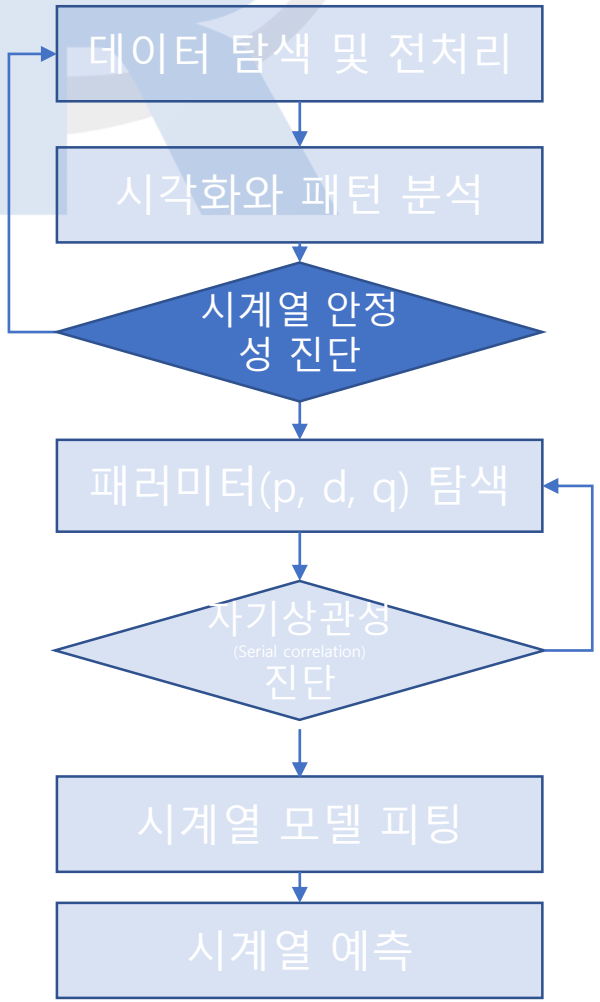
실습1: 월별 국제항공노선 이용자수(1949-1960)




```
> plot(decompose(AirPassengers)) # 시계열 분해 그래프 생성
```

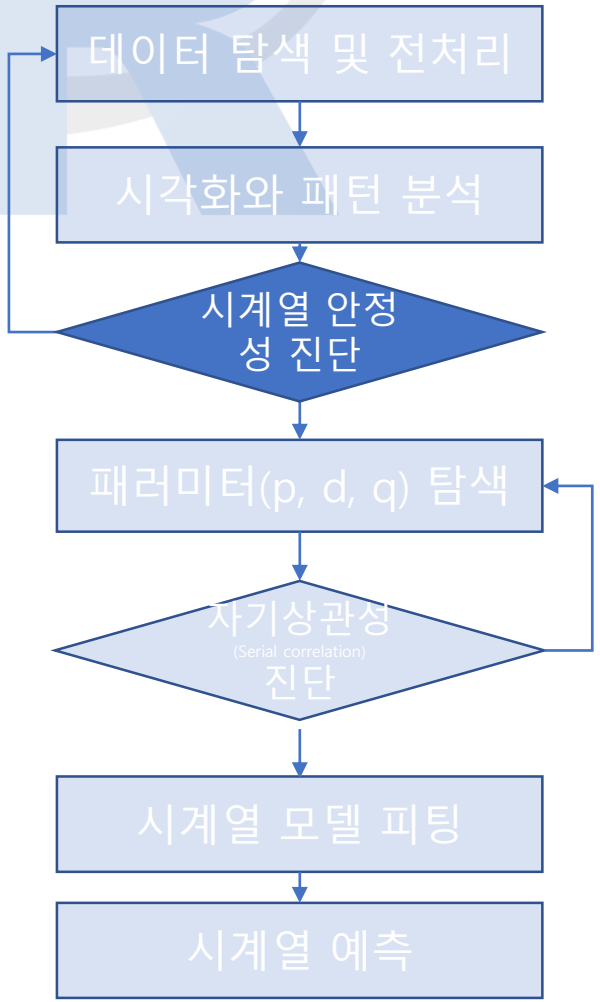


실습1: 월별 국제항공노선 이용자수(1949-1960)

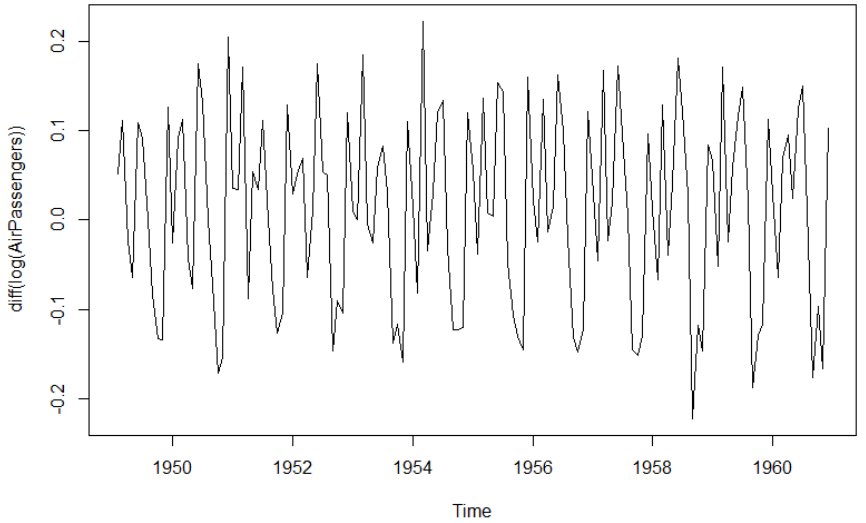



```
> ## 3. 시계열 안정성 진단 및 검증
> library(tseries) # install.packages("tseries")
> ?adf.test() # Augmented Dickey-Fuller Test for the null hypothesis of a unit root of a univarate tim
> adf.test(AirPassengers,
+          alternative="stationary", #  stationary enough to do any kind of time series modelling.
+          k=0) # the lag order to calculate the test statistic

        Augmented Dickey-Fuller Test

data:  AirPassengers
Dickey-Fuller = -4.6392, Lag order = 0, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(AirPassengers, alternative = "stationary", k = 0) :
  p-value smaller than printed p-value
> ?Box.test() # Box-Pierce and Ljung-Box Tests
> Box.test(AirPassengers,
+          type = "Ljung-Box")

        Box-Ljung test

data:  AirPassengers
X-squared = 132.14, df = 1, p-value < 2.2e-16
```

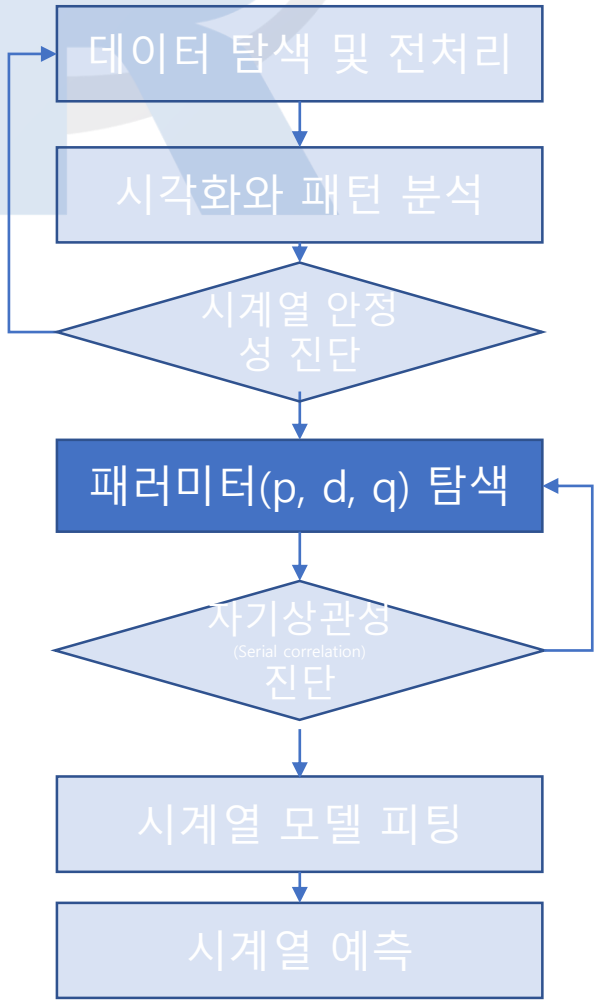
실습1: 월별 국제항공노선 이용자수(1949-1960)



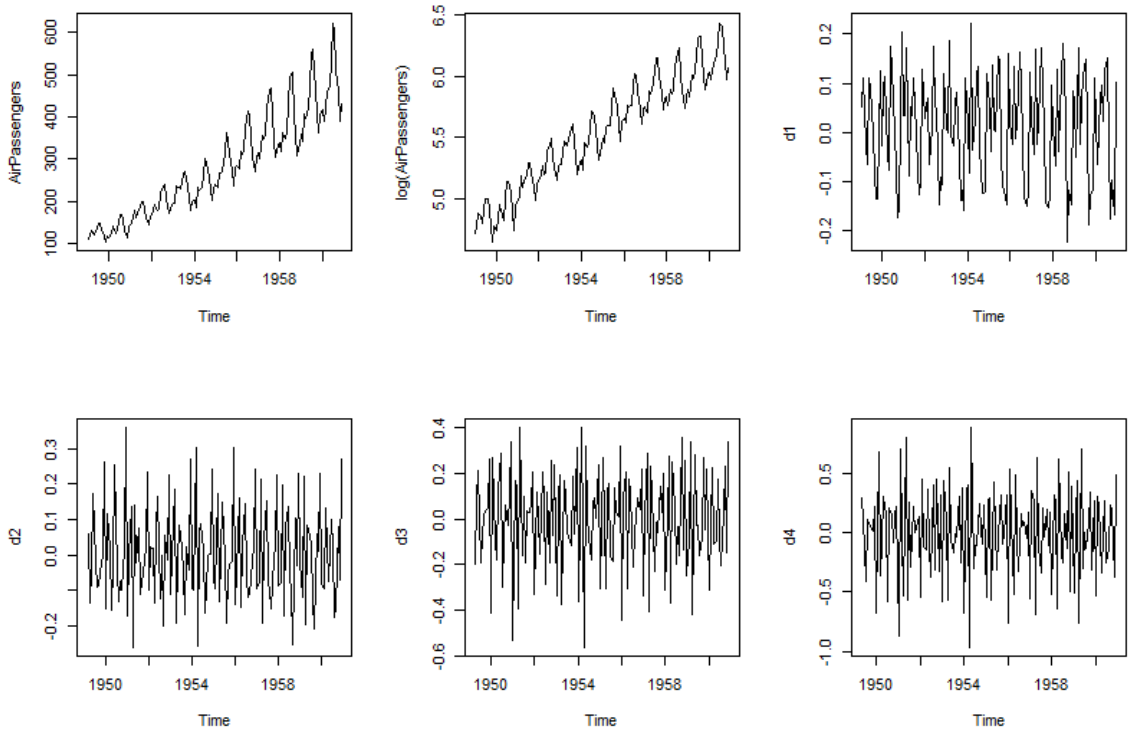

```
> ?diff() # Lagged Differences
> plot(diff(log(AirPassengers))) # 로그 차분
> adf.test(diff(log(AirPassengers)),
+          alternative="stationary", #  stationary enough to do any kind of time series modelling.
+          k=0)

        Augmented Dickey-Fuller Test

data:  diff(log(AirPassengers))
Dickey-Fuller = -9.6003, Lag order = 0, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(diff(log(AirPassengers)), alternative = "stationary",  :
  p-value smaller than printed p-value
> Box.test(diff(log(AirPassengers)),
+          type = "Ljung-Box")

        Box-Ljung test

data:  diff(log(AirPassengers))
X-squared = 5.8263, df = 1, p-value = 0.01579
```



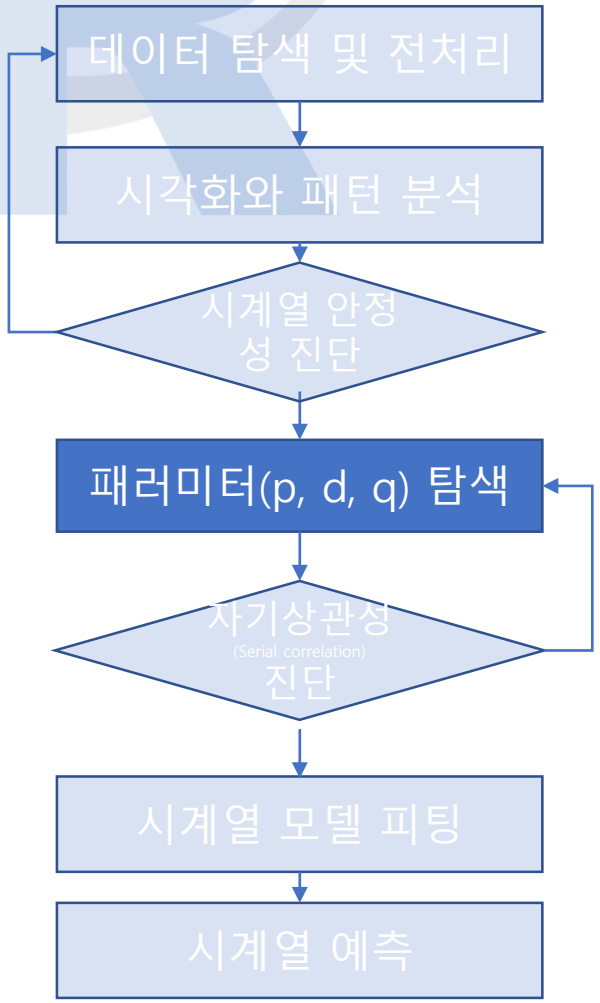
실습1: 월별 국제항공노선 이용자수(1949-1960)



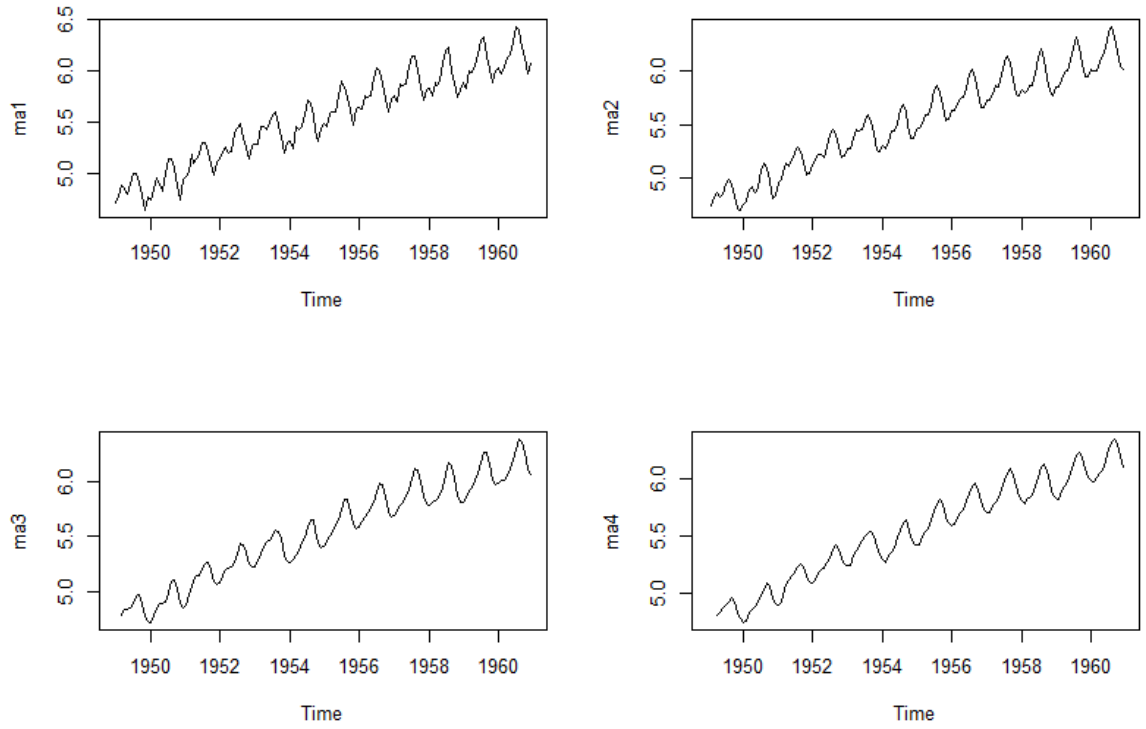

```
> ## 4. ARIMA모델 패러미터 탐색
> #### 차분(d=?) 정상화 시도
> ?diff() # Lagged Differences, suitably lagged and iterated differences.
> d1 <- diff(log(AirPassengers), differences = 1)
> d2 <- diff(log(AirPassengers), differences = 2)
> d3 <- diff(log(AirPassengers), differences = 3)
> d4 <- diff(log(AirPassengers), differences = 4)
> par(mfrow = c(2,3))
> plot(AirPassengers)
> plot(log(AirPassengers))
> plot(d1)
> plot(d2)
> plot(d3)
> plot(d4)
> par(mfrow = c(1,1))
```



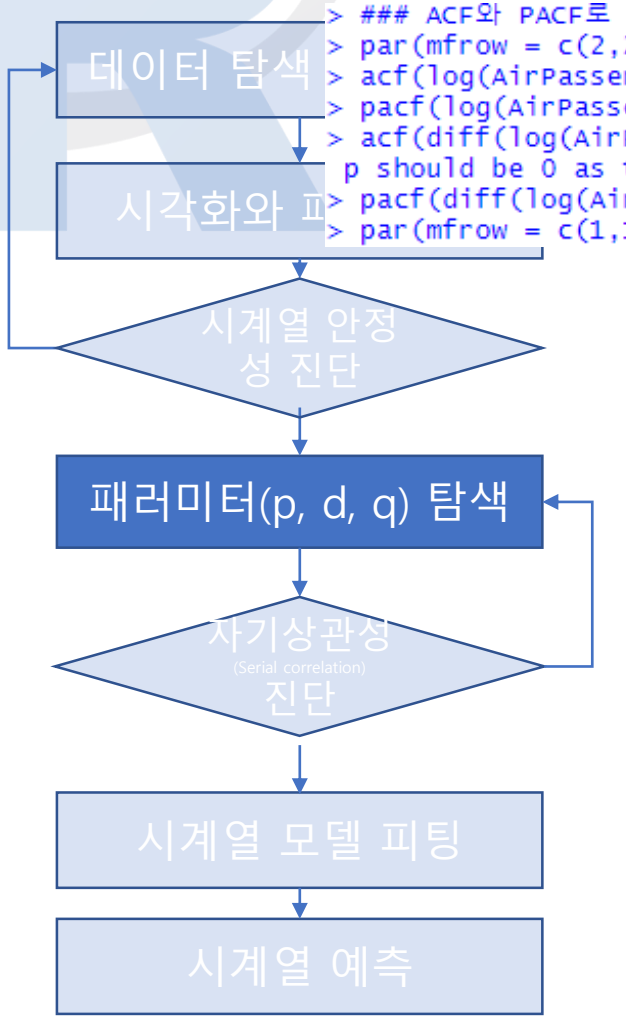
실습1: 월별 국제항공노선 이용자수(1949-1960)



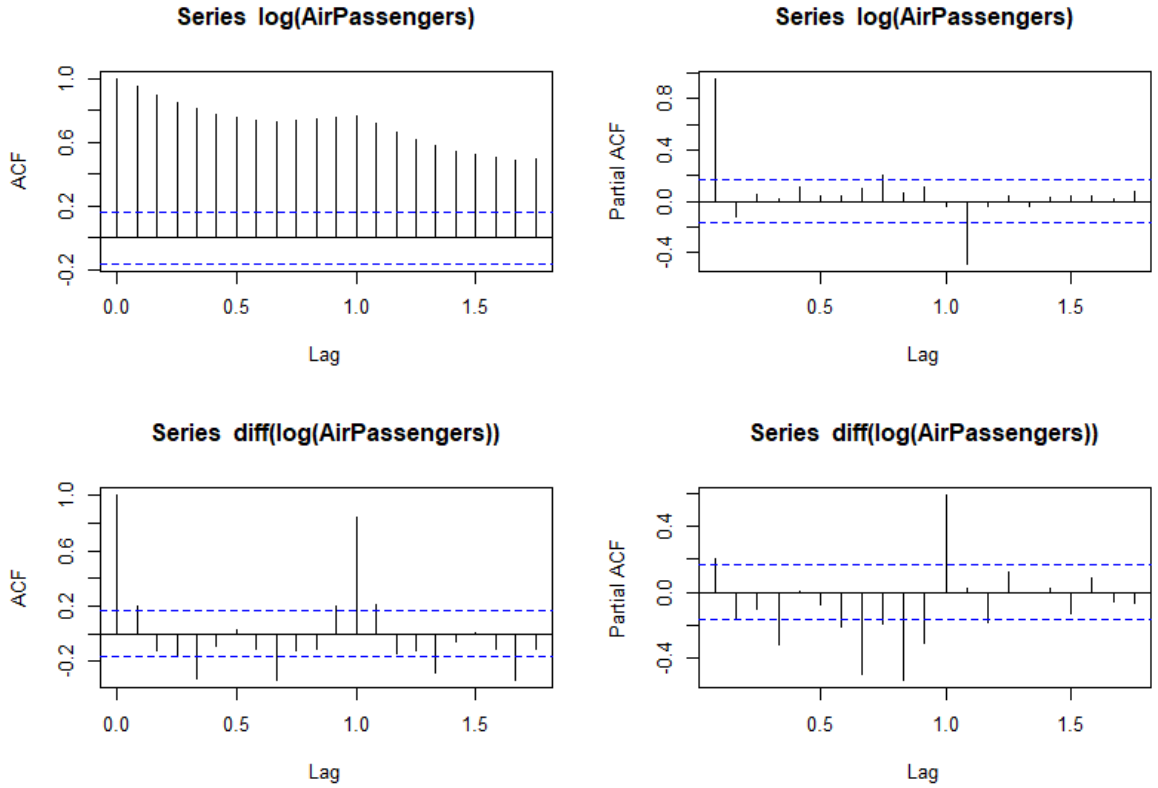

```
> ### p, q 탬색: 직접 확인하기(예: q)
> #### 이동평균 평활화
> library(TTR) # 미설치시 install.packages("TTR")
> ?SMA() # Calculate various moving averages (MA) of a series.
> ma1 <- SMA(log(AirPassengers), n = 1) # q = ?
> ma2 <- SMA(log(AirPassengers), n = 2)
> ma3 <- SMA(log(AirPassengers), n = 3)
> ma4 <- SMA(log(AirPassengers), n = 4)
> par(mfrow = c(2,2))
> plot(ma1)
> plot(ma2)
> plot(ma3)
> plot(ma4)
> par(mfrow = c(1,1))
```



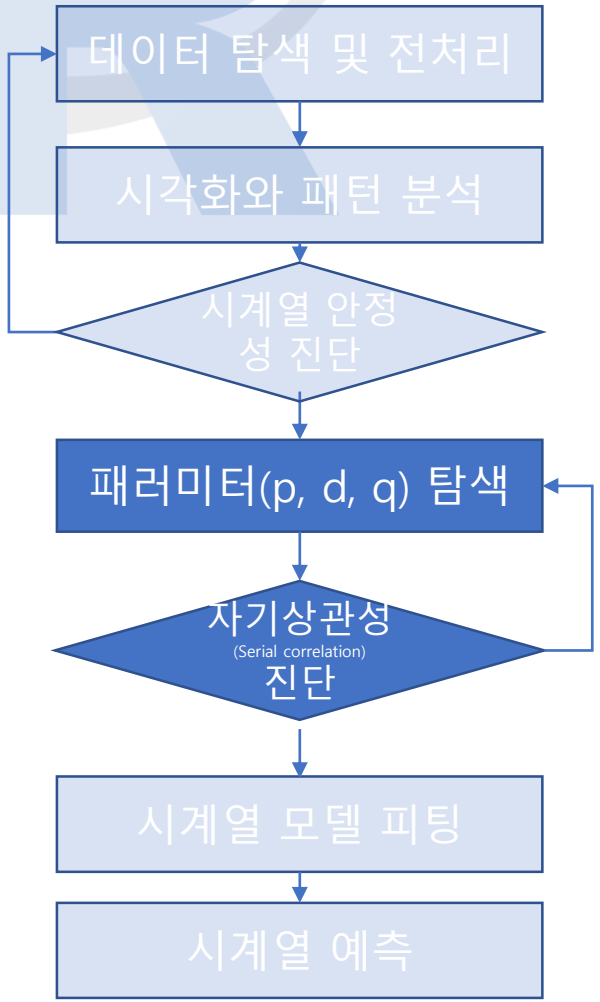
실습1: 월별 국제항공노선 이용자수(1949-1960)




```
> ### ACF와 PACF로 확인
> par(mfrow = c(2,2))
> acf(log(AirPassengers)) # the decay of ACF chart is very slow, which means that the population is not stationary.
> pacf(log(AirPassengers))
> acf(diff(log(AirPassengers))) #  # lag 2부터 점선 안에 존재. lag 절단값 = 2. --> MA(1): after regressing on the differe
  p should be 0 as the ACF is the curve getting a cut off
> pacf(diff(log(AirPassengers))) # value of q should be 1 or 2.
> par(mfrow = c(1,1))
```

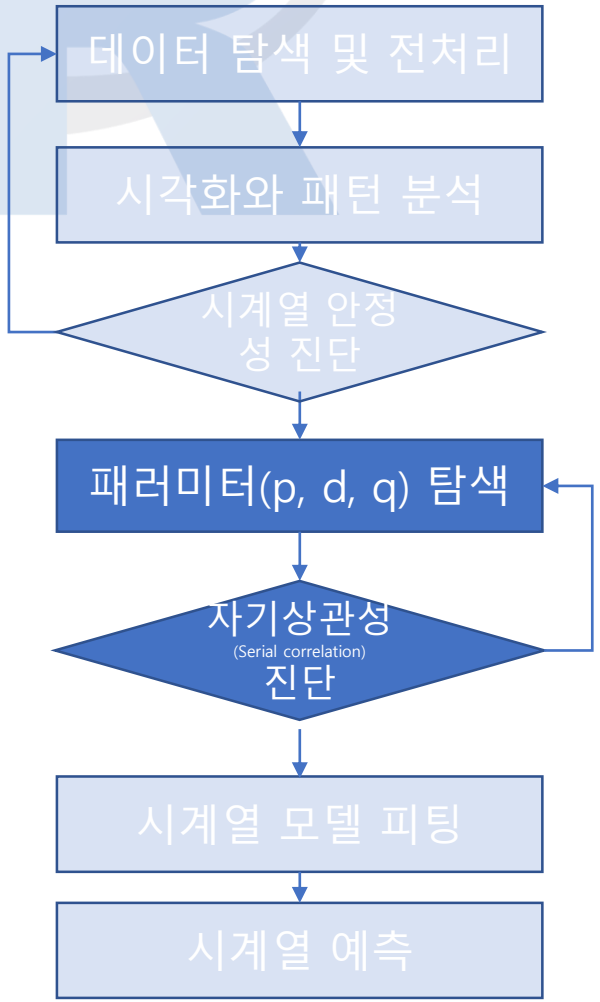


실습1: 월별 국제항공노선 이용자수(1949-1960)



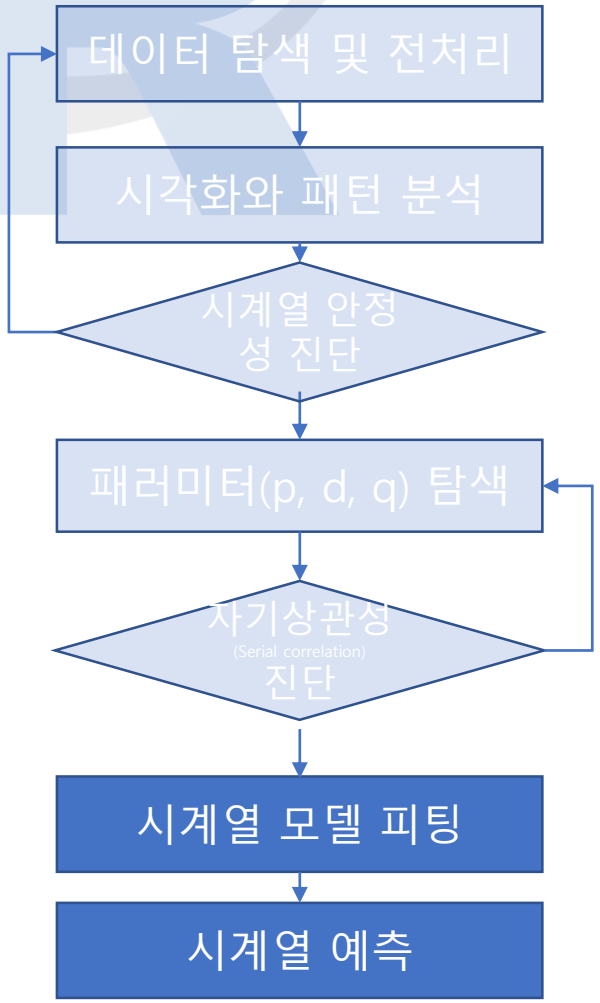

```
> ## 5. ARIMA 모델링
> ?arima() # ARIMA Modelling of Time Series
> (fit <- arima(log(AirPassengers),
+               c(0, 1, 1)))

Call:
arima(x = log(AirPassengers), order = c(0, 1, 1))

Coefficients:
          ma1
      0.2768
s.e.  0.0944

sigma^2 estimated as 0.01072:  log likelihood = 121.36,  aic = -238.73
> Box.test(fit$residuals, type="Ljung-Box")

        Box-Ljung test

data:  fit$residuals
X-squared = 0.15568, df = 1, p-value = 0.6932
```

실습1: 월별 국제항공노선 이용자수(1949-1960)

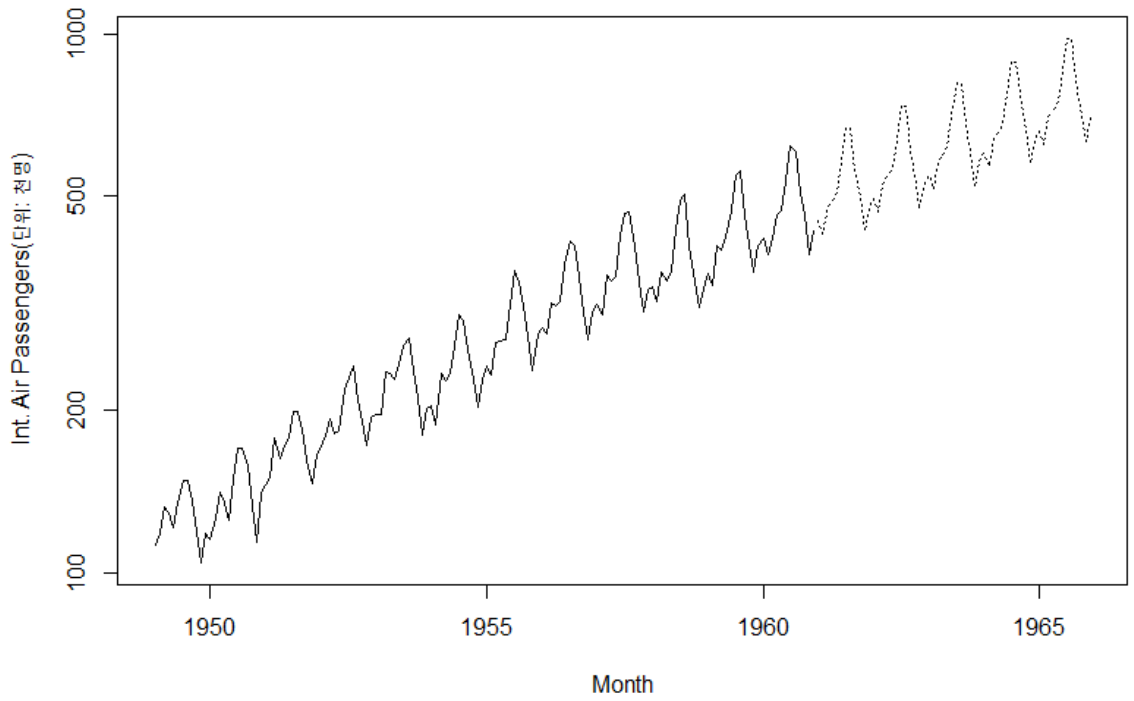



```
> library(forecast) #install.packages("forecast")
> ?auto.arima() # Fit best ARIMA model to univariate time series
> fit2 <- auto.arima(log(AirPassengers)) # 자동으로 ARIMA 모형 p,d,q 확인
> summary(fit2)
Series: log(AirPassengers)
ARIMA(0,1,1)(0,1,1)[12]

Coefficients:
          ma1     sma1
      -0.4018  -0.5569
s.e.   0.0896   0.0731

sigma^2 estimated as 0.001371:  log likelihood=244.7
AIC=-483.4   AICc=-483.21   BIC=-474.77

Training set error measures:
                      ME       RMSE        MAE         MPE      MAPE      MASE        ACF1
Training set 0.0005730622 0.03504883 0.02626034 0.01098898 0.4752815 0.2169522 0.01443892
> Box.test(fit2$residuals, type="Ljung-Box")

        Box-Ljung test

data:  fit2$residuals
X-squared = 0.030651, df = 1, p-value = 0.861
```

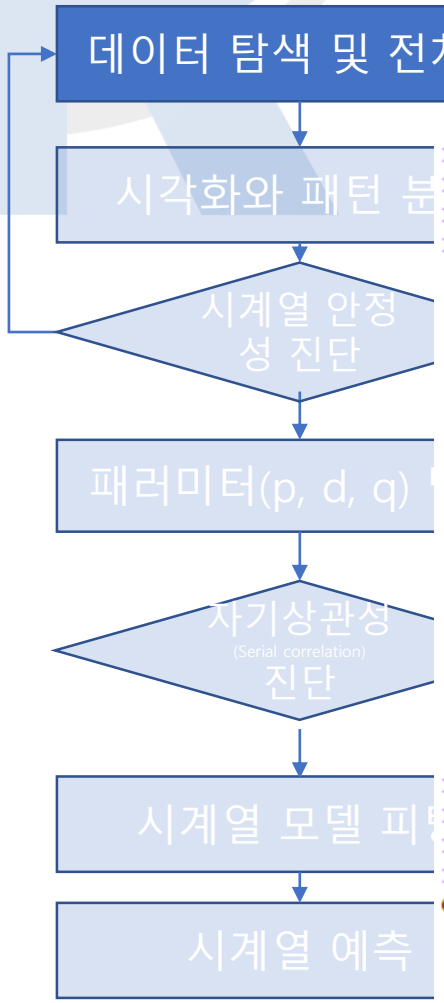
실습1: 월별 국제항공노선 이용자수(1949-1960)




```
> ## 6. 시계열 예측
> ?predict()
> pred <- predict(fit2, n.ahead = 5*12) # 5년 후 까지 예측
> ?ts.plot() # Plot Multiple Time Series
> ts.plot(AirPassengers,
+         exp(pred$pred), # 자연로그 e (= 2.718...)
+         log = "y",
+         lty = c(1,3), # 선형태(1=실측치, 2=예측치)
+         gpars=list(xlab="Month", ylab="Int. Air Passengers(단위: 천명)"))
```

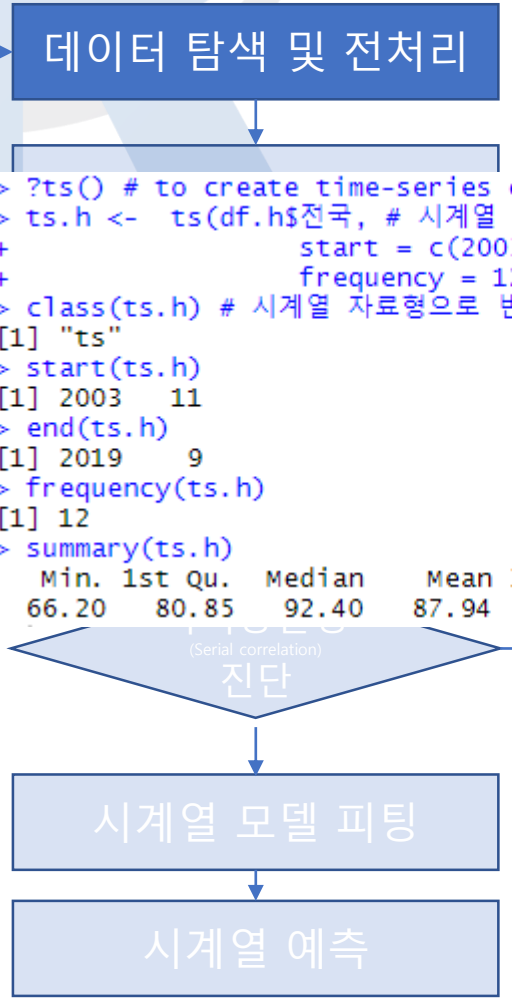


실습2: 한국 월별 주택매매가격 지수




```
> # 실습 2:한국 월별 주택매매가격 지수
> ### 1. 데이터 불러오기와 탐색하기
> setwd("K:\\기타\\2019년2학기\\수치해석\\실습데이터")
> list.files() # 파일 확인
 [1] "(2018년 기준) 서울서베이 조사표_가구용.pdf"      "apt2.csv"
 [3] "apt3.csv"                                        "chicago_crime_data_2019.csv"
 [5] "data_by_sigungu (1).csv"                         "data_by_sigungu_2018.xlsx"
 [7] "df.seoul.csv"                                    "df.seoul.riders.csv"
 [9] "df.seoul.worker.csv"                             "df.서울서베이.가구원.csv"
[11] "df.서울서베이.가구원.xlsx"                        "gdp_growth_rate.xlsx"
[13] "GIS_Data"                                        "ridership_weather_gasprice.xlsx"
[15] "년도별_항공운송통계.xlsx"                         "대기오염"
[17] "데이터_아파트(매매)_실거래가_충남대전세종.xlsx"  "데이터_아파트매매가격.csv"
[19] "데이터_아파트매매가격.xlsx"                       "서울_기상대중교통_데이터분석.R"
[21] "서울서베이_가구원.xlsx"                           "서울서베이_가구원_데이터_가공_v1_2019-09-26.R"
[23] "서울서베이_가구원_데이터_가공_v2_2019-09-26.R"   "서울서베이_가구주.xlsx"
[25] "서울시대중교통"                                   "서울시도로교통량"
[27] "실습자료_서울서베이_가구원.xlsx"                  "실습자료_서울서베이_가구원_v2.xlsx"
[29] "월별_서울시_주민등록인구.xlsx"                    "월별_주택매매가격지수_시도.xlsx"
[31] "인구10만명당_교통사고사망자수_OECD.xlsx"          "주유소_서울시_평균판매가격.xlsx"
[33] "청주_아파트_거래자료.csv"
> library(readxl)
> # 주택매매가격지수: 주택매매가격을 기준시점(2017.11=100.0)과 조사시점의 가격비를 이용하여 기준시점이 100인 수치로 환산한 값
> df.h <- read_xlsx("월별_주택매매가격지수_시도.xlsx", 1) # 엑셀자료 불러오기
> str(df.h) # 데이터 구조 확인하기
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':       191 obs. of  19 variables:
 $ id_ym         : num  2e+05 2e+05 2e+05 2e+05 2e+05 ...
 $ 전국          : num  68.6 68.1 67.7 67.8 67.9 68 68 67.8 67.6 67.4 ...
 $ 서울특별시    : num  63.5 63 62.9 63.1 63.4 63.6 63.6 63.5 63.3 62.9 ...
```

실습2: 한국 월별 주택매매가격 지수




```
> ?ts() # to create time-series objects
> ts.h <-  ts(df.h$전국, # 시계열 자료로 변환: 전국 데이터 열 할당
+                   start = c(2003, 11), # 시작 주기
+                   frequency = 12) # 빈도
> class(ts.h) # 시계열 자료형으로 변환되었는 지 확인
[1] "ts"
> start(ts.h)
[1] 2003   11
> end(ts.h)
[1] 2019    9
> frequency(ts.h)
[1] 12
> summary(ts.h)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  66.20   80.85   92.40   87.94   97.35  101.20
```

Description

The function `ts` is used to create time-series objects.

`as.ts` and `is.ts` coerce an object to a time-series and test whether an object is a time series.

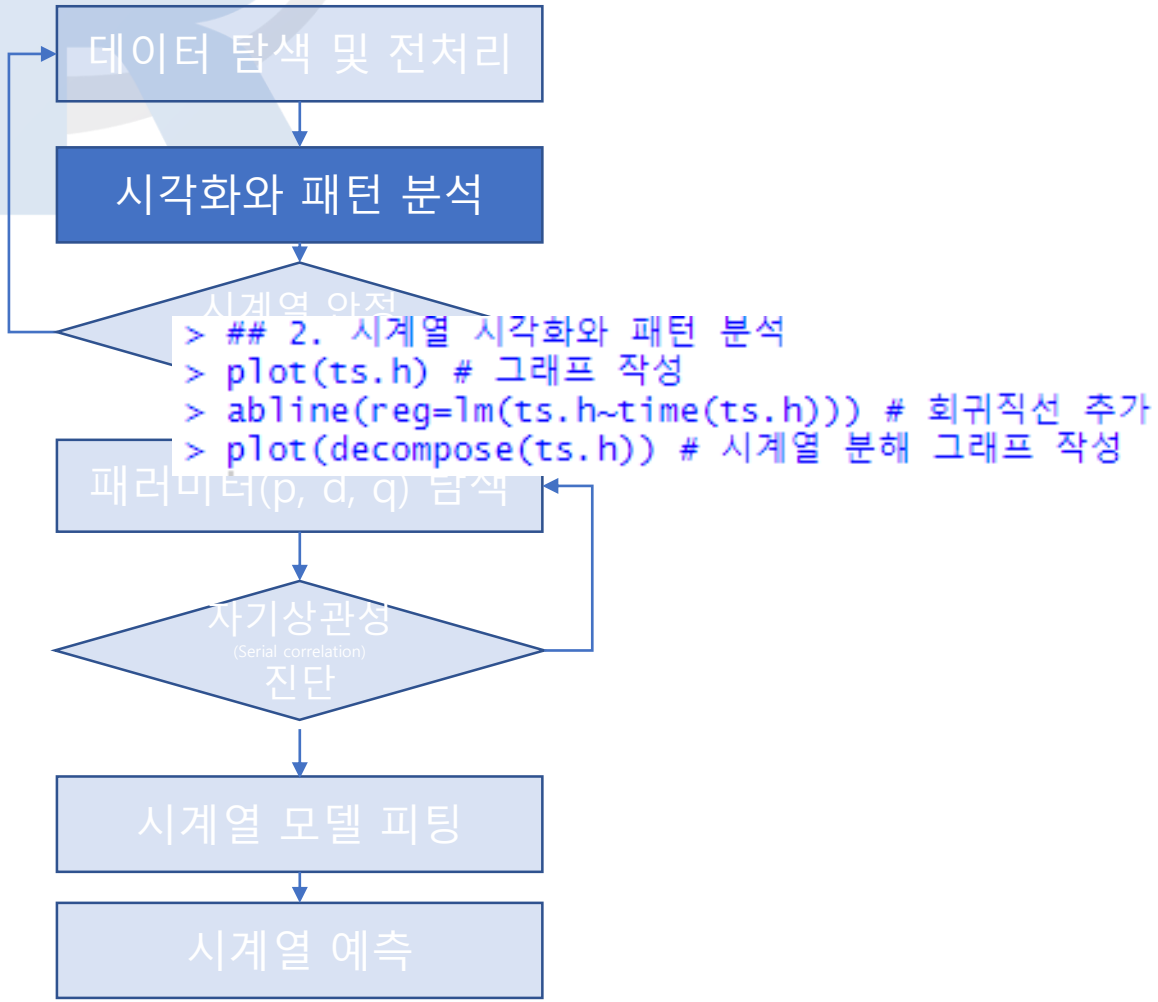
Usage

```
ts(data = NA, start = 1, end = numeric(), frequency = 1,
   deltat = 1, ts.eps = getOption("ts.eps"), class = , names = )
as.ts(x, ...)
is.ts(x)
```

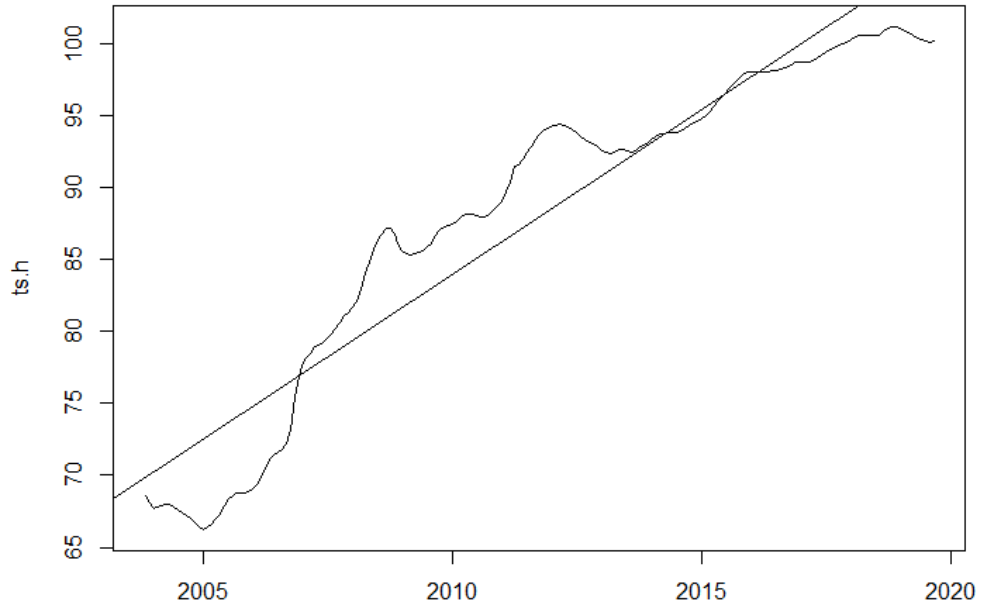
Arguments

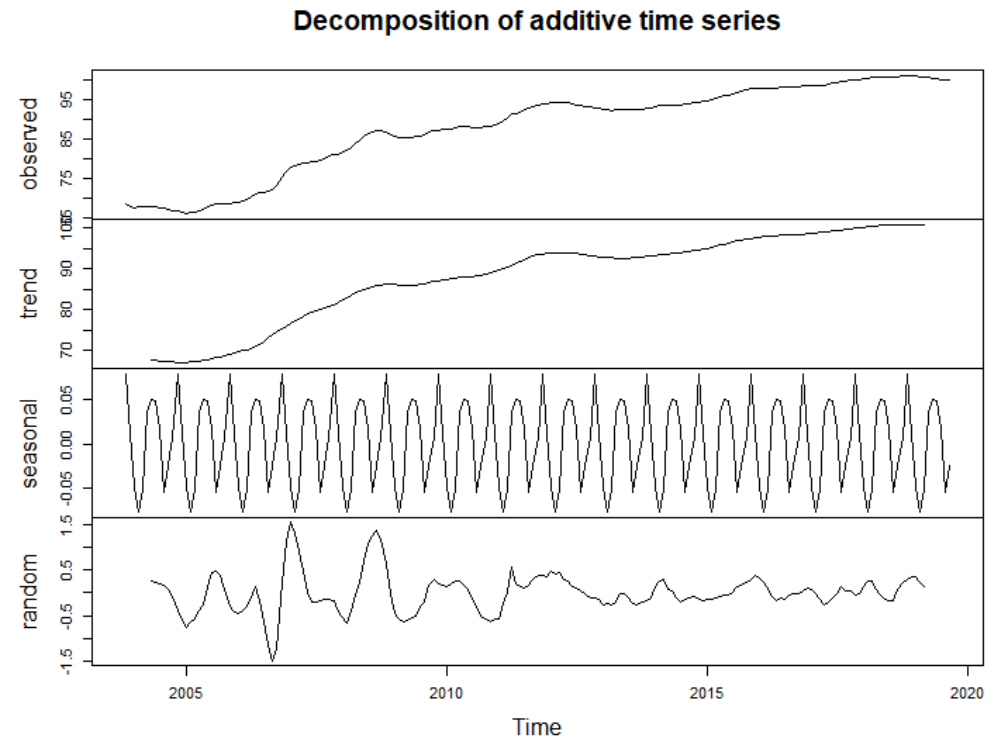
`data`	a vector or matrix of the observed time-series values. A data frame will be coerced to a numeric matrix via `data.matrix`. (See also 'Details'.)
`start`	the time of the first observation. Either a single number or a vector of two integers, which specify a natural time unit and a (1-based) number of samples into the time unit. See the examples for the use of the second form.
`end`	the time of the last observation, specified in the same way as `start`.
`frequency`	the number of observations per unit of time.
`deltat`	the fraction of the sampling period between successive observations; e.g., 1/12 for monthly data. Only one of `frequency` or `deltat` should be provided.

실습2: 한국 월별 주택매매가격 지

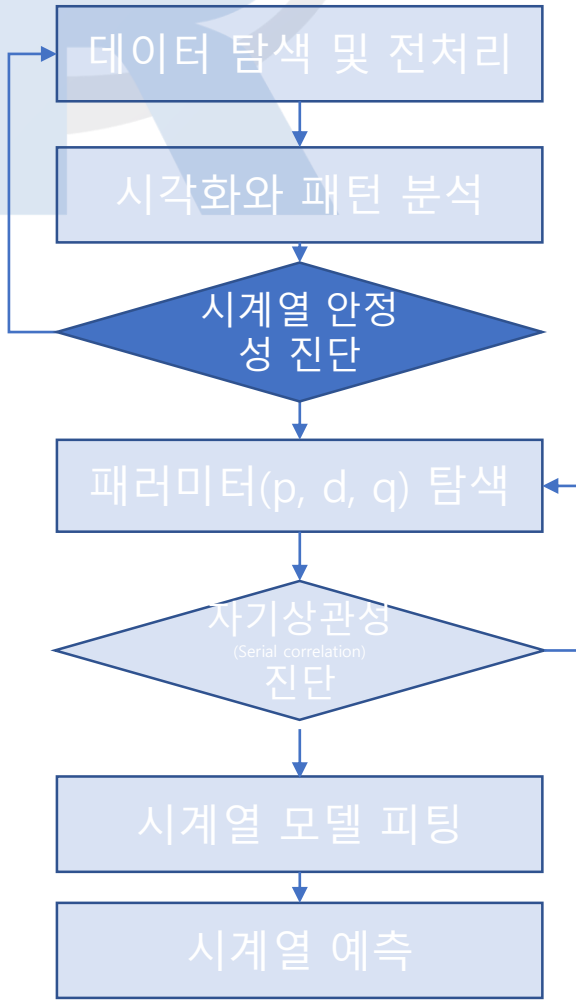



```
> ## 2. 시계열 시각화와 패턴 분석
> plot(ts.h) # 그래프 작성
> abline(reg=lm(ts.h~time(ts.h))) # 회귀직선 추가
> plot(decompose(ts.h)) # 시계열 분해 그래프 작성
```

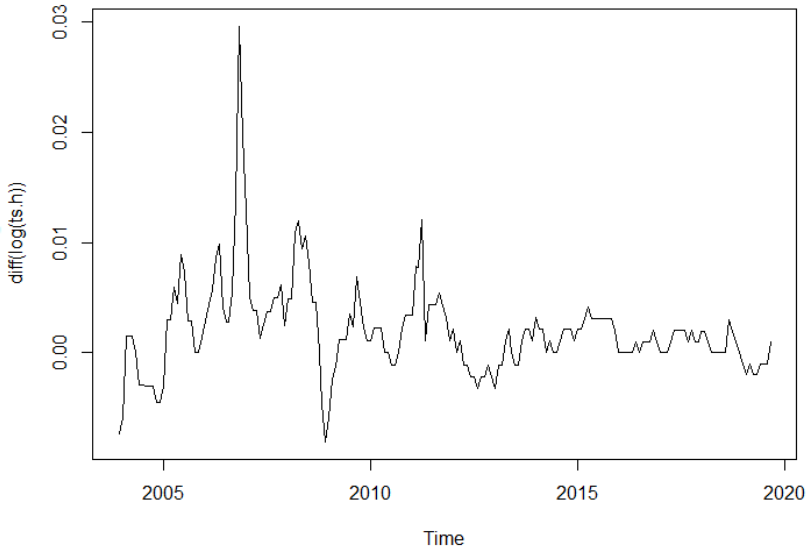




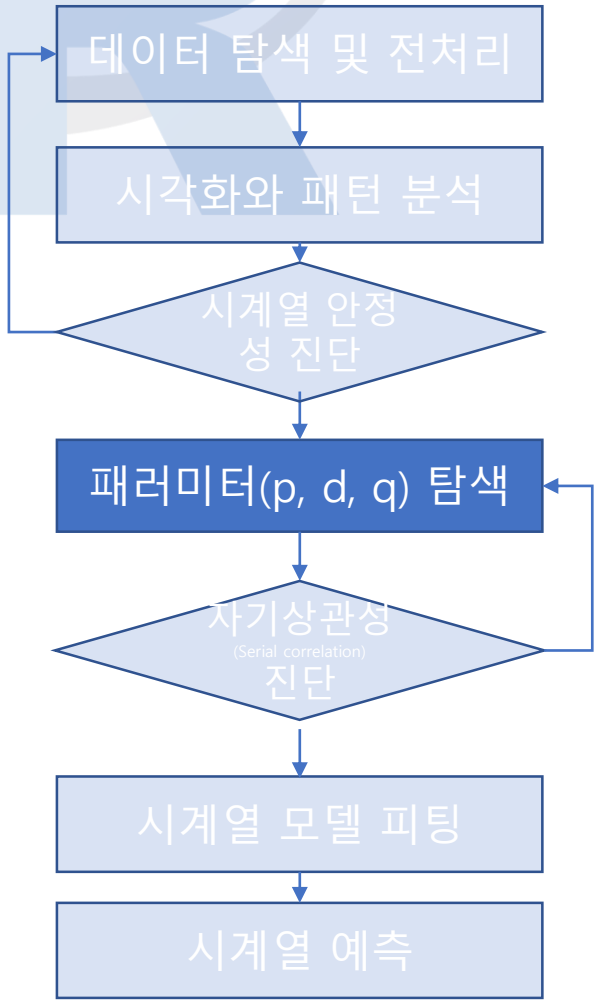
실습2: 한국 월별 주택매매가격 지수



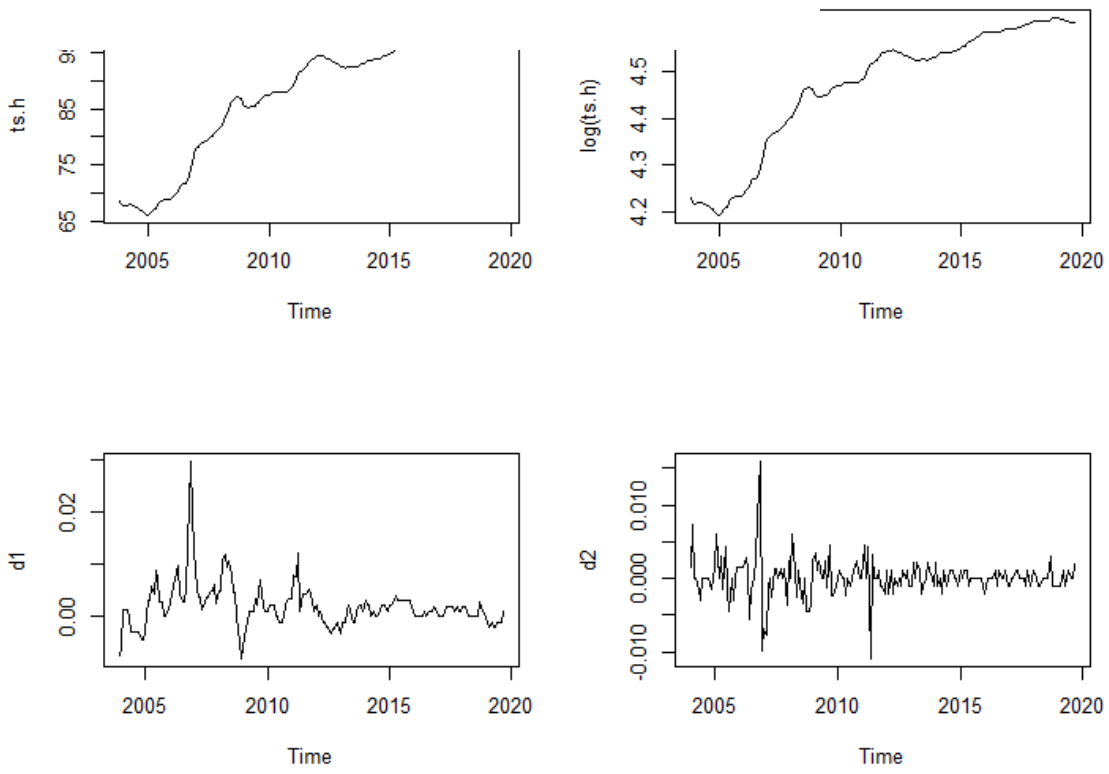

```
> ## 3. 시계열 안정성 진단 및 섬승
> adf.test(ts.h, # 단위근 검정
+          alternative="stationary", #  stationary enough to do any kind of time series modelling.
+          k=0)

        Augmented Dickey-Fuller Test

data:  ts.h
Dickey-Fuller = 0.42797, Lag order = 0, p-value = 0.99
alternative hypothesis: stationary

Warning message:
In adf.test(ts.h, alternative = "stationary", k = 0) :
  p-value greater than printed p-value
> plot(diff(log(ts.h))) # 로그 차분 그래프 작성
> adf.test(diff(log(ts.h)), # 로그 차분 단위근 검정
+          alternative="stationary", #  stationary enough to do any kind of time series modelling.
+          k=0)

        Augmented Dickey-Fuller Test

data:  diff(log(ts.h))
Dickey-Fuller = -5.0163, Lag order = 0, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(diff(log(ts.h)), alternative = "stationary"
  p-value smaller than printed p-value
```



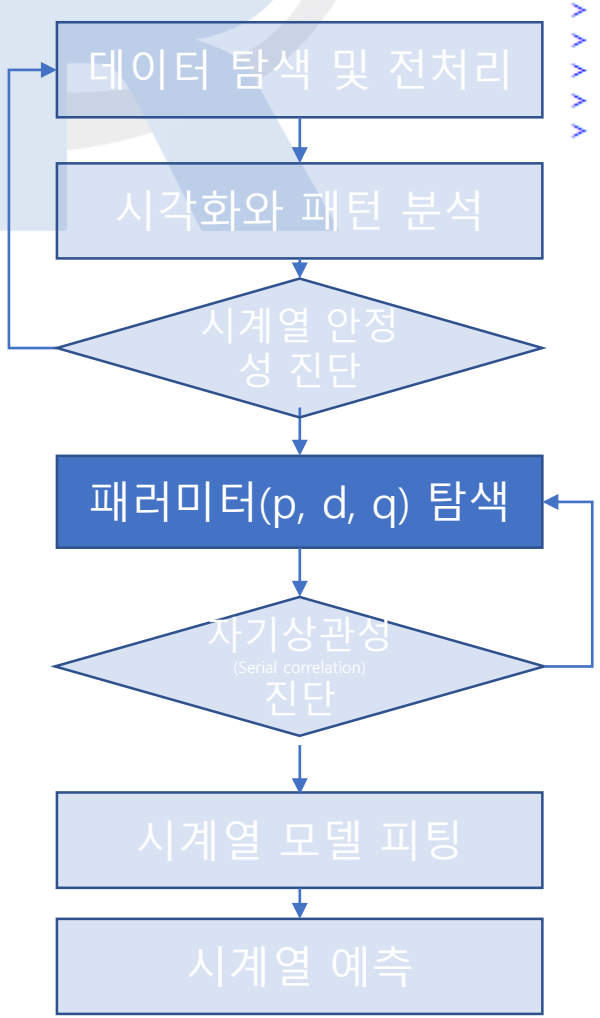
실습2: 한국 월별 주택매매가격 지수



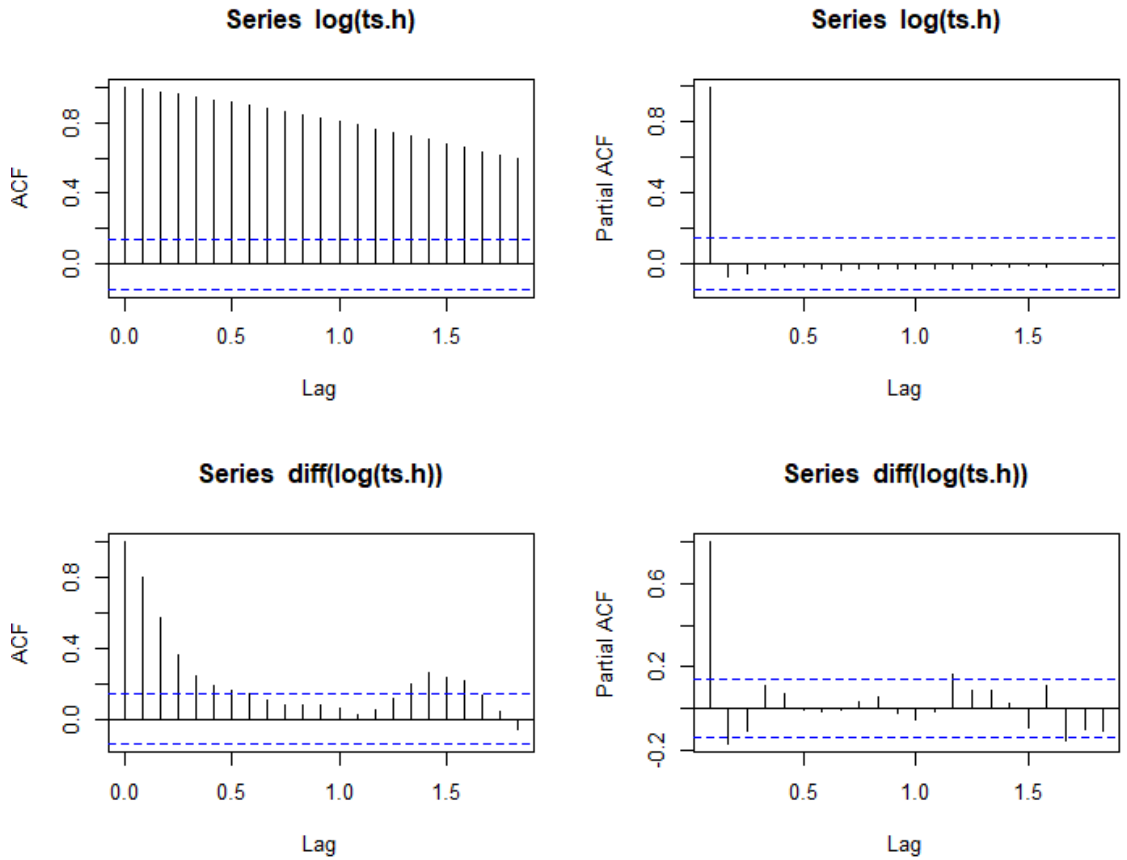

```
> ## 4. ARIMA모델 패러미터 탐색
> # 차분 정상화
> ?diff() # Lagged Differences, suitably lagged and iterated differences.
> d1 <- diff(log(ts.h), differences = 1)
> d2 <- diff(log(ts.h), differences = 2)
> par(mfrow = c(2,2))
> plot(ts.h)
> plot(log(ts.h))
> plot(d1)
> plot(d2)
> par(mfrow = c(1,1))
```



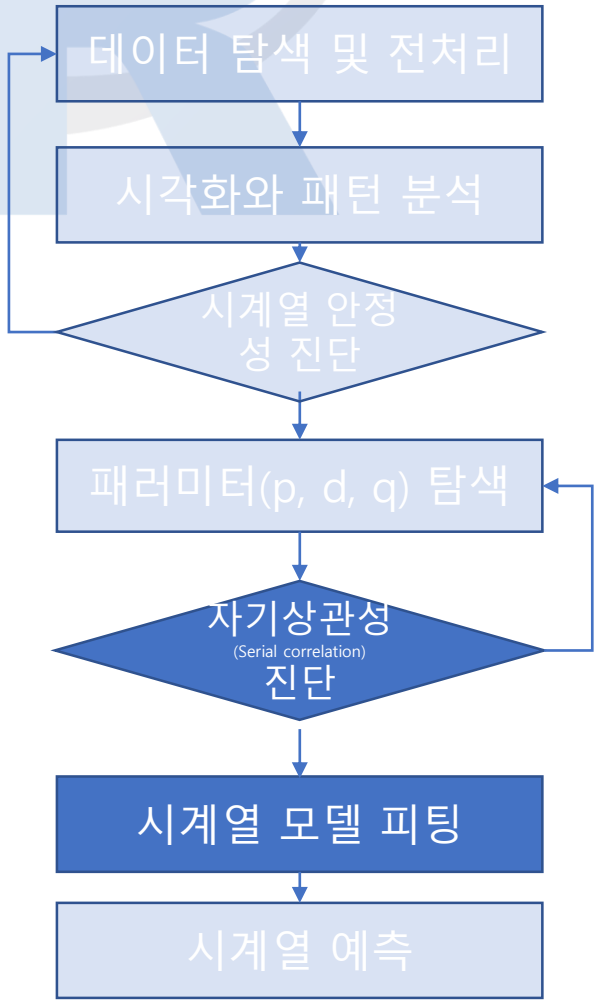
실습2: 한국 월별 주택매매가격 지수




```
> par(mfrow = c(2,2))
> acf(log(ts.h)) # the decay of ACF chart is very slow, which means that the population is not stationary.
> pacf(log(ts.h))
> acf(diff(log(ts.h))) #  # lag 7부터 점선 안에 존재. lag 절단값 = 6 ?
> pacf(diff(log(ts.h))) # lag 2 -> 1
```

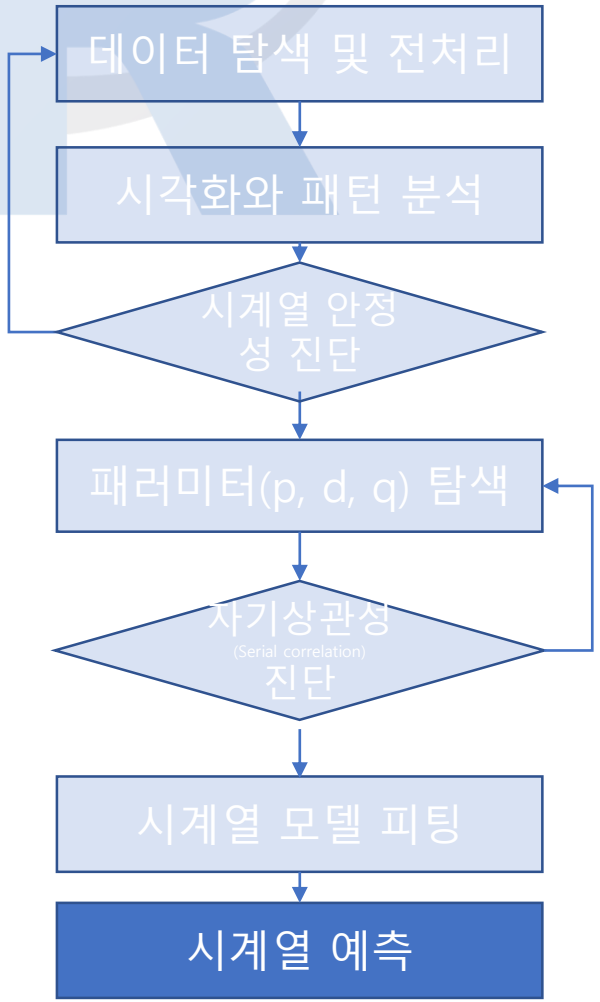


실습2: 한국 월별 주택매매가격 지수

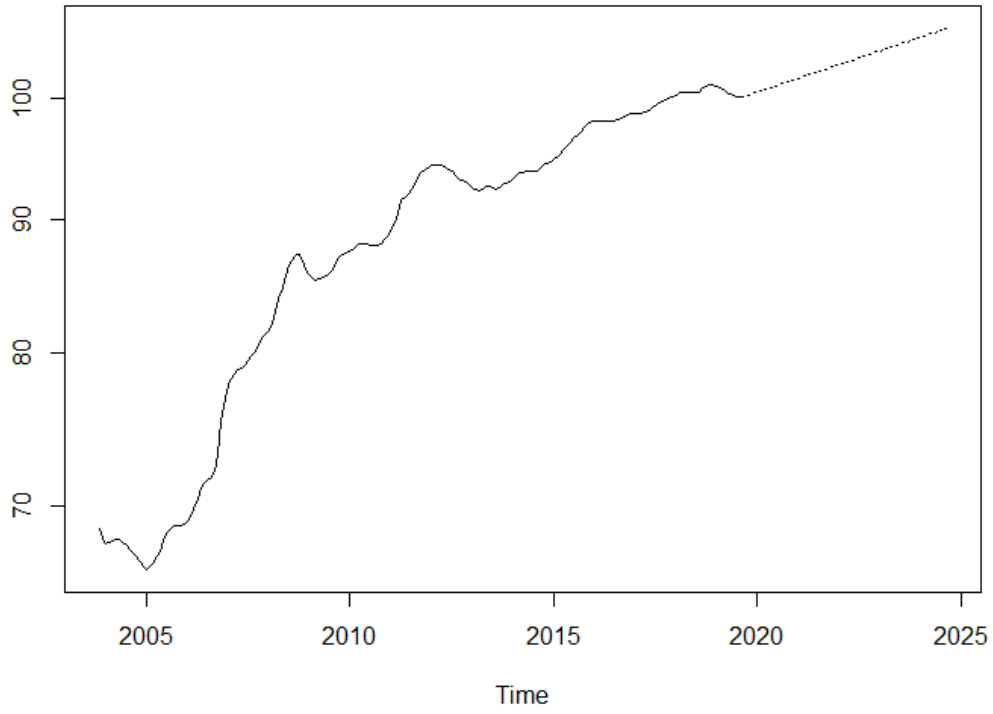



```
> ## 5. ARIMA 모델링
> auto.arima(log(ts.h)) # # 자동으로 ARIMA 모형 확인
Series: log(ts.h)
ARIMA(0,2,0)

sigma^2 estimated as 6.881e-06:  log likelihood=857.76
AIC=-1713.53   AICc=-1713.51   BIC=-1710.29
> fit <- arima(log(ts.h),
+              c(0, 2, 0),
+             seasonal = list(order = c(0, 0, 0),
+              period = 12))
> summary(fit)

Call:
arima(x = log(ts.h), order = c(0, 2, 0), seasonal = list(order = c(0, 0, 0),
    period = 12))


sigma^2 estimated as 6.691e-06:  log likelihood = 857.76,  aic = -1713.53

Training set error measures:
                      ME        RMSE         MAE           MPE       MAPE      MASE       ACF1
Training set 2.364179e-05 0.002609423 0.001528972 0.0006435289 0.03468371 0.5019742 0.05006917
> Box.test(fit$residuals)

        Box-Pierce test

data:  fit$residuals
X-squared = 0.47882, df = 1, p-value = 0.489
```

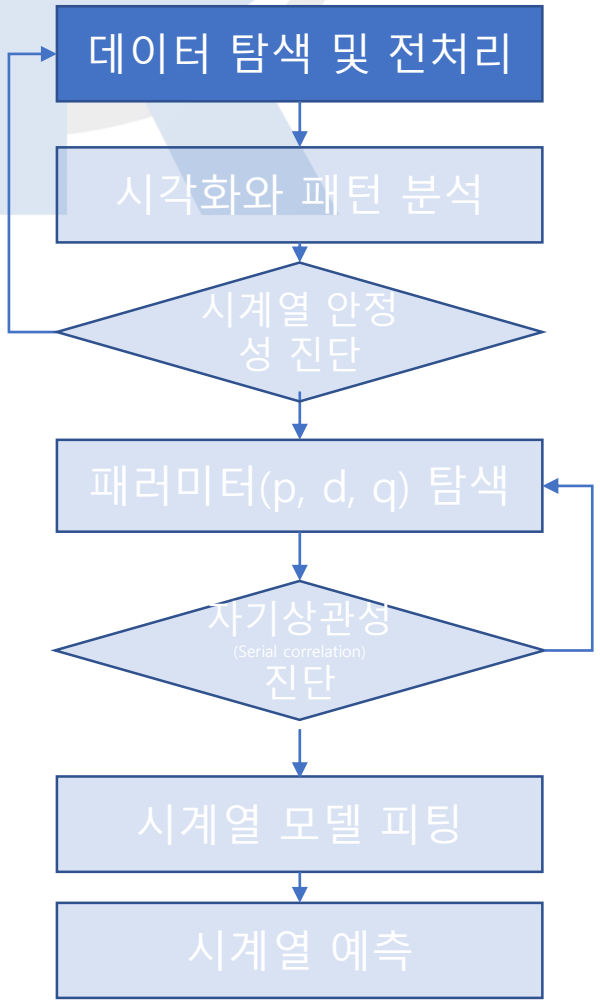
실습2: 한국 월별 주택매매가격 지수




```
> ## 6. 시계열 예측
> pred <- predict(fit, n.ahead = 5*12)
> ts.plot(ts.h,
+         exp(pred$pred), # 자연로그 e (= 2.718...)
+         log = "y",
+         lty = c(1,3))
```

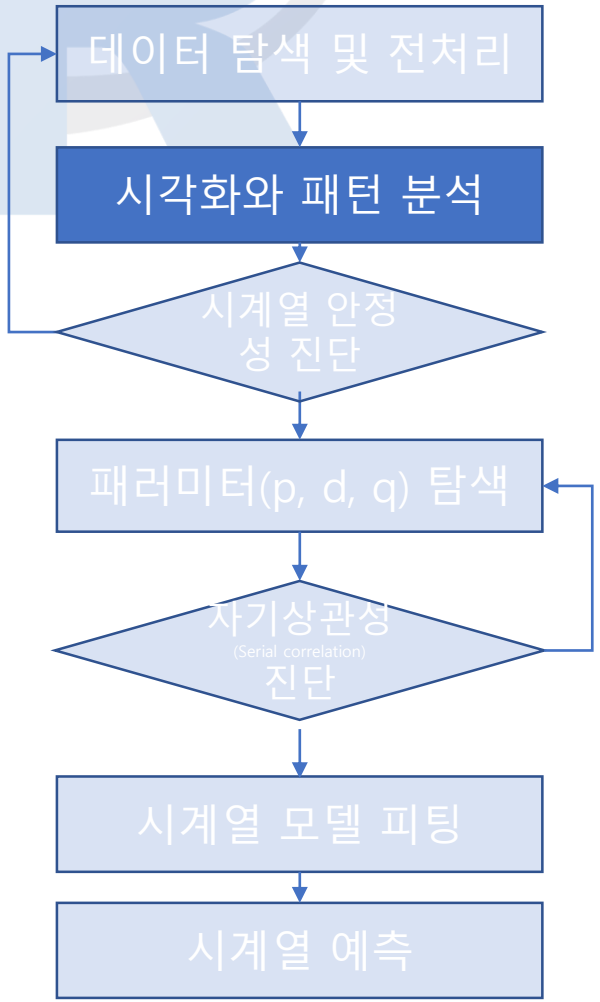


실습3: 다중시계열 분석: 서울시 일별 지하철 이용자수

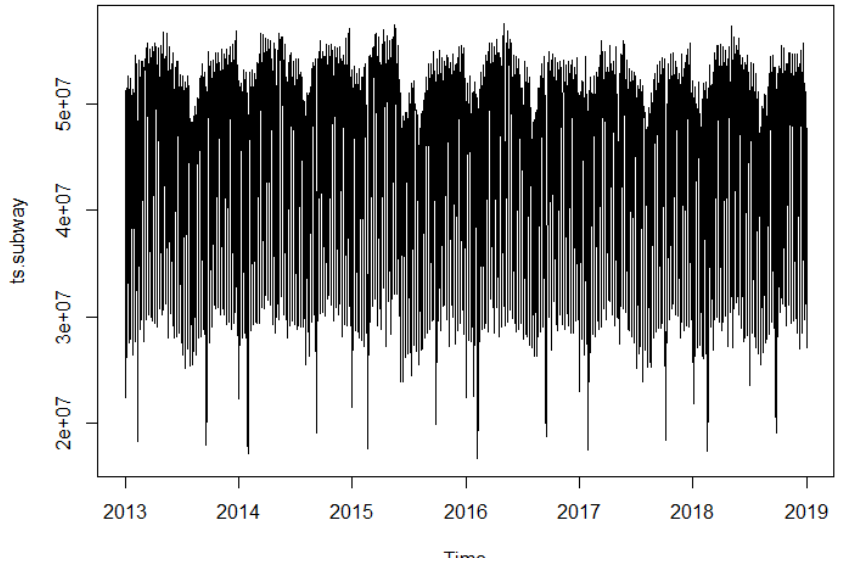


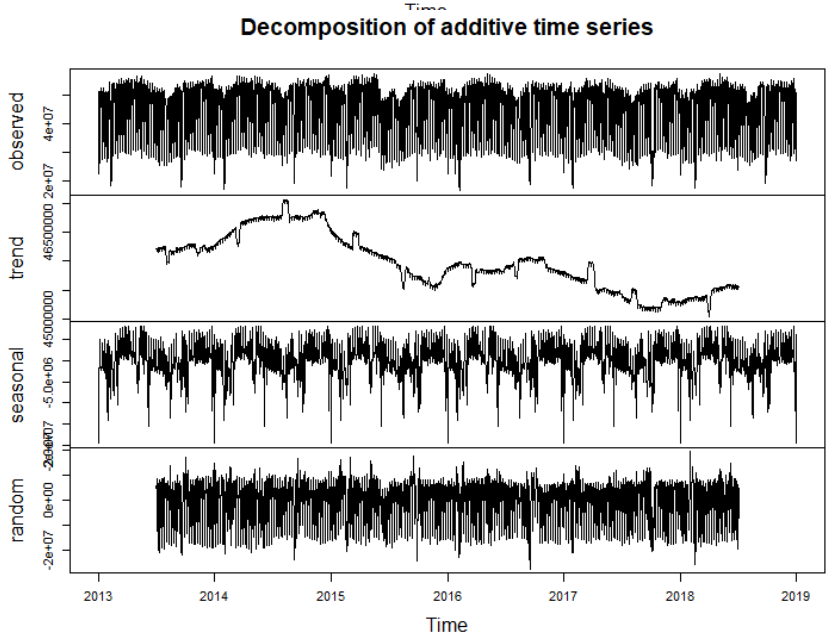

```
> # 실습 3: 다중시계열 분석: 서울시 지하철 이용자수
> setwd("K:\\기타\\2019년2학기\\수치해석\\실습데이터\\서울시대중교통")
> list.files()
 [1] "airquality_transit_ridership.pdf"
 [2] "BUS_STATION_BOARDING_MONTH_201501.csv"
 [3] "BUS_STATION_BOARDING_MONTH_201502.csv"
 [4] "BUS_STATION_BOARDING_MONTH_201503.csv"
 [5] "BUS_STATION_BOARDING_MONTH_201504.csv"
 [6] "BUS_STATION_BOARDING_MONTH_201505.csv"
> df.subway <- read.csv("seoul_subway_total_ridership.csv")
> summary(df.subway)
       X                  날짜          total            tot_amPeak
 Min.   :   1.0   2013-01-01:   1   Min.   :16675175   Min.   : 307653
 1st Qu.: 548.5   2013-01-02:   1   1st Qu.:40033317   1st Qu.:1095628
 Median :1096.0   2013-01-03:   1   Median :50949779   Median :2656469
 Mean   :1096.0   2013-01-04:   1   Mean   :45969564   Mean   :2134055
 3rd Qu.:1643.5   2013-01-05:   1   3rd Qu.:52887589   3rd Qu.:2831405
 Max.   :2191.0   2013-01-06:   1   Max.   :57604832   Max.   :3097891
                  (other)   :2185
> ts.subway <- msts(df.subway$total,
+                   seasonal.periods = c(7,365.25),
+                   start = c(2013, 1, 1))
> plot(ts.subway, main="Daily subway ridership", xlab="Year", ylab="Daily Admissions")
> class(ts.subway)
[1] "msts" "ts"
> start(ts.subway)
[1] 2013    1
> end(ts.subway)
[1] 2019    1
> frequency(ts.subway)
[1] 365
> summary(ts.subway)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
16675175 40033317 50949779 45969564 52887589 57604832
```

실습3: 다중시계열 분석: 서울시 일별 지하철 이용자수

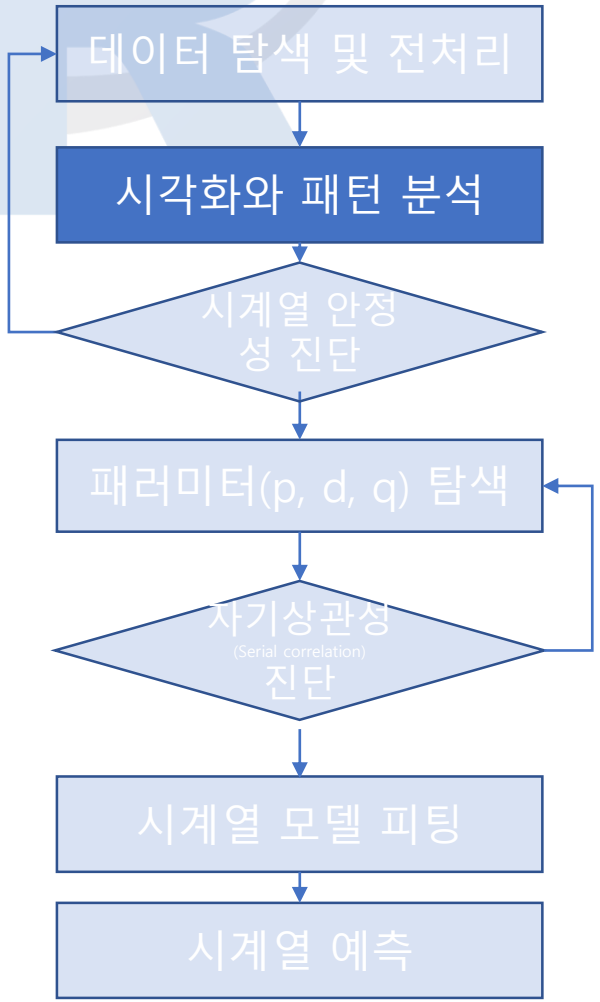



```
> ## 2. 시계열 시각화와 패턴 분석
> plot(ts.subway) #
> library(forecast)
> ## 2. 시계열 시각화와 패턴 분석
> plot(ts.subway)
> plot(decompose(ts.subway))
```

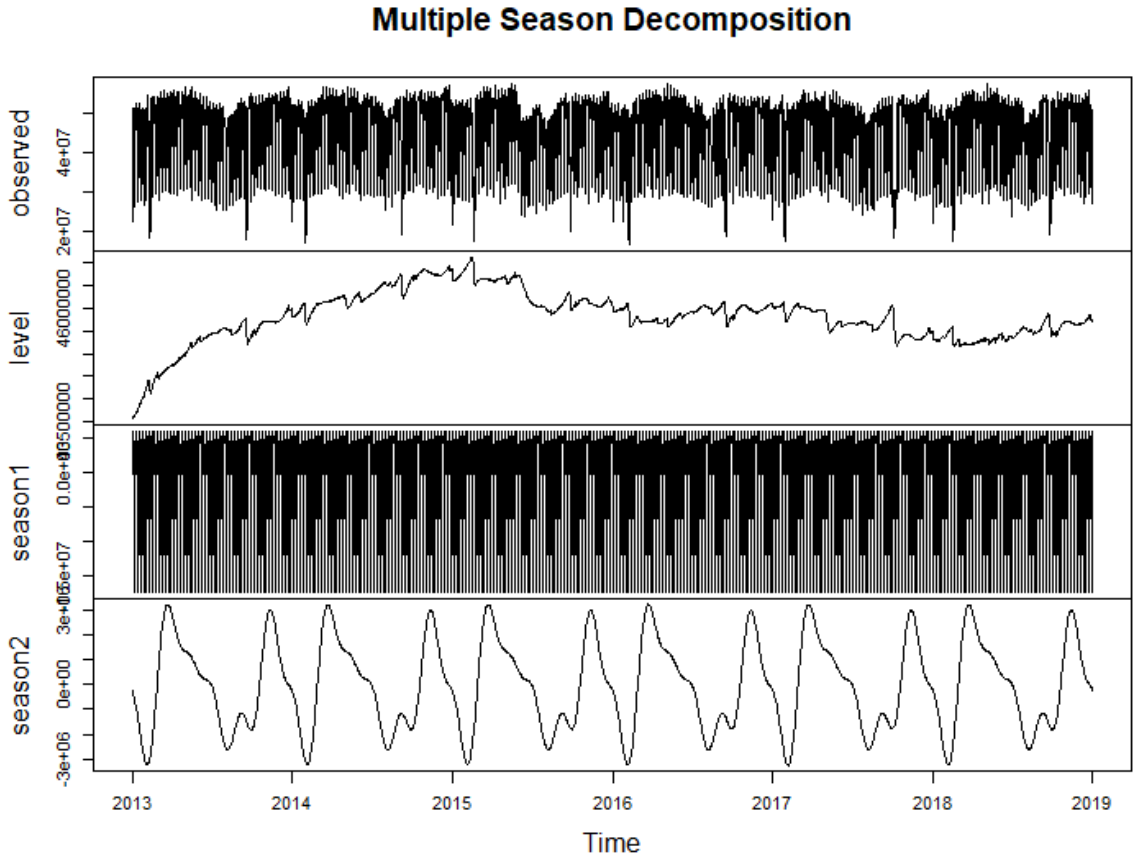




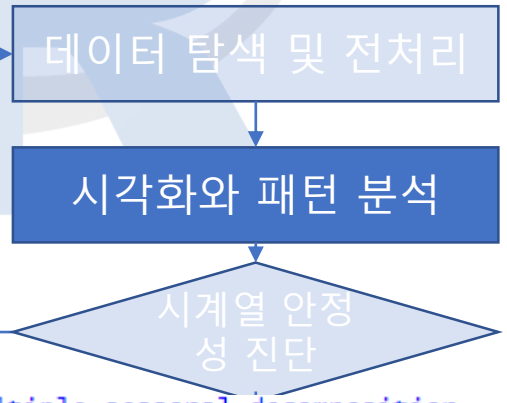
실습3: 다중시계열 분석: 서울시 일별 지하철 이용자수



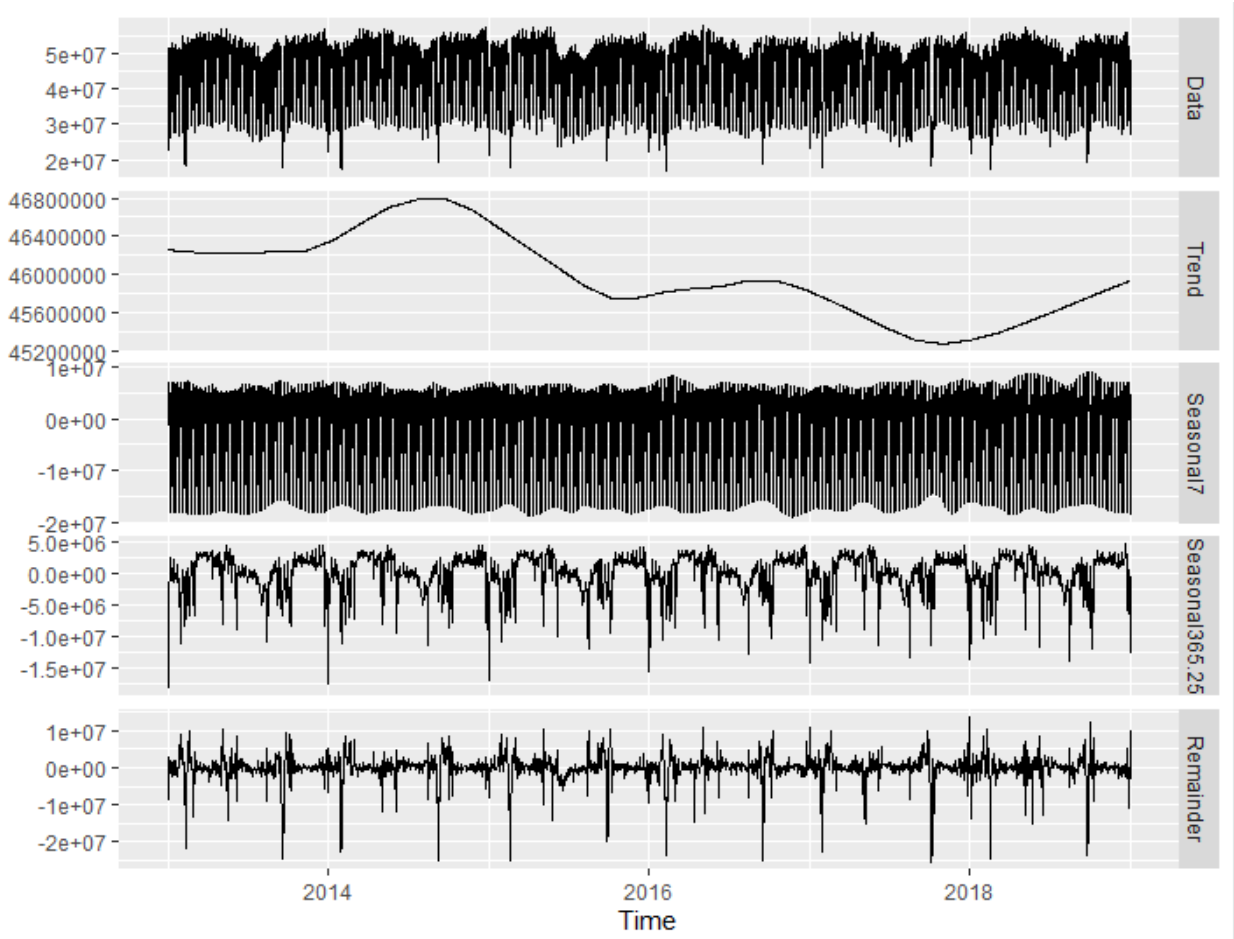

```
> library(forecast)
> ?tbats() # TBATS model (Exponential smoothing state space model with Box-Co
> tbats <- tbats(ts.subway)
> plot(tbats, main="Multiple Season Decomposition") # 시간이 오래 걸림
```



실습3: 다중시계열 분석: 서울시 일별 지하철 이용자수

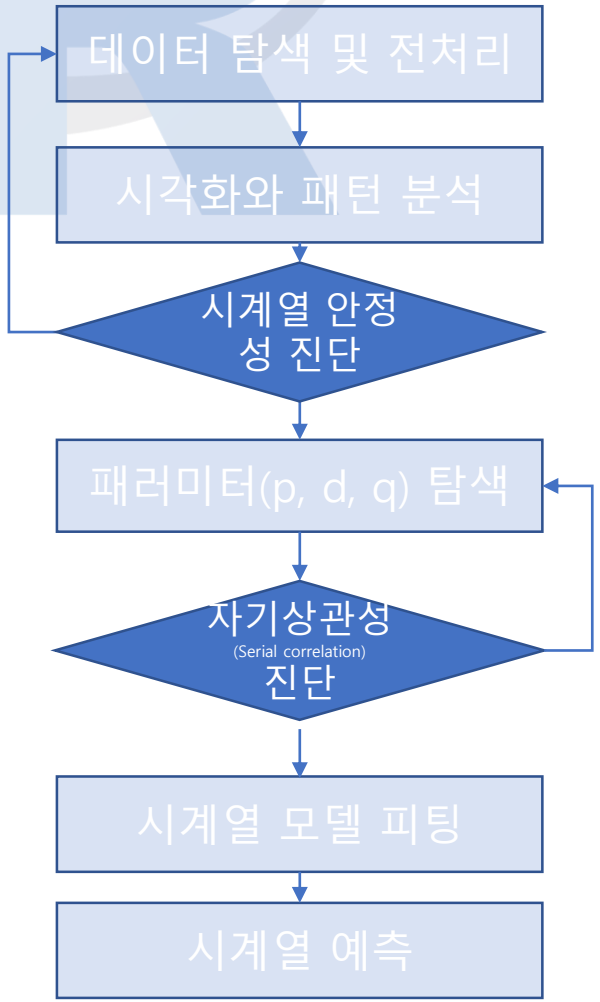



```
> ?mstl() # Multiple seasonal decomposition
> head(mstl(ts.subway))
Multi-Seasonal Time Series:
Start: 2013 1
Seasonal Periods:
Data:
         Data    Trend Seasonal7 Seasonal365.25  Remainder
[1,] 22461407 46255951   3293895       -18319180 -8769259.3
[2,] 49522564 46255561   7179216        -6724070  2811856.9
[3,] 48314717 46255171   6813456        -2967470 -1786440.5
[4,] 51312257 46254781   5968798        -1414159   502836.9
[5,] 36274484 46254391  -8321771        -1065809  -592327.2
[6,] 26229672 46254001 -18118246        -1213188  -692894.5
attr(,"seasonal.periods")
[1]   7.00 365.25
```



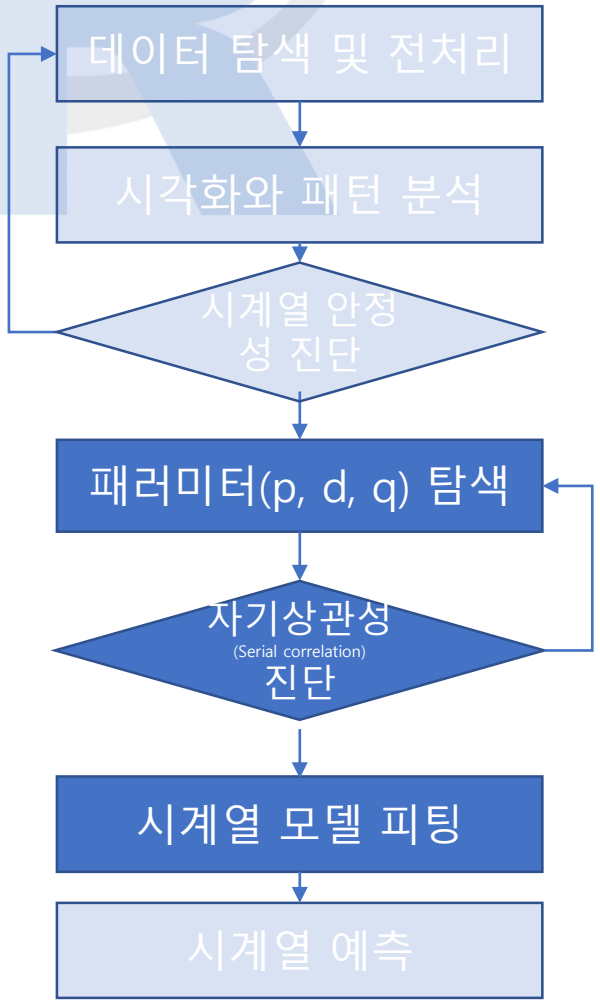

```
> ?autoplot() #  ggplot2 to draw a particular plot for an object of a particular class in a single command.
> autoplot(mstl(ts.subway))
```

실습3: 다중시계열 분석: 서울시 일별 지하철 이용자수



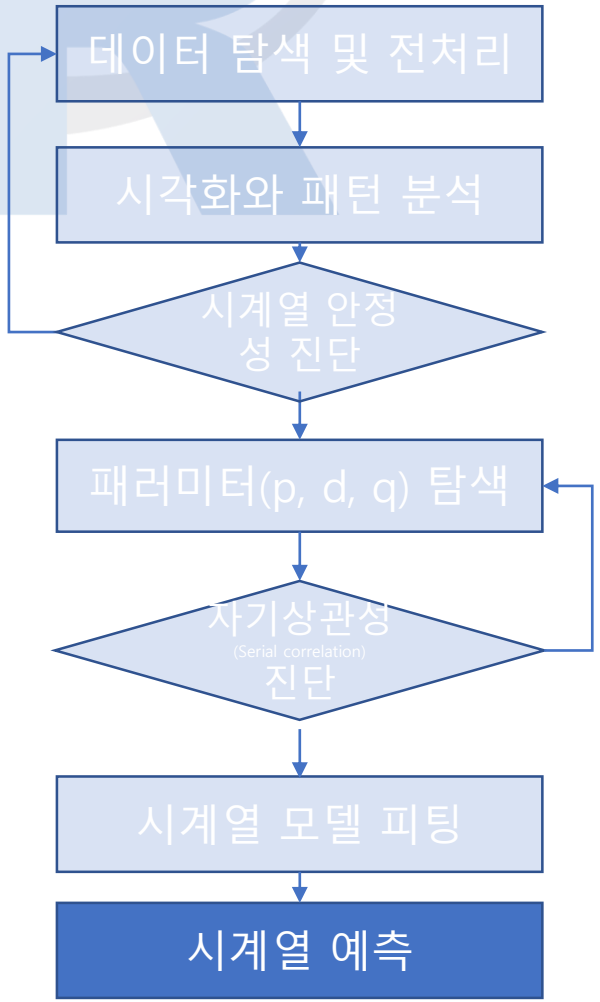

```
> ## 3. 시계열 안정성 진단 및 검증
> adf.test(ts.subway,
+          alternative="stationary", #  stationary enough to do any kind of time series modelling.
+          k=0)

        Augmented Dickey-Fuller Test

data:  ts.subway
Dickey-Fuller = -34.1, Lag order = 0, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(ts.subway, alternative = "stationary", k = 0) :
  p-value smaller than printed p-value
> Box.test(ts.subway)

        Box-Pierce test

data:  ts.subway
X-squared = 207.71, df = 1, p-value < 2.2e-16
```

실습3: 다중시계열 분석: 서울시 일별 지하철 이용자수




```
> ## 4. ARIMA모델 패러미터 탐색
> auto.arima(ts.subway) # 자동으로 ARIMA 모형 패러미터 확인: 시간이 오래 걸림
Series: ts.subway
ARIMA(5,0,0)(0,1,0)[365]

Coefficients:
          ar1      ar2     ar3     ar4      ar5
      0.1105  -0.1617  0.0651  0.0285  -0.2593
s.e.  0.0226   0.0227  0.0230  0.0227   0.0226

sigma^2 estimated as 1.258e+14:  log likelihood=-32230.12
AIC=64472.23   AICc=64472.28   BIC=64505.29

> ## 5. ARIMA 모델링
> fit <- arima(ts.subway, # 시간 오래 걸림
+              c(5, 0, 0),
+              seasonal = list(order = c(0, 1, 0),
+                              period = 365))
> fit

Call:
arima(x = ts.subway, order = c(5, 0, 0), seasonal = list(order = c(0, 1, 0),
    period = 365))

Coefficients:
          ar1      ar2     ar3     ar4      ar5
      0.1105  -0.1617  0.0651  0.0285  -0.2593
s.e.  0.0226   0.0227  0.0230  0.0227   0.0226

sigma^2 estimated as 1.255e+14:  log likelihood = -32230.12,  aic = 64472.23

> Box.test (fit$residuals, lag = 1, type = "Ljung")

        Box-Ljung test

data:  fit$residuals
X-squared = 0.23836, df = 1, p-value = 0.6254
```

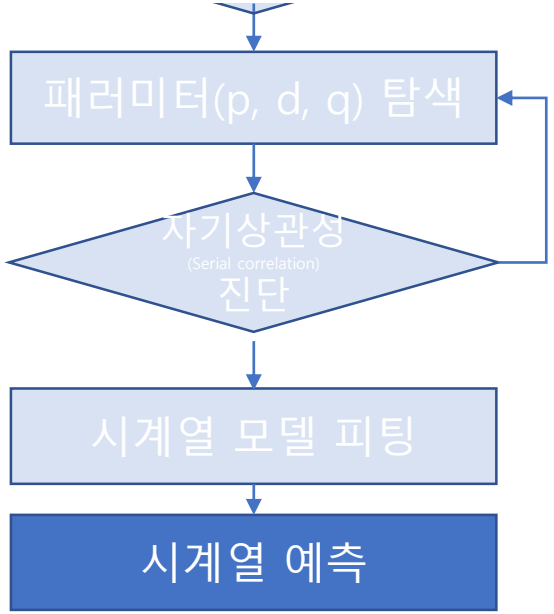
실습3: 다중시계열 분석: 서울시 일별 지하철 이용자수

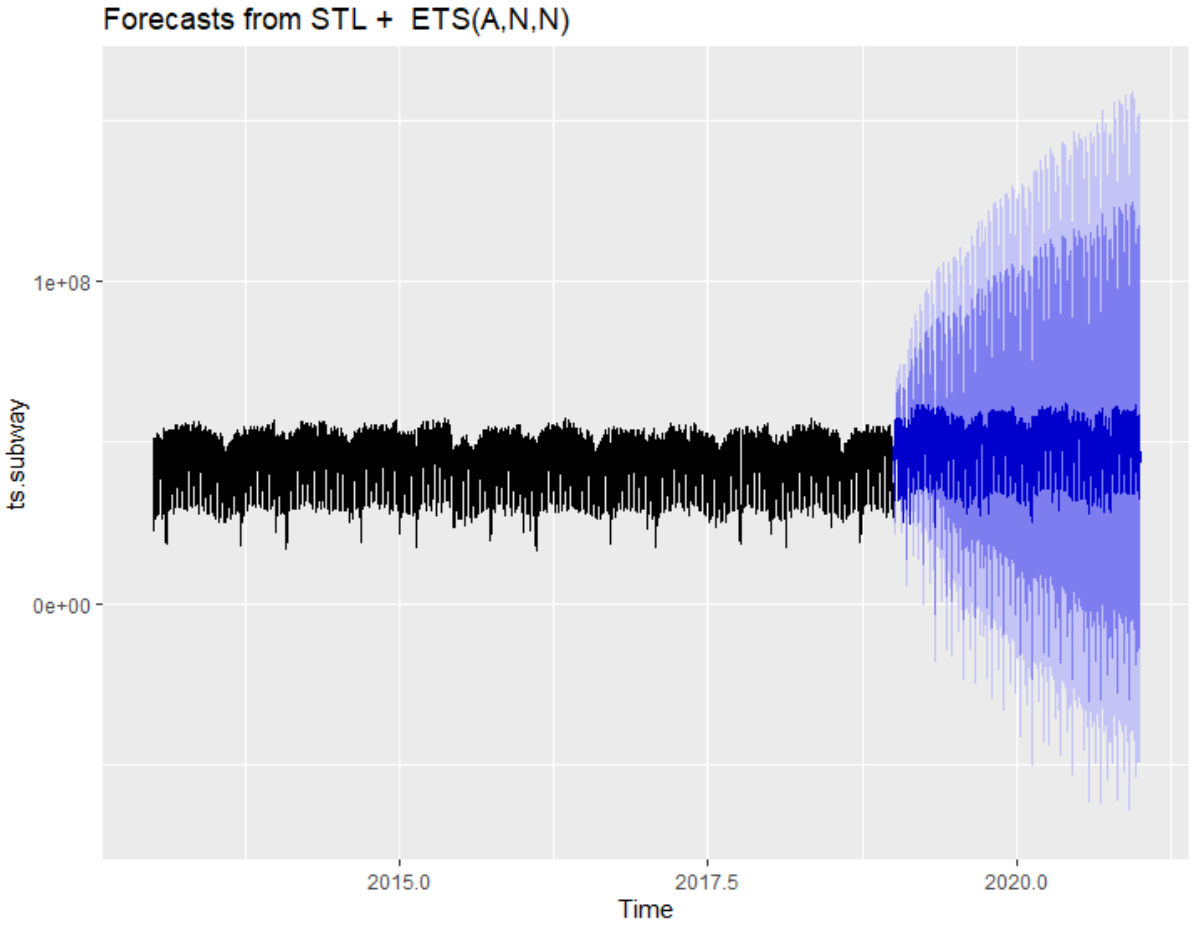



```
> ## 6. 시계열 예측
> ?stlf() # Forecasting using stl objects
> forecast <- stlf(ts.subway)
> summary(forecast)

Forecast method: STL +  ETS(A,N,N)

Model Information:
ETS(A,N,N)

Call:
 ets(y = x, model = etsmodel, allow.multiplicative.trend = allow.multiplicative.trend)

  Smoothing parameters:
    alpha = 0.5463

  Initial states:
    l = 45060655.0989

  sigma:  3473210

     AIC     AICc      BIC
82852.92 82852.93 82870.00

Error measures:
                    ME     RMSE      MAE         MPE     MAPE      MASE       ACF1
Training set 4944.234 3471624 1960656 -0.7640662 5.248306 0.2396305 0.0183632

Forecasts:
          Point Forecast      Lo 80    Hi 80      Lo 95    Hi 95
2019.0027       44148746 39697648.6 48599844  37341379.8 50956113
2019.0055       53783989 48712039.4 58855939  46027111.3 61540867
2019.0082       55466933 49842247.6 61091618  46864719.2 64069146
```

실습3: 다중시계열 분석: 서울시 일별 지하철 이용자수

```
2019.5041       56655747 23462638.9 89848854   5891269.1 107420224
2019.5068       57791085 24509036.0 91073135   6890583.4 108691587
2019.5096       57354311 23983557.3 90725065   6318147.3 108390475
2019.5123       43675659 10216436.2 77134883  -7495806.6  94847125
2019.5151       32382669 -1164790.6 65930128 -18923742.8  83689080
2019.5178       55631971 21996507.1 89267435   4190968.1 107072974
2019.5205       53166600 19443360.8 86889838   1591356.6 104741843
2019.5233       56628797 22818011.1 90439583   4919662.2 108337932
2019.5260       57665261 23767153.9 91563367   5822580.1 109507941
2019.5288       57697483 23712280.1 91682687   5721600.2 109673367
2019.5315       43709982  9637904.6 77782059  -8398763.6  95818728
2019.5342       32905585 -1253145.1 67064316 -19335684.7  85146855
2019.5370       55763506 21518341.8 90008670   3390047.0 108136965
2019.5397       52696853 18365472.7 87028233    191537.8 105202168
2019.5425       56584811 22167431.1 91002192   3947970.5 109221653
2019.5452       57811578 23308412.2 92314745   5043539.3 110579618
2019.5479       57886714 23297974.3 92475453   4987801.7 110785625
 [ reached 'max' / getOption("max.print") -- omitted 530 rows ]
> autoplot(stlf(ts.subway))
```





연습문제 05

- 서울시 일별 지하철 이용자수 데이터를 7일 주기가 아닌 1년 주기(365.25)로만 설정하여 ARIMA의 적합한 패터미터 값이 얼마인지 auto.arima()함수를 활용하여 확인하시오.

연습문제 06

- 서울시 월별 주택매매가격지수(df.h$서울)를 이용하여 ARIMA의 적합한 패러미터 값이 얼마 인지 auto.arima()함수를 활용하여 확인하시오.

연습문제 07

- 서울시 일별 버스이용자수 데이터 중 일별 승차자수(on_bus) 자료를 활용하여 ARIMA 모형의 p, d, q를 적합하게 auto.arima() 함수 구축 후 이 모델링 후의 잔차에 대한 자기상관성 검정을 실시하시오.

요약

- 시계열 자료와 모델링 개요
  - 시계열 자료와 모델
  - 시계열 자료 분해
  - 시계열 자료와 정상성
  - 불안정 시계열과 모델링
  - 시계열 모델: AR과 MA
  - 시계열 모델: ARIMA와 분해 시계열
  - 정상성 검정
  - 백색 잡음과 자기상관성 검정
  - 시계열자료와 다중 계절성
- 시계열 모델링 분석절차
- 실습
  - 실습 1: 월별 국제항공노선 이용자수
  - 실습 2: 월별 주택매매가격 지수
  - 실습 3: 다중시계열 분석: 서울시 지하철 이용자수

끝

- 질의와 토의(Question & Discussion)
  - 이번 강의 내용을 시청하고, 실행하면서 궁금한 점이나 어려운 점에 대하여 토의해봅시다.