R의 데이터 유형 - 1) 데이터 생성

1. 벡터(vector) : c( ) 함수 이용

==> 동일한 데이터 유형(수치 또는 문자)의 데이터들이 일차원적으로 구성된 것

예)
(x <- c(1,2,3,4))
(y <- c("A", "B", "C", "D"))

 # 연속적인 값들의 벡터 만들기 : p. 68

(x1 <- seq(1,10)) # 1부터 10까지의 연속적인 벡터 생성
(x2 <- seq(10,1))

(x3 <- seq(1, 10, by=3)) # 1부터 3을 더한 숫자를 10까지 출력 = (1, 4, 7, 10)
(y1 <- seq(1, 10, length.out=5)) # 1~10 중, 1부터 같은 간격으로 5개의 벡터 생성
 # 간격 값 = (a2 - a1) / (length - 1)
<간격 값> = (10-1)/(5-1) = 2.25 ===> [1,(1+2.25),(1+2.25*2),(1+2.2.5*3),(1+2.25*4)]

(x4 <- 1:10)

 # 반복적인 값들의 벡터 만들기 : p. 69

(x5 <- c(1, 2, 3))
(y2 <- rep(x5, times=2)) # y2는 x5를 두 번 연속한 값들로 된 벡터 -> (1,2,3,1,2,3)

(y3 <- rep(x5, each=2)) # y3는 x5 각각의 요소를 2번씩 반복한 벡터 -> (1,1,2,2,3,3)

[보충자료]
http://sjh836.tistory.com/112
https://wsyang.com/2011/05/data-set-in-r/

2. 배열(array) : array( ) 함수 이용

2-1) 1차원 배열 : 벡터와 동일함

```
(x <- array(1:3, dim=c(3)))              # 교재 75, array() 함수 설명
```

2-2) 2차원 배열 : 행렬

① 기존의 벡터를 이용하여 행렬 만들기 : cbin( ), rbind( ) 함수 이용

```
( a <- c(1,2,3) )
( b <- c(4,5,6) )
( c <- c(7,8,9) )
( d <- cbind(a,b,c) )
( e <- rbind(a,b.c) )
```

② array( ) 함수를 이용하여 행렬 만들기

```
( x <- array(1:6, dim=c(2,3)) )
( x <- array(c(2,4,6,8,10,12), dim=c(2,3)) )
```

2차원 배열의 행과 열에 이름 붙이기 – 교재 77쪽

```
( names <- list( c("1행", "2행"), c("1열", "2열", "3열") ) )
( x <- array(c(2,4,6,8,10,12), dim=c(2,3), dimnames = names) )

# 또는
( y <- array(c(2,4,6,8,10,12), dim=c(2,3)) )
( rownames(y) <-  c("1행", "2행") )
( colnames(y) <- c("1열", "2열", "3열") )
y
```

2-3) 3차원 배열

```
( x <- array(1:24, dim=c(2, 3, 4)) )        # 2 × 3 행렬을 4개 생성
```

3. 행렬(matrix) : matrix( ) 함수 사용

벡터의 계층화 -> 행(row)과 열(column)으로 구성. 모든 요소들이 같은 데이터 유형임.

 # 예 1) 직접 행렬 만들기 : 열 우선

```
( x1 <- matrix(1:6, nrow=2, ncol=3)) )
```

 # 예 2) 직접 행렬 만들기 : 행 우선

```
( x2 <- matrix(1:6, nrow=2, byrow=TRUE) )
```

 # 예 3) 벡터 x, y 결합으로 행렬 만들기 : 벡터 결합 -> rbind( ), cbind( ) 함수 이용

```
( x <- c(1,2,3,4) )          # 벡터 x 생성
( y <- c(5,6,7,8) )          # 벡터 y 생성

( a <- rbind(x,y) )          # 행으로 결합하기
( b <- cbind(x,y) )          # 열로 결합하기
```

 # 예 4) 열과 행의 추가

```
( z <- c(9,10,11,12) )        # 벡터 z 생성
( c <- rbind(a,z) )           # 행으로 결합하기
( d <- cbind(b,z) )           # 열로 결합하기
```

 # 예 5) 행과 열에 이름 달기

```
( names <- list(c("1행", "2행"), c("1열", "2열", "3열")) )
( c <- matrix(1:6, nrow=2, byrow=TRUE, dimnames=names) )
```

4. 리스트(list) : list( ) 함수 이용

 # 각 요소/원소들이 이름을 가지거나, 서로 다른 데이터 유형으로 구성될 수 있다.

예)

 # 4개의 요소로 구성된 리스트 : 요소 각각에 이름(key)을 지정하지 않은 경우

```
( x <- list("홍길동", "2016001", 20, c("IT융합", "데이터 관리")) )
```

 # 4개의 요소 각각에 이름(key)을 지정한 리스트

```
( y <- list("성명"="홍길동", "학번"="2016001", "나이"=20, "수강과목" = c("IT융합" ,
"데이터 관리")) )
```

```
( x1 <- list("홍길동", "2016001", 20, c("IT융합", "데이터 관리")) )
( x2 <- list("갑돌이", "2016002", 21, c("IT 실습", "모바일 기술")) )
( x3 <- list(홍길동=x1, 갑돌이=x2) )
```

```
( y1 <- list("성명"="홍길동", "학번"="2016001", "나이"=20, "수강과목" = c("IT융합" ,
"데이터 관리")) )
( y2 <- list("성명"="갑돌이", "학번"="2016002", "나이"=21, "수강과목" = c("IT 실습" ,
"모바일 기술")) )
( y3 <- list(홍길동=y1, 갑돌이=y2) )
```

#--------------------

 # 2개의 요소가 모두 리스트인 리스트 만들기

 # (방법 1)

```
( a <- list( val=c(1, 2, 3) ) )                  # 리스트 a의 키는 val (a$val) 한 개
( b <- list( val=c(1, 2, 3, 4) ) )               # 리스트 b의 키는 val (b$val) 한 개
( c <- list(a=a, b=b) )        # 리스트 c의 키는 a와 b (c$a, c$b) 두 개 => 두 요소 출력
```

 # (방법 2)

```
( x <- list(a=list(val=c(1, 2, 3)), b=list(val=c(1, 2, 3, 4))) )
```

(예 3)

```
( a <- c(1,2,3) )
( b <- c(4,5,6) )
( c <- c(7,8,9) )
( d <- cbind(a,b,c) )
```

```
( z <- list(x,d) )
```

5. 데이터 프레임 (data frame) : data.frame( ) 함수 사용

2차원 표 형식의 데이터 구조 : 엑셀의 워크시트와 유사

 # 예 1) 데이터 프레임 x1 생성하기

```
( x1 <- data.frame(성명=c("홍길동", "손오공"), 나이=c(20, 30), 주소=c("서울", "부산")) )
```

 # 예 2) 데이터 프레임 x2 생성하기 -> x1과 동일

```
( x2 <- data.frame("성명"=c("홍길동", "손오공"), "나이"=c(20, 30), "주소"=c("서울", "부산")) )
```

 # 예 3) 데이터 프레임 x1 에 새로운 행과 열 추가하기 : rbind( ) 와 cbind( ) 이용

```
( x1 <- cbind(x1, 학과=c("전산학", "경영학")) )                    # 새로운 열 추가

                                                                  # 새로운 행 추가
( x1 <- rbind(x1, data.frame(성명="장발장", 나이=25, 주소="파리", 학과="전산학")) )
```

R의 데이터 유형 - 2) 데이터의 요소 접근

1. 벡터(vector)

==> 동일한 데이터 유형(수치 또는 문자)의 데이터들이 일차원적으로 구성된 것

 # 예 1)

(x <- c(1,2,3,4))

 # 1) 벡터 x의 3번째 요소 접근

(x[3])

 # 2) 벡터 x의 2번째 요소를 제외한 나머지 접근

(x[-2])

 # 3) 벡터 x의 1번째와 3번째 요소 접근

(x[c(1,3)])

 # 예 2)

(y <- c("A", "B", "C", "D"))

 # 1) 벡터 y의 3번째 요소 접근

(y[3])

 # 2) 벡터 x의 2번째 요소를 제외한 나머지 접근

(y[-2])

 # 3) 벡터 x의 2번째와 4번째 요소 접근

(y[c(2,4)])

2. 배열(array) : array( ) 함수 이용

2-1) 1차원 배열 : 벡터의 데이터 접근과 동일함

```
(x <- array(1:3, dim=c(3)))    # 교재 75, array() 함수 설명

# 예 1) 벡터 x의 3번째 요소 접근
( x[3] )

# 예 2) 벡터 x의 2번째 요소를 제외한 나머지 접근
( x[-2] )

# 예 3) 벡터 x의 1번째와 3번째 요소 접근
( x[c(1,3)] )
```

2-2) 2차원 배열 : 행렬과 같은 의미.

```
( x1 <- array(c(2,4,6,8,10,12), dim=c(2,3)) )

# 예1)
x1[1]       # 2차원 배열의 첫 번째 값
x1[2]       # 2차원 배열의 두 번째 값
x1[-3]       # 2차원 배열의 세 번째 값을 제외한 나머지 값들

x1[1,2]      # 2차원 배열의 첫 번째 행, 두 번째 열의 값
x1[2,3]      # 2차원 배열의 두 번째 행, 세 번째 열의 값

x1[1,]       # 첫 번째 행의 값들
x1[,2]       # 두 번째 열의 값들

x1[-1,]      # 첫 번째 행을 제외한 나머지 행들
x1[,-2]      # 두 번째 열을 제외한 나머지 열들


# 2차원 배열의 행과 열에 이름 붙이기 – 교재 77쪽
( names <- list(c("1행", "2행"), c("1열", "2열", "3열")) )
( x2 <- array(c(2,4,6,8,10,12), dim=c(2,3), dimnames = names) )

x2[ , "1열"]
x2[ "1행", ]
```

```
rownames(x2)        # 행에 부여된 이름(key)를 출력
colnames(x2)        # 열에 부여된 이름(key)를 출력


rownames(x2) <- c("Row #1", "Row #2")               # 행 이름 변경
colnames(x2) <- c("Col #1", "Col #2", "Col #3")     # 열 이름 변경
x2                  # 행과 열의 이름이 변경되어 있음.
```

2-3) 3차원 배열

```
# 배열변수[행 인덱스, 열 인덱스, 차원 인덱스] 로 데이터 접근.

( x3 <- array(1:24, dim=c(2, 3, 4)) )     # 2 × 3 행렬을 4개 생성

x3

x3[1]
x3[5]


x3[ , , 1]        # 첫 번째 행렬
x3[ , , 3]        # 세 번째 행렬

x3[, 2, 1]        # 첫 번째 행렬의 두 번째 열
x3[1, , 4]        # 네 번째 행렬의 첫 번째 행

x3[1,1,]          # 모든 행렬의 첫 번째 행, 첫 번째 열의 값들
x3[2,2,]          # 모든 행렬의 두 번째 행, 두 번째 열의 값들

x3[ , , -1]        # 첫 번째 행렬을 제외한 나머지 행렬들
x3[ , , -3]        # 세 번째 행렬 제외한 나머지 행렬들

x3[, -2, 1]        # 첫 번째 행렬의 두 번째 열을 제외한 나머지 값들
x3[-1, , 4]        # 네 번째 행렬의 첫 번째 행을 제외한 나머지 값들
```

3. 행렬(matrix) : matrix( ) 함수 사용

행렬의 데이터 접근은 2차원 배열과 동일함.

다음의 예들 각각에 대하여 데이터 접근 스크립트를 작성해 보라.

예1)

```
(x1 <- matrix(1:6, nrow=2, ncol=3)) )
```

예2)

```
(x2 <- matrix(1:6, nrow=2, byrow=TRUE) )
```

예3)

```
(x <- c(1,2,3,4))
(y <- c(5,6,7,8))

(a <- rbind(x,y))
(b <- cbind(x,y))
```

4. 리스트(list) : list( ) 함수 이용

리스트에 저장된 데이터는 인덱스 또는 키를 사용해 접근할 수 있다.
x$key : 리스트 x에서 키 값 key에 해당하는 값
x[[n]] : 리스트 x에서 n 번째 저장된 값
x[n] : 리스트 x에서 n번째 데이터의 서브리스트

 # 예 1) 리스트에 key 가 지정되지 않은 경우
4개의 요소로 구성된 리스트
(x <- list("홍길동", "2016001", 20, c("IT융합", "데이터 관리")))

x   # 리스트 x 출력
x[[1]]   # 리스트 x의 첫 번째 값 출력
x[[3]]   # 리스트 x의 세 번째 값 출력

x[1]   # 리스트 x의 첫 번째 데이터의 서브리스트
x[3]   # 리스트 x의 세 번째 데이터의 서브리스트

예제 1)의 리스트 x는 key가 지정되어 있지 않기 때문에 **x&key**를 사용할 수 없음.

 # 예제 2) 리스트에 key 가 지정된 경우

4개의 요소 각각에 key를 지정한 리스트
(y <- list("**성명**"="홍길동", "**학번**"="2016001", "**나이**"=20, "**수강과목**" = c ("IT융합" , "데이터 관리")))

y   # 리스트 y 출력
y[[1]]   # 리스트 y의 첫 번째 값 출력
y[[4]]   # 리스트 y의 세 번째 값 출력

y[1]   # 리스트 y의 첫 번째 데이터의 서브리스트
y[4]   # 리스트 y의 세 번째 데이터의 서브리스트

y$**성명**   # 리스트 y의 key='성명'에 해당하는 값
y$**수강과목**   # 리스트 y의 key='수강과목'에 해당하는 값

y$**수강과목[1]** # 리스트 y의 key='수강과목'의 첫 번째 요소에 해당하는 값
y$**수강과목[2]** # 리스트 y의 key='수강과목'의 두 번째 요소에 해당하는 값

y$수강과목[[1]]

```
y$수강과목[[2]]
y$수강과목[[1]][[1]]
y$수강과목[[1]][[2]]      # 에러가 남

y$수강과목[[2]][[1]]
y$수강과목[[1]][[2]]      # 에러가 남
```

 # 예제 3)

```
( x1 <- list("홍길동", "2016001", 20, c("IT융합", "데이터 관리")) )
( x2 <- list("갑돌이", "2016002", 21, c("IT 실습", "모바일 기술")) )
( x3 <- list(홍길동=x1, 갑돌이=x2) )

x3

x3[[1]]       # 리스트 x3의 첫 번째 값 출력
x3[[2]]       # 리스트 x3의 두 번째 값 출력

x3[[1]][[2]]   # 리스트 x3의 첫 번째 리스트 요소의 두 번째 값 출력 : 학번 2016001
x3[[2]][[3]]   # 리스트 x3의 두 번째 리스트 요소의 세 번째 값 출력 : 21

x3$홍길동     # 리스트 x3의 key='홍길동'에 해당하는 값  ==> x3[[1]]과 같음
x3$갑돌이     # 리스트 x3의 key='갑돌이'에 해당하는 값  ==> x3[[2]]와 같음
```

 # 예제 4)

```
( y1 <- list("성명"="홍길동", "학번"="2016001", "나이"=20, "수강과목" = c("IT융합" ,
"데이터 관리")) )
( y2 <- list("성명"="갑돌이", "학번"="2016002", "나이"=21, "수강과목" = c("IT 실습" ,
"모바일 기술")) )
( y3 <- list(홍길동=y1, 갑돌이=y2) )
```

 # 각자 앞의 예를 참고하여 데이터를 참조하는 스크립트를 작성해 보기 바람.

5. 데이터 프레임 (data frame) : data.frame( ) 함수 사용

데이터프레임에 저장된 데이터는 인덱스 또는 키를 사용해 접근할 수 있다.

d$colname : 데이터 프레임 d의 컬럼 이름 colname에 저장된 데이터.
d[m, n, drop=T] : 데이터 프레임 d의 m행 n 컬럼에 저장된 데이터. 특정 칼럼을 가져올 때 형변환을 원치 않으면 drop=F를 하면 데이터프레임을 반환한다.

(x <- data.frame(성명=c("홍길동", "손오공"), 나이=c(20, 30), 주소=c("서울", "부산")))
이 데이터 프레임 x는 3개의 키(성명, 나이, 주소)를 가지고 있다.
이 데이터 프레임 x에는 두 명(홍길동과 손오공)의 데이터가 있다.

예 1)
x$성명 # key="성명" 인 값
x[1]

예 2)
x$주소 # key="주소" 인 값
x[3]

예 3)
x[[3]][1] # 세 번째 키(주소)의 첫 번째 값
x$주소[1]

예 4)
x[[1]][2] # 첫 번째 키(주소)의 두 번째 값
x$성명[2]

예 5) --- 다음과 같이 인덱스로 참고하는 경우에는 데이터 프레임을 엑셀 시트로 생각하여 접근함.
x[1,2] # 첫 번째 데이터(홍길동)의 두 번째 값 ==> 나이를 출력함 (20)
x[2,3] # 두 번째 데이터(손오공)의 세 번째 값 ==> 주소를 출력함 (부산)