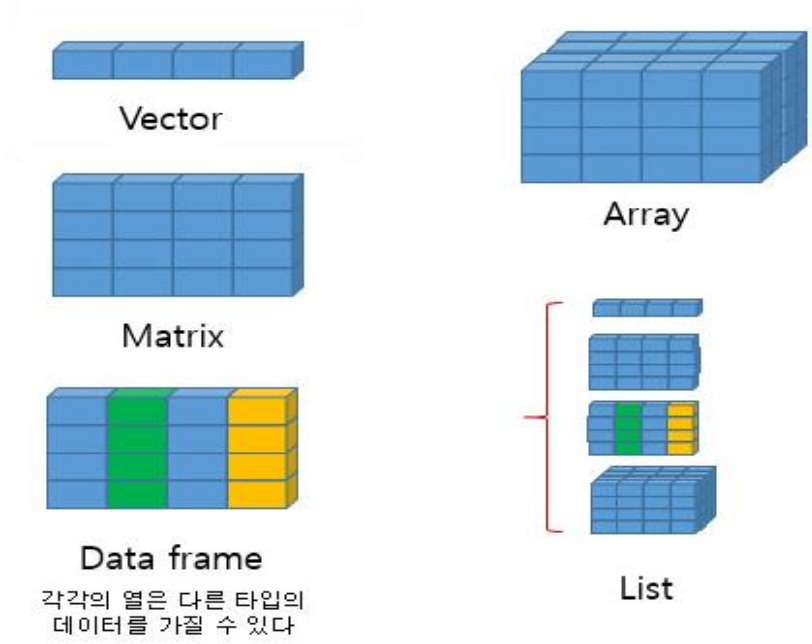
ch.2 데이터 유형과 구조

R에서 제공하는 자료구조

1) Vector (1차원 배열)
2) Matrix (2차원 배열)
3) Array (다차원 배열)
4) Data Frame (2차원 테이블 구조)
5) List (자료구조 중첩)



1. Vector자료구조
index로 접근 가능

1.1 벡터 객체
벡터데이터 생성: c(), seq(), rep()함수 이용
c()함수
인수는 콜론(:)이나 콤마(,)를 이용하여 표현
콜론(:): 범위
콤마(,): 개별 데이터 구분

실습 (c()함수 이용 벡터 생성)
c()함수
https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/c

# 실습: c() 함수를 이용한 벡터 객체 생성
c(1:20)
1:20
c(1, 2, 3, 4, 5)

실습 (seq()함수 이용 벡터 생성)
seq()함수
https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/seq

# 실습: seq() 함수를 이용한 벡터 객체 생성
seq(1, 10, 2)

실습 (rep()함수 이용 벡터 생성)
rep()함수: replicate value
https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/rep

```
# rep() 함수를 이용한 벡터 생성
rep(1:3, 3)
rep(1:3, each = 3)
```

1.2 벡터 자료 처리

setdiff()함수, intersect()함수

실습 (벡터 자료 처리)

union()

https://www.rdocumentation.org/packages/prob/versions/1.0-1/topics/union

setdiff()

https://www.rdocumentation.org/packages/prob/versions/1.0-1/topics/setdiff

intersect()

https://www.rdocumentation.org/packages/prob/versions/1.0-1/topics/intersect

```
# 실습: union(), setdiff() 그리고 intersect() 함수를 이용한 벡터 자료 처리
x <- c(1, 3, 5, 7)
y <- c(3, 5)
union(x, y)
setdiff(x, y)
intersect(x, y)
```

실습 (숫자형, 문자형, 논리형 벡터 생성)

* 반드시 같은 유형의 자료만 하나의 변수에 저장할 수 있다.
* 다른 유형의 자료형이 혼합된 경우 자료형이 변경될 수 있다.

```
v1 <- c(33, -5, 20:23, 12, -2:3)
v2 <- c("홍길동", "이순신", "유관순")
v3 <- c(T, TRUE, FALSE, T, TRUE, F, T)
v1; v2; v3
```

```
# 실습: 자료형이 혼합된 경우
v4 <- c(33, 05, 20:23, 12, "4")
```

v4

실습 (한 줄에 여러개의 스크립트 명령문 사용)
세미콜론(;) 사용으로 한줄에 여러개의 명령문 사용 가능

```
v1; mode(v1); class(v1)
v2; mode(v2); class(v2)
v3; mode(v3); class(v3)
```

실습 (칼럼명 지정)
names()함수: 벡터에 컬럼명 지정
https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/names

벡터에 NULL추가 시 객체 제거

```
age <- c(30, 35, 40)
age
names(age) <- c("홍길동", "이순신", "강감찬")
age
age <- NULL
```

실습 (벡터 참조)
index이용 벡터에 접근
index는 1부터 시작

```
a <- c(1:50)
a[10:45]
a[19: (length(a) - 5)]
```

실습 (잘못된 index사용)
콤마(,)는 2차원 이상 다차원 배열 시 사용

a[1, 2] → error

실습 (c()함수에서 콤마 사용 예)
콤마(,)는 개별 index 접근 시 사용
콜론(:)은 범위 설정 시 시용

```
v1 <- c(13, -5, 20:23, 12, -2:3)
v1[1]
v1[c(2, 4)]
v1[c(3:5)]
v1[c(4, 5:8, 7)]
```

실습 (음수 index)
index가 음수 --> 해당 index 제외
-c(2:5) : index 2-5 제외

```
v1[-1]; v1[-c(2, 4)]; v1[-c(2:5)]; v1[-c(2, 5:10, 1)]
```

1.3 벡터 객체 데이터 셋

실습 (RSADBE패키지)

```
install packages("RSADRE")
library(RSADBE)
data(Severity_Counts) #RSADRE 패키지에서 제공되는 데이터셋 로딩
str(Severity_Counts) # 데이터셋 구조 보기
```

https://www.rdocumentation.org/packages/utils/versions/3.6.2/topics/str

실습 (데이터 셋)

```
Severity_Counts
```

Severity_Counts 데이터 셋 설명

https://cran.r-project.org/web/packages/RSADBE/RSADBE.pdf

2. Matrix 자료 구조

행렬(Matrix) 자료구조
2차원
동일한 타입의 데이터만 저장 가능
행렬 생성 함수: matrix(), rbind(), cbind()
행렬 자료 처리 함수: apply()

2.1 벡터 행렬 객체 생성

c()함수는 기본적으로 열을 기준으로 객체 생성

실습 (행렬 객체 생성)

```
m <- matrix(c(1:5))
m
```

실습 (행렬 객체 생성)

```
m <- matrix(c(1:10), nrow = 2)
m
```

속성 nrow = 2 : 행의 수 = 2

실습 (행과 열의 수가 불일치)

matrix 생성 시 행과 열의 수가 불일치 --> 경고 --> 모자라는 데이터는 첫번째 데이터로부터 재사용

```
m <- matrix(c(1:11), nrow = 2)
m
```

실습 (행 우선으로 행렬 생성)

```
m <- matrix(c(1:10), nrow = 2, byrow = T)
m
```

byrow = T: 행 우선으로 배열

2.2 행 또는 열 묶음으로 행렬 생성

rbind()함수: 행 묶음
cbind()함수: 열 묶음

실습 (행 묶음)

```
x1 <- c(m, 40, 50:52)
x2 <- c(30, 5, 6:8)
mr <- rbind(x1, x2)
mr
```

실습 (열 묶음)

```
mc <- cbind(x1, x2)
mc
```

2.3 matrix()함수 이용 행렬 생성

형식:
marix(data=NA, nrow=1, ncol=1, byrow=FALSE, dimnames =NULL)

실습 (2행으로 행렬 생성)

```
m3 <- matrix(10:19, 2)
m4 <- matrix(10:20, 2)
m3
mode(m3); class(m3)
```

실습 (index이용 행렬에 접근)

형식: 변수명[행 index, 열 index]
특정 행 접근: 변수명[행 index,]
특정 열 접근: 변수명[,열 indx]

```
m3[1, ]
m3[ , 5]
m3[2, 3]
m3[1, c(2:5)]
```

실습 (3행 3열 행렬 생성)
행 우선으로 생성 시 byrow = T 속성 추가

```
x <- matrix(c(1:9), nrow = 3, ncol = 3)
x
```

2.4 행렬 객체 자료 처리 함수

실습 (자료의 개수)
length()함수: 행렬의 전체 원소 개수
https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/length

ncol()함수: 열의 수
https://www.rdocumentation.org/packages/hyperSpec/versions/0.98-20140523/topics/ncol

nrow()함수: 행의 수
https://www.rdocumentation.org/packages/hyperSpec/versions/0.98-20140523/topics/ncol

```
length(x)
ncol(x)
```

실습 (apply()함수)

apply()함수
https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/apply

형식: apply(X, MARGIN, FUN, ...)
where X: 행렬
MARGIN: 1(행 단위) or 2(열 단위)
FUN: 행렬에 적용할 함수(function)

```
apply(x, 1, max)
apply(x, 1, min)
apply(x, 2, mean)
```

실습 (사용자 정의 함수 적용)

사용자 정의 함수
형식: function(x) {
}
전치행렬: 행렬의 행과 열의 구조가 서로 바뀐 행렬

```
# 실습: 사용자 정의 함수 적용하기
f <- function(x) {
  x * c(1, 2, 3)
}
result <- apply(x, 1, f)
result
```

실습 (열 우선 순서로 사용자 정의 함수 적용)
행 우선 순서로 적용해 볼 것

```
result <- apply(x, 2, f)
result
```

실습 (행렬에 컬럼명 지정)
colnames()함수
https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/row%2Bcolnames

```
colnames(x) <- c("one", "two", "three")
x
```

벡터 객체에 컬럼명 지정 함수는?

3. Array 자료구조

3차원 배열 형태
배열 생성 함수: array()

실습 (배열 객체 생성)

```
vec <- c(1:12)
arr <- array(vec, c(3, 2, 2))
arr
```

실습 (배열 객체의 자료 조회)

```
arr[ , , 1]
arr[ , , 2]
mode(arr); class(arr)
```

index 사용

실습 (데이터 셋 가져오기)

```
library(RSADBE)
data("Bug.Metrices_Software")
```

실습 (데이터 셋 구조)

```
str(Bug.Metrices_Software)
```

실습 (데이터 셋 자료 보기)

```
Bug.Metrices_Software
```

Bug.Metrices_Software 데이터 셋

https://cran.r-project.org/web/packages/RSADBE/RSADBE.pdf

4. DataFrame 자료구조

R에서 가장 많이 쓰는 자료구조
DB의 테이블 구조와 유사
컬럼 단위로 서로 다른 데이터형의 저장이 가능
컬럼은 리스트, 컬럼 내 데이터는 벡터 구조
DataFrame 생성함수: data.frame(), read.table(), read.csv()
DataFrame 자료 처리 함수: str(), ncol(), nrow(), apply(), summary(), subset()

4.1 data.frame 객체 생성

형식:
data.frame(컬럼1 = 자료1, 컬럼2 = 자료2, ..., 컬럼n = 자료n)

실습 (벡터 이용 data.frame 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)
vemp
```

c()함수 이용
컬럼 길이는 동일해야 함

실습 (matrix 이용 data.frame 생성)

```
m <- matrix(
  c(1, "hong", 150,
    2, "lee", 250,
```

```
    3, "kim", 300), 3, by = T)
memp <- data.frame(m)
memp
```

matrix()함수 이용

https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/matrix

실습 (text파일 이용 data.frame 생성)

```
getwd()
txtemp <- read.table('emp.txt', header = 1, sep = "")
txtemp
```

read.table()함수: 텍스트 파일을 읽어 data.frame 생성

실습 (csv파일 이용 data.frame 생성)

```
csvtemp <- read.csv('emp.csv', header = T)
csvtemp
help(read.csv)
name <- c("사번", "이름", "급여")
read.csv('emp2.csv', header = F, col.names = name)
```

read.csv()함수: csv파일을 읽어 data.frame 생성

실습 (data.frame 만들기)

```
df <- data.frame(x = c(1:5), y = seq(2, 10, 2), z = c('a', 'b', 'c', 'd', 'e'))
df
```

실습 (data.frame 컬럼명 참조)
형식:
변수명$컬럼명

df$x

4.2 data.frame 객체 자료 처리 함수

str(), summary(), apply() 함수

실습(data.frame의 자료구조)

str(df)
ncol(df)
nrow(df)
names(df)
df[c(2:3), 1]

str() 함수: data.frame의 구조 보여준다.
https://www.rdocumentation.org/packages/utils/versions/3.6.2/topics/str

실습 (요약통계량)

summary(df)

summary()함수: 최소값, 최대값, 중위수, 평균, 사분위수값을 요약하여 보여준다.
https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/summary

실습 (함수 적용)
apply()함수
형식: apply(data.frame, 행/열, 함수)
where 행: 1, 열: 2

```
apply(df[ , c(1, 2)], 2, sum)
```

실습 (data.frame의 부분객체)
subset() 함수: data.frame()에서 조건에 만족하는 행을 추출하여 독립된 객체인 subset 생성
형식: 변수 <-subset(data.frame, 조건)
https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/subset

```
x1 <- subset(df, x >= 3)
x1
```

실습 (2개의 조건 이용 부분 객체 만들기)

```
y1 <-subset(df, y<=8)
xyand <- subset(df, x<=2 & y <=6)
xyor <- subset(df, x<=2 | y <=0)
y1
xyand
xyor
```

실습 (student data.frame 만들기)

```
sid = c("A", "B", "C", "D")
score = c(90, 80, 70, 60)
subject = c("컴퓨터", "국어국문", "소프트웨어", "유아교육")
```

```
student <- data.frame(sid, score, subject)
student
```

실습 (자료형과 자료구조 보기)

```
mode(student); class(student)
str(sid); str(score); str(subject)
str(student)
```

실습 (data.frame 병합)

```
1단계: 병합할 data.frame 생성
height <- data.frame(id = c(1, 2), h = c(180, 175))
weight <- data.frame(id = c(1, 2), w = c(80, 75))

2단계: data.frame 병합
user <- merge(height, weight, by.x = "id", by.y = "id")
user
```

merge()함수: data.frame 병합
https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/merge

4.3 data.frame 객체 데이터 셋

실습 (galton 데이터 셋)

```
install.packages("UsingR")
library(UsingR)
```

data(galton)

실습 (galton 데이터 셋 구조)

str(galton)
dim(galton)
head(galton, 15)

head()함수: 데이터 셋의 앞부분 일부분만 보여준다.
https://www.rdocumentation.org/packages/utils/versions/3.6.2/topics/head

galton 데이터 셋 정보
https://www.randomservices.org/random/data/Galton.html

5. List 자료구조

성격이 다른 자료형(문자열, 숫자형, 논리형)과 자료구조(벡터, 행렬, 리스트, 데이터프레임)를 객체로 생성 가능

특징:
키(key)와 값(value)이 한쌍으로 저장
python의 dictionary와 유사
key를 통해 value를 불러오는데 value에 해당하는 자료는 Vector, Matrix, Array, data.frame 등 대부분의 R 객체가 저장 가능
함수 내에서 여러 값을 하나의 키로 묶어서 반환하는 경우 유용하다.
리스트 생성 함수: list()
리스트 자료 처리 함수: unlist(), lapply(), sapply()

5.1 key 생략

키 생략 시 자동으로 기본키 생성
기본키는 [[n]]형식으로 출력 (n은 index)
값은 [n] 형식으로출력 (n은 index)

실습 (key생략한 list 생성)

```
list <- list("lee", "이순신", 95)
list
```

실습 (리스트를 벡터 구조로 변경)

```
unlist <- unlist(list)
unlist
```

unlist()함수: 리스트 형태의 자료구조를 벡터 형식의 자료구조로 변경
https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/unlist

실습 (1개 이상의 값을 갖는 리스트 객체 생성)

```
num <- list(c(1:5), c(6, 10))
num
```

5.2 key와 value형식

형식: list(key1 = value1, key2 = value2, ..., keyn = valuen)

실습 (key와 value형식으로 리스트 객체 생성)

```
member <- list(name = c("홍길동", "유관순"), age = c(35, 25),
               address = c("한양", "충남"), gender = c("남자", "여자"),
               htype = c("아파트", "오피스텔"))
member

member$name
member$name[1]
member$name[2]
```

실습 (key를 이용하여 value에 접근)
리스트 원소의 key를 호출하기 위하여 변수명$키 형식으로 작성

```
member$age[1] <- 45
member$id <- "hong"
```

```
member$pwd <- "1234"
member
member$age <- NULL
member
length(member)
mode(member); class(member)
```

5.3 리스트 객체의 자료 처리 함수

실습 (리스트 객체에 함수 적용)

```
a <- list(c(1:5))
b <- list(c(6:10))
lapply(c(a, b), max)
```

lapply()함수: 함수 적용 후 리스트 형태로 반환
https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/lapply

실습 (리스트 형식을 벡터 형식으로 반환)

```
sapply(c(a, b), max)
```

sapply()함수: 함수 적용 후 벡터 형식으로 반환
https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/lapply
cf.
https://www.rdocumentation.org/packages/memisc/versions/0.99.27.3/topics/Sapply

5.4 다차원 리스트 객체 생성

다차원 리스트(중첩): 리스트 자료구조에 다른 리스트가 중첩된 자료구조

실습 (다차원 리스트 객체 생성)

```
multi_list <- list(c1 = list(1, 2, 3),
                   c2 = list(10, 20, 30),
                   c3 = list(100, 200, 300))
multi_list$c1; multi_list$c2; multi_list$c3
```

실습 (다차원 리스트를 열 단위로 바인딩)

```
do.call(cbind, multi_list)
class(do.call(cbind, multi_list))
```

do.call()함수: 다차원 리스트를 구성하는 리스트를 각각 분해한 후 지정된 함수를 호출하여 리스트 자료를 처리

https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/do.call

6. 문자열 처리

stringr 패키지
https://www.rdocumentation.org/packages/stringr/versions/1.4.0

6.1 stringr 패키지 제공 함수

문자열 연산에 필요한 다양한 함수 제공

[표 2.1] stringr에서 제공 함수
https://www.rdocumentation.org/packages/stringr/versions/1.4.0

실습 (문자열 추출)
str_extract()함수와 정규표현식
https://www.rdocumentation.org/packages/stringr/versions/1.4.0/topics/str_extract

```
install.packages("stringr")
library(stringr)
str_extract("홍길동35이순신45유관순25", "[1-9]{2}")
str_extract_all("홍길동35이순신45유관순25", "[1-9]{2}")
```

```
str_extract("홍길동35이순신45유관순25", "[1-9]{2}")
숫자가 2개 연속된 첫번째 문자열 : 35
```

정규표현식: 추출하려는 문자열의 패턴을 지정

6.2 정규표현식

문자열 처리 관련 함수는 대부분 정규표현식(약속된 기호에 의해 표현) 이용

(1) 반복 관련 정규표현식
정규표현식에서 [ ] 기호는 대괄호 안의 문자가 {n}에서 n만큼 반복된다.
ex. [a-z]{3} : 영문 소문자가 연속으로 3개 발생
[a-z]{3,} : 영문 소문자가 3자 이상 연속하는 것 추출
[a-z]{3,5} : 영문 소문자가 3-5자 연속하는 것 추출

# 실습: 반복 수를 지정하여 영문자 추출하기
string <- "hongkd105leess1002you25강감찬2005"
str_extract_all(string, "[a-z]{3}")
str_extract_all(string, "[a-z]{3,}")
str_extract_all(string, "[a-z]{3,5}")

(2) 문자와 숫자 관련 정규표현식

실습 (문자열에서 한글, 영문자, 숫자 추출)

str_extract(string, "hong") # 해당 문자열 추출
str_extract(string, "35") #해당 숫자 추출
str_extract(string, "[가-힣]{3}") #연속된 3개의 한글문자열 추출
str_extract_all(string, "[a-z]{3}") # 무엇을 의미하는가?
str_extract_all(string, "[0-9]{4}") # 무엇을 의미하는가?

(3) 특정 문자열을 제외하는 정규표현식
형식:
"[^제외문자열]": 헤딩 문자열을 제외하고 나머지 추출
"[^제외문자열]{n}": 문자열을 제외하고 연속된 n글자 추출

# 실습: 문자열에서 한글, 영문자, 숫자를 제외한 나머지 추출하기
str_extract_all(string, "[^a-z]")
str_extract_all(string, "[^a-z]{4}")

```
str_extract_all(string, "[^가-힣]{5}")
str_extract_all(string, "[^0-9]{3}")
```

(4) 한 개의 숫자와 단어 관련 정규표현식
숫자와 단어를 패턴으로 지정할 수 있는 정규표현식

실습 (주민등록번호 검사)
" \ \ d{6}" : 숫자가 6개 연속된 패턴을 지정하는 정규표현식. "[0-9]{6}" 과 같은 결과 산출
*자판에서는 ₩ 사용

```
jumin <- "123456-1234567"
str_extract(jumin, "[0-9]{6}-[1234][0-9]{6}")
str_extract_all(jumin, "₩₩d{6}-[1234]₩₩d{6}")
```

실습 (지정된 길이의 단어 추출)
" \ \ w{7}" : 단어의 길이가 7이상인 패턴을 지정하는 정규표현식
* 특수문자 제외

```
name <- "홍길동1234,이순신5678,강감찬1012"
str_extract_all(name, "₩₩w{7,}")
```

6.3 문자열 연산

(1) 문자열 길이와 위치

실습 (문자열 길이)

```
string <- "hongkd105leess1002you25강감찬2005"
```

```
len <- str_length(string)
len
```

str_length()함수: 문자열의 길이
https://www.rdocumentation.org/packages/stringr/versions/1.4.0/topics/str_length

실습 (문자열 내 특정 문자열의 index)

```
string <- "hongkd105leess1002you25강감찬2005"
str_locate(string, "강감찬")
```

str_locate()함수: 문자열 내에서 특정 단어의 위치 표시
https://www.rdocumentation.org/packages/stringr/versions/1.4.0/topics/str_locate

(2) 부분 문자열 만들기
부분 문자열(substring)
https://www.rdocumentation.org/packages/stringr/versions/1.4.0/topics/str_sub

실습 (부분 문자열 만들기)

```
string_sub <- str_sub(string, 1, len - 7)
string_sub
string_sub <- str_sub(string, 1, 23)
string_sub
```

(3) 대・소문자 변경하기

실습 (대문자, 소문자 변경)
str_to_upper()함수: 주어진 문자열에서 소문자를 대문자로 변경
str_to-lower()함수: 주어진 문자열에서 대문자를 소문자로 변경

https://www.rdocumentation.org/packages/stringr/versions/1.4.0/topics/case

```
# 실습: 대문자, 소문자 변경하기
ustr <- str_to_upper(string_sub); ustr
str_to_lower(ustr)
```

(4) 문자열 교체・결합・분리

실습 (문자열 교체)
str_replace(문자열, 변경될 인자, 변경할 인자)함수
https://www.rdocumentation.org/packages/stringr/versions/1.4.0/topics/str_replace

```
# 실습: 문자열 교체하기
string_sub
string_rep <- str_replace(string_sub, "hongkd105", "홍길동35,")
string_rep <- str_replace(string_rep, "leess1002", "이순신45,")
string_rep <- str_replace(string_rep, "you25", "유관순25,")
string_rep
```

실습 (문자열 결합하기)
str_c(문자열, 추가인자)함수
https://www.rdocumentation.org/packages/stringr/versions/1.4.0/topics/str_c

```
string_rep
string_c <- str_c(string_rep, "강감찬55")
string_c
```

실습 (문자열 분리)

str_split(문자열, 구분자)함수
https://www.rdocumentation.org/packages/stringr/versions/1.4.0/topics/str_split

```
string_c
string_sp <- str_split(string_c, ",")
string_sp
```

(5) 문자열 합치기

Paste()함수 이용 여러 개의 문자열로 구성된 벡터 객체를 대상으로 구분자를 적용하여 하나의 문자열을 갖는 벡터 객체로 합칠 수 있다.
https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/paste

실습 (문자열 합치기)

```
1단계: 문자열 벡터 만들기
string_vec <- c("홍길동35", "이순신45", "유관순25", "강감찬55")
string_vec

2단계: 구분자를 기준으로 문자열 벡터 합치기
string_join <- paste(string_vec, collapse = ",")
string_join
```

paste(문자열, collapse=",")
* 문자열 자료가 많은 경우 매우 유용 (ex. 문자열 100개가 1개씩 원소를 차지하는 벡터객체를 대상으로 paste()함수 적용)

연습문제 #2

Reference:

김진성, 빅데이터분석을 위한 R 프로그래밍, 2nd edit, 2020

https://intro2r.com/

https://rstudio-education.github.io/hopr/index.html

https://dbrang.tistory.com/1032?category=710879