

Data Frame

Data Frame

- ▶ **Data Frame**은 데이터를 나타내기에 가장 이상적인 객체
 - ▶ 여러 개의 **vector**를 열(**column**)로 이어놓은 형태
 - ▶ 이때, **Vector**의 길이는 모두 같아야 한다
 - ▶ 각 벡터의 자료형은 다를 수도 있음

X1 (character) ▾	mpg (double) ▾	cyl (integer) ▾	disp (double) ▾	hp (integer) ▾
Mazda RX4	21.0	6	160.0	110
Mazda RX4 Wag	21.0	6	160.0	110
Datsun 710	22.8	4	108.0	93
Hornet 4 Drive	21.4	6	258.0	110
Hornet Sportabout	18.7	8	360.0	175

5개 열과 5개 행으로 구성된 데이터 프레임의 예

Data Frame


: 열(Column)

▶ 열(Column) : 세로로 나열되는 데이터의 집합

▶ 변수(Variable) 혹은 특성(feature)을 나타냄

▶ '속성(attribute)' 이라 부르기도 함

▶ 셀(Cell) : 하나의 셀에는 하나의 값을 담는다




X1 (character) ▾	mpg (double) ▾	cyl (integer) ▾	disp (double) ▾	hp (integer) ▾
Mazda RX4	21.0	6	160.0	110
Mazda RX4 Wag	21.0	6	160.0	110
Datsun 710	22.8	4	108.0	93
Hornet 4 Drive	21.4	6	258.0	110
Hornet Sportabout	18.7	8	360.0	175

Data Frame

: 행(Rows)

- ▶ 행(Row) : 가로로 나열되는 데이터의 집합
 - ▶ 관측치(Observation)을 나타냄
 - ▶ 'Case' 라 부르기도 함
- ▶ 하나의 데이터는 여러 개의 속성을 가지고 있으며 이 속성들이 조합되어 하나의 데이터(관측치)를 식별하게 된다



X1 (character) ▾	mpg (double) ▾	cyl (integer) ▾	disp (double) ▾	hp (integer) ▾
Mazda RX4	21.0	6	160.0	110
Mazda RX4 Wag	21.0	6	160.0	110
Datsun 710	22.8	4	108.0	93
Hornet 4 Drive	21.4	6	230.0	110
Hornet Sportabout	18.7	8	360.0	175

Data Frame

: *tidy* 데이터

- ▶ 분석을 위한 데이터는 가능하면 깔끔하게 잘 정돈되어야 한다 (*tidy data*)
- ▶ 데이터가 *tidy* 하면 얻게 되는 장점
 - ▶ 일괄성 유지
 - ▶ 모든 *R* 함수에서 열(*column*) = 변수(*Variable*), 행(*row*) = 관측치(*observation*)라는 룰을 적용할 수 있음

X1 (character) ▾	mpg (double) ▾	cyl (integer) ▾	disp (double) ▾	hp (integer) ▾
Mazda RX4	21.0	6	160.0	110
Mazda RX4 Wag	21.0	6	160.0	110
Datsun 710	22.8	4	108.0	93
Hornet 4 Drive	21.4	6	258.0	110
Hornet Sportabout	18.7	8	360.0	175

Data Frame

: 데이터 프레임 만들기

▶ *data.frame()*을 이용

- ▶ 데이터 프레임을 구성할 변수를 괄호 안에 콤마(,)로 나열

```
> name <- c("홍길동", "전우치", "임꺽정", "장길산")
> height <- c(175.8, 170.2, 186.7, 188.3)
> weight <- c(73.2, 66.3, 88.2, 90)
> thieves <- data.frame(name, height, weight)
> thieves
```

	name	height	weight
1	홍길동	175.8	73.2
2	전우치	170.2	66.3
3	임꺽정	186.7	88.2
4	장길산	188.3	90.0

Data Frame

: 데이터 프레임의 활용

- ▶ **Vector**와 **Matrix**에서 사용했던 **indexing**, **slicing**, 통계 함수 등을 그대로 활용 가능
 - ▶ 단, 통계 함수의 경우, 수치 데이터만을 대상으로 하므로 적절한 **slicing**이 필요할 것
 - ▶ **\$** 기호를 이용, 데이터 프레임 내의 변수에 접근 가능

```
> colMeans(thieves[c("height", "weight")])
height weight
180.250  79.425
> colMeans(thieves[c(2, 3)])
height weight
180.250  79.425
```

```
> thieves
```

	name	height	weight
1	홍길동	175.8	73.2
2	전우치	170.2	66.3
3	임꺽정	186.7	88.2
4	장길산	188.3	90.0

```
> length(thieves)
```

```
[1] 3
```

```
> nrow(thieves)
```

```
[1] 4
```

```
> mean(thieves$height)
```

```
[1] 180.25
```

```
> mean(thieves$weight)
```

```
[1] 79.425
```

```
> colMeans(thieves[2:3])
```

	height	weight
180.250	79.425	



Data Frame

: 데이터 프레임 한번에 만들기

- ▶ **data.frame()** 함수 안에 변수 명과 값을 나열하여 한번에 만들 수도 있음

```
> thieves <- data.frame(name = c("홍길동", "전우치", "임꺽정", "장길산"),  
+                        height = c(175.8, 170.2, 186.7, 188.3),  
+                        weight = c(73.2, 66.3, 88.2, 90))  
>  
> thieves
```

	name	height	weight
1	홍길동	175.8	73.2
2	전우치	170.2	66.3
3	임꺽정	186.7	88.2
4	장길산	188.3	90.0

Data Frame

: 데이터 프레임에 새 컬럼 추가

- ▶ \$ 기호를 이용하여 새로운 컬럼을 추가할 수 있음

```
> thieves$bmi <- thieves$weight / (thieves$height / 100) ^ 2  
> thieves
```

	name	height	weight	bmi
1	홍길동	175.8	73.2	23.68500
2	전우치	170.2	66.3	22.88729
3	임꺽정	186.7	88.2	25.30346
4	장길산	188.3	90.0	25.38294

List

List

: 다 받아 주어라

▶ List

- ▶ R에서 가장 범용적인 자료 구조
- ▶ 모든 데이터 구조를 포함하는 데이터 구조. 여러 데이터 구조를 합해 하나의 리스트로 만들 수 있음
- ▶ *list* 함수를 이용하여 만들 수 있음

```
lst <- list(name = "홍길동", # 문자열  
            physical = c(175.8, 73.2), # 벡터  
            scores = data.frame(intellect = 95, health = 90)  
            # 데이터프레임  
)
```

```
> lst  
$name  
[1] "홍길동"  
  
$physical  
[1] 175.8  73.2  
  
$scores  
  intellect health  
1         95     90
```

List

: indexing

- ▶ 원소 구해오기 -> 리스트를 반환

리스트['원소명']

- ▶ 원소의 숫자 인덱스를 주어도 무방

- ▶ 원소 내 객체 얻어오기 -> 내부 객체 반환

리스트[['원소명']]

- ▶ 원소 내 객체를 얻기 위해서는 \$를 이용해도 된다

```
> lst['name']
```

```
$name
```

```
[1] "홍길동"
```

```
> lst[1]
```

```
$name
```

```
[1] "홍길동"
```

```
> lst[['name']]
```

```
[1] "홍길동"
```

```
> lst$name
```

```
[1] "홍길동"
```

```
> is(lst['name'])
```

```
[1] "list"      "vector"
```

```
> is(lst[['name']])
```

```
[1] "character"
```

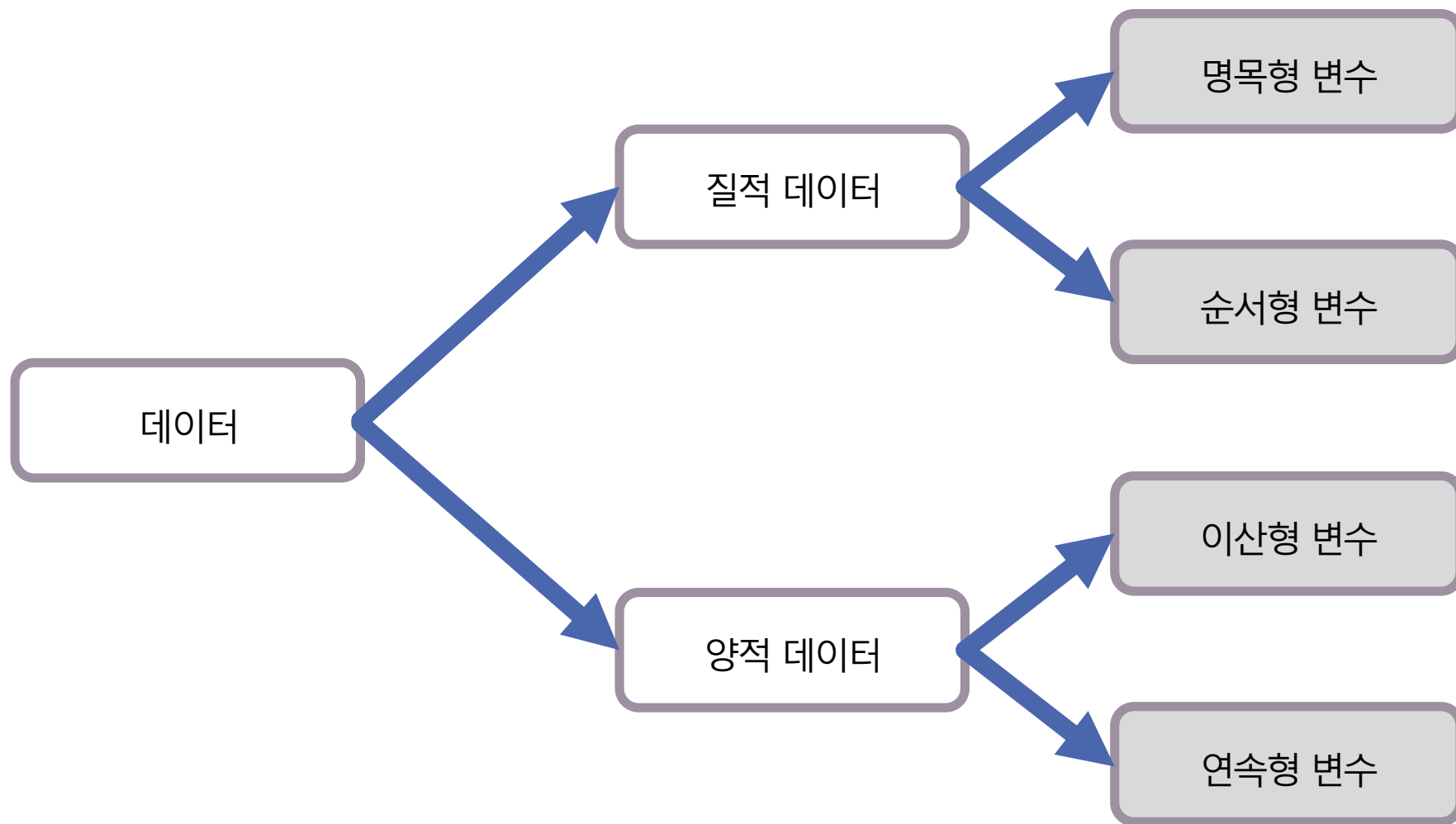
```
"data.frameRowLabels"
```

```
[4] "SuperClassMethod"
```

"vector"

*Data*의 종류와 *Factor*

*Data*의 종류



Data의 종류

명목형 변수

-> 성별(남, 여), 혈액형(A/B/O/AB) 등. 순서를 정할 수 없음

순서형 변수

-> 성적 (A/B/C/D/F) 등. 순서를 정할 수 있음

질적 변수

이산형 변수

-> 인원수(0, 1, 2, ...), 나이(13, 21, 37...) 등. (정수, 자연수)

연속형 변수

-> 키(176.3cm ...), 체중(73.30kg) 등. (실수)

양적 변수

Factor

- ▶ 질적 변수를 다루는 *R*의 자료형
- ▶ 대상을 분류하는 의미를 지닌 변수로 범주형 변수(*Categorical Variable*)이라 부르기도 함
- ▶ *factor* 형의 변수는 연산할 수 없음
- ▶ *factor()* 함수로 선언 가능
- ▶ 기존 변수를 *factor*로 변환할 때는 *as.factor()*를 이용

```
> var1 <- c(1, 2, 3, 2, 1)
> var2 <- factor(c(1, 2, 3, 2, 1))
> var1
[1] 1 2 3 2 1
> var2
[1] 1 2 3 2 1
Levels: 1 2 3
```

```
> var1 * 2 # numeric 변수 연산
[1] 2 4 6 4 2
> var2 * 2 # factor 변수 연산
[1] NA NA NA NA NA
```

Warning message:
In Ops.factor(var2, 2) : '*' not meaningful for factors

Factor

: 범주 구성 확인

- ▶ **levels()** 를 이용하면 **factor** 형 변수의 범주를 확인할 수 있음

- ▶ **factor** 형 변수가 아닌 경우, **NULL**을 반환

- ▶ **levels()**를 이용, **factor** 형 변수의 범주에 순서를 바꿀 수 있다

```
> sizes <- factor(c("medium", "small", "large", "huge", "small"))
> levels(sizes)
[1] "huge"    "large"   "medium"  "small"
> levels(sizes) <- c("small", "medium", "large", "huge")
> sizes
[1] large huge medium small huge
Levels: small medium large huge
> levels(sizes)
[1] "small" "medium" "large" "huge"
```

Ordered Factor

- ▶ *factor()* 대신 *ordered()*를 이용하면 순서형 변수를 만들 수 있음
- ▶ *ordered*로 만들어진 범주형 변수는 대소를 비교할 수 있게 된다

```
> sizes <- ordered(c("medium", "small", "large", "huge", "small"),  
+                  levels = c("small", "medium", "large", "huge"))  
> sizes  
[1] medium small large huge small  
Levels: small < medium < large < huge  
> sizes > "medium"  
[1] FALSE FALSE TRUE TRUE FALSE  
> sizes[sizes > "medium"] # 순서형이므로 대소를 비교할 수 있음  
[1] large huge  
Levels: small < medium < large < huge
```