



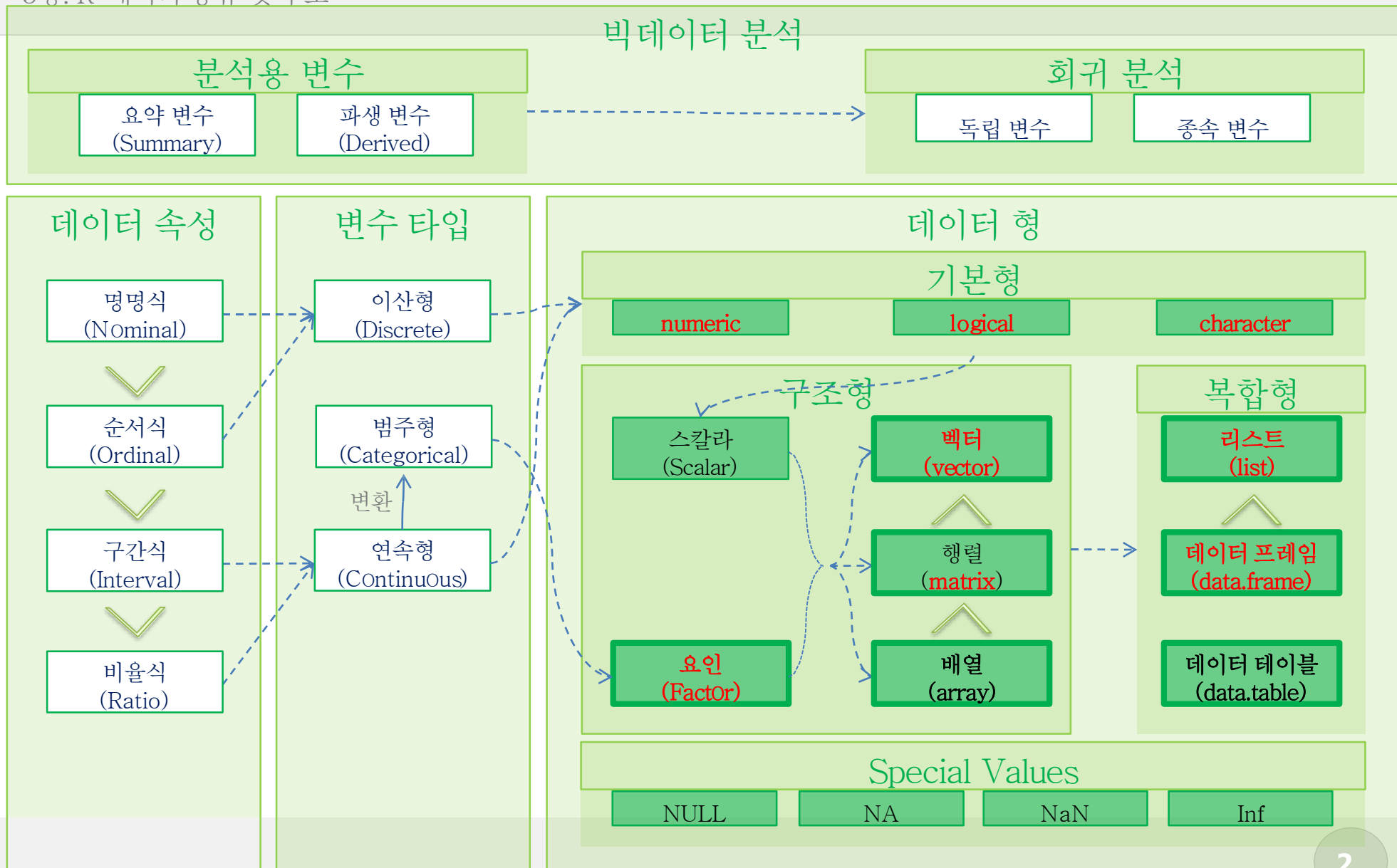
---



## 3장. R 데이터 종류 및 구조

# R 데이터 종류 및 구조의 이해

## 3장. R 데이터 종류 및 구조



# 1. 데이터 종류

## 3장. R 데이터 종류 및 구조

R은 데이터 속성에 따라 명명식, 순서식, 구간식, 비율식으로 데이터를 구분합니다. 명명식(Nominal)은 명목척도를 나타내며 이름으로 명명되는 자료를 의미합니다. 예를 들면 성별(남,녀)이 명명식에 해당합니다. 순서식(Ordinal)은 서열척도를 나타내며 순서가 있는 명명식을 의미합니다. 예를 들면 상, 중, 하 값 중 하나를 갖는 소득 데이터는 순서식에 해당합니다. 구간식(Interval)은 간격척도를 나타내며 순서의 간격을 측정할 수 있는 순서식을 의미합니다. 예를 들면 온도가 구간식에 해당합니다. 비율식(Ratio)은 비율척도를 나타내며 절대 영점이 존재해 비율이 의미 있는 구간식에 해당합니다. 예를 들면 체중이 비율식에 해당합니다.

R은 변수 타입에 따라 연속형, 이산형, 범주형 변수로 구분합니다. 연속형 변수(Continuous)는 체중처럼 연속적인 값을 갖는 변수입니다. 이산형 변수(Discrete)는 동전의 앞, 뒤처럼 값의 개수가 정의된 변수입니다. 범주형 변수(Categorical)는 연령대 10대, 20대, 등처럼 연속형 변수를 구간으로 묶어서 이산형 변수로 만든 것입니다.

# 데이터 종류

## 3장. R 데이터 종류 및 구조

R은 데이터의 품질에 따라 특이값(Outlier)과 결측값(Missing)으로 구분합니다. 특이값(Outlier)은 정상적이지 않은 데이터를 의미합니다. 잘못 측정 된 값이나 오차에의 한 값일 확률이 높습니다. 특이값은 데이터 분석 처리 시 제거하고 처리해야 할 수 있습니다. 결측값(Missing)은 아직 측정되지 않은 값입니다. R에서는 NA로 표기합니다.

R에서 분석용 변수는 요약 변수와 파생 변수로 구분합니다. 요약 변수(Summary Variables)는 데이터 분석을 위해서 1차 가공한 변수를 의미합니다. 요약 변수는 연속형 변수를 구간화하여 가공했거나, 시계열 데이터를 기간별, 요일별, 주중/주말별로 가공한 데이터입니다. 그 외에서 트랜트의 증가액 또는 증감비율, 그리고 SNS 데이터에서 단어의 빈도수 등이 요약변수에 해당합니다. 고객의 나이를 구간화하여 연령대로 가공했다면 이는 요약변수입니다. 파생 변수(Derived Variables)는 분석자의 판단에 따라 특정 조건을 만족하는 변수입니다. 예를 들면 고객에게 상품을 추천하기 위한 시스템의 경우 고객이 구매할 가능성이 있는 상품을 찾기 위해 가중치를 부여하는 변수<sup>1)</sup>들입니다. 파생변수의 예에는 근무시간 구매지수, 주 구매 매장, 주 활동 지역, 주 구매 상품(10개), 가격 선호대, 시즌 선호 고객, 행사 민감도, 고객 생애 가치(CLV) 등이 있습니다.

## 2. R 기본 데이터타입

### 3장. R 데이터 종류 및 구조

R은 문자(character), 숫자(numeric), 논리(logical)의 3가지 기본적인 데이터타입을 제공합니다.

- 문자형은 “(쌍 따옴표) 또는 ‘(홀 따옴표)로 묶으며, 문자와 문자열의 구분이 없습니다.
- 숫자형은 정수형과 부동소수점(실수)형 구분이 없습니다. 기본값은 0입니다.
- 논리형은 참은 TRUE, 거짓을 FALSE 값을 갖습니다. 논리형의 값은 전역변수인 T 또는 F 값을 가질 수 있습니다. 기본값은 FALSE입니다.

```
> a <- "Hello"
> is.character(a)
[1] TRUE

> b <- 10
> is.numeric(b)
[1] TRUE

> c <- TRUE
> is.logical(c)
[1] TRUE
```

values	
a	"Hello"
b	10
c	TRUE

# 타입 및 구조 확인

## 3장. R 데이터 종류 및 구조

`class()` 함수는 타입을 확인합니다.

```
> class(a)
[1] "character"
> class(b)
[1] "numeric"
> class(c)
[1] "logical"
```

`str()` 함수는 R 객체의 내부 구조를 간결하게 표시합니다.

```
> str(a)
chr "Hello"
> str(b)
num 10
> str(c)
logi TRUE
```

### 3.Special Values : 기본형 외에 특별한 의미로 사용되는 예약어

3장. R 데이터 종류 및 구조

Special Values	상세
NULL	Empty value로 값이 없음을 의미합니다. <code>is.null(변수)</code> 함수를 사용하여 변수가 가진 값이 NULL인지 확인합니다.
NA	Not Available Missing value(결측치)를 의미합니다. 값이 없다는 것이 아니라 측정이 되지 않아 값이 무엇인지 모른다는 의미로 사용합니다. <code>is.na(변수)</code> 함수를 사용하여 결측치 여부를 확인합니다. <code>mean(변수, na.rm=TRUE)</code> : 함수의 <code>na.rm</code> 파라미터라 TRUE이면 NA 값은 계산에서 제외합니다.
NaN	Not a Number 변수의 값이 숫자가 아니라는 것을 의미합니다. <code>is.nan(변수)</code> 함수를 사용하여 NaN 여부를 확인합니다.
Inf	Infinite number 무제한으로 큰 값을 의미합니다. <code>is.finite(변수)</code> 함수를 사용하여 Inf 여부를 확인합니다.



## 4. Factor

3장. R 데이터 종류 및 구조

팩터(Factor, 요인)는 범주형(Categorical) 변수를 의미합니다. 팩터는 미리 정해진 여러 개의 값 중 하나의 값을 가집니다. 팩터는 명명식(Nominal) 또는 순서식(Ordinal) 데이터를 저장합니다.

다음 코드는 명명식 데이터를 저장하는 팩터 변수를 선언하는 예입니다. `nlevels()` 함수는 범주의 수를 출력하고, `levels()` 함수는 범주의 목록을 출력합니다.

```
> gender <- factor(c("남","남","여","남","여"), levels=c("남","여"))
> gender
[1] 남 남 여 남 여
Levels: 남 여

> nlevels(gender)    # 범주의 수
[1] 2

> levels(gender)     # 범주의 목록
[1] "남" "여"

> class(gender)
[1] "factor"

> str(gender)
Factor w/ 2 levels "남","여": 1 1 2 1 2
```



# Factor

## 3장. R 데이터 종류 및 구조

다음 코드는 순서식 데이터를 저장하는 팩터 변수를 선언하는 예입니다. 순서식 데이터는 `factor()` 함수의 `ordered` 속성을 이용하거나 `ordered()` 함수를 이용하여 선언할 수 있습니다.

```
> gender <- ordered(c("남","남","여","남","여"), levels=c("남","여"))
> gender
[1] 남 남 여 남 여
Levels: 남 < 여

> nlevels(gender)
[1] 2
> levels(gender)
[1] "남" "여"
> class(gender)
[1] "ordered" "factor"
> str(gender)
Ord.factor w/ 2 levels "남"<"여": 1 1 2 1 2
```

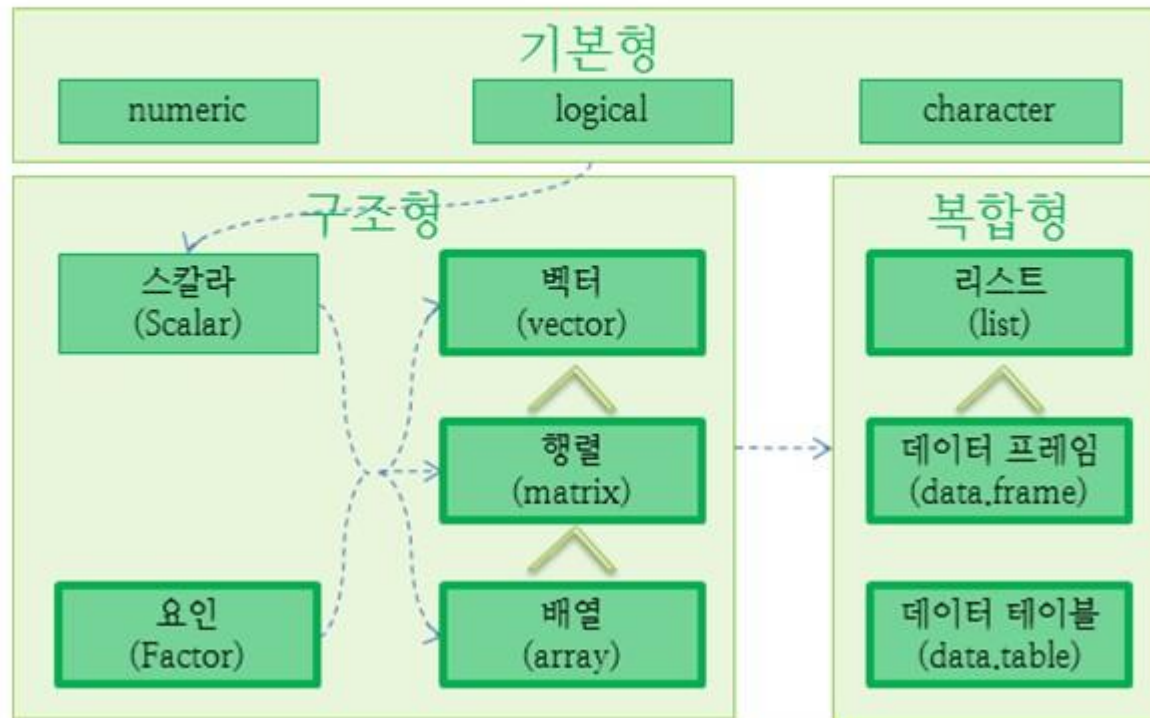
`ordered()` 함수를 이용하는 것과 `factor()` 함수의 `ordered` 속성을 이용하여 순서식 데이터를 저장하는 결과는 같습니다.

```
> gender <- factor(c("남","남","여","남","여"), levels=c("남","여"), ordered=TRUE)
```

# 5. 구조형 변수와 복합형 변수

## 3장. R 데이터 종류 및 구조

R은 변수가 한 가지 데이터타입 값만 가질 수 있는 구조형 변수와, 서로 다른 데이터타입 값을 가질 수 있는 복합형 변수로 나뉩니다.



# 구조형 변수와 복합형 변수

구분	구조형 변수	복합형 변수
정의	동일한 데이터타입의 데이터만 저장할 수 있는 변수	서로 다른 데이터타입의 데이터를 저장할 수 있는 변수
	스칼라(scalar) : 기본형 데이터를 저장하는 변수 요인(factor) : 여러 개의 값을 가질 수 있는 범주형(Categorical) 변수	
1차원 (배열)	벡터(vector)	리스트(list)
2차원 (배열)	행렬(matrix)	데이터 프레임(data.frame) 데이터 테이블(data.table)
n차원	배열(Array)	



# Scalar (스칼라)와 Factor (요인)

3장. R 데이터 종류 및 구조

- Scalar (스칼라)
  - 하나의 기본형 데이터를 가지는 변수입니다.
  - `data <- "Scalar Data"`
- Factor (요인)
  - 범주형 (Categorical) 변수로, 미리 정해진 여러 개의 값 중 하나의 값을 가집니다.
  - 명명식(Nominal) 또는 순서식(Ordinal) 데이터를 저장합니다.
  - `sex <- factor("남", levels=c("남", "여"))`
  - `nlevels(sex)` : 범주의 수
  - `levels(sex)` : 범주의 목록
  - `sex[1]` : `sex` 변수에 저장된 첫 번째 항목

# 6. Vector (벡터)

## 3장. R 데이터 종류 및 구조

벡터(Vector)는 여러 개의 동일한 형태의 데이터를 모아서 함께 저장되는 세트 또는 집합입니다. R에서 가장 많이 사용하는 데이터 구조입니다.

- 벡터는 `c()` 함수 안에 원하는 인자들을 나열하여 정의합니다(`c`는 Combined Value의 약어).
- 나열하는 인자들은 한 가지 유형의 스칼라 타입이어야 합니다. 문자, 숫자, 논리 타입이 섞여 있을 경우 문자타입으로 자동 형변환 됩니다. 만일 숫자와 논리타입이 섞여 있을 경우 논리값 `TRUE`는 숫자 1로, 논리값 `FALSE`는 숫자 0으로 형변환 됩니다.
- 벡터 내 데이터 접근은 `[ ]` 안에 색인 요소를 이용합니다. **색인이 음수일 경우 해당 데이터를 제외합니다. R의 색인은 1부터 시작합니다.**
- 벡터의 길이는 `length()` 또는 `NROW()`를 이용하여 알 수 있습니다.
- `%in%` 연산자는 어떤 값이 벡터에 포함되어 있는지를 알려줍니다.

# Vector (벡터)

## 3장. R 데이터 종류 및 구조

```
> data <- c(1, 2, 3)
> NROW(data)          # 항목의 개수를 출력합니다. length(data)와 같습니다.
[1] 3
> names(data) <- c("colA", "colB", "colC")    # 각 항목에 이름을 지정합니다.
> data["colA"]         # 이름으로 데이터 반환합니다.
colA
  1
> data[2]              # 두 번째 값을 반환합니다.
colB
  2
> data[-1]             # 첫 번째 값을 제외하고 반환합니다.
colB colC
  2    3
> data[c(1, 3)]        # 첫 번째와 세 번째 값 반환합니다.
colA colC
  1    3
> data[data > 2]       # 2보다 큰 값만 반환합니다.
colC
  3
> data[c(FALSE, FALSE, TRUE)] # TRUE로 표시된 순서의 데이터를 반환합니다.
colC
  3
> 1 %in% data          # data에 1이 포함되어 있으면 TRUE를 반환합니다.
[1] TRUE
```

# character()

## 3장. R 데이터 종류 및 구조

character() 함수는 문자열을 저장하는 벡터를 생성합니다.

```
data <- character(항목의수)
```

다음 코드는 문자열을 저장할 수 있는 벡터를 생성하고 값을 대입하는 예입니다.<sup>14)</sup>

```
> charArr <- character()
> charArr
character(0)
> charArr[1] <- "Hello"; charArr[2] <- "Nice"; charArr[3] <- "Good"
> charArr
[1] "Hello" "Nice"  "Good"
```



# numeric()

3장. R 데이터 종류 및 구조

numeric() 함수는 숫자를 저장하는 벡터를 생성합니다.

```
data <- numeric(항목의수)
```

다음 코드는 숫자를 저장할 수 있는 벡터를 생성하고 값을 대입하는 예입니다.

```
> intArr <- numeric()  
> intArr[1] <- 10; intArr[2] <- 20; intArr[3] <- 30;  
> intArr  
[1] 10 20 30
```

# logical()

3장. R 데이터 종류 및 구조

logical() 함수는 논리값을 저장하는 벡터를 생성합니다.

```
data <- logical(항목의수)
```

다음 코드는 논리값을 저장할 수 있는 벡터를 생성하고 값을 대입하는 예입니다.

```
> logicArr <- logical()
> logicArr[1] <- TRUE; logicArr[2] <- T; logicArr[3] <- FALSE
> logicArr
[1] TRUE TRUE FALSE
```

# 벡터의 결합

## 3장. R 데이터 종류 및 구조

c() 함수에 의해 여러 벡터를 결합하여 하나의 벡터로 만들 수 있습니다. 이 경우에는 각 벡터들의 데이터 타입이 같아야 하며, 만일 결합되는 벡터들의 타입이 다를 경우 문자>숫자>논리 순으로 형변환 됩니다.

```
> a <- c(1,2,3)
> b <- c("Hello", "World")
> c <- c(TRUE, FALSE, TRUE, TRUE)
> (z <- c(a,b,c))
[1] "1"      "2"      "3"      "Hello" "World" "TRUE"  "FALSE" "TRUE"  "TRUE"
```

Values	
a	num [1:3] 1 2 3
b	chr [1:2] "Hello" "World"
c	logi [1:4] TRUE FALSE TRUE TRUE
z	chr [1:9] "1" "2" "3" "Hello" "World" "TRUE" "FALSE" "TRUE"...

# append, union, intersect, setdiff, setequal

3장. R 데이터 종류 및 구조

append() 함수를 이용하여 벡터에 새로운 벡터를 추가할 수 있습니다.

```
> a <- c(1,2,3)
> append(a, c(4,5,6))
[1] 1 2 3 4 5 6
```

union(~, ~), intersect(~, ~), setdiff(~, ~)는 각각 Vector간 합집합, 교집합, 차집합을 구합니다.

```
> a <- c(1,2,3,4,5,6)
> b <- c(2,4,6,8,10,12)
> union(a, b)
[1] 1 2 3 4 5 6 8 10 12
> intersect(a, b)
[1] 2 4 6
> setdiff(a, b)
[1] 1 3 5
```

setequal(~, ~)은 Vector를 비교합니다. 만일 두 벡터가 동일하면 TRUE를 리턴합니다.

```
> setequal(a, b)
[1] FALSE
> setequal(a, c(intersect(a, b), setdiff(a, b)))
[1] TRUE
```

# 시퀀스

## 3장. R 데이터 종류 및 구조

규칙적인 시퀀스를 생성합니다. `seq()`는 시퀀스를 생성하는 기본함수입니다.  
구문에서 `by`는 시퀀스의 증가값입니다.

```
seq(from=1, to=1, by=((to-from)/(length.out-1)),  
     length.out=NULL, along.with=NULL, ...)
```

# 시퀀스

## 3장. R 데이터 종류 및 구조

```
> seq(0, 1, length.out=11)
[1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
> seq(stats::rnorm(10)) # effectively 'along'
[1] 1 2 3 4 5 6 7 8 9 10
> seq(1, 9, by=2)      # matches 'end'
[1] 1 3 5 7 9
> seq(1, 9, by=pi)     # stays below 'end'
[1] 1.000000 4.141593 7.283185
> seq(1, 6, by=3)
[1] 1 4
> seq(1.50, 5.15, by=0.5)
[1] 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
> seq(10)
[1] 1 2 3 4 5 6 7 8 9 10
```

# 리피트

## 3장. R 데이터 종류 및 구조

`rep()` 함수는 `x` 데이터를 반복 출력하는 기본 함수입니다. `rep.int()` 및 `rep_len()` 함수는 두 가지 경우에 더 빠른 함수입니다.

```
rep(x, times=1, length.out=NA, each=1)
```

```
rep.int(x, times)
```

```
rep_len(x, length.out)
```

구문에서...

- `x` : 반복 출력할 데이터입니다.
- `times` : 반복 출력할 횟수입니다.
- `each` : `each`가 지정되면 각 항목이 `each` 만큼 복제된 다음 `times` 또는 `length.out` 만큼 반복 출력합니다.
- `length.out` : 반복 출력할 길이입니다. `length.out`와 `times` 가 같이 사용되면 `length.out`의 우선 순위를 취하고 `times`는 무시됩니다.



# 리피트

## 3장. R 데이터 종류 및 구조

```
> rep(1:4, 2)
[1] 1 2 3 4 1 2 3 4

> rep(1:4, each=2)
[1] 1 1 2 2 3 3 4 4

> rep(1:4, c(2,2,2,2))
[1] 1 1 2 2 3 3 4 4

> rep(1:4, c(2,1,2,1))
[1] 1 1 2 3 3 4

> rep(1:4, each=2, len=4) # 처음 4개 만 출력합니다.
[1] 1 1 2 2

> rep(1:4, each=2, len=10) # 정수 8개와, 다시 반복된 1 두 개가 출력됩니다.
[1] 1 1 2 2 3 3 4 4 1 1

> rep(1:4, each=2, times=3) # 각 숫자가 두 번씩 3번 반복된 24개가 출력됩니다.
[1] 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4
```

## 7. List (리스트)

3장. R 데이터 종류 및 구조

리스트(List)는 복합 구조형의 벡터에 해당하는 데이터 타입입니다.

- 리스트는 다른 언어에서 흔히 보는 해싱 또는 딕셔너리에 해당합니다.
- (키, 값) 형태의 데이터를 담는 연관 배열(associative array)입니다.
- 리스트는 `list(키=값, 키=값, ...)` 형태로 데이터를 나열해 정의합니다.
- '[색인]'의 형태는 각 값이 아니라 '(키, 값)' 을 담고 있는 서브 리스트를 반환합니다.

```
> data <- list(name="홍길동", age=25)
```

리스트의 데이터를 조회하기 위해서 '리스트변수명\$키' 또는 각 요소를 순서대로 '리스트변수[[색인]]'와 같이 접근할 수 있습니다.

<code>list[[1]]</code>	: 첫 번째 항목인 name의 값 조회
<code>list\$name</code>	: name 항목의 값 조회
<code>list\$name &lt;- NULL</code>	: name 항목 삭제

# 리스트

## 3장. R 데이터 종류 및 구조

unlist는 리스트를 벡터로 변환합니다.

```
> unlist(data)
  name    age
"홍길동"  "25"
```

리스트 항목의 개수는 NROW() 또는 length()를 이용합니다.

```
> data <- list(name="홍길동", age=25)

> NROW(data)
[1] 2

> length(data)
[1] 2
```

# 리스트

## 3장. R 데이터 종류 및 구조

```
> x <- c(1,2,3,4,5)

> y <- c("Hello", "World")

> z <- c(TRUE, FALSE, TRUE)

> a <- c(x,y,z)  # 벡터

> a
[1] "1"      "2"      "3"      "4"      "5"      "Hello"
[7] "World"  "TRUE"   "FALSE"  "TRUE"

> b <- list(x,y,z)  #리스트

> b
[[1]]
[1] 1 2 3 4 5

[[2]]
[1] "Hello" "World"

[[3]]
[1] TRUE FALSE TRUE
```

## 8. Matrix (행렬)

### 3장. R 데이터 종류 및 구조

행렬(Matrix)은 행과 열을 가지는 2차원 배열입니다. 벡터와 마찬가지로 행렬에는 한 가지 유형의 스칼라 데이터들만 저장할 수 있습니다. 그러므로 모든 요소가 숫자인 행렬은 가능하지만, '1열은 숫자, 2열은 문자열'과 같은 형태는 불가능합니다.

다음은 행렬의 특징들입니다.

- 행렬은 `matrix()`를 사용해 표현합니다.
- 열 우선으로 채워집니다.(행 우선으로 채우기 위해선 `byrow=TRUE` 속성을 이용합니다.)
- 행렬의 행과 열에 명칭을 부여하고 싶다면 `dimnames()`를 사용합니다.
- 행렬의 차원은 `ncol()` 또는 `nrow()`을 이용합니다.
- 행렬끼리의 덧셈이나 뺄셈은 `+` 또는 `-` 를 사용합니다.
- 행렬 곱은 `%*%` 를 사용합니다.
- 전치행렬은 `t()`로 구합니다.
- 역행렬은 `solve(행렬)`로 계산합니다.

# 행렬은 열 우선

3장. R 데이터 종류 및 구조

```
> colMatrix <- matrix(1:15, nrow=5, ncol=3,  
+                      dimnames=list(c("R1", "R2", "R3", "R4", "R5"),  
+                      c("C1", "C2", "C3")))  
> colMatrix  
  C1 C2 C3  
R1  1  6 11  
R2  2  7 12  
R3  3  8 13  
R4  4  9 14  
R5  5 10 15  
  
> rowMatrix <- matrix(1:15, nrow=5, ncol=3, byrow=TRUE,  
+                      dimnames=list(c("R1", "R2", "R3", "R4", "R5"),  
+                      c("C1", "C2", "C3")))  
> rowMatrix  
  C1 C2 C3  
R1  1  2  3  
R2  4  5  6  
R3  7  8  9  
R4 10 11 12  
R5 13 14 15
```

# 행과 열 조회

## 3장. R 데이터 종류 및 구조

```
> dim(rowMatrix)           # 행과 열의 개수 조회
[1] 5 3
> nrow(rowMatrix)          # 행의 개수
[1] 5
> NROW(rowMatrix)          # 행의 개수
[1] 5
> ncol(rowMatrix)          # 열의 개수
[1] 3
> NCOL(rowMatrix)          # 열의 개수
[1] 3
> length(rowMatrix)        # 행 * 열
[1] 15
> dimnames(rowMatrix)      # 행과 열의 이름 조회
[[1]]
[1] "R1" "R2" "R3" "R4" "R5"

[[2]]
[1] "C1" "C2" "C3"

> rownames(rowMatrix)      # 행 이름 조회
[1] "R1" "R2" "R3" "R4" "R5"
> colnames(rowMatrix)      # 열 이름 조회
[1] "C1" "C2" "C3">
```



# 행렬 데이터 조회

## 3장. R 데이터 종류 및 구조

```
> rowMatrix[1:2, ]           # 1행과 2행 데이터 반환
  C1 C2 C3
R1  1  2  3
R2  4  5  6
> rowMatrix["R1", "C1"]      # R1행 C1열 데이터 반환
[1] 1
> rowMatrix[-3, c(1, 2)]     # 3행을 제외한 행의 1열, 2열 데이터 반환
  C1 C2
R1  1  2
R2  4  5
R4 10 11
R5 13 14
> rowMatrix[c(T, T, F, F, T), ] # 1, 2, 5행 반환
  C1 C2 C3
R1  1  2  3
R2  4  5  6
R5 13 14 15
> rowMatrix["R1", "C1", drop=FALSE] # R1행, C1열 데이터를 Matrix 형태로 반환
  C1
R1  1
```

# 전치행렬

## 3장. R 데이터 종류 및 구조

```
> payMatrix <- matrix(c(12000, 26000, 18000), ncol=3)
> payMatrix
      [,1] [,2] [,3]
[1,] 12000 26000 18000

> workerMatrix <- matrix(c(c(5, 4, 9), c(7, 3, 2)), ncol=2)
> workerMatrix
      [,1] [,2]
[1,]    5    7
[2,]    4    3
[3,]    9    2

> payMatrix %*% workerMatrix           # 행렬 곱
      [,1] [,2]
[1,] 326000 198000
```

# n차 대각행렬의 값

3장. R 데이터 종류 및 구조

```
> rowMatrix
```

```
  C1 C2 C3
```

```
R1  1  2  3
```

```
R2  4  5  6
```

```
R3  7  8  9
```

```
R4 10 11 12
```

```
R5 13 14 15
```

```
> t(rowMatrix)
```

```
  R1 R2 R3 R4 R5
```

```
C1  1  4  7 10 13
```

```
C2  2  5  8 11 14
```

```
C3  3  6  9 12 15
```

```
> diag(rowMatrix)
```

```
[1] 1 5 9
```

# 전치 행렬 (행과 열을 교환)

# n차 대각선 행렬

# 데이터 프레임으로 변환

3장. R 데이터 종류 및 구조

```
> dataFrame <- as.data.frame(rowMatrix)           # data.frame으로 변환
> class(dataFrame)
[1] "data.frame"
> dataFrame
  C1 C2 C3
R1  1  2  3
R2  4  5  6
R3  7  8  9
R4 10 11 12
R5 13 14 15
```

# 행렬을 이용한 선형방정식 풀이

3장. R 데이터 종류 및 구조

```
> x <- c(32,64,96,118,126,144,152.5,158)
> y <- c(18,24,61.5,49,52,105,130.3,125)
> plot(x, y, col=2, pch=19, ylim=c(0,150))

> (A <- matrix(c(x,rep(1,NROW(x))), ncol=2)
      [,1] [,2]
[1,]  32.0   1
[2,]  64.0   1
[3,]  96.0   1
[4,] 118.0   1
[5,] 126.0   1
[6,] 144.0   1
[7,] 152.5   1
[8,] 158.0   1

> (ab <- solve(t(A)%*%A) %*% t(A) %*% matrix(y, ncol=1))
      [,1]
[1,]  0.8749313
[2,] -26.7907862
> lines(x, x*ab[1] + ab[2])
```

## 9. Array (배열)

### 3장. R 데이터 종류 및 구조

배열(Array)은 3차원 이상의 데이터를 다룰 경우에 사용합니다. 배열은 `array()` 함수를 이용하여 만들 수 있으며, `dim` 속성을 이용해 차원의 크기를 지정합니다.

```
dataArray <- array(data, dim=c(행의수, 열의수, 면의수))
```

다음 코드는 3차원 데이터를 갖는 배열 객체를 생성하는 예입니다.

```
> dataArray <- array(1:24, dim=c(3, 4, 2))
```

```
> dataArray
```

```
, , 1
```

	[,1]	[,2]	[,3]	[,4]
[1,]	1	4	7	10
[2,]	2	5	8	11
[3,]	3	6	9	12

```
, , 2
```

	[,1]	[,2]	[,3]	[,4]
[1,]	13	16	19	22
[2,]	14	17	20	23
[3,]	15	18	21	24

# dim, nrow, NROW, ncol, NCOL, length

3장. R 데이터 종류 및 구조

```
> dataArray <- array(1:24, dim=c(3, 4, 2))
> dim(dataArray)      # 차원의 크기 조회
[1] 3 4 2
> nrow(dataArray)      # 행(1차원)의 개수
[1] 3
> NROW(dataArray)     # 행(1차원)의 개수
[1] 3
> ncol(dataArray)     # 열(2차원)의 개수
[1] 4
> NCOL(dataArray)     # 열(2차원)의 개수
[1] 4
> length(dataArray)    # 각 차원 값이 곱
[1] 24
```



# dimnames(), rownames(), colnames()

3장. R 데이터 종류 및 구조

배열의 행과 열 이름 조회는 `dimnames()`, `rownames()`, `colnames()` 등 함수를 이용합니다. `dimnames()`를 이용해 배열의 행/열/면의 이름을 지정하려면 `list()`를 이용해야 합니다. 다음의 예는 `dataArray` 배열이 3행/4열/2면 이기 때문에 그에 맞도록 이름 개수를 다르게 지정했습니다.

```
> dimnames(dataArray) <- list(c(1,2,3), c("c1", "c2", "c3", "c4"), c("x", "y"))
> dimnames(dataArray)
[[1]]
[1] "1" "2" "3"

[[2]]
[1] "c1" "c2" "c3" "c4"

[[3]]
[1] "x" "y"

> rownames(dataArray)
[1] "1" "2" "3"
> colnames(dataArray)
[1] "c1" "c2" "c3" "c4"
```

# 객체의 속성 변경

## 3장. R 데이터 종류 및 구조

`attr()` 함수를 이용하면 객체의 속성을 변경할 수 있습니다.

```
attr(객체, "속성이름") <- 속성값
```

만일 `dataAttr` 배열을 행렬 구조로 바꾸고 싶다면 다음처럼 합니다.

```
> (attr(dataArray, "dim") <- c(3, 8))
[1] 3 8
> dataArray
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
[1,]    1    4    7   10   13   16   19   22
[2,]    2    5    8   11   14   17   20   23
[3,]    3    6    9   12   15   18   21   24
```

# 10. 데이터 프레임

## 3장. R 데이터 종류 및 구조

데이터 프레임(data.frame)은 2차원 구조이며, 복합형 데이터타입입니다. 데이터 프레임은 처리할 데이터를 마치 엑셀의 스프레드시트와 같이 표 형태로 정리한 모습을 하고 있습니다. 데이터 프레임의 각 열에는 관측 값의 이름이 저장되고, 각 행에는 매 관측 단위마다 실제 얻어진 값이 저장됩니다. 행렬은 행렬 안의 모든 값이 같은 스칼라타입으로 지정되지만 데이터 프레임은 열 단위로 고유한 타입을 가질 수 있다는 점에서 차이가 있습니다.

데이터 프레임은 데이터를 가장 자연스럽게 표현하는 데이터타입이기 때문에 R에서 가장 중요한 데이터타입이며, 많은 R 함수에서 인자로 데이터 프레임을 받습니다.

```
data.frame(..., row.names=NULL, check.rows=FALSE,  
            check.names=TRUE, fix.empty.names=TRUE,  
            stringsAsFactors=default.stringsAsFactors())
```

# 데이터 프레임 생성 및 구조 확인

3장. R 데이터 종류 및 구조

다음 코드는 데이터 프레임 객체를 생성하는 예입니다.

```
> student_name <- c("Jin", "Eric", "Den", "Kei")
> student_eng <- c(60, 85, 90, 95)
> student_kor <- c(70, 90, 85, 90)

> studentData <- data.frame(student_name, student_eng, student_kor)
> studentData
  student_name student_eng student_kor
1         Jin          60          70
2         Eric          85          90
3         Den          90          85
4         Kei          95          90
```

다음 코드는 데이터 프레임의 구조를 확인합니다.

```
> str(studentData)
'data.frame': 4 obs. of 3 variables:
 $ student_name: Factor w/ 4 levels "Den","Eric","Jin",...: 3 2 1 4
 $ student_eng : num 60 85 90 95
 $ student_kor : num 70 90 85 90
```

# 데이터 프레임 열 추가 및 삭제

3장. R 데이터 종류 및 구조

데이터 프레임 객체에 새로운 열 이름을 지정하고 벡터를 이용해 값을 할당할 수 있습니다.

```
data$newColumn <- c(...)
```

열을 삭제하려면 데이터 프레임의 열에 NULL을 할당합니다.

```
data$column <- NULL
```

# 열 타입 변경 및 열 이름 지정

## 3장. R 데이터 종류 및 구조

열의 타입을 변경할 수 있습니다. 다음 구문은 student\_name 열의 타입을 Factor에서 문자열로 변경하는 예입니다. 데이터프레임의 구조를 str() 함수로 확인해 보세요.

```
> studentData$student_name <- as.character(studentData$student_name)
> str(studentData)
'data.frame': 4 obs. of 3 variables:
 $ student_name: chr  "Jin" "Eric" "Den" "Kei"
 $ student_eng : num  60 85 90 95
 $ student_kor : num  70 90 85 90
```

names() 함수를 이용하면 열 이름을 새로 지정할 수 있습니다. 다음 코드는 새로운 열을 추가하고 열 이름을 새로 지정합니다.

```
> studentData$student_math <- c(80, 95, 95, 90)
> names(studentData) <- c("Name", "English", "Korean", "Mathematics")
> studentData
```

	Name	English	Korean	Mathematics
1	Jin	60	70	80
2	Eric	85	90	95
3	Den	90	85	95
4	Kei	95	90	90

# rename 함수를 이용한 열 이름 변경

## 3장. R 데이터 종류 및 구조

rename() 함수를 이용하면 특정 열의 이름을 바꿀 수 있습니다. rename() 함수를 사용하려면 reshape 패키지를 설치하고 로드해야 합니다.

```
> newData <- rename(studentData, c(No="StudentID"))
Error in rename(studentData, c(No = "StudentID")) :
  could not find function "rename"
> install.packages("reshape")
... 생략
* installing *source* package 'reshape' ...
** 패키지 'reshape'는 성공적으로 압축해제되었고, MD5 sums 이 확인되었습니다
** R
** data
*** moving datasets to lazyload DB
** inst
** preparing package for lazy loading
** help
*** installing help indices
** building package indices
** testing if installed package can be loaded
* DONE (reshape)

The downloaded source packages are in
  'C:\Users\JK\AppData\Local\Temp\RtmpwvVVtJ\downloaded_packages'
> library(reshape)
> names(studentData)
[1] "Name"      "English"   "Korean"    "Mathmatics"
> newData <- rename(studentData, c(Name="StudentName"))
> names(newData)
[1] "StudentName" "English"     "Korean"      "Mathmatics"
```

# 데이터 프레임 합치기

3장. R 데이터 종류 및 구조

`cbind()` 함수는 두 데이터 프레임을 열 단위로 합쳐줍니다. 두 객체는 행의 수가 같아야 합니다.

`rbind()` 함수는 두 데이터 프레임을 행 단위로 합쳐줍니다. 두 객체는 열의 수가 같아야 합니다.

```
cbind(a, b)      : 열 단위 병합  
rbind(a, b)     : 행 단위 병합
```

```
> student_no <- data.frame(1:NROW(studentData$English))  
> names(student_no) <- c("No")  
> student_no  
  No  
1  1  
2  2  
3  3  
4  4  
  
> studentData <- cbind(student_no, studentData)  
> studentData  
  No Name English Korean Mathematics  
1  1  Jin     60     70         80  
2  2 Eric     85     90         95  
3  3  Den     90     85         95  
4  4  Kei     95     90         90
```



# 부분 데이터셋 조회

## 3장. R 데이터 종류 및 구조

데이터 프레임에서 부분 데이터셋을 조회하는 방법은 매우 다양합니다. [ ]와 색인을 이용하여 원하는 위치의 열과 행 정보를 조회할 수 있습니다. [ ]를 이용하여 부분 데이터셋을 조회할 때 조회하고 싶은 열 또는 행을 지정할 수 있습니다. 색인을 음수로 표현하면 해당 행 또는 열을 제외합니다. 벡터함수인 c()를 이용하면 여러 행 또는 열을 지정할 수 있습니다.

```
> studentData[, 2]           # 2열 데이터만 추출
[1] Jin  Eric Den  Kei
Levels: Den Eric Jin Kei
```

# 부분 데이터셋 조회

3장. R 데이터 종류 및 구조

```
> studentData[, -2]           # 2열을 제외하고 모든 열 추출
  No English Korean Mathematics
1  1      60      70          80
2  2      85      90          95
3  3      90      85          95
4  4      95      90          90

> studentData[3, ]           # 3행 데이터만 추출
  No Name English Korean Mathematics
3  3  Den      90      85          95

> studentData[-3, ]          # 3행을 제외하고 모든 행 추출
  No Name English Korean Mathematics
1  1  Jin      60      70          80
2  2  Eric     85      90          95
4  4  Kei      95      90          90
```

# 부분 데이터셋 조회

3장. R 데이터 종류 및 구조

```
> studentData[-3, -2]           # 3행 2열을 제외하고 모든 행 추출
  No English Korean Mathematics
1  1         60      70          80
2  2         85      90          95
4  4         95      90          90

> studentData[,c(1,3,4)]        # 1,3,4열 추출
  No English Korean
1  1         60      70
2  2         85      90
3  3         90      85
4  4         95      90

> studentData[, -c(2,5)]        # 2,5열을 제외하고 모든 열 추출
  No English Korean
1  1         60      70
2  2         85      90
3  3         90      85
4  4         95      90
```

# 부분 데이터셋 조회

3장. R 데이터 종류 및 구조

```
> studentData[,c(-2,-5)]      # 2,5열을 제외하고 모든 열 추출
  No English Korean
1  1      60      70
2  2      85      90
3  3      90      85
4  4      95      90

> studentData[,c(TRUE, FALSE, TRUE, FALSE, TRUE)] # 1,3,5열 추출
  No English Mathematics
1  1      60          80
2  2      85          95
3  3      90          95
4  4      95          90
```

- [ ] 표현식에서 콤마(,)를 제외하면 열을 추출합니다.

# subset

## 3장. R 데이터 종류 및 구조

```
> subset(studentData, studentData$Korean >= 90)
  No Name English Korean Mathematics
2  2 Eric      85      90           95
4  4 Kei       95      90           90
> subset(studentData, select=c("Name", "English", "Korean"))
  Name English Korean
1  Jin      60      70
2 Eric      85      90
3  Den      90      85
4  Kei      95      90
> subset(studentData, select=c(2,3,4))
  Name English Korean
1  Jin      60      70
2 Eric      85      90
3  Den      90      85
4  Kei      95      90
```

# subset

## 3장. R 데이터 종류 및 구조

```
> subset(studentData, select=-c(1,5))
  Name English Korean
1  Jin      60      70
2 Eric      85      90
3  Den      90      85
4  Kei      95      90
> subset(studentData, select=c(2,3,4), studentData$Korean >= 90)
  Name English Korean
2 Eric      85      90
4  Kei      95      90
> subset(studentData, select=c(2,3,4), subset=(studentData$Korean >= 90))
  Name English Korean
2 Eric      85      90
4  Kei      95      90
```

# head, tail

3장. R 데이터 종류 및 구조

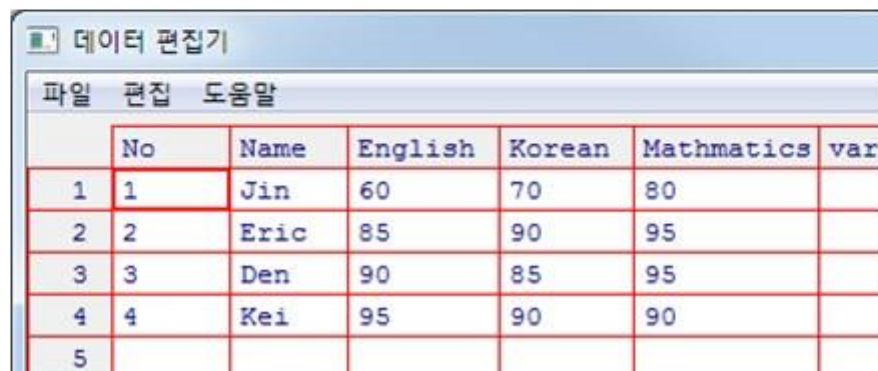
head(), tail() 함수를 이용하여 맨 처음, 또는 마지막 일부 행을 조회할 수 있습니다. head(), tail() 함수 파라미터 n의 기본값은 6입니다. n 값을 지정하지 않으면 6개 행이 조회됩니다.

```
> head(studentData, n=3)
  No Name English Korean Mathematics
1  1  Jin      60      70           80
2  2  Eric      85      90           95
3  3  Den      90      85           95
> tail(studentData, n=3)
  No Name English Korean Mathematics
2  2  Eric      85      90           95
3  3  Den      90      85           95
4  4  Kei      95      90           90
```

# edit

## 3장. R 데이터 종류 및 구조

`edit()` 함수를 이용하면 데이터 편집기를 실행시킬 수 있습니다. `edit()` 함수는 편집한 데이터를 반환하기 때문에 반드시 변수에 할당을 해야 데이터가 데이터 편집기에서 수정한 데이터가 반영됩니다. 데이터 편집기에서 작업한 내용은 되돌리기(Ctrl+z)로 취소하지 못합니다. 그러므로 임시 변수에 데이터를 할당한 다음 수정한 데이터가 올바른지 확인 하 후 원본 데이터를 변경시키는 것을 권장합니다.



	No	Name	English	Korean	Mathmatics	var
1	1	Jin	60	70	80	
2	2	Eric	85	90	95	
3	3	Den	90	85	95	
4	4	Kei	95	90	90	
5						



# edit

## 3장. R 데이터 종류 및 구조

```
> temp <- edit(studentData)      # 5, Soo, 85, 90, 95 추가
> temp
  No Name English Korean Mathematics
1  1  Jin     60     70         80
2  2  Eric    85     90         95
3  3  Den    90     85         95
4  4  Kei    95     90         90
5  5  Soo    85     90         95
> studentData <- temp
> rm(temp)
```



	No	Name	English	Korean	Mathematics	va:
1	1	Jin	60	70	80	
2	2	Eric	85	90	95	
3	3	Den	90	85	95	
4	4	Kei	95	90	90	
5	5	Soo	85	90	95	
6						

# 11. 타입 판별

3장. R 데이터 종류 및 구조

다음 코드는 `class()` 함수와 `is.*()` 형태 함수를 사용하여 타입을 확인하는 예입니다.

```
> class(iris17)  
[1] "data.frame"  
> class(iris$Sepal.Length)  
[1] "numeric"  
> class(iris$Species)  
[1] "factor"  
> is.factor(iris$Species)  
[1] TRUE  
> is.numeric(iris$Sepal.Length)  
[1] TRUE  
> is.character(iris$Species)  
[1] FALSE  
> is.data.frame(iris)  
[1] TRUE
```

# 11. 타입 변환

3장. R 데이터 종류 및 구조

타입 변환은 각 타입에 인자로 변환할 데이터를 넘기거나, `as.*()` 형식 함수를 사용하여 수행할 수 있습니다. `as.*()` 함수는 `as.numeric`, `as.factor`, `as.data.frame`, `as.matrix` 등이 있습니다.

다음 구문은 iris데이터의 Species 열의 타입을 바꾸고 확인하는 예입니다.

```
> str(iris$Species)
Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...
> iris$Species <- as.character(iris$Species)
> str(iris$Species)
chr [1:150] "setosa" "setosa" "setosa" "setosa" "setosa" "setosa" "setosa" ...
> iris$Species <- as.factor(iris$Species)
> str(iris$Species)
Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...
```

# 12. 문자열과 날짜

3장. R 데이터 종류 및 구조

## 1. 문자열의 길이 알아내기

`nchar()` 사용

`length()` 는 '벡터'의 길이를 반환, 문자열 "Moe"를 단일개체 벡터로 봄

## 2. 문자열 연결하기

`paste()` 사용 : 여러 개의 문자열들을 연결

문자열 사이사이에 기본적으로 공백 삽입 -> `sep` 인수로 변경 가능)

## 3. 하위 문자열 추출하기

`substr(string, start, end)` : `start`에서 시작해서 `end`에서 끝나는 하위 문자열 추출

## 4. 구분자로 문자열 분할하기

`strsplit(string, delimiter)` 사용

`delimiter`(구분자) : 간단한 문자열이나 정규표현식

# 문자열과 날짜

3장. R 데이터 종류 및 구조

## 5. 하위 문자열 대체하기

`sub(Old, new, string)` : 첫 번째 하위 문자열을 대체

`gsub(Old, new, string)` : 모든 하위 문자열 대체

## 6. 문자열에서 특수문자 보기

출력되지 않는 특수문자들이 문자열에 포함

`print()` : OK! 문자열에 있는 특수문자 볼 수 있음

`cat()` : No. 볼 수 없음

## 7. 문자열의 모든 쌍별 조합 만들기

`outer(문자열1, 문자열2, paste, sep="")`

: 외적을 계산하기 위한 함수 -> 세번째인자에 곱셈대신 다른 함수 허용(paste)

: 행렬이 반환됨 (`as.vector`) 사용

# 문자열과 날짜

3장. R 데이터 종류 및 구조

## 8. 현재 날짜 알아내기

`Sys.Date()` : 현재 날짜(Date 객체)를 반환

## 9. 문자열을 날짜로 변환하기

`as.Date` 사용 - 기본 문자열 형식 : `yyyy-mm-dd`

`yyyy-mm-dd` 형식이 아닌 경우 `as.Date`의 `format` 인자를 지정

ex) 미국스타일 - `format="%m/%d/%Y"`

지원되는 형식에 관해서는 `strptime` 함수의 도움말 페이지 참고

Y : 4자리 연도 / y : 2자리 연도

## 10. 날짜를 문자열로 변환하기

`format`이나 `as.character` 사용

# R 데이터 종류 및 구조의 이해 실습

1. iris 데이터를 사용하여 data.frame의 특성을 살펴봅니다.
  - 1) 행과 열에 대한 다양한 참조 방식을 사용하여 데이터를 조회합니다.
    - ✓ iris 데이터의 차원 확인, 컬럼이름 확인, 구조확인, 속성들
  - 2) 행과 열 정보를 조회합니다.
    - ✓ iris의 요약통계 정보
    - ✓ 꽃받침의 길이 처음 10개 조회
  - 3) 부분 데이터셋을 추출해 봅니다.
    - ✓ virginica종만 추출 => virginica
    - ✓ setosa종만 추출 => setosa
  - 4) 추출한 부분 데이터셋을 다시 결합해 봅니다.
2. setosa 종의 꽃 받침(Sepal)의 폭과 길이 부분 데이터 셋을 추출하세요.
3. 작업내용에 따른 급여가 차등 지급됩니다.(행렬 문제)  
A작업은 시급 12000원, B작업은 시급 26000원, C작업은 시급 18000원 입니다.  
두 사람이 각 작업을 수행하는 데 있어서 실제 작업한 시간이 작업 내역에 따라 다릅니다. 갑은 A작업을 5시간, B작업을 4시간, C작업을 9시간  
그리고 을은 A작업을 7시간, B작업을 3시간, C작업을 2시간 작업 했습니다.  
갑과 을의 급여를 계산하세요.

#힌트 : 행렬 두 개, 작업당 급여를 저장하는 행렬, 근무자들이 근무한 시간

#행렬의 곱은 %\*% 를 이용한다.

계산한 갑과 을의 급여는 각각 326000원, 198000원 입니다.