1. 데이터 종류

기본 데이터타입 : 문자, 숫자, 논리, 복소수 <- 스칼라 타입

factor(남, 여), vector(동일 자료형), list(동일 자료형이 아니어도 됨)

matrix, array, data.frame, 날짜, 그 외 특별한 값(결측치)

2. 기본데이터 타입 : character, numeric, logical(TRUE:T, FALSE:F), complex

```
class(a)    # 타입 출력
class(a)

mode(a)
typeof(a)
methods(is)    # is로 시작하는 함수들
is.character(a) # a 변수가 character인지 여부
as.character(100) # 인수를 강제로 character타입으로 형변환한 결과 반환
```

numeric : 정수, 실수

```
b <- 10.1
is.numeric(b) TRUE
as.character(b) "10.1"
as.integer(10.9) 10 # 매개변수를 정수로(내림)
```

logical

```
c <- TRUE;
is.logical(c) TRUE
class(c) "logical"
```

3. 특별한 값(Null, NA; 결측치, NaN, Inf)

```
result <- 0
add <- function(a,b){
  result <<- a+b
  return(result)
#  return()                    # 로 변경하면 add(1,2)를 반환하려 시도해도 Null값이 반환됨.
}
result
(add(1,2))
(add(b=10, a=1))
```

```
# 결측치(NA) 관련함수
d <- c(2, 4, 6, NA, 8, 10)
is.na(d) : 결측치인지 아닌지 여부 반환
complete.case(d) : d변수에 결측치가 아니면 TRUE
na.omit(d) : 결측치 제외
na.pass(d) : NA 여부와 상관없이 처리
d[complete.cases(d)]   # 벡터d에서 결측치가 아닌 값만 뽑기
mean(d, na.rm=T)
boxplot(d)
```

4. 팩터(factor) : 범주형 데이터(미리 정해진 여러개 값 중 하나 eg:성별 - 남여, 소득수준 - 상중하)

```
gender2 <- factor(c('남', '여', '여', '남'), levels=c('남','여'))
level <-factor(c('좋음', '보통', '보통'), levels=c('매우 싫음', '싫음', '보통', '좋음', '매우 좋음'), ordered=TRUE)
class(gender2)
nlevels(gender2)   # level의 개수
levels(gender2) # 카테고리 출력
```

5. 구조형 변수와 복합형 변수 : 변수 하나에 여러 값 할당

(1) 구조형 변수 : 동일 자료형

ex) 벡터(1차원), 행렬(회귀분석 용이)(2차원), 배열(n차원)

(2) 복합형 변수 : 서로 다른 자료형을 가질 수 있음

ex) 리스트, 데이터프레임(csv)

6. 벡터 : 동일자료형

자동 형변환 룰 : 문자 < 복소수 < 숫자 < logic

```
data <- c(11, 22, 33, 'TEST') # 동일자료형이 아닌 경우 자동형변환이 발생
```

```
data <- 1:10
콘솔 맨 왼쪽 [1]. [34] 같은 값은 바로 오른쪽에 나타나는 값의 index
names(data) <- c('A열', 'B열', 'C열', 'D열', 'E열', 'F열', 'G열', 'H열', 'I열', 'J열')
열이름을 부여함.
data[c(1,2)]   # 1번째, 2번째 값을 반환
data[c('A열', 'B열', 'C열', 'D열')]
data[c(1:4)]   # 1,2,3,4번째 값을 반환
data[1:4]      # 1,2,3,4번째 값을 반환
# data[1,2] # 에러 / 2차원 행렬이었다면 data[1][2]가 가능할 듯?
data[-c(2,3)] # 2, 3번째 값 제외
data[data>4] # 조건을 만족하는 대상만 반환
```

```
# 항목갯수
length(data) --- 글자수:nchar(데이터/변수)
NROW(data)
nrow(data)   # 2차원 데이터부터는 작동함
```

```
2 %in% data # data안에 2가 포함되어 있는지 T/F

avg <- mean(score, na.rm=TRUE)
cat('평균 =', avg)

(score[complete.cases(score)])
스코어 내에 결측치(NA)가 있을 경우 제외하고 나머지 유효한 값들만 출력
```

6.1 character()

```
charVec <- character(5)
    [1] "" "" "" "" "" ""
charVec[1] <- '안녕'; charVec[2] <- 'R';
charVec[3] <- '수업중' charVec[7] <- 'TEST'
[1] "안녕" "R" "수업중" "" "" NA "TEST"
```

6.2 numeric()

```
intVec <- numeric(2)
[1] 0 0
intVec[3] <- 12.2
intVec[5] <- 3.3
[1]  0.0  0.0 12.2   NA  3.3
```

```
intVec <- c(1:10)
class(intVect) : [1] "integer"
intVect <- c(12.2, 3.3)
class(intVect) : [1] "numeric"
```

벡터의 집합연산 : 합집합(union), 교집합(intersect), 차집합(setdiff), 비교(setequal)

```
union(a,b) # 합집합
intersect(a, b) # 교집합
setdiff(a,b) # 차집합
setequal(a,b)
setequal(a, c(intersect(a,b), setdiff(a,b)))
```

6.3 logical()

```
logiVec <- logical(4)
logiVec FALSE FALSE FALSE FALSE
length(logiVec) 4
```

6.4 seq

```
(a <- seq(from=1, to=10))
seq(1, 10)
seq(1, 10, 2)
seq(from=1, to=10, by=2)
a <- seq(from=1, to= 100, length.out=11)
is.vector(a) # seq는 vector를 생성하는
class(a)
seq(10, -10, -2) # 감소하는 원소들도 가능
seq(0, 1, 0.1)
seq(1, 9, pi)
```

6.5 rep()

```
rep(x, times=1, length.out= NA, each=1)
rep(1:4, 2) # rep(1:4, times=2)와 동일
    1 2 3 4 1 2 3 4
rep(1:4, each=2)
    1 1 2 2 3 3 4 4
rep(1:4, 2, 5, 2) # rep(1:4, times=2, length.out=5, each=2)
    1 1 2 2 3
```

7. 리스트 : 복합 구조형(키 값 형태로 데이터를 담는 복합구조형)

```
student <- list(name='홍길동', age=25)
$name
[1] "홍길동"
$age
[1] 25
```

```
# 키값이 name인 자료에 access하는 방법
student['name']
student$name      # 가장 많이 쓰이는 방법★
student[1]
student[[1]]
```

```
student$age <- NULL # student의 age 제거
student$score <- 88  # student의 score 추가
student$expel <- FALSE # student의 expel 추가가
student
NROW(student)    #항목갯수
length(student)    #항목갯수
```

```
studentVect <- unlist(student)
is.vector(studentVect)
studentVect # 각 원소의 type이 문자(동일 자료형)로 통일됨

student2 <- as.list(studentVect)
student2
```

list를 unlist하면 벡터가 되고 벡터를 as.list하면 list로의 형변환이 이루어진다. ㅇㅅㅇ...

8. 행렬

```
colMatrix <- matrix(1:15, nrow <- 5, ncol <-3)
   세로방향으로 먼저 데이터를 할당 (1열의 5개 행을 모두 채운 뒤 2열로 이동하는 식)
# dimnames는 반드시 리스트
rowMatrix <- matrix(1:15, nrow=5, ncol=3, byrow=TRUE,
                    dimnames = list(c("R1","R2","R3","R4","R5"),c("C1","C2","C3")))
                                  근데 행이름만 혹은 열이름만 입력하는건 안되는 듯?
                                      The dimnames of a data frame are its row.names and its names.
rownames(rowMatrix) <- c('1월','2월','3월','4월','5월') # 행이름 선언
colnames(rowMatrix) <- c('kim','lee','park')            # 열이름 선언
dim(rowMatrix) # 차원수 (행, 열의 수) / NROW(rowMatrix), nrow(rowMatrix) # 행의 수
NCOL(rowMatrix), ncol(rowMatrix)    # 열의 수
```

```
# 행렬 값 조회
rowMatrix['1월', 'kim', drop=FALSE]    #1월 kim열의 데이터를 Matrix형태로 반환
rowMatrix['1월',]    # 1월 데이터가 벡터 형태로 반환
rowMatrix['1월', , drop=FALSE] # 1월 데이터가 Matrix 형태로 반환
rowMatrix[c(1:3), c(1:2)]
rowMatrix[c('1월':'3월'), c('kim':'park')] # 에러
```

```
rowMatrix[-3, c('kim', 'lee')]  # 3행 제외한 kim열, lee열
rowMatrix[-3,]                  # 3행을 제외한 매트릭스 전체
rowMatrix[-3]                   # 3행을 제외한 모든 데이터가 벡터형태로 1차원 반환
```

```
#행렬의 곱(%*%)
payMatrix <- matrix(c(12000,26000,18000), ncol=3, nrow=1)
payMatrix
dimnames(payMatrix) <- list(c('시간당수당'),c('Yuna', 'Choo','Rachel'))
workerMatrix <- matrix(c(c(5,4,9), c(7,3,2)), nrow=3,
                       dimnames=list(c('Yuna','Choo','Rachel'), c('5월','6월')))
# payMatrix(1x3 행렬) workerMatrix(3x2 행렬)
cost <- payMatrix %*% workerMatrix
cost
rownames(cost) <-c('인건비')
cost
cost['인건비',]
```

```
#전치행렬
o <- rowMatrix    # 5x3
t <- t(rowMatrix)# 3x5
# 정방행렬이 아닌 경우 전치행렬을 곱하면 정방행렬이 도출됨
o %*% t
```

```
# 역행렬
X <- matrix(1:4, nrow=2, ncol=2, byrow=F)
X
solve(X) # 역행렬
X %*% solve(X)
solve(X) %*% X
```

```
# 행렬의 곱을 이용한 선형회귀식 도출

x <- c(2,4) # 독립변수 : 공부량
y <- c(40, 60) # 종속변수 : 점수
# 2시간을 공부하하면 40점, 4시간을 공부하면 60점의 성과를 내는 학생인듯.
X <- matrix(c(x, rep(1,NROW(x))), nrow=NROW(x), byrow=FALSE)
어째서 2X2행렬로 구성하는거지? 값에 영향을 미치는 바 없는 건가? ### 모르겠음.
양변에 역행렬을 곱하는 것도 엄밀하게는 행렬식으로 확인이 먼저 이루어 져야함.

x <-c(32, 64, 96, 118, 126, 144, 152.5, 158)  # 독립변수
y <- c(18, 24, 61.5, 49, 52, 105, 130.3, 125)
X <- matrix(c(x,rep(1, NROW(x))), nrow=NROW(x), ncol=2)
Y <- matrix(y, ncol=1)
ab <- solve(t(X) %*% X) %*% t(X) %*% Y
ab
lm(y~x)
plot(x, y)
lines(x, x*ab[1]+ab[2], col=2, lty='dotdash', lwd=4)
정방행렬을 만들기 위해 양변에 전치행렬을 곱하는 것도 가능한 경우가 제한적일 듯.

역행렬의 요소를 선형회귀의 기울기와 y절편 값으로 활용

# 행렬의 곱을 이용한 다변량 선형회귀식 도출(독립변수 3개)
# 행렬을 데이터프레임으로 변환
# 행렬연산( *, +, -, /, %%, %*%)
```

9. 배열(n차원 동일 자료형의 집합)

```
dataArray <- array(1:24, dim=c(3,4,2)) # 3차원 numeric 배열 (3행4열2면)
dimnames(dataArray) <- list(c('1행', '2행', '3행'),
                            c('1열', '2열', '3열', '4열'),
                            c('1면', '2면'))
```

10. 데이터 프레임 ★★★★★★★★★★★★

(1) 데이터 프레임 생성

```
d <- data.frame(id=c(1,2,3), name=c('김','홍','이'))


student_data  <- data.frame(student_id, student_name, student_kor, student_eng, student_gender)
                                                    student_id, student_name 등의 데이터는 이미 입력된 상태.
edit(student_data)
st <- edit(student_data) #를 실행하여 데이터를 추가할 수 있다.
```

누락된 열 삽입 / 열의 타입변경 / 일부행,열 삭제

(2) 데이터 프레임에 열 추가 및 삭제

```
student_data$math <- c(100, 95, 94, 80)     # 열 추가
student_data$age <- c(19, 31, 26, 29)       # 열 추가
student_data$age <- NULL                    # 열 삭제
```

(3) 데이터프레임의 열 형 변환

```
student_id를 numeric으로 형변환
student_data$student_id <- as.numeric(student_data$student_id)


student_gender를 factor타입으로 형변환
student_data$student_gender <- factor(student_data$student_gender, labels=c('남','여'))
                                                    데이터는 빈 칸 등 예상할 수 없는 형태로도 존재할 수 있으므로...★★★
student_data$student_gender <- as.factor(student_data$student_gender)
                                                    # 데이터에 '남','여'만 존재한다고 확신할 수 있는 경우에만
                    # 결측치가 없다고 확신하는 경우에만 사용, 데이터를 다운받을 경우 NA로 추정되는 값이 많으므로 비추
student_data$student_gender <- as.character(student_data$student_gender)
```

(4) 데이터프레임의 열이름 변경 (모든 열이름 변경, 특정 열이름만 변경)

```
.libPaths()
install.packages("reshape") # 1. 패키지 설치
library(reshape)            # 2. 메모리에 로드
detach("package:reshape", unload=TRUE)
```

```
student <- rename(student_data, c("student_id"="id")) # 특정 열 이름 변경
student <- rename(student, c("student_name"="name",
                             "student_kor"="kor",
                             "student_eng"="eng",
                             "student_gender"="gender"))
```

(5) 데이터 프레임 합치기 : cbind(행이 동일해야함), rbind(열이 동일해야함)

(6) 데이터 프레임의 부분데이터 조회

행렬마냥 간편히 []기호를 활용하는 방법이 있음.

```
# 1행 1열
student[1,1]

# 1행 'id'열
student[1, 'id']

# 1행 모든 열 데이터
student[1, ]

# 1~3행까지의 모든 열 데이터
student[1:3,]

student[,1]
                # 벡터 형태로 반환
student[, 1, drop=FALSE]   # 데이터프레임 형태로 반환
  (위에서는 매트릭스형태로 변환한다고 써놈, 아래에 emp를 벡터 vs df형식으로 호출함)

# 모든 행의 2,3,4,5열 데이터
student[, 2:5]
student[, c(2,3,4,5)]

# 2,4,6 행 외의 모든 데이터
student[c(-2,-4,-6),]
student[-c(2,4,6),]

# 1~3행 데이터에서 1열과 5열을 제외
student[1:3, c(-1, -5)]

# kor점수가 90점 이상인 대상자의 이름과 kor점수
student[student$kor>=90, c('name','kor')]
```

subset 함수 (여전히 데이터프레임에서의 부분데이터 조회영역 중 하나!)

```
subset(student, subset=(student$kor>=90))    # kor이 90이상인 모든 열 데이터
subset(student, student$kor>=90)
subset(student, subset = (student$kor>=90) & (student$gender=='남')) # kor이 90 이상인 남자의 모든 열 데이터
subset(student, select = c(1,4)) # 1번째 4번째 열
subset(student, select = c('id','name','kor','eng'))
subset(student, select = c(-1, -4)) # 1열과 4열 제외
subset(student, select = -c(1, 4))
subset(student, select = -c('id', 'eng')) # 에러!
# math가 90점 이상인 여학생 데이터 (id, name, math, kor열만 출력)
subset(student, subset = (math>=90 & gender == '여'), select=c('id','name','math','kor'))
student[c((nrow(student)-3):nrow(student)),] # 마지막 4행
```

부분데이터 조회 반복연습

같은 부분데이터 조회인데도 위에서는 student로만 하다가 emp로 하니 완전 새롭;;;

```
emp <- read.csv(file.choose()) # 탐색기에서 선택한 csv화일을 emp 변수에 할당
① emp$ename
② emp[, 'ename'] # 벡터형식
③ emp[,2,drop=FALSE] # 데이터프레임 형태
④ subset(emp, select=c('ename'), DROP=FALSE)
```

11. 타입판별 및 타입변환

class(), is.factor(), str() mode(), typeof()

12. 문자열과 날짜

```
names <- c('Eric', 'Larray', 'Curly')
length(names) # 요소의 갯수 : 3
nchar(names)  # 요소 하나하나의 문자길이 : 4 5 6

# 하위 문자열 추출하기 : substr
substr('ABCDEF', 1, 4) # 1~4번째 문자추출
substr('ABCDEF', 1, 16)
substring('ABCDEF', c(1,3), c(1,5)) # 1번째부터 1번째까지, 3번째부터 5번째까지
substring(names, 1, 2) # names안에 각 문자 첫번째~두번째 추출 : "Er" "La" "Cu"

# 문자 분할 (구분자로 분할하기)
customers <- c('jin@gamil.com, 010-8888-8888, Seoul Korea',
               'yis@gmail.com, 010-777-7777, Incheon Korea',
               'kim@naver.com, 02-456-2341, Mapo Korea')
strsplit(customers, ',[0-9]{2,3}-[0-9]{3,4}-[0-9]{4}')
strsplit(customers, ',')
```

끊는 단위가 어떻게 되는거지? ',' 는 이해가 쉬움...

```
#문자열 대체 sub(oldStr, new Str, string) 처음 나오는 oldStr을 newStr로 바꿈
#           gsub(oldStr, new Str, string) 다 바꿈.
s <- 'Impossible is just a big word thrown around by small men who find it easier to live in the world they\'ve been given than to explore the power they have to change it. Impossible is not a fact. It\'s an opinion. Impossible is not a declaration. It\'s a dare. Impossible is potential. Impossible is temporary.
```

```
Impossible is nothing.'
sub('Impossible', 'blank', s)
gsub('Impossible', 'Blank', s)
```

총괄 연습문제

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(Sepal.Length) 처음 10개 조회
   3) 부분 데이터셋을 추출해 봅니다.
      virginica종만 virginica 변수에 추출
      Setosa종 만 setosa 변수에 추출
   4) 추출한 부분 데이터셋(virginica와 setosa)을 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 근무한시간
   #행렬의 곱은 %*% 를 이용한다.