

# ADsP

인공지능소프트웨어학과

강환수 교수

# R 기초와 데이터 마트

DONGYANG MIRAE UNIVERSITY  
Dept. of Artificial Intelligence



# R 기본 자료형

character > numeric > logical

- **character**
  - 문자
- **numeric**
  - 숫자
- **logical**
  - 논리값 TRUE, T, FLASE, F
- **특수한 상수(literals)**
  - NULL
  - NA
    - not available
  - NaN
    - not a number
  - Inf, -Inf
    - infinity

표 2-3 R에서 사용할 수 있는 값의 자료형

자료형	사용 예	비고
숫자형	1, 2, 3, -4, 12.8	정수와 실수 모두 가능
문자형	'Tom', "Jane"	작은따옴표나 큰따옴표로 묶어서 표현
논리형	TRUE, FALSE	반드시 따옴표가 없는 대문자로 표기하며, T나 F로 줄여서 사용하는 것도 가능
특수값	NULL	정의되어 있지 않음을 의미하며, 자료형도 없고 길어도 0임
	NA	결측값(missing value)
	NaN	수학적으로 정의가 불가능한 값 예 sqrt(-3)
	Inf, -Inf	양의 무한대(Inf), 음의 무한대(-Inf)

## 주요 산술 연산자

표 2-1 산술연산자

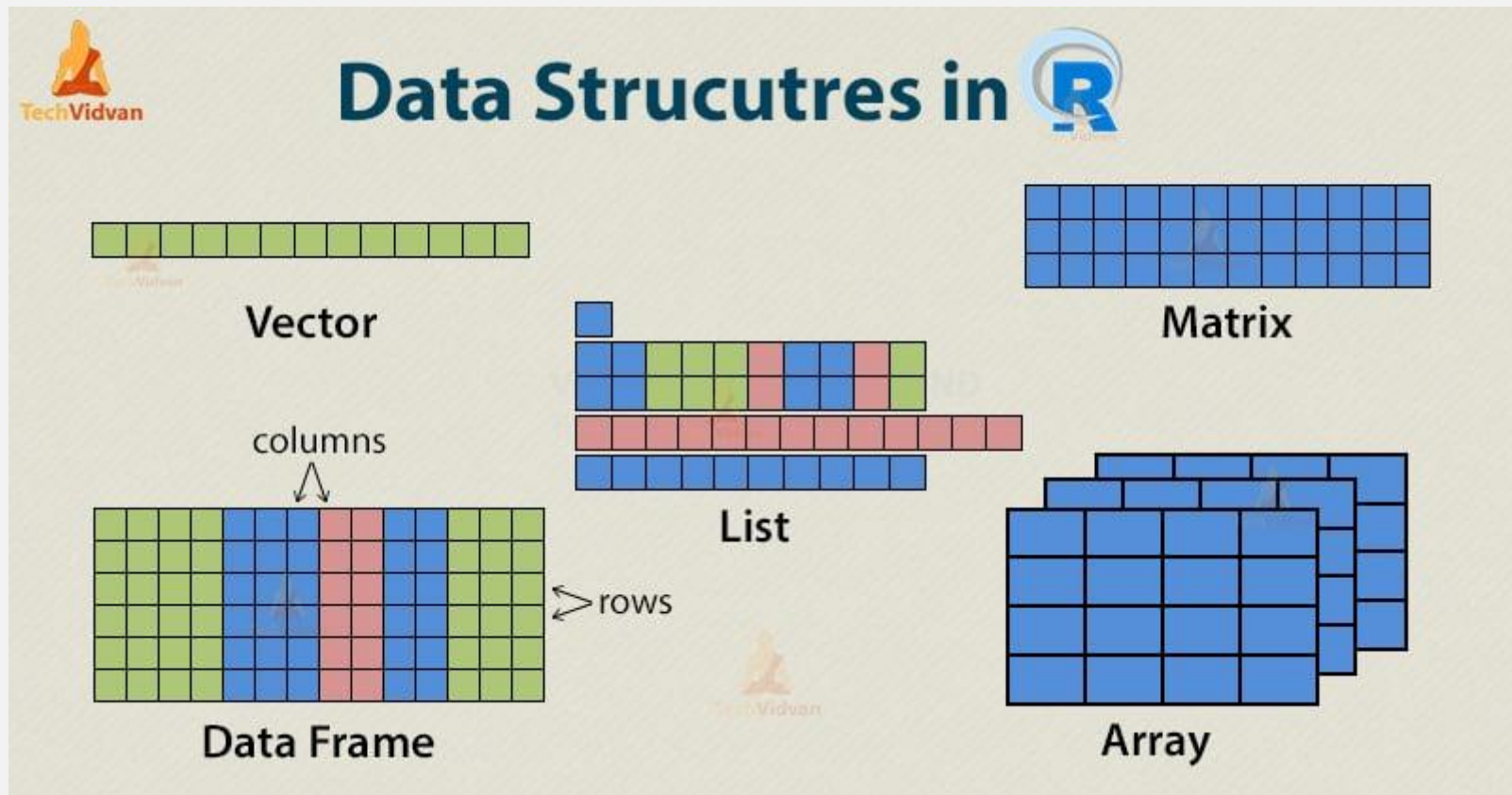
연산자	의미	사용 예
+	덧셈	3+5+8
-	뺄셈	9-3
*	곱셈	7*5
/	나눗셈	8/3
%%	나눗셈의 나머지	8%%3
^	제곱	2^3

## 주요 산술연산 함수

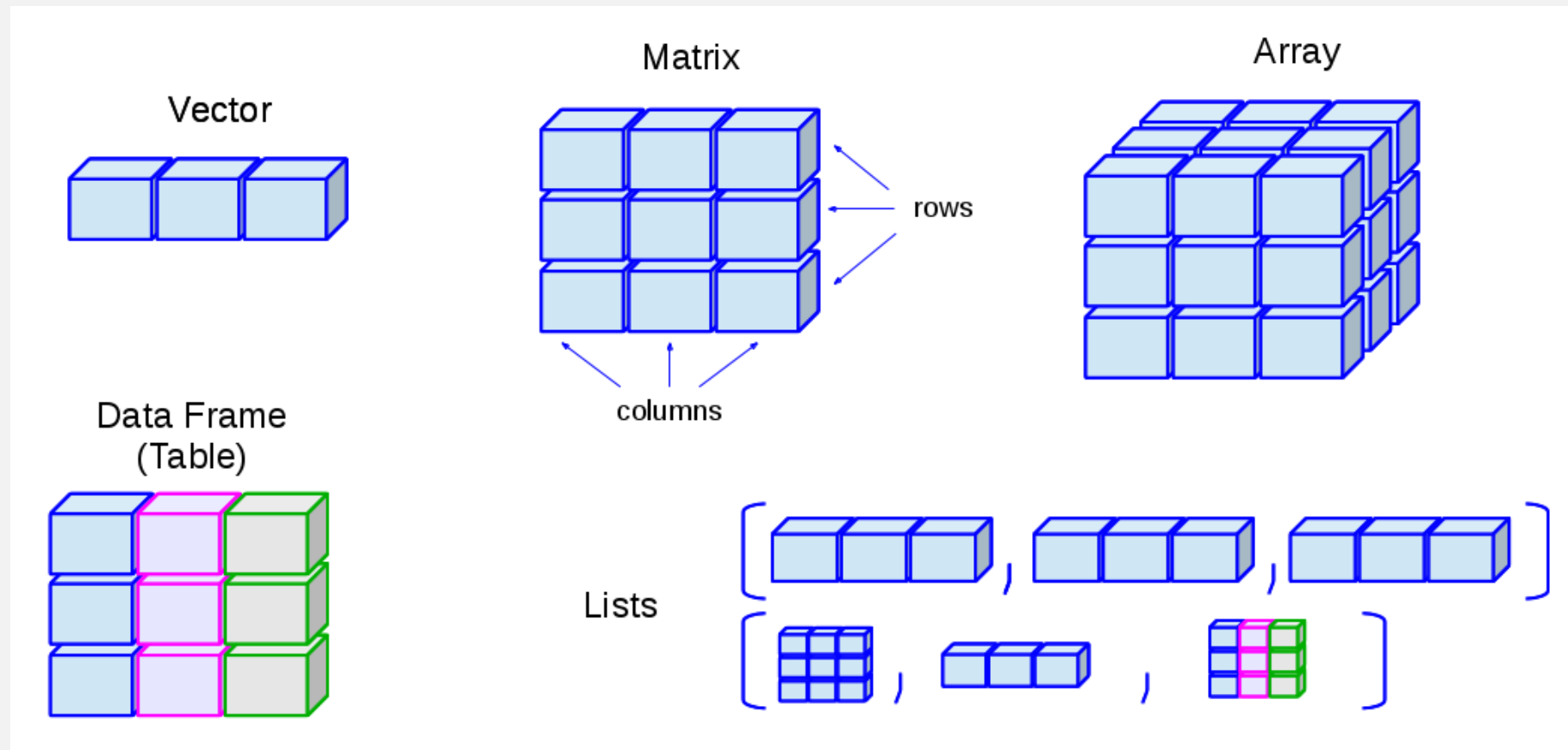
표 2-2 산술연산 함수

함수	의미	사용 예
log( )	로그함수	log(10), log(10, base=2)
sqrt( )	제곱근	sqrt(36)
max( )	가장 큰 값	max(3,9,5)
min( )	가장 작은 값	min(3,9,5)
abs( )	절댓값	abs(-10)
factorial( )	팩토리얼	factorial(5)
sin( ), cos( ), tan( )	삼각함수	sin(pi/2)

## R 자료 구조 (1)



## R 자료 구조 (2)



## 도움말

- > `help()`
- > `help(cat)`
- > `?cat`
- > `??cat`
- > `help(package=ggplot2)`

## 자료형과 내부구조 확인 함수

- 함수 `str()`
  - 객체의 구조를 간결하게 보여 줌
  - 데이터프레임, 리스트, 벡터, 팩터 등 다양한 객체의 내부 구조를 요약하여 출력
- 함수 `mode()`
  - 객체의 기본 자료형을 반환
  - 객체가 기본적으로 어떤 자료형인지 확인
- 함수 `class()`
  - 객체의 클래스 속성을 반환
    - 객체의 클래스(타입)를 확인
    - 데이터프레임, 팩터, 리스트 등
    - [S3] 객체 지향 프로그래밍에서 객체의 클래스 확인에 주로 사용
- 함수 `typeof()`
  - 객체의 내부 저장 유형을 반환
    - 객체가 메모리에서 어떻게 저장되는지를 확인
    - 좀 더 기술적인 정보 제공 (R의 내부 자료형)



## 4개 함수 비교

함수	목적	반환값	용도
<code>`str()`</code>	객체의 구조 요약	객체의 구조 요약 텍스트	데이터 요약 및 디버깅
<code>`mode()`</code>	객체의 기본 자료형	"numeric", "character" 등	기본 자료형 확인
<code>`class()`</code>	객체의 클래스 속성	"data.frame", "factor" 등	객체의 클래스 확인, S3 객체 지향 프로그래밍
<code>`typeof()`</code>	객체의 내부 저장 유형	"integer", "double" 등	객체의 메모리 저장 유형 확인

## R 객체 속성 관련 함수 비교 (배열 예시: `arr = array(1:12, dim = c(2,3,2))`)

함수	결과 예시	반환값 유형	설명	주요 특징 및 목적	관련 함수 / 참고
<code>str(arr)</code>	int [1:2, 1:3, 1:2] 1 2 3 ...	문자열 요약 출력	객체의 구조(structure)를 요약 출력하며 내부 구조, 데이터 타입, 차원, 값 일부를 보여줌	객체를 빠르게 이해하는 데 유용한 요약 정보 제공. 탐색적 데이터 분석(EDA)에 많이 사용됨.	<code>attributes()</code> , <code>dim()</code> , <code>summary()</code>
<code>mode(arr)</code>	"numeric"	문자열	객체의 개념적 데이터 타입을 반환. R의 구형 시스템에서 사용됨	"numeric", "character", "list", "function" 등 추상적인 분류 중심.	R의 전통적 타입 시스템 (LISP 스타일). <code>is.numeric()</code> 등과 연계됨.
<code>typeof(arr)</code>	"integer"	문자열	객체가 메모리에 어떻게 저장되는지를 알려주는 저수준 내부 타입 반환	"integer", "double", "character" 등 실제 저장 구조 기반. <code>mode()</code> 보다 더 구체적임.	R 내부 구현에 가까운 정보 제공. <code>storage.mode()</code> 와 비교 가능
<code>class(arr)</code>	"array"	문자열 또는 문자열 벡터	객체가 어떤 클래스(객체 유형)인지 반환. S3 객체지향 시스템에서 메서드 디스패치에 사용됨	"array", "matrix", "data.frame", "lm" 등. S3 메서드 호출 시 사용됨.	<code>methods()</code> , S3 method dispatch, <code>inherits()</code>

# 자료형과 자료 구조 확인 함수

## • 함수 str(): 내부 구조 확인

객체	예시	mode 함수 적용	class 함수 적용	typeof 함수
숫자	3	numeric	numeric	double
문자열	"hi"	character	character	character
숫자벡터	c(1,2,3)	numeric	numeric	double
문자열벡터	c("hi","hello")	character	character	character
Factor	factor(c("A","A","B"))	numeric	factor	integer
리스트	list(c(1,2),c("A"))	list	list	list
데이터프레임	data.frame(c(1,2))	list	data.frame	list
행렬	matrix(c(12,3,4),2,2)	numeric	matrix	double
배열	array(c(1,2,3,4),dim=c(2,2,3))	numeric	array	integer
테이블	table(c(1,1,2,3))	numeric	table	integer
함수	mean	function	function	closure

## 4개 함수 코드 비교

```
# 데이터프레임 생성
df <- data.frame(name = c("Alice", "Bob"), age = c(25, 30))

# str() 출력
str(df)
# 'data.frame':  2 obs. of  2 variables:
# $ name: Factor w/ 2 levels "Alice","Bob": 1 2
# $ age : num  25 30

# mode() 출력
mode(df)
# [1] "list"

# class() 출력
class(df)
# [1] "data.frame"

# typeof() 출력
typeof(df)
# [1] "list"
```

## 벡터 활용 함수

표 2-4 벡터에 적용 가능한 함수

함수명	설명
sum()	벡터에 포함된 값들의 합
mean()	벡터에 포함된 값들의 평균
median()	벡터에 포함된 값들의 중앙값
max(), min()	벡터에 포함된 값들의 최댓값, 최솟값
var()	벡터에 포함된 값들의 분산
sd()	벡터에 포함된 값들의 표준편차
sort()	벡터에 포함된 값들을 정렬(오름차순이 기본)
range()	벡터에 포함된 값들의 범위(최솟값~ 최댓값)
length()	벡터에 포함된 값들의 개수(길이)

# 함수 as.OOO()

## 변환 함수

함수	설명	예시	변환 결과
as.numeric()	객체를 숫자형(numeric)으로 변환	as.numeric("3.14")	3.14
as.integer()	객체를 정수형(integer)으로 변환	as.integer(3.14)	3
as.character()	객체를 문자형(character)으로 변환	as.character(123)	"123"
as.factor()	객체를 팩터(factor)로 변환	as.factor(c("apple", "banana", "apple"))	factor("apple", "banana", "apple")
as.data.frame()	객체를 데이터프레임(data.frame)으로 변환	as.data.frame(matrix(1:4, ncol = 2))	data.frame
as.matrix()	객체를 행렬(matrix)로 변환	as.matrix(data.frame(a = 1:3, b = 4:6))	matrix
as.list()	객체를 리스트(list)로 변환	as.list(c(1, 2, 3))	list(1, 2, 3)
as.logical()	객체를 논리형(logical)으로 변환	as.logical(c(1, 0, 1))	TRUE, FALSE, TRUE
as.complex()	객체를 복소수형(complex)으로 변환	as.complex("3+4i")	3+4i
as.Date()	객체를 날짜형(Date)으로 변환	as.Date("2021-01-01")	2021-01-01
as.ts()	객체를 시계열(time series)형으로 변환	as.ts(c(1, 2, 3, 4))	ts
as.environment()	객체를 환경(environment)으로 변환	as.environment(list(a = 1, b = 2))	environment
as.factor()	객체를 범주형 데이터(factor)로 변환	as.factor(c("low", "medium", "high"))	factor("low", "medium", "high")
as.tibble()	객체를 티블(tibble, tidyverse 패키지)로 변환	as_tibble(data.frame(a = 1:3, b = 4:6))	tibble
as.ordered()	팩터(factor)를 순서형 팩터(ordered factor)로 변환	as.ordered(factor(c("low", "medium", "high")))	ordered factor

# 역행렬

함수 solve()

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

$$A^{-1} = \frac{1}{\det(A)} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

$$\det(A) = ad - bc$$

따라서, 역행렬  $A^{-1}$ 는 다음과 같습니다:

$$A^{-1} = \frac{1}{ad-bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

## 예시

행렬  $A$ 가 다음과 같다고 가정합니다:

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

1. 행렬식 계산:

$$\det(A) = 1 \cdot 4 - 2 \cdot 3 = 4 - 6 = -2$$

2. 역행렬 계산:

$$A^{-1} = \frac{1}{-2} \begin{pmatrix} 4 & -2 \\ -3 & 1 \end{pmatrix} = \begin{pmatrix} \frac{4}{-2} & \frac{-2}{-2} \\ \frac{-3}{-2} & \frac{1}{-2} \end{pmatrix} = \begin{pmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{pmatrix}$$

## 확인

행렬  $A$ 와  $A^{-1}$ 을 곱하여 항등행렬이 되는지 확인합니다:

$$\begin{aligned} A \times A^{-1} &= \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \times \begin{pmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{pmatrix} = \\ &= \begin{pmatrix} (1 \cdot -2 + 2 \cdot \frac{3}{2}) & (1 \cdot 1 + 2 \cdot -\frac{1}{2}) \\ (3 \cdot -2 + 4 \cdot \frac{3}{2}) & (3 \cdot 1 + 4 \cdot -\frac{1}{2}) \end{pmatrix} = \begin{pmatrix} -2 + 3 & 1 - 1 \\ -6 + 6 & 3 - 2 \end{pmatrix} = \\ &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I \end{aligned}$$

항등행렬  $I$ 이므로,  $A^{-1}$ 이 올바르게 계산되었음을 확인할 수 있습니다.



## 공분산과 상관(관계)계수 (1)

- 비교 두 변수의 단위의 차이
  - 공분산은 두 변수의 단위에 따라 달라지므로, 비교하기 어려움
- 공분산의 해석
  - 공분산의 크기는 절대적인 기준이 없으므로, 비교하거나 해석하기 어려움이 있음
    - 양의 공분산: 두 변수가 같은 방향으로 변동
    - 음의 공분산: 두 변수가 반대 방향으로 변동
    - 공분산이 0에 가까움: 두 변수 간의 선형 관계가 거의 없음
- 상관계수의 해석
  - 해석의 용이성: 상관계수는 두 변수 간의 관계의 강도와 방향을 직관적으로 이해하기 쉬움
    - 무단위 척도이므로 값의 범위가 -1에서 1 사이로 고정
  - 상관계수는 공분산을 표준화하여 해석을 쉽게 만든 지표
    - 1: 완벽한 양의 선형 관계
    - -1: 완벽한 음의 선형 관계
    - 0: 선형 관계가 없음
  - 비교 기준
    - 상관계수의 절댓값이 0.7 이상이면 강한 관계
    - 0.3 이상 0.7 미만
      - 약한 관계
    - 0.3 미만이면
      - 매우 약한 관계 또는 거의 관계가 없다고 해석

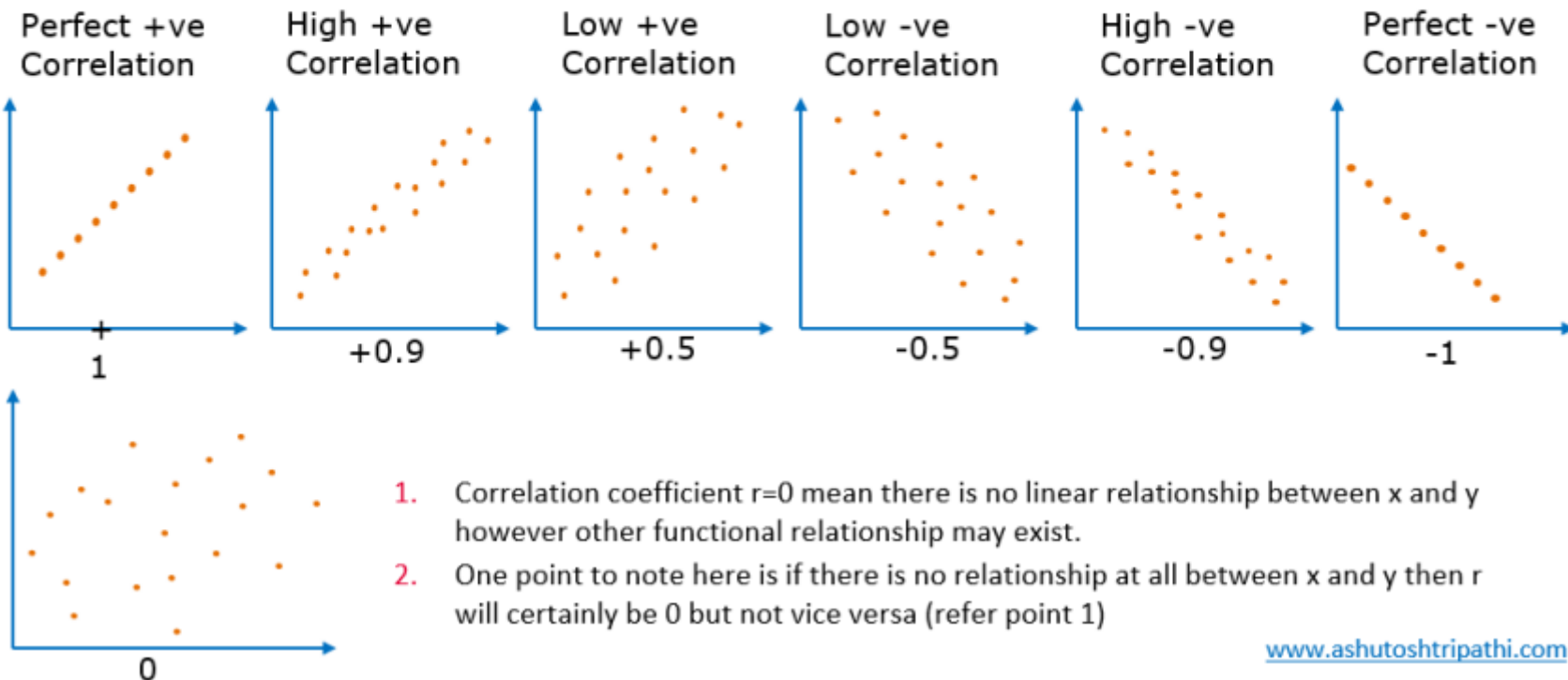
## 공분산과 상관(관계)계수 (2)

Correlation coefficient  $r$  is number between -1 to +1 and tells us how well a regression line fits the data and defined by

$$r_{xy} = \frac{s_{xy}}{s_x s_y}$$

where,

- $s_{xy}$  is the covariance between  $x$  and  $y$
- $s_x$  and  $s_y$  are the standard deviations of  $x$  and  $y$  respectively.

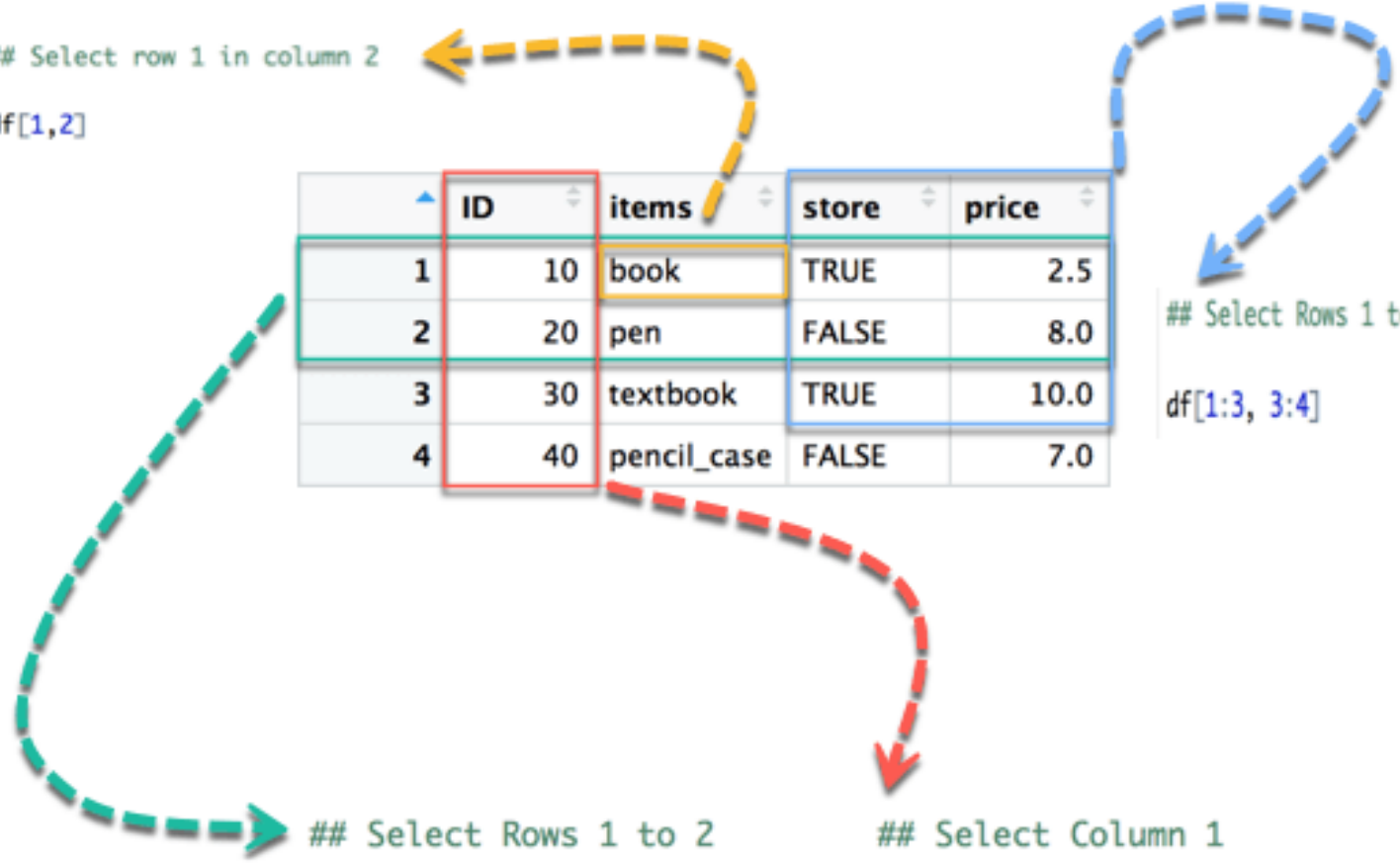


[www.ashutoshtripathi.com](http://www.ashutoshtripathi.com)

## data.frame 참조 방식

## Select row 1 in column 2

df[1,2]



	ID	items	store	price
1	10	book	TRUE	2.5
2	20	pen	FALSE	8.0
3	30	textbook	TRUE	10.0
4	40	pencil_case	FALSE	7.0

The diagram illustrates various indexing methods on a data frame table. A yellow dashed arrow points from the text '## Select row 1 in column 2' to the cell containing 'book' (row 1, column 2). A blue dashed arrow points from the text '## Select Rows 1 to 3 and columns 3 to 4' to the sub-table containing rows 1-3 and columns 3-4. A green dashed arrow points from the text '## Select Rows 1 to 2' to the sub-table containing rows 1-2. A red dashed arrow points from the text '## Select Column 1' to the column containing 'ID'.

## Select Rows 1 to 3 and columns 3 to 4

df[1:3, 3:4]

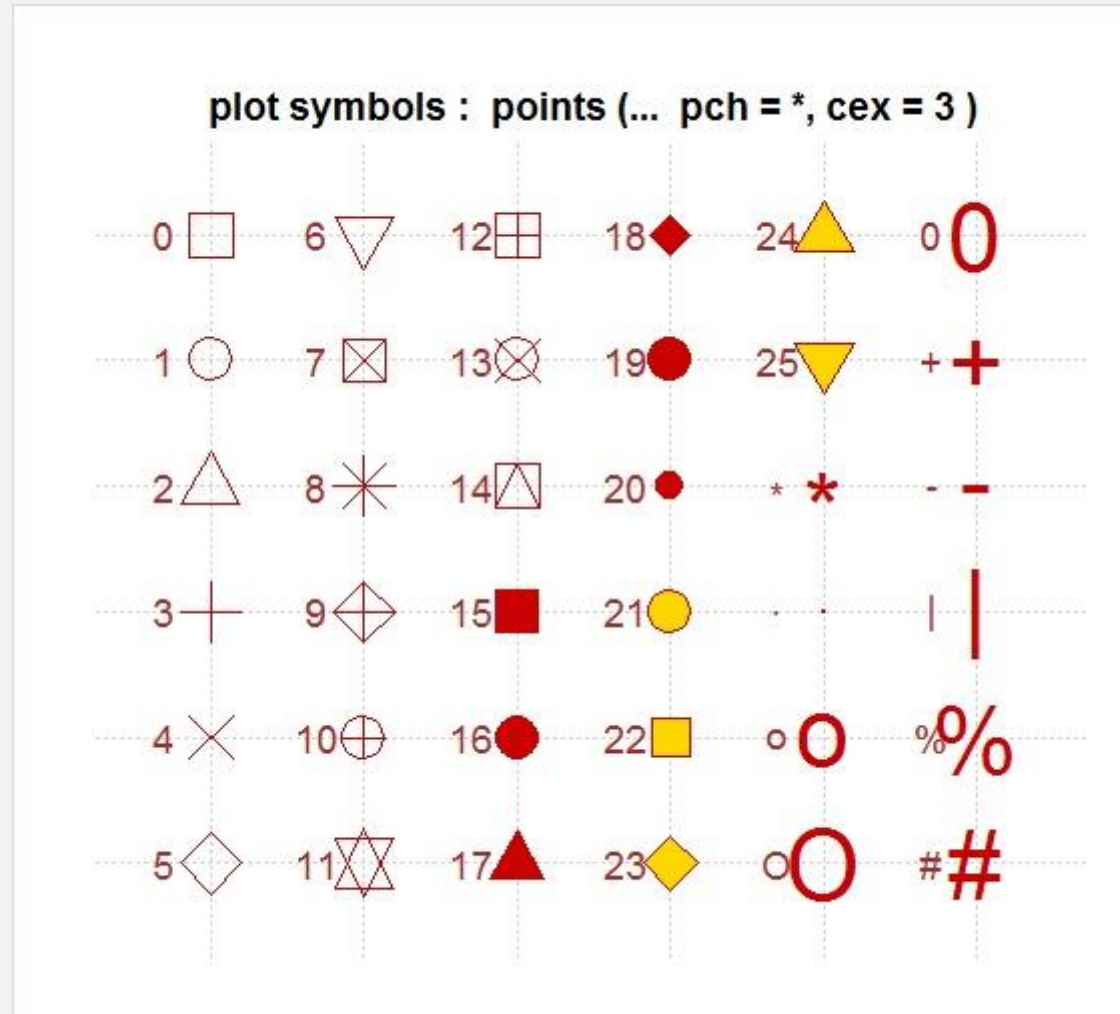
## Select Rows 1 to 2

df[1:2,]

## Select Column 1

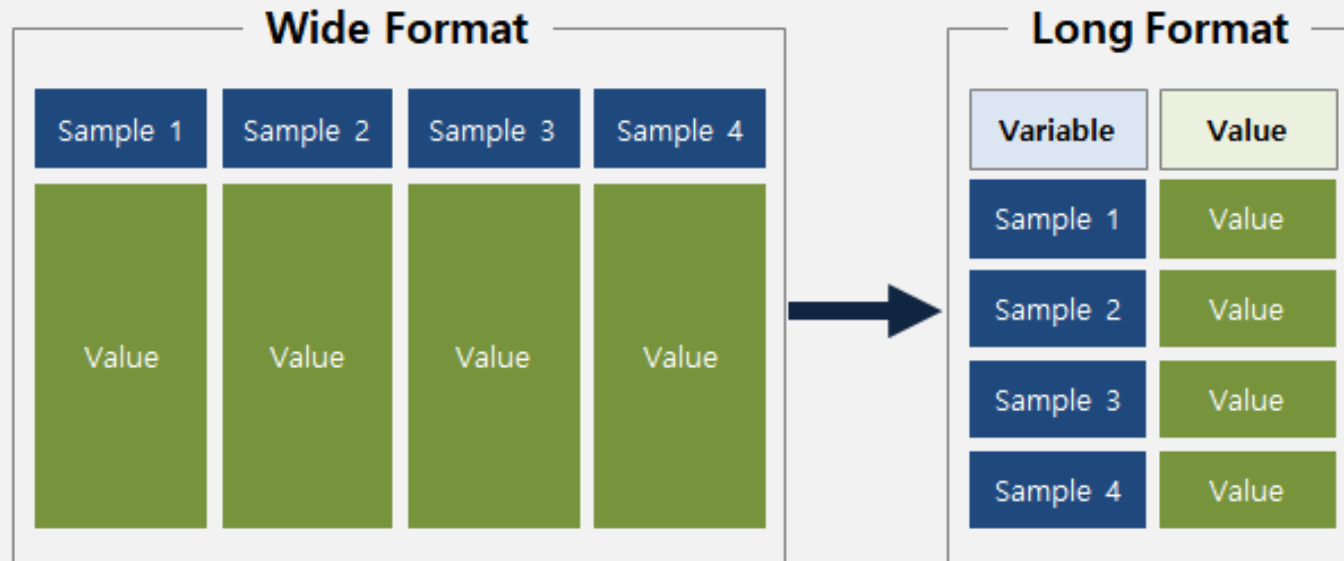
df[,1]

## 함수 `plot(... , pch = 21)`



## 함수 melt() cast() {reshape} (1)

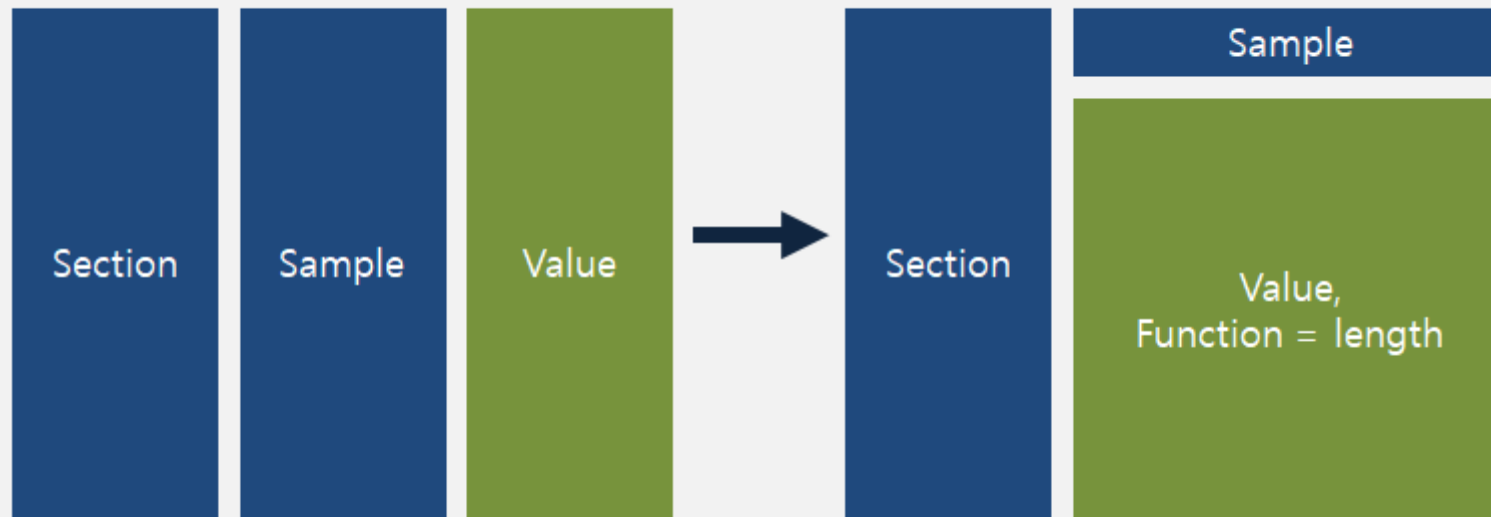
**melt(data)**



<https://chloe-with-data.tistory.com/>

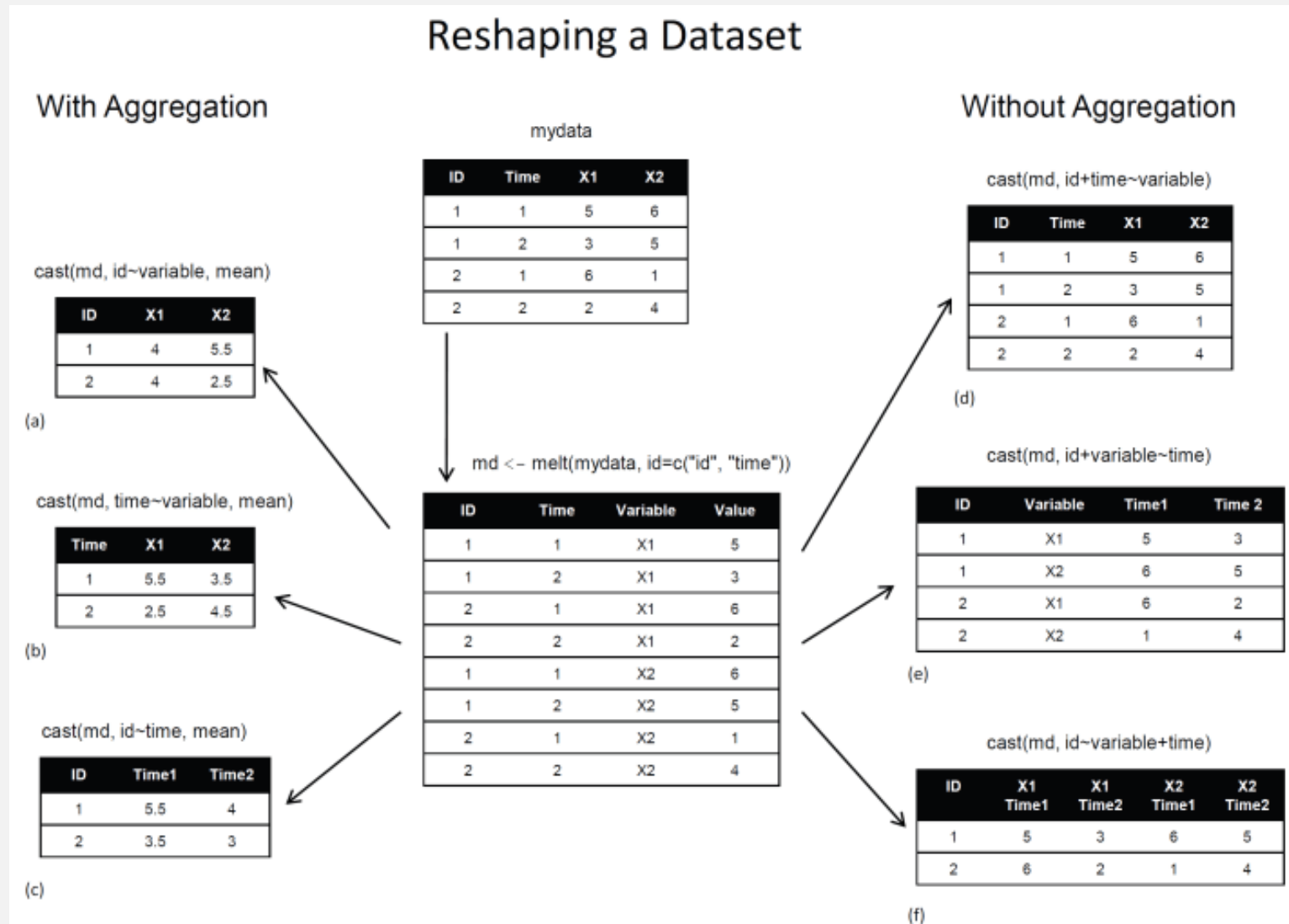
## 함수 melt() cast() {reshape} (2)

**cast(data, Section~Sample, fun)**



<https://chloe-with-data.tistory.com/>

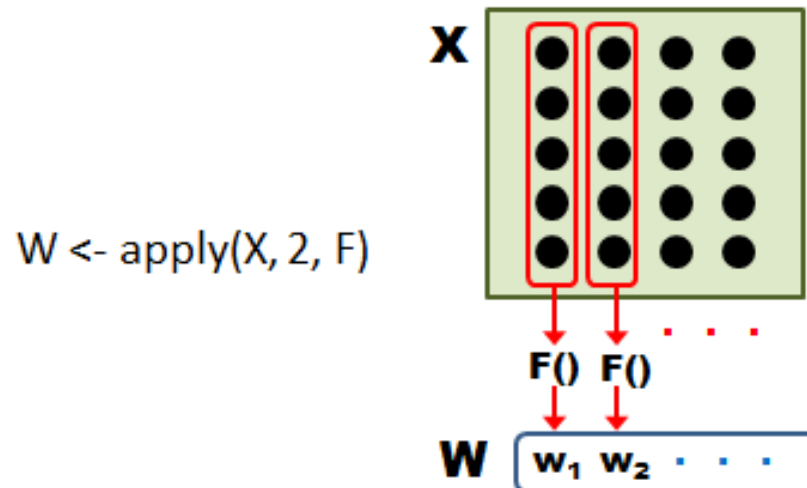
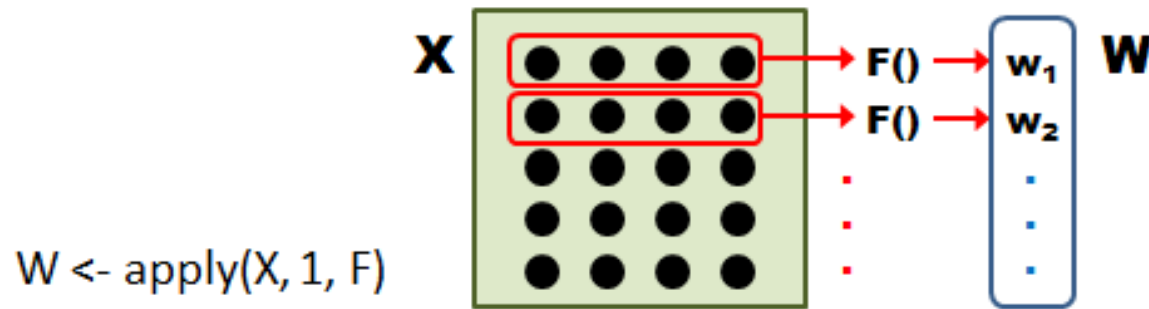
## 함수 melt() cast() {reshape} (3)



## 함수 `apply(X, margin = 1 | 2, func)`

for 루프를 사용하지 않고 다량 데이터의 일괄 처리를 간단하게 쓸 수 있는 `apply` 계의 함수

- 두번째 인자 `margin`: 1, 2





# ddply()

- ddply 인자

- [1] 대상이 되는 데이터 프레임
- [2] 그룹을 나눠서 계산하고 싶은 그룹명 변수
- [3] 그룹 전체에 수행하고자 하는 함수(예를 들면, summarise 또는 transform),
- [4] 그룹 내에 수행하고자 하는 함수(예를 들면, sum, max, min, mean 등))

## 데이터셋 freetrade {amelia}

1980년부터 1999년까지 아시아 9개 개발도상국에 대한 경제 및 정치 데이터

- 각 국가의 경제적 지표와 정치 체제의 특성을 나타내며, 주로 관세와 무역 관련 분석에 사용

- year: 데이터가 수집된 연도, 예: 1981, 1982, 1983, 등
  - 시계열 데이터
    - 일정 시간 간격으로 측정된 데이터를 의미하며, 일반적으로 시간에 따른 변화나 추세를 분석하는 데 사용
- country: 해당 데이터가 어떤 나라에 대한 것인지 나타냄, 예: SriLanka (스리랑카)
  - 교차 섹션 변수
    - 데이터가 어떤 단위를 기준으로 측정되었는지를 나타내며, 다중 대체 과정에서 각 단위별로 다른 패턴이나 특성을 고려하는 데 사용
- tariff: 관세율, 관세는 국가 간 무역에서 부과되는 세금, 예: 41.3, 31.0
- polity: Polity Score, 정치 체제의 민주성/비민주성을 평가하는 지표, -10에서 10까지의 값
- pop: 해당 연도의 국가 인구, 예: 14988000, 15189000
- gdp.pc: 1인당 GDP, 국가의 총생산을 인구수로 나눈 값, 예: 461.0236, 473.7634
- intresmi: 실질 이자율(Real Interest Rate), 명목 이자율에서 인플레이션율을 뺀 값, 예: 1.937347, 1.964430.
- signed: 자유무역 협정(Free Trade Agreement, FTA)이 체결 여부, 1은 체결, 0은 미체결
- fiveop: 경제적 개방도를 나타내는 지표, 특정 방법론을 통해 계산, 12.4, 12.5.
- usheg: 미국과의 경제 관계를 나타내는 지표, 예: 0.2593112, 0.2558008.

- 요약

- year: 연도, country: 국가 이름, tariff: 관세율, polity: 정치 체제의 민주성/비민주성 점수
- pop: 인구 수, gdp.pc: 1인당 GDP, intresmi: 실질 이자율, signed: 자유무역 협정 체결 여부
- fiveop: 경제적 개방도 지표, usheg: 미국과의 경제 관계 지표

# 함수 amelia() 결과 해석

## • 해석

- Message: Normal EM convergence
  - EM 알고리즘(Expectation-Maximization algorithm)이 정상적으로 수렴하였음을 나타냅니다.
- Chain Lengths
  - 각 대체 데이터셋을 생성하는 데 걸린 EM 알고리즘의 반복 횟수를 나타냅니다.
- Rows after Listwise Deletion: 96
  - 리스트와이즈 삭제(Listwise Deletion) 후 남은 행의 수는 96
  - 결측값이 있는 행을 완전히 제거하는 방법
- Rows after Imputation: 171
  - 대체 후의 행 수는 171
  - 원래 데이터셋에 있던 모든 행(결측값이 있거나 없는)을 포함
- Patterns of missingness in the data: 8
  - 데이터에서 결측값의 패턴은 8개
  - 결측값이 존재하는 변수들의 조합이 8가지라는 의미
- Fraction Missing for original variables
  - 원래 데이터셋의 각 변수에 대해 누락된 값의 비율

```
> #####
> # 대체된 데이터셋 확인
> summary(a.out)
```

Amelia output with 5 imputed datasets.  
Return code: 1  
Message: Normal EM convergence.

Chain Lengths:  
-----

```
Imputation 1: 13
Imputation 2: 18
Imputation 3: 21
Imputation 4: 14
Imputation 5: 13
```

Rows after Listwise Deletion: 96  
Rows after Imputation: 171  
Patterns of missingness in the data: 8

Fraction Missing for original variables:  
-----

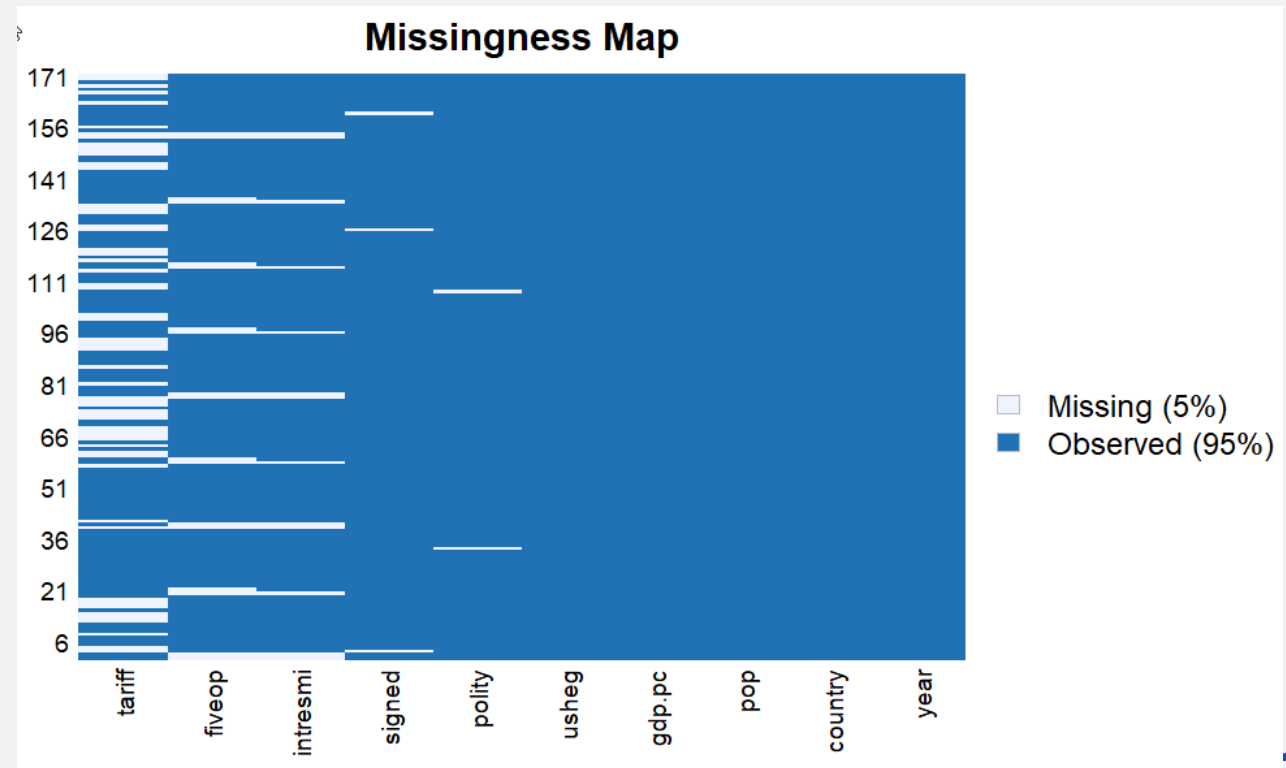
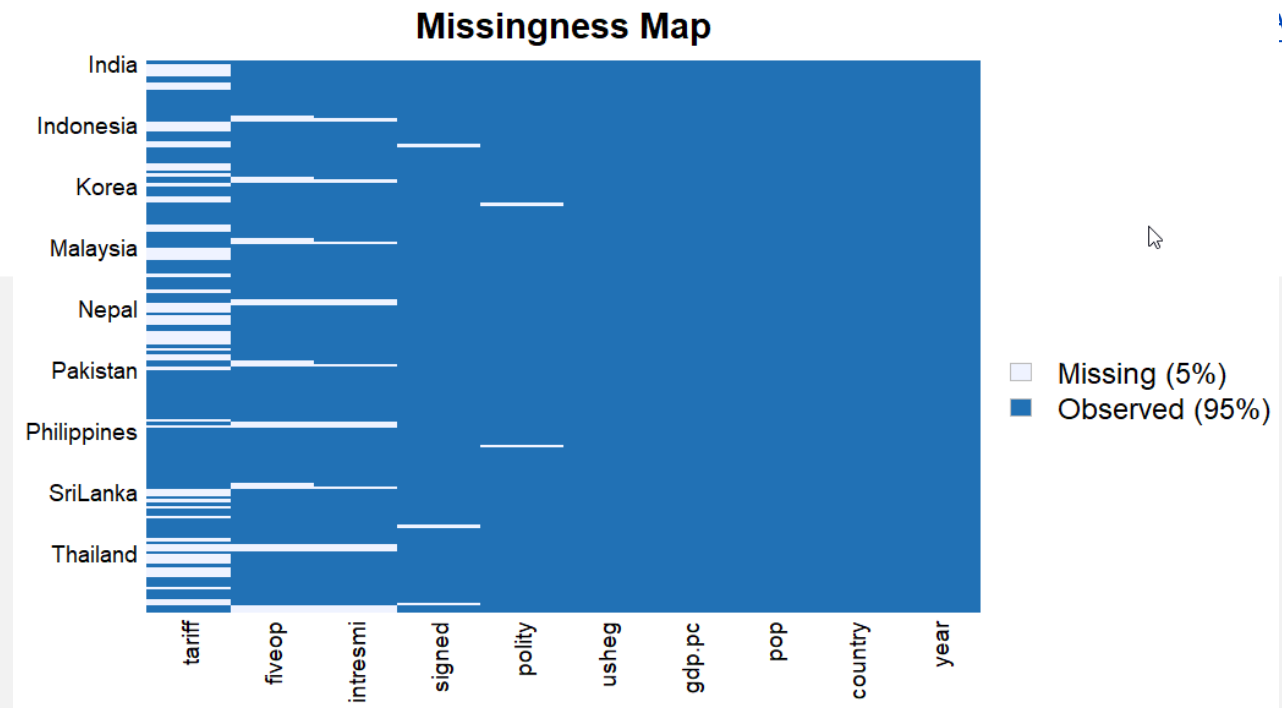
	Fraction Missing
year	0.00000000
country	0.00000000
tariff	0.33918129
polity	0.01169591
pop	0.00000000
gdp.pc	0.00000000
intresmi	0.07602339
signed	0.01754386
fiveop	0.10526316
usheg	0.00000000

```
> |
```

## 결측값 시각화

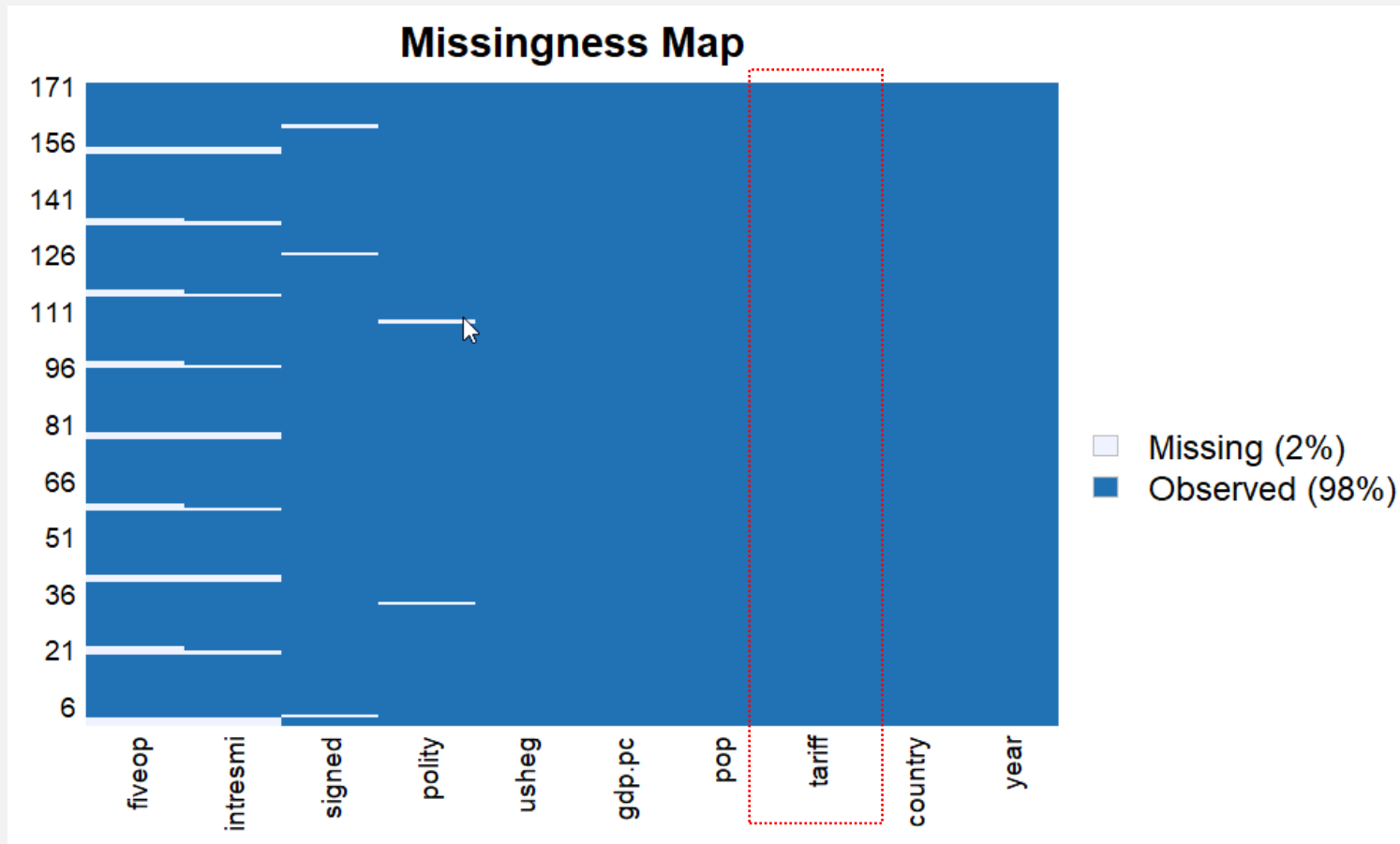
함수 `missmap()`

- > `missmap(a.out)`
- > `missmap(freetrade)`



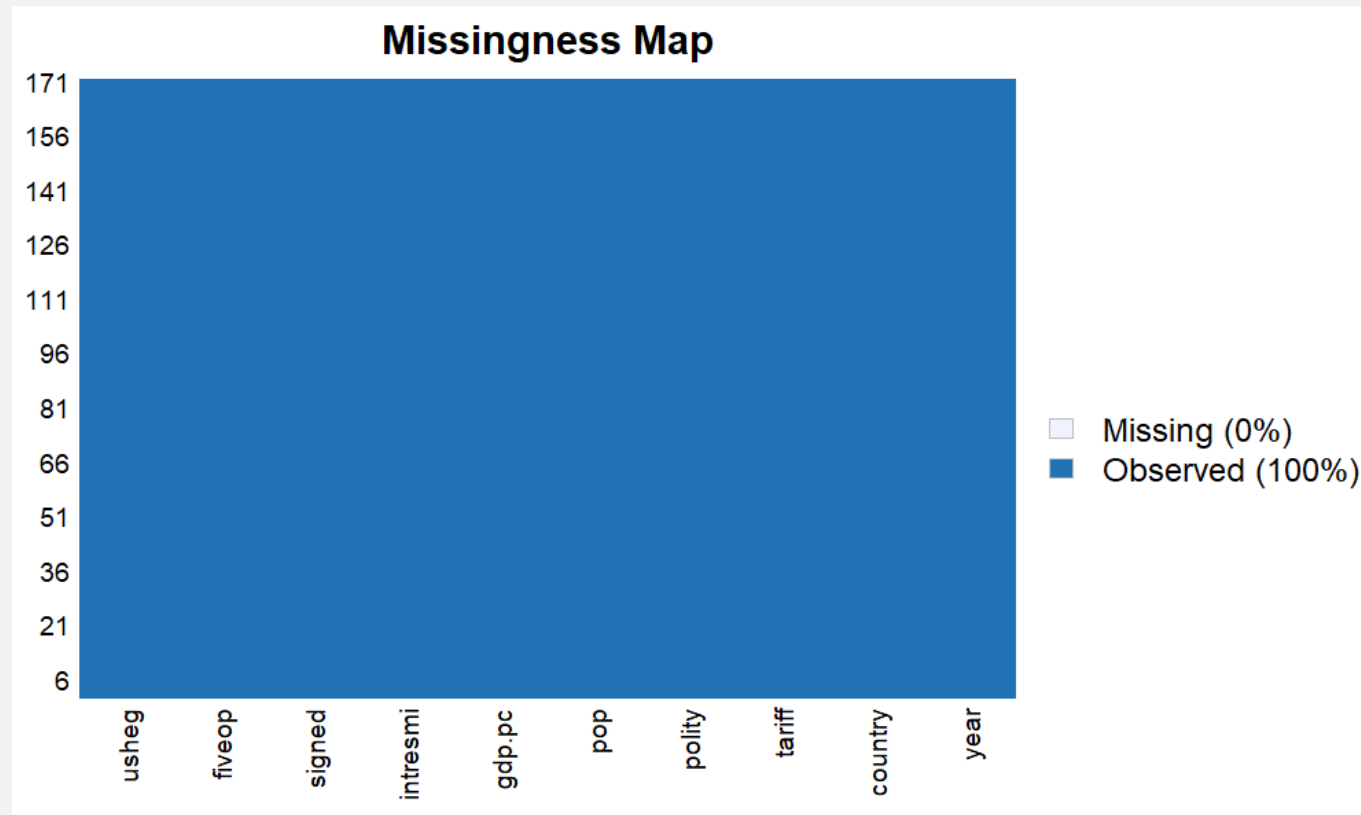
## 열 tariff 대체

- > freetrade\$tariff <- a.out\$imputation[[5]]\$tariff
- > missmap(freetrade)



## 모든 na 대체

- > freetrade.imputed <- a.out\$imputation[[5]]
- > missmap(freetrade.imputed)



# 이상값 검색

- 이상값의 발생 경우의 수

- 다음 a1,a2는 bad data

- a1 : 의도하지 않게 잘못 입력한 경우

- a2 : 의도하지 않게 입력했으나 분석 목적에 부합되지 않아 제거해야 할 경우

- POS로 수작업으로 일괄적인 거래정보 입력 시 시간대별 매출 분석에선 제외

- a3 : 의도하지 않으나 분석에 포함해야 하는 경우

- 고객행동 분석에는 a3에 포함되는데

- 이때 경우의 조합이 다양해져서 변수가 많아지면

- 표준 데이터 기준으로 작업 못하고 경우에 따라 데이터가 달라지므로 의사결정을 해야 함

- a3와 b1을 이상값

- b1 : 의도된 이상값

- 대부분 사기(fraud)

- 카드 분실 시 이상 사용

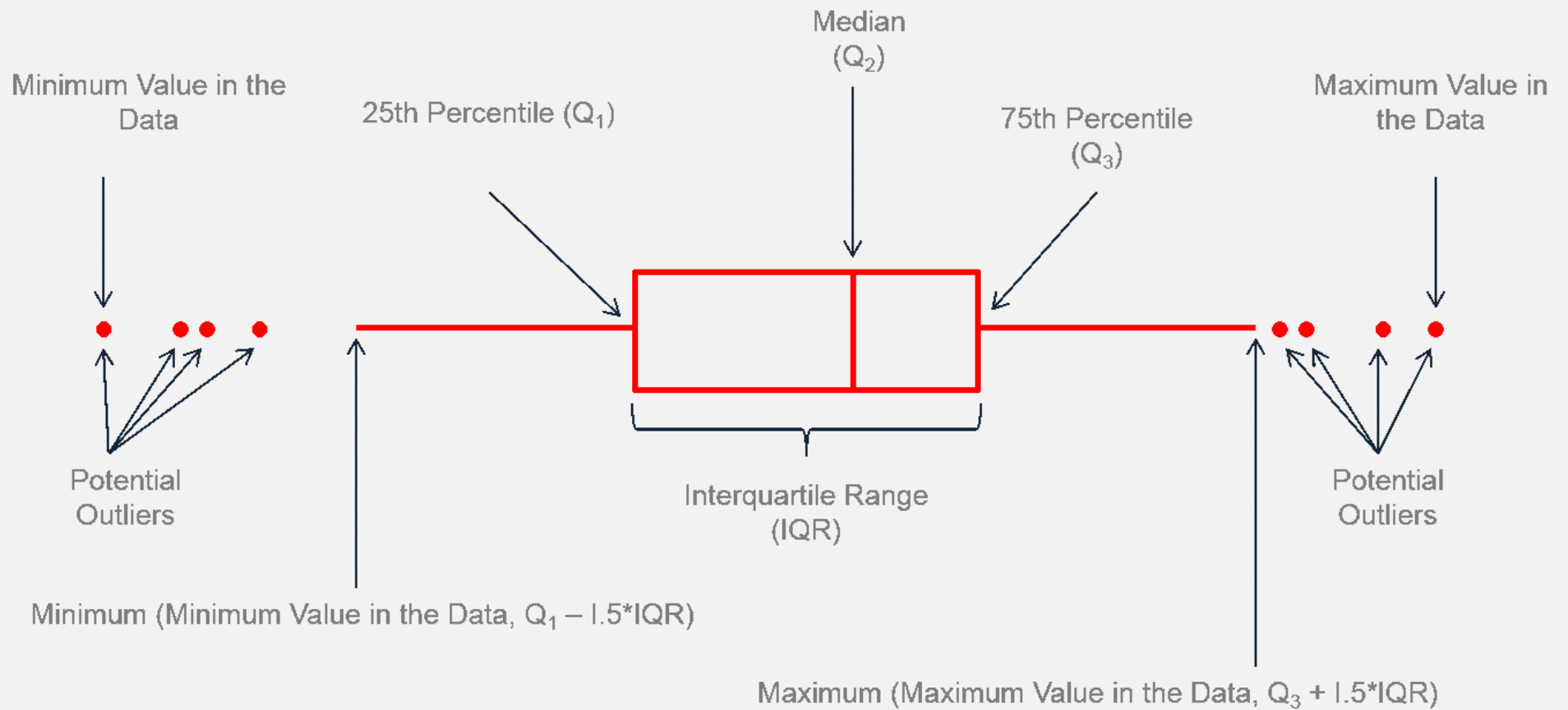
- 부정사용 방지 프로젝트

## 이상값 관련 알고리즘

- ESD(extreme studentized deviation), MADM(Median Absolute Deviation Method) 등이 있고 보통 ESD 사용
- ESD
  - 평균으로 부터  $k$ \*표준편차만큼 떨어져 있는 값들을 이상값으로 판단
    - $[\text{mean} - 3*sd, \text{mean} + 3*sd]$  이외의 값을 이상값으로 판단
  - 일반적으로  $k$ 는 3(그러나 이것도 매우 민감)
- 실제, 이상값 계산에 많은 시간 쓸 필요는 없음
- 단 부정사용 방지 프로젝트(fraud detection project)엔 많이 써야함
  - 보통 summary 정도로 평균과 중위수 파악해 1사분위, 3사분위 보고 1차 판단
  - 시간이 되면 주요 변수(dimension) 별로 플롯해보면서 특성 파악 가능.



## Boxplot() 이상값



# 데이터웨어하우스(Data Warehouse)와 데이터마트(Data Mart)

모두 데이터 분석과 비즈니스 인텔리전스를 위한 데이터 저장소

- 데이터웨어하우스 (Data Warehouse)

- 대규모 데이터를 중앙집중적으로 저장하고 관리하기 위한 시스템
- 주로 대기업에서 사용하며, 다양한 출처의 데이터를 통합하여 분석하고 의사결정을 지원

- 주요 특징

- 통합
  - 데이터웨어하우스는 다양한 운영 시스템(예: ERP, CRM 등)에서 데이터를 추출하여 통합
- 주제 지향적
  - 특정 비즈니스 주제(예: 판매, 고객, 재무 등) 중심으로 데이터가 구성
- 비휘발성
  - 데이터웨어하우스에 저장된 데이터는 변경되지 않고 읽기 전용
  - 데이터는 주기적으로 업데이트되지만 기존 데이터는 그대로 유지
- 시간 가변성
  - 시간에 따라 변하는 데이터를 저장
  - 예를 들어, 매출 데이터는 여러 시간 단위(일, 주, 월, 분기 등)로 저장
- 규모
  - 테라바이트(TB)에서 페타바이트(PB)까지 매우 큰 규모의 데이터를 저장 가능

## DW 아키텍처 구성 요소

- **ETL (Extract, Transform, Load):**
  - 데이터를 추출, 변환, 적재하는 프로세스
  - 다양한 소스에서 데이터를 가져와 데이터웨어하우스에 맞게 변환하고 적재
- **데이터 저장소**
  - 정제된 데이터를 저장하는 중앙 저장소
- **데이터 마트**
  - 특정 부서나 비즈니스 기능을 위한 서브셋 데이터베이스
- **OLAP (Online Analytical Processing)**
  - 다차원 분석을 가능하게 하는 시스템으로, 빠른 조회와 분석을 지원
- **BI 도구**
  - 비즈니스 인텔리전스 도구로, 데이터를 시각화하고 보고서를 작성하는 데 사용
    - 태블로, MS BI

# 데이터마트 (Data Mart)

- 정의
  - 데이터웨어하우스의 작은 부분집합으로
    - 특정 부서나 사용자 그룹의 요구를 충족시키기 위해 설계된 데이터 저장소
  - 데이터의 한 부분으로서
    - 특정 사용자가 관심을 갖는 데이터들을 담은 비교적 작은 규모의 데이터 웨어하우스
- 주요 특징
  - 소규모
    - 데이터웨어하우스에 비해 훨씬 작은 규모로, 기가바이트(GB)에서 수 테라바이트(TB) 정도의 데이터를 저장
  - 주제 중심
    - 특정 비즈니스 주제나 부서(예: 마케팅, 판매, 인사 등)에 맞춘 데이터를 저장
  - 신속성
    - 특정 사용자의 요구를 신속하게 지원하기 위해 데이터마트는 빠른 접근과 쿼리를 제공
  - 비용 효율성
    - 데이터웨어하우스보다 구축과 유지 비용이 저렴
- 유형
  - 독립형 데이터마트
    - 데이터웨어하우스와 독립적으로 운영되는 데이터마트로, 자체적인 ETL 프로세스를 통해 데이터를 가져옴
  - 의존형 데이터마트
    - 데이터웨어하우스에서 데이터를 추출하여 사용하는 데이터마트

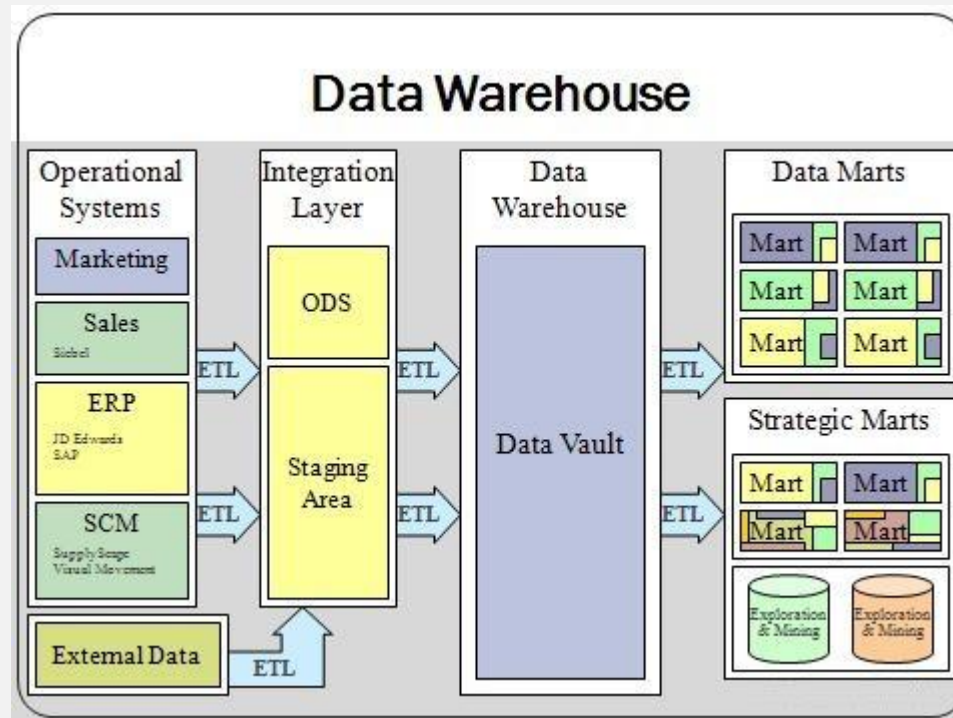
# 데이터웨어하우스와 데이터마트 비교

데이터웨어하우스와 데이터마트는 모두 비즈니스 인텔리전스와 데이터 분석을 위해 중요

- **범위**
  - 데이터웨어하우스는 기업 전체 데이터를 다루는 반면, 데이터마트는 특정 부서나 기능에 초점을 맞춤
- **크기**
  - 데이터웨어하우스는 매우 큰 규모의 데이터를 다루고, 데이터마트는 비교적 작은 규모의 데이터를 다룸
- **구축 비용**
  - 데이터웨어하우스는 구축과 유지 비용이 높지만, 데이터마트는 비용이 낮음
- **복잡성**
  - 데이터웨어하우스는 복잡한 데이터 통합과 관리가 필요하지만, 데이터마트는 비교적 간단
- **결론**
  - 데이터웨어하우스는 통합된 대규모 데이터를 제공하여 전사적인 의사결정을 지원
  - 데이터마트는 특정 부서나 사용자 그룹의 요구를 빠르게 충족
  - 두 시스템을 적절히 활용하면 효율적인 데이터 관리와 분석이 가능

## 데이터 마트, data mart (IT용어사전)

- 데이터의 한 부분으로서 특정 사용자가 관심을 갖는 데이터들을 담은 비교적 작은 규모의 데이터 웨어하우스
  - 즉 일반적인 데이터베이스 형태로 갖고 있는 다양한 정보를 사용자의 요구 항목에 따라 체계적으로 분석하여 기업의 경영 활동을 돕기 위한 시스템
  - 데이터 웨어하우스는 정부 기관 혹은 정부 전체의 상세 데이터를 포함하는데 비해
    - 데이터 마트는 전체적인 데이터 웨어하우스에 있는 일부 데이터를 가지고 특정 사용자를 대상으로 한다.
    - 데이터 웨어하우스와 데이터 마트의 구분은 사용자의 기능 및 제공 범위를 기준으로 한다.



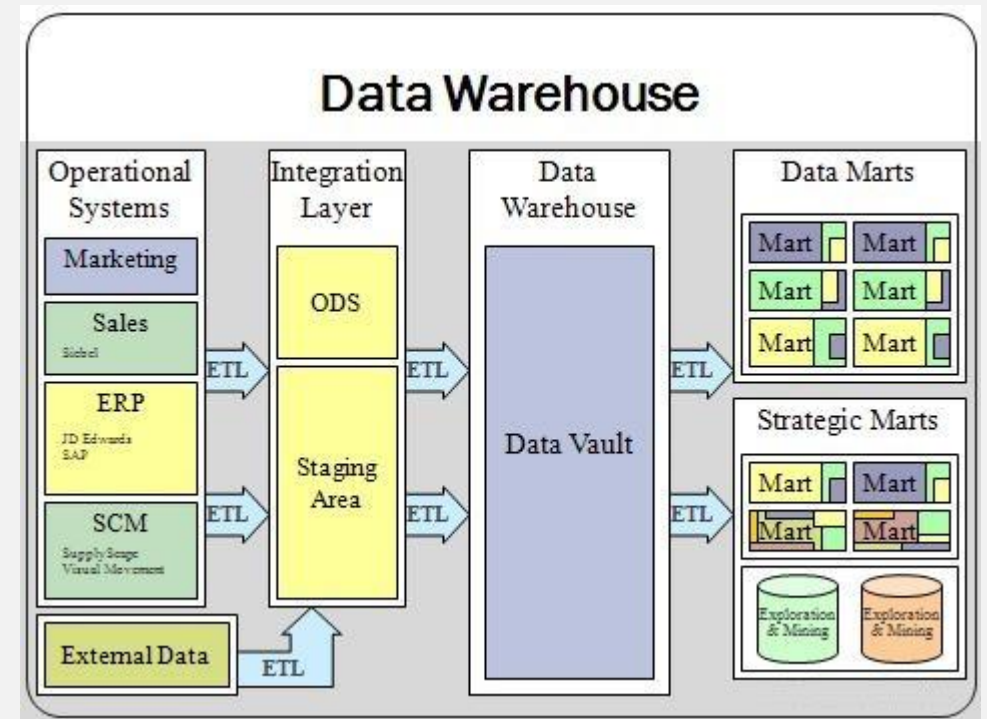
# ODS(Operational Data Store)

## • 운영 데이터 스토어

- Operational Data Store, ODS
- 기업의 운영 데이터를 통합하는 중간 저장소
- 주로 일상적인 비즈니스 운영을 지원하기 위해 실시간 데이터나 최근의 데이터를 저장하고 관리

## • ODS의 주요 특징

- 실시간 데이터 처리
  - ODS는 운영 데이터베이스에서 실시간으로 데이터를 가져오고 이를 중앙 저장소에 통합
- 데이터 통합
  - 다양한 소스에서 데이터를 수집하여 하나의 통합된 뷰를 제공
- 단기 저장소:
  - 최근 데이터를 저장하며, 장기 저장을 목적으로 하지 않음
- 운영 지원
  - 비즈니스 프로세스와 일상 운영을 지원하기 위해 사용



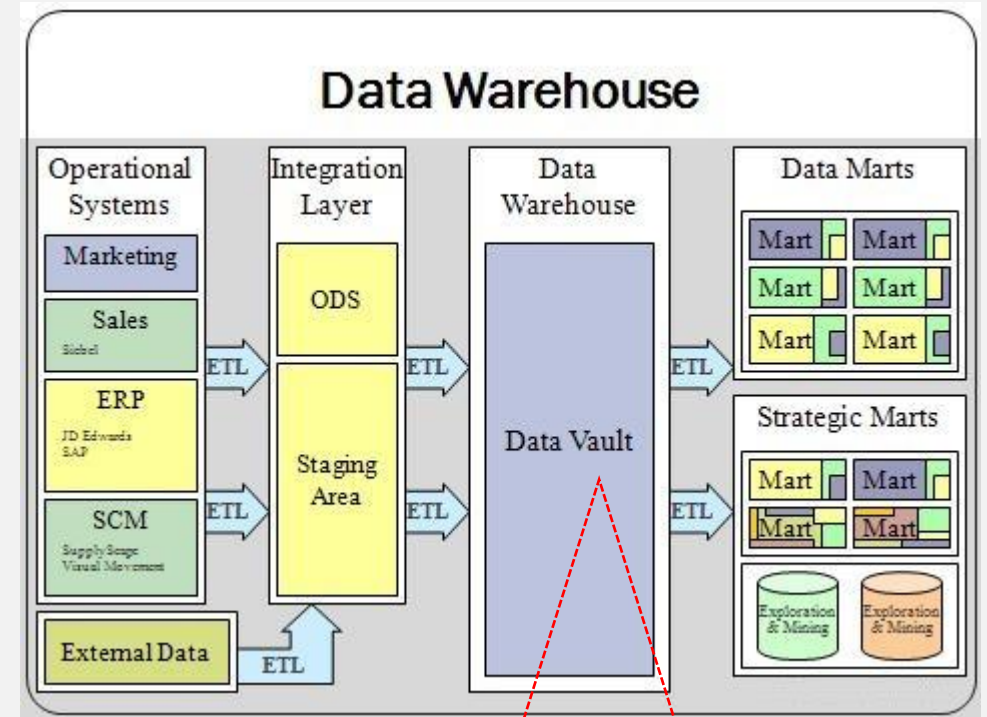
# Staging Area

## • 스테이징 영역(Staging Area)

- 데이터웨어하우스 환경에서 데이터를 변환하고 로드하기 전에 임시로 저장하는 영역
- 주요 목적은 데이터 통합 및 정제를 지원

## • 주요 특징

- 데이터 수집
  - 다양한 소스 시스템에서 데이터를 수집하여 임시로 저장
- 데이터 변환
  - 데이터를 정제하고 변환하여 데이터웨어하우스에 적합한 형태로 생성
- 데이터 로드:
  - 변환된 데이터를 데이터웨어하우스에 로드
- 임시 저장소
  - 데이터를 일시적으로 저장하며, 데이터웨어하우스에 로드되면 데이터를 삭제하는 경우가 많음



데이터웨어하우스 설계 방법론 중 하나로, 데이터를 추적하고 변화 관리를 쉽게 하기 위해 설계



# 수고하셨습니다!