



## 2. 데이터 유형과 구조

### chap02\_DataStructure 수업내용

- R에서 제공하는 주요 자료구조(객체 타입)
  1. Vector : 동일 데이터 타입을 갖는 1차원 배열
  2. Matrix : 동일 데이터 타입을 갖는 2차원 배열
  3. Array : 동일 데이터 타입을 갖는 3차원 배열
  4. Data Frame : 열 단위로 서로 다른 데이터 타입을 갖는 배열
    - > 2차원 테이블 구조(DB 테이블과 유사)
  5. List : 서로 다른 데이터 구조(Vector, Data Frame, Array, List) 중첩



## 2. 데이터 유형과 구조

### 1) Vector 자료구조

- ✓ R의 기본 데이터 구조
- ✓ 1차원 배열 형태
  - 접근 : `[index]` : 1부터 시작
- ✓ 동일한 타입의 데이터만 저장 가능
- ✓ 벡터 데이터 생성 함수 : `c()`, `seq()`, `rep()`
- ✓ 벡터 데이터 처리 함수 : `setdiff()`, `intersect()`



Vector



## 2. 데이터 유형과 구조

### 2) Matrix 자료구조

- ✓ 동일 데이터 타입을 갖는 2차원 배열
- ✓ 행렬(matrix) 객체
- ✓ Matrix 데이터 생성 함수
  - `rbind()` : 행 묶음
  - `cbind()` : 컬럼 묶음
- ✓ Matrix 데이터 처리 함수
  - `apply()` : 함수적용

1	2	3	4	5
---	---	---	---	---

Vector

1	2	3	4	5
1	2	3	4	5
1	2	3	4	5
1	2	3	4	5

Matrix



## 2. 데이터 유형과 구조

- c()함수 이용 matrix 생성

```
m <- matrix(c(1:5))  
m # 열 기준으로 행렬 생성  
#####  
      [,1]  
[1,]    1  
[2,]    2  
[3,]    3  
[4,]    4  
[5,]    5  
#####
```

```
m <- matrix(c(1:10), nrow=2) # 열 우선 2행2열 생성  
m  
#####  
      [,1] [,2] [,3] [,4] [,5]  
[1,]    1    3    5    7    9  
[2,]    2    4    6    8   10  
#####  
m <- matrix(c(1:10), nrow=2, byrow=T) # by=T : 행 우선  
m  
#####  
      [,1] [,2] [,3] [,4] [,5]  
[1,]    1    2    3    4    5  
[2,]    6    7    8    9   10  
#####
```



## 2. 데이터 유형과 구조

### 3) Array : 동일 데이터 타입을 갖는 다차원 배열

- ✓ 3차원 배열 객체 생성
- ✓ R에서 활용도 낮음

```
d <- c(1:12) # 12개 벡터 객체 생성
arr <- array(d, c(3,2,2)) # 3행2열 구조 2개
arr #1~6(1면), 7~12(2면) -> 3차원 배열 객체
#####
```

```
,, 1
```

```
[,1] [,2]
[1,] 1 4
[2,] 2 5
[3,] 3 6
```

```
,, 2
```

```
[,1] [,2]
[1,] 7 10
[2,] 8 11
[3,] 9 12
```

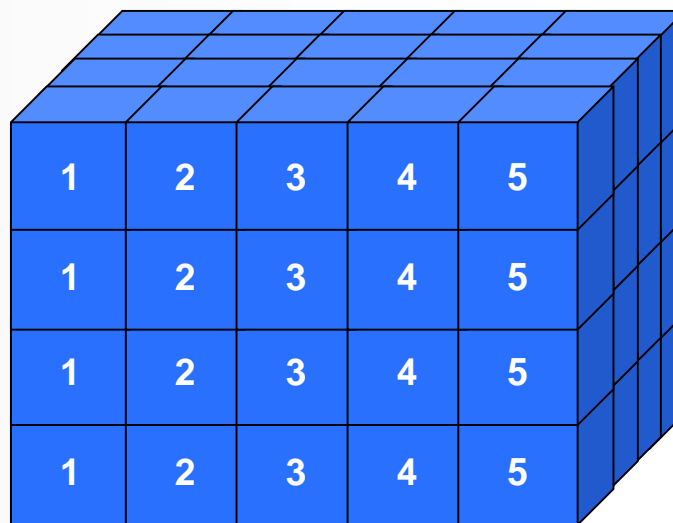
```
#####
```



Vector



Matrix



Array



## 2. 데이터 유형과 구조

### 4) Data Frame

- ✓ 리스트 보다 활용범위 넓다.
- ✓ DBMS의 테이블 구조와 유사  
(서로 다른 데이터 타입을 갖는 컬럼)
- ✓ 가장 많이 사용하는 객체 타입
- ✓ list와 Vector 혼합형
  - 컬럼 구성 : list, list 구성 : vector
- ✓ data frame 생성방법
  - Vector, Matrix, txt/excel/csv 파일



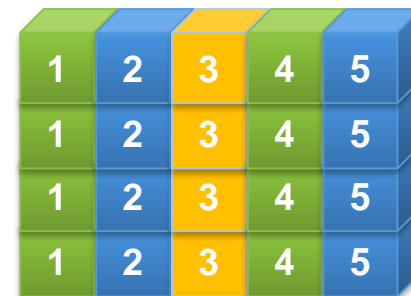
Vector



Array



Matrix



Data Frame



## 2. 데이터 유형과 구조

- Data Frame 특징

1. 형식) data.frame(컬럼1, 컬럼2.. 컬럼n)
2. 컬럼 단위로 서로 다른 자료형 가능
3. 모든 컬럼은 크기가 동일해야 함

[ 컬럼 구성 예 ]

```
id <- c("hong", "lee", "kang")
```

```
name <- c("홍길동", "이순신", "강감찬")
```

```
age <- c(30, 35, 45)
```



## 2. 데이터 유형과 구조

- Data Frame 생성

- 1) Vector이용 객체 생성

```
no <- c(1,2,3)
```

```
name <- c("hong", "lee", "kim")
```

```
pay <- c(150,250,300)
```

```
vemp <- data.frame(NO=no, Name=name, Pay=pay) #컬럼명 지정
```

```
vemp
```

```
#####
```

```
  NO Name  Pay
```

```
1  1 hong 150
```

```
2  2 lee  250
```

```
3  3 kim  300
```

```
#####
```





## 2. 데이터 유형과 구조

- Data Frame 생성

2) matrix이용 객체 생성

```
m <- matrix(
  c(1,"hong",150,
    2, "lee", 250,
    3, "kim", 300) ,3 ,by=T) # 행우선, 3개 리스트 생성
memp <- data.frame(m)
memp
#####
  X1  X2  X3  <- 기본 컬럼명
1  1 hong 150
2  2  lee 250
3  3  kim 300
#####
```



## 2. 데이터 유형과 구조

- Data Frame 생성

- 3) txt파일 이용 객체 생성

```
getwd()
```

```
setwd("c:/Rwork/Part-I")
```

```
txttemp <- read.table('emp.txt', header=T, sep="")
```

```
# 제목있음, 공백구분
```

```
txttemp
```

```
#####
```

```
사번 이름 급여
```

```
1 101 hong 150
```

```
2 201 lee 250
```

```
3 301 kim 300
```

```
#####
```



## 2. 데이터 유형과 구조

- Data Frame 생성

- 4) csv파일 이용 객체 생성

```
getwd()
```

```
csvtemp <- read.csv('emp.csv', header=T) # 제목있음,  
컴마구분
```

```
csvtemp
```

```
#####
```

```
사번  이름  급여
```

```
1  101  홍길동  150
```

```
2  102  이순신  450
```

```
3  103  강감찬  500
```

```
4  104  유관순  350
```

```
5  105  김유신  400
```

```
#####
```



## 2. 데이터 유형과 구조

### 5) List : 서로 다른 데이터 구조

- ✓ Vector, Data Frame, Array, List의 중첩 구조
- ✓ c(구조체), python(딕셔너리)
- ✓ 함수 내에서 여러 값을 하나의 키로 묶어서 반환할 경우 유용함

Key	Value
name	홍길동
age	35
address	한양시
gender	남자
htype	아파트

```
member <- list(name="홍길동",  
               age = 35,address="한양",  
               gender="남자", htype="아파트")
```



## 2. 데이터 유형과 구조

# 데이터 프레임 검색 -> 벡터 결과 반환

df\$x # 형식)변수\$컬럼

# 데이터 프레임 처리함수

str(df) # 테이블 구조보기

summary(df) # 요약함수

#####

	x	y	z
Min.	:1	Min. :2	a:1
1st Qu.:	2	1st Qu.: 4	b:1
Median	:3	Median : 6	c:1
Mean	:3	Mean : 6	d:1
3rd Qu.:	4	3rd Qu.: 8	e:1
Max.	:5	Max. :10	

#####



## 2. 데이터 유형과 구조

```
apply(df[, c(1,2)],2, sum) # 컬럼(열)단위 합계
```

```
# x y
```

```
# 15 30
```

```
# 데이터프레임 대상 조건에 만족하는 서브셋 만들기
```

```
x1 <- subset(df, x>=3) # x가 3이상인 레코드 대상 서브셋 생성
```

```
x1
```

```
y1 <- subset(df, y<=8) # y가 8이하인 레코드 대상 서브셋 생성
```

```
y1
```

```
xy <- subset(df, x>=2 & y<=6) # 2개 조건이 참인 레코드 대상 서브셋 생성
```

```
xy
```



## 2. 데이터 유형과 구조

### ● Data Join

```
h <- data.frame(id=c(1,2), h=c(180,175))  
w <- data.frame(id=c(1,2), w=c(80,75))  
  
# id 컬럼으로 data.frame 조인  
user3 <- merge(h, w, by.x="id", by.y="id")  
user3  
# id  h  w  
#  1  1 80  
#  2  2 75
```



## 2. 데이터 유형과 구조

- stringr()과 정규표현식

```
install.packages("stringr") # 패키지 설치
```

```
library(stringr) # in memory
```

```
str_extract("abcd12aaa33", "[1-9]{2}") # 연속된 숫자2개가 첫번째 발견 항목
```

```
str_extract_all("abcd12aaa33", "[1-9]{2}") # 모두
```

```
d <- c("김길동","유관순","강감찬","김길동")
```

```
str_replace(d, "김길동","홍길동") # 문자열 교체
```

```
subs <- str_sub("abcd12aaa33", 3,6) # 서브스트링 생성
```

```
subs # "cd12"
```





## 2. 데이터 유형과 구조

- Stringr 패키지에서 제공하는 주요 함수

**str\_length ()** : 문자열 길이 리턴

**str\_join** : 문자열 연결

**str\_sub()** : 범위에 해당하는 부분 문자열 출력

**str\_split()** : 기준문자를 중심으로 부분 문자열 리스트 출력

**str\_replace** : 문자열 교체

**str\_extract()** : 문자열의 위치(index) 리턴

**str\_locate()** : 문자열에서 특정 문자열 패턴의 첫번째 위치 찾기

**str\_locate\_all()** : 문자열에서 특정 문자열 패턴의 전체 위치 찾기

✓ 문자열 패턴은 정규표현식(Regular Expression) 이용

✓ 참고 사이트 : 위키백과 정규표현식



## 2. 데이터 유형과 구조

### ● 위키백과 정규 표현식

[http://ko.wikipedia.org/wiki/%EC%A0%95%EA%B7%9C\\_%ED%91%9C%ED%98%84%EC%8B%9D](http://ko.wikipedia.org/wiki/%EC%A0%95%EA%B7%9C_%ED%91%9C%ED%98%84%EC%8B%9D)

식	기능	설명
.	문자	1개의 문자와 일치한다. 단일행 모드에서는 <b>새줄 문자</b> 를 제외한다.
\w	이스케이프	특수 문자를 식에 문자 자체로 포함한다.
	선택	여러 식 중에서 하나를 선택한다. 예를 들어, "abc adc"는 abc와 adc 문자열을 모두 포함한다.
^	부정	문자 클래스 안의 문자를 제외한 나머지를 선택한다. 예를 들면 [^abc]d는 ad, bd, cd는 포함하지 않고 ed, fd 등을 포함한다. [^a-z]는 알파벳 소문자로 시작하지 않는 모든 문자를 의미한다.
[]	문자 클래스	"["과 "]" 사이의 문자 중 하나를 선택한다. " "를 여러 개 쓴 것과 같은 의미이다. 예를 들면 [abc]d는 ad, bd, cd를 뜻한다. 또한, "-" 기호와 함께 쓰면 범위를 지정할 수 있다. "[a-z]"는 a부터 z까지 중 하나, "[1-9]"는 1부터 9까지 중의 하나를 의미한다.
()	하위식	여러 식을 하나로 묶을 수 있다. "abc adc"와 "a(b d)c"는 같은 의미를 가진다.
*	0회 이상	0개 이상의 문자를 포함한다. "a*b"는 "b", "ab", "aab", "aaab"를 포함한다.
+	1회 이상	"a+b"는 "ab", "aab", "aaab"를 포함하지만 "b"는 포함하지 않는다.
?	0 또는 1회	"a?b"는 "b", "ab"를 포함한다.
{m}	m회	"a{3}b"는 "aaab"만 포함한다.
{m,}	m회 이상	"a{2,}b"는 "aab", "aaab", "aaaab"를 포함한다. "ab"는 포함되지 않는다.
{m, n}	m회 이상 n회 이하	"a{1,3}b"는 "ab", "aab", "aaab"를 포함하지만, "b"나 "aaaab"는 포함하지 않는다.