1_Rbasics(a)

YoungJin Lee

3/8/2018

변수와 상수 and assign

```
1234
```

```
## [1] 1234
```

```
"abcd"
```

```
## [1] "abcd"
```

```
x <- 5678
y <- "efgh"
```

상수(constant)는 숫자, 문자, 로직(T/F)등이 될 수 있고, 이러한 constant를 어딘가에 저장한다면 그 저장되는 공간을 변수라고 합니다.

데이터 타입: numeric

```
typeof(10.5)
```

```
## [1] "double"
```

```
typeof(10)
```

```
## [1] "double"
```

```
typeof(10L)
```

```
## [1] "integer"
```

R에서 데이터 타입에는 numeric(integer or double), Boolean(or logical), Text(or string / character)이 있습니다. 숫자 integer, double은 모두 numerics에 포함되기 때문에 이 둘의 차이에 대해서는 크게 신경쓰지 않고 numeric 만 기억해도 큰 문제는 없습니다.

데이터 타입: Boolean

```
typeof(10>3)
```

```
## [1] "logical"
```

```
typeof(F)
```

```
## [1] "logical"
```

logical 데이터는 TRUE / FALSE를 반환합니다.

데이터 타입: string

```
typeof("Hello")

## [1] "character"

typeof("1234")

## [1] "character"
```

string data를 입력할 땐 큰 따옴표 혹은 작은 따옴표로 묶어 줄 수 있습니다.

데이터 타입: NA(Not Available)

```
my.grade <- 100
your.grade <- 50
his.grade <- NA
is.na(my.grade)

## [1] FALSE

is.na(his.grade)

## [1] TRUE
```

결측치(NA)는 때에 따라 결측치 그 자체가 의미가 있을 수 있습니다. NULL과 헷갈릴 수 있는데, NULL은 프로그래밍적 측면에서 준비가 되어있지 않다는 의미이고, NA는 준비가 되어있지만 값이 없다라고 이해하면 되겠습니다. 따라서 NA를 발견한다면, 그 의미를 해석하고 적절한 값으로 대치를 하거나, 해당 변수 혹은 케이스(data point)를 삭제할 수 있습니다.

데이터 타입: Special Values

```
typeof(Inf)

## [1] "double"

typeof(-Inf)

## [1] "double"

typeof(NA)

## [1] "logical"
```

Math Operators

```
a <- 10.5
b <- 20
c <- 4

a + b ## addition
```

```
## [1] 30.5
```

```
a - c ## subtraction
```

```
## [1] 6.5
```

```
a * c ## multiplication
```

```
## [1] 42
```

```
a / c ## division
```

```
## [1] 2.625
```

```
a %% c ## remainder
```

```
## [1] 2.5
```

Boolean Operators

```
a > b ## inequality
```

```
## [1] FALSE
```

```
a*2 == b ## equality
```

```
## [1] FALSE
```

```
!(a > b) ## negation
```

```
## [1] TRUE
```

```
(b > a) & (b > c) ## logical AND
```

```
## [1] TRUE
```

```
(a > b) | (a > c) ## logical OR
```

```
## [1] TRUE
```

변수를 저장하는 여러가지 방법

R에는 변수를 저장하는 여러가지 방법이 있습니다. Vector vs Matrix vs list vs dataframe 중 Vector와 Dataframe이 현재 단계에서는 중요하다고 할 수 있습니다.

Vector 1

```
numeric_vector <- c(1, 10, 49)
character_vector <- c("a", "b", "c")
boolean_vector <- c(TRUE, FALSE, TRUE)

length(numeric_vector) ## number of members in the vector
```

```
## [1] 3
```

```
new_vector <- c(numeric_vector, 50)
new_vector
```

```
## [1]  1 10 49 50
```

다른 타입이 섞여서 들어 갈 수는 없고, 한가지 타입만이 한 개이상 들어가 있는 변수의 형태를 Vector 라고 합니다. 정의: A vector is a sequence of data elements of the same basic type. typeof() 함수를 사용해서 위 세 개 vector가 어떤 type인지 확인 할 수 있습니다. vector의 특성; 예를 들면 길이는 length()라는 함수를 사용해 확인 할 수 있으며, vector의 다른 property도 여러 함수를 통해 확인 할 수 있습니다. 또, vector와 vector, vector와 constant를 이어 붙이고 싶다면 처음에 vector를 만드는 것과 같이(see new_vector above) 만들 수 있습니다.

Vector 2

```
name_vector = c("John","Bob","Sarah","Alice")
name_vector[1]
```

```
## [1] "John"
```

```
name_vector[5]
```

```
## [1] NA
```

```
name_vector[1:3]
```

```
## [1] "John"  "Bob"   "Sarah"
```

```
name_vector[-2]
```

```
## [1] "John"  "Sarah" "Alice"
```

```
name_vector[c(-1,-2)]
```

```
## [1] "Sarah" "Alice"
```

```
name_vector[c(1,3,4)]
```

```
## [1] "John"  "Sarah" "Alice"
```

Vector 인덱싱은 나중에 유용하게 쓰이게 됩니다. 다른 언어와는 다르게 인덱스는 1부터 시작합니다. [1]이 첫번째 element를 가리키는 것 입니다. 원하는 데이터를 골라내는데 유용한 인덱싱은 익숙해질 필요가 있습니다.

Vector 3

```
some_vector <- c("John Doe", "poker player")
names(some_vector) <- c("Name", "Profession")

some_vector
```

```
##           Name     Profession
##     "John Doe" "poker player"
```

```
some_vector['Name']

##       Name
## "John Doe"

some_vector['Profession']

##     Profession
## "poker player"

some_vector[1]

##       Name
## "John Doe"
```

vector에 “John Doe”, “poker player” 라는 element를 집어넣고 이들을 “Name”, “Profession”이라는 카테고리(names) 아래에 넣을 수 있습니다. 인덱싱하는 방법이 조금 헷갈릴 수 있으니 위 예시처럼 이것 저것 해보세요.

Vector 4

```
some_vector <- c(some_vector, "new1")
some_vector

##           Name     Profession
##     "John Doe" "poker player"         "new1"

some_vector <- c(some_vector, "newname2" = "new2")
some_vector

##           Name     Profession                       newname2
##     "John Doe" "poker player"         "new1"         "new2"

names(some_vector)[3] = "newname3"
some_vector

##           Name     Profession       newname3       newname2
##     "John Doe" "poker player"         "new1"         "new2"
```

vector에 element를 추가하고 name도 인덱싱을 통해 추가 할 수 있습니다.

Vector 5

```
weather_vector <- c("Mon" = "Sunny", "Tues" = "Rainy",
                    "Weds" = "Cloudy", "Thur" = "Foggy",
                    "Fri" = "Sunny", "Sat" = "Sunny",
                    "Sun" = "Cloudy")

weather_vector

##      Mon     Tues     Weds     Thur      Fri      Sat      Sun
##  "Sunny"  "Rainy" "Cloudy"  "Foggy"  "Sunny"  "Sunny" "Cloudy"

names(weather_vector)
```

```
## [1] "Mon"  "Tues" "Weds" "Thur" "Fri"  "Sat"  "Sun"
```

이전 예시처럼 1) element를 넣는 작업, 2) names를 정해주는 작업 을 한꺼번에 할 수 있습니다. = 표시를 통해 name과 element를 mapping 할 수 있습니다.

Shortcut to make numeric vector

```
a_vector <- 1:10 ## numbers from 1 to 10
b_vector <- seq(1, 10, 2) ## numbers from 1 to 10 increasing by 2

a_vector

##  [1]  1  2  3  4  5  6  7  8  9 10

b_vector

## [1] 1 3 5 7 9

c_vector <- rep(1:3, 3)
d_vector <- rep(1:3, each = 3)

c_vector

## [1] 1 2 3 1 2 3 1 2 3

d_vector

## [1] 1 1 1 2 2 2 3 3 3

c(a_vector, d_vector)

##  [1]  1  2  3  4  5  6  7  8  9 10  1  1  1  2  2  2  3  3  3
```

여러가지 방법을 통해 numeric vector를 손쉽게 만들어 낼 수 있습니다.

QUIZ

서로 다른 type의 vector를 합치면 어떻게 될까요 ?

some useful functions (sets)

```
a_vector <- c(1,5,2,6,7,2,3)
b_vector <- seq(1, 10, 3)

intersect(a_vector, b_vector)

## [1] 1 7

union(a_vector, b_vector)

## [1]  1  5  2  6  7  3  4 10

setdiff(a_vector, b_vector)
```

```
## [1] 5 2 6 3
```

```
unique(a_vector)
```

```
## [1] 1 5 2 6 7 3
```

intersect는 교집합, union은 합집합, setdiff는 차집합, unique는 중복을 제거한 unique한 수를 반환 합니다.

vector scalar computations

```
a_vector + 10
```

```
## [1] 11 15 12 16 17 12 13
```

```
a_vector > 4
```

```
## [1] FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE
```

```
sum(a_vector > 4)
```

```
## [1] 3
```

vector에 스칼라를 더하면 vector안의 모든 element에 값을 더한 값을 반환합니다.

vector vector computations

```
a_vector <- c(1,5,2,7,8)
b_vector <- seq(1,10,2)
a_vector - b_vector
```

```
## [1]  0  2 -3  0 -1
```

```
a_vector == b_vector
```

```
## [1]  TRUE FALSE FALSE  TRUE FALSE
```

```
sum(a_vector > (mean(a_vector)))
```

```
## [1] 3
```

vector의 길이가 다른데도 계산이 되기도 합니다. 하지만 그런 계산이 필요한 경우는 흔치 않습니다.

Vector Indexing(Selection)

```
sample_vector <- c(1,4,NA,2,1,NA,4,NA)
sample_vector[1:5]
```

```
## [1]  1  4 NA  2  1
```

```
sample_vector[c(1,3,5)]
```

```
## [1]  1 NA  1
```

```
sample_vector[-1]
```

```
## [1]  4 NA  2  1 NA  4 NA
```

```
sample_vector[c(T,T,F,T,F,T,F,T)]
```

```
## [1]  1  4  2 NA NA
```

```
sum(is.na(sample_vector))
```

```
## [1] 3
```

```
# Selecting non NA elements only(both work)
index <- !is.na(sample_vector)
index
```

```
## [1]  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE FALSE
```

```
index <- which(!is.na(sample_vector))
index
```

```
## [1] 1 2 4 5 7
```

```
sum(index)
```

```
## [1] 19
```

```
sample_vector[index]
```

```
## [1] 1 4 2 1 4
```

```
# Selecting those greater than the mean value
sample_vector <- c(1,7,8,99,5,15,17)
sample_vector[sample_vector > mean(sample_vector)]
```

```
## [1] 99
```

Matrix

```
new_hope <- c(460.998,314.4)
empire_strikes <- c(290.475,247.900)
return_jedi <- c(309.306, 165.8)

star_wars_matrix <- matrix(c(new_hope, empire_strikes, return_jedi), nrow = 3
, byrow =TRUE)
star_wars_matrix
```

```
##         [,1]  [,2]
## [1,] 460.998 314.4
## [2,] 290.475 247.9
## [3,] 309.306 165.8
```

```
region <- c("US", "non-US")
titles <- c("A New Hope", "The Empire Strikes Back","Return of the Jedi")
colnames(star_wars_matrix) <- region
```

```
rownames(star_wars_matrix) <- titles
star_wars_matrix
```

```
##                              US non-US
## A New Hope                460.998  314.4
## The Empire Strikes Back 290.475  247.9
## Return of the Jedi       309.306  165.8
```

```
rowSums(star_wars_matrix)
```

```
##              A New Hope The Empire Strikes Back      Return of the Jedi
##                 775.398                 538.375                 475.106
```

```
colSums(star_wars_matrix)
```

```
##       US   non-US
## 1060.779  728.100
```

```
rowMeans(star_wars_matrix)
```

```
##              A New Hope The Empire Strikes Back      Return of the Jedi
##                387.6990                269.1875                237.5530
```

```
colMeans(star_wars_matrix)
```

```
##      US  non-US
## 353.593 242.700
```

Adding new column with cbind

```
worldwide_vector <- rowSums(star_wars_matrix)
all_wars_matrix <- cbind(star_wars_matrix, worldwide_vector)
all_wars_matrix
```

```
##                              US non-US worldwide_vector
## A New Hope                460.998  314.4          775.398
## The Empire Strikes Back 290.475  247.9          538.375
## Return of the Jedi       309.306  165.8          475.106
```

Adding new rows with rbind

```
box_office <- c(474.5, 552.5, 310.7, 338.7, 380.3, 468.5)

star_wars_matrix2 <- matrix(box_office, nrow = 3, byrow = TRUE, dimnames = li
st(c("The Phantom Menace", "Attack of the Clones", "Revenge of the Sith"), c(
"US", "non-US")))

star_wars_matrix
```

```
##                              US non-US
## A New Hope                460.998  314.4
## The Empire Strikes Back 290.475  247.9
## Return of the Jedi       309.306  165.8
```

```
star_wars_matrix2

##                          US non-US
## The Phantom Menace   474.5  552.5
## Attack of the Clones 310.7  338.7
## Revenge of the Sith  380.3  468.5

all_wars_matrix <- rbind(star_wars_matrix, star_wars_matrix2)
all_wars_matrix

##                               US non-US
## A New Hope                460.998  314.4
## The Empire Strikes Back   290.475  247.9
## Return of the Jedi        309.306  165.8
## The Phantom Menace        474.500  552.5
## Attack of the Clones      310.700  338.7
## Revenge of the Sith       380.300  468.5
```

dimenames 는 행 -> 열 순으로 이름을 지어주는 역할을 합니다.

Selection of matrix elements

```
all_wars_matrix[1:3,1]

##              A New Hope The Empire Strikes Back      Return of the Jedi
##                 460.998                 290.475                 309.306

all_wars_matrix[1:3, 'non-US']

##              A New Hope The Empire Strikes Back      Return of the Jedi
##                   314.4                   247.9                   165.8

all_wars_matrix[,'non-US']

##              A New Hope The Empire Strikes Back      Return of the Jedi
##                   314.4                   247.9                   165.8
##      The Phantom Menace    Attack of the Clones     Revenge of the Sith
##                   552.5                   338.7                   468.5

all_wars_matrix[c(1,3,5),]

##                           US non-US
## A New Hope           460.998  314.4
## Return of the Jedi   309.306  165.8
## Attack of the Clones 310.700  338.7

all_wars_matrix[c(1,1,1),] ## *******

##                 US non-US
## A New Hope 460.998  314.4
## A New Hope 460.998  314.4
## A New Hope 460.998  314.4
```

대괄호 안에 인덱싱은 [행, 열] 입니다.

Matrix multiplication

```
A.mat <- matrix(1:9, byrow = TRUE, nrow = 3)
A.mat
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
```

```
B.mat <- matrix(rep(1:3, each = 3), byrow = TRUE, nrow = 3)
B.mat
```

```
##      [,1] [,2] [,3]
## [1,]    1    1    1
## [2,]    2    2    2
## [3,]    3    3    3
```

```
C.mat <- matrix(rep(1:3, 2), byrow = F, ncol = 2) # bycolumn
C.mat
```

```
##      [,1] [,2]
## [1,]    1    1
## [2,]    2    2
## [3,]    3    3
```

```
A.mat * 2
```

```
##      [,1] [,2] [,3]
## [1,]    2    4    6
## [2,]    8   10   12
## [3,]   14   16   18
```

```
A.mat -10
```

```
##      [,1] [,2] [,3]
## [1,]   -9   -8   -7
## [2,]   -6   -5   -4
## [3,]   -3   -2   -1
```

```
A.mat /5
```

```
##      [,1] [,2] [,3]
## [1,]  0.2  0.4  0.6
## [2,]  0.8  1.0  1.2
## [3,]  1.4  1.6  1.8
```

```
A.mat * B.mat
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
```

```
## [2,]    8   10   12
## [3,]   21   24   27

A.mat - B.mat

##      [,1] [,2] [,3]
## [1,]    0    1    2
## [2,]    2    3    4
## [3,]    4    5    6

A.mat / B.mat

##          [,1]     [,2] [,3]
## [1,] 1.000000 2.000000    3
## [2,] 2.000000 2.500000    3
## [3,] 2.333333 2.666667    3

## matrix multiplication
A.mat %*% C.mat

##      [,1] [,2]
## [1,]   14   14
## [2,]   32   32
## [3,]   50   50
```

Vector******

```
# Sex vector
sex_vector <- c("Male", "Female", "Female", "Female", "Male", "Male")
typeof(sex_vector)

## [1] "character"

# Convert sex_vector to a factor
factor_sex_vector <- factor(sex_vector)

# Print out factor_sex_vector
print(factor_sex_vector)

## [1] Male   Female Female Female Male   Male
## Levels: Female Male

typeof(factor_sex_vector)

## [1] "integer"

str(factor_sex_vector)

##  Factor w/ 2 levels "Female","Male": 2 1 1 1 2 2

levels(factor_sex_vector)

## [1] "Female" "Male"
```

```r
# Code to build factor_survey_vector
survey_vector <- c("M", "F", "F", "M", "M")
survey_vector
```

```
## [1] "M" "F" "F" "M" "M"
```

```r
factor_survey_vector <- factor(survey_vector)
factor_survey_vector
```

```
## [1] M F F M M
## Levels: F M
```

```r
# Specify the levels of factor_survey_vector
levels(factor_survey_vector) <- c('Female', 'Male')
factor_survey_vector
```

```
## [1] Male   Female Female Male   Male
## Levels: Female Male
```

```r
# Generate summary for survey_vector
summary(survey_vector)
```

```
##    Length     Class      Mode
##         5 character character
```

```r
typeof(survey_vector)
```

```
## [1] "character"
```

```r
# Generate summary for factor_survey_vector
summary(factor_survey_vector)
```

```
## Female   Male
##      2      3
```

```r
typeof(factor_survey_vector)
```

```
## [1] "integer"
```

DataFrame

```r
# Observing Data frame
head(mtcars,10)
```

```
##                    mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
```

```
## Merc 230          22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280          19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
```

```
tail(mtcars,10)
```

```
##                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## AMC Javelin      15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28       13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird 19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Fiat X1-9        27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2    26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa     30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ford Pantera L   15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino     19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora    15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Volvo 142E       21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

```
str(mtcars)
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

```
dim(mtcars)
```

```
## [1] 32 11
```

```
## Creating Data frame
name <- c("Mercury","Venus","Earth","Mars","Jupiter", "Saturn", "Uranus", "Ne
ptune")
type <- c("Terrestrial planet","Terrestrial planet","Terrestrial planet","Ter
restrial planet","Gas giant","Gas giant","Gas giant","Gas giant")

diameter <- c(0.382, 0.949, 1, 0.532, 11.209, 9.449, 4.007, 3.883)
rotation <- c(58.64, -243.02, 1, 1.03, 0.41, 0.43, -0.72, 0.67)
rings <- c(FALSE,FALSE,FALSE,FALSE,TRUE,TRUE,TRUE,TRUE)

## Create a data frame from the vectors
planets_df <- data.frame(name, type, diameter, rotation, rings)
planets_df
```

```
##      name               type diameter rotation rings
## 1 Mercury Terrestrial planet    0.382    58.64 FALSE
```

```
## 2   Venus Terrestrial planet    0.949  -243.02 FALSE
## 3   Earth Terrestrial planet    1.000     1.00 FALSE
## 4    Mars Terrestrial planet    0.532     1.03 FALSE
## 5 Jupiter          Gas giant   11.209     0.41  TRUE
## 6  Saturn          Gas giant    9.449     0.43  TRUE
## 7  Uranus          Gas giant    4.007    -0.72  TRUE
## 8 Neptune          Gas giant    3.883     0.67  TRUE
```

```
my.df <- data.frame(name = c('John', 'Kim', 'Kaith'),
                    job = c('Teacher', 'Policeman', 'Secretary'),
                    age = c(32,25,28))
my.df
```

```
##    name       job age
## 1  John   Teacher  32
## 2   Kim Policeman  25
## 3 Kaith Secretary  28
```

Selection of data frame elements - tricky

```
planets_df[1,3]
```

```
## [1] 0.382
```

```
planets_df[4, ]
```

```
##   name               type diameter rotation rings
## 4 Mars Terrestrial planet    0.532     1.03 FALSE
```

```
planets_df[1:5, 'diameter']
```

```
## [1]  0.382  0.949  1.000  0.532 11.209
```

```
#planets_df[,3]
#planets_df[,"diameter"]
#planets_df$diameter
planets_df[planets_df$rings, ] # rings -> T/F
```

```
##      name      type diameter rotation rings
## 5 Jupiter Gas giant   11.209     0.41  TRUE
## 6  Saturn Gas giant    9.449     0.43  TRUE
## 7  Uranus Gas giant    4.007    -0.72  TRUE
## 8 Neptune Gas giant    3.883     0.67  TRUE
```

```
planets_df[planets_df$rings, 'name']
```

```
## [1] Jupiter Saturn  Uranus  Neptune
## Levels: Earth Jupiter Mars Mercury Neptune Saturn Uranus Venus
```

```
planets_df$diameter > 1
```

```
## [1] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
```

```
planets_df[planets_df$diameter > 1, ]
```

```
##      name      type diameter rotation rings
## 5 Jupiter Gas giant   11.209     0.41  TRUE
## 6  Saturn Gas giant    9.449     0.43  TRUE
## 7  Uranus Gas giant    4.007    -0.72  TRUE
## 8 Neptune Gas giant    3.883     0.67  TRUE
```

List

```
# Vector with numerics from 1 up to 10
my_vector <- 1:10
# Matrix with numerics from 1 up to 9
my_matrix <- matrix(1:9, ncol = 3)
# First 10 elements of the built-in data frame mtcars
my_df <- mtcars[1:3,]
# Adapt list() call to give the components names
my_list <- list(vec = my_vector, mat = my_matrix, df = my_df)
# Print out my_list
my_list
```

```
## $vec
##  [1]  1  2  3  4  5  6  7  8  9 10
##
## $mat
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
##
## $df
##                mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4     21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710    22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
```

Selecting elements from a list

```
my_list[[1]]
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
my_list[['mat']]
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```
my_list$mat
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

Adding more componemts to the list

```
my_list$new_vector <- c(1,3,5,7,9)
str(my_list)
```

```
## List of 4
##  $ vec       : int [1:10] 1 2 3 4 5 6 7 8 9 10
##  $ mat       : int [1:3, 1:3] 1 2 3 4 5 6 7 8 9
##  $ df        :'data.frame':  3 obs. of  11 variables:
##   ..$ mpg : num [1:3] 21 21 22.8
##   ..$ cyl : num [1:3] 6 6 4
##   ..$ disp: num [1:3] 160 160 108
##   ..$ hp  : num [1:3] 110 110 93
##   ..$ drat: num [1:3] 3.9 3.9 3.85
##   ..$ wt  : num [1:3] 2.62 2.88 2.32
##   ..$ qsec: num [1:3] 16.5 17 18.6
##   ..$ vs  : num [1:3] 0 0 1
##   ..$ am  : num [1:3] 1 1 1
##   ..$ gear: num [1:3] 4 4 4
##   ..$ carb: num [1:3] 4 4 1
##  $ new_vector: num [1:5] 1 3 5 7 9
```

```
my_list[['new_vector']] # 리스트는 이렇게만 해도 값이 assign 됩니다.
```

```
## [1] 1 3 5 7 9
```

```
str(my_list)
```

```
## List of 4
##  $ vec       : int [1:10] 1 2 3 4 5 6 7 8 9 10
##  $ mat       : int [1:3, 1:3] 1 2 3 4 5 6 7 8 9
##  $ df        :'data.frame':  3 obs. of  11 variables:
##   ..$ mpg : num [1:3] 21 21 22.8
##   ..$ cyl : num [1:3] 6 6 4
##   ..$ disp: num [1:3] 160 160 108
##   ..$ hp  : num [1:3] 110 110 93
##   ..$ drat: num [1:3] 3.9 3.9 3.85
##   ..$ wt  : num [1:3] 2.62 2.88 2.32
##   ..$ qsec: num [1:3] 16.5 17 18.6
##   ..$ vs  : num [1:3] 0 0 1
##   ..$ am  : num [1:3] 1 1 1
##   ..$ gear: num [1:3] 4 4 4
##   ..$ carb: num [1:3] 4 4 1
##  $ new_vector: num [1:5] 1 3 5 7 9
```

Data Type - Scala

```
# Factor
gender <- factor("male", c("male", "female"))
gender
```

```
## [1] male
## Levels: male female
```

```r
nlevels(gender)
```

```
## [1] 2
```

```r
levels(gender)
```

```
## [1] "male"   "female"
```

```r
levels(gender)[1]
```

```
## [1] "male"
```

```r
# Ordered Factor **** ?
grade1 <- factor("A0", c("A+","A0","B+","B0","C+","C0","D+","D0","F"), ordere
d = T)
grade2 <- ordered("B+", c("A+","A0","B+","B0","C+","C0","D+","D0","F"))
grade1
```

```
## [1] A0
## Levels: A+ < A0 < B+ < B0 < C+ < C0 < D+ < D0 < F
```

```r
grade2
```

```
## [1] B+
## Levels: A+ < A0 < B+ < B0 < C+ < C0 < D+ < D0 < F
```

```r
grade1 > grade2
```

```
## [1] FALSE
```

```r
nlevels(grade1)
```

```
## [1] 9
```

```r
levels(grade2)
```

```
## [1] "A+" "A0" "B+" "B0" "C+" "C0" "D+" "D0" "F"
```

```r
x <- NULL
is.null(x)
```

```
## [1] TRUE
```

```r
is.null(1.5)
```

```
## [1] FALSE
```

```r
is.null(NA)
```

```
## [1] FALSE
```

```r
is.na(x)
```

```
## Warning in is.na(x): is.na() applied to non-(list or vector) of type 'NULL
'
```

```
## logical(0)
```

2_RBasics(b)

YJLEE

2018 3 19

IF Statement

```
medium <- "LinkedIn"
num_views <- 14

if(medium == "LinkedIn")
{
  print("Showing LinkedIn information")
}

## [1] "Showing LinkedIn information"

if(num_views > 15) # condition is not met
{
  print("You're popular!")
}
```

IF ELSE Statement

```
if(medium == "LinkedIn")
{
  print("Showing LinkedIn information")
}else{
  print("Unknown medium")
}

## [1] "Showing LinkedIn information"

if(num_views > 15)
{
  print("You're popular")
}else{
  print("try to be more visible")
}

## [1] "try to be more visible"
```

For Loop

```
cities <- c("New York", "Paris", "London", "Tokyo",
            "Rio de Janeiro", "Cape Town")

for(city in cities)
{
  print(city)
}
```

```
## [1] "New York"
## [1] "Paris"
## [1] "London"
## [1] "Tokyo"
## [1] "Rio de Janeiro"
## [1] "Cape Town"
```

Vectorized Operation 1

How could you make 'numbers_even_odd' vector from numbers_vector?

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

numbers_even_odd <- ifelse(numbers_vector %% 2 == 0, 'even', 'odd')
numbers_even_odd
```

```
## [1] "odd"  "odd"  "even" "even" "even" "even" "odd"  "odd"
```

```
table(numbers_even_odd)
```

```
## numbers_even_odd
## even  odd
##    4    4
```

이 연산을 하기 위해 for 문을 사용하게 되면 loop 횟수 만큼 벡터를 생성하게 됩니다(비효율적). 예를 들어 각 element에 10을 더하는 연산으로 loop가 3번 돈다고 하면 c(1,3,4) -> c(11,3,4),c(11,13,4),c(11,13,14) 이렇게 3개의 벡터가 메모리에 생성됩니다. 3개의 element를 한 번에 병렬 연산(multi-core; 여러 개의 CPU)해주는 것이 c(1,3,4) -> c(11,13,14) vectorized operation입니다.

멀티 코어(multi-core) CPU는 두 개 이상의 독립 코어를 단일 집적 회로로 이루어진 하나의 패키지로 통합한 것.

Vectorized Operation 2

DataFrame에서 한 column도 vector이기 때문에 DF에도 vectorized operation을 적용할 수 있다.

Adding New Variable "Fuel_efficiency" to mtcars

```
avg_mpg <- mean(mtcars$mpg)
new_var <- ifelse(mtcars$mpg >= avg_mpg, 'good', 'bad')
```

```r
# adding new variable to mtcars
mtcars$fuel_efficiency <- new_var
dim(mtcars)
```

```
## [1] 32 12
```

```r
head(mtcars[,c(10:12)])
```

```
##                   gear carb fuel_efficiency
## Mazda RX4            4    4            good
## Mazda RX4 Wag        4    4            good
## Datsun 710           4    1            good
## Hornet 4 Drive       3    1            good
## Hornet Sportabout    3    2             bad
## Valiant              3    1             bad
```

Function(User-defined Functions 1)

```r
cube <- function(n)
{
  return(n*n*n)
}

cube(10)
```

```
## [1] 1000
```

```r
cube(1:5)
```

```
## [1]   1   8  27  64 125
```

Function(User-defined Functions 2)

```r
is.even.number <- function(n)
{
  n %% 2 == 0
}

is.even.number(10)
```

```
## [1] TRUE
```

```r
is.even.number(c(1,2,5,6,7,9,15))
```

```
## [1] FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE
```

```r
ifelse(is.even.number(numbers_vector), 'even', 'odd')
```

```
## [1] "odd"  "odd"  "even" "even" "even" "even" "odd"  "odd"
```

Function(User-defined Functions 3)

```r
diff.max.min <- function(...)
{
```

```
  a <- c(...)
  largest <- max(a)
  smallest <- min(a)

  largest - smallest
}

diff.max.min(6,5,6,23,4,25)

## [1] 21

diff.max.min(-55,100,23,-7)

## [1] 155

diff.max.min(c(1,2,3,4), c(10,20,30,40))  # 두 개의 벡터를 넣어도 전체의 min max를 사용한다.

## [1] 39
```

Vectorized operations 3

```
my.vector <- c(1,3,5,8,13)
my.vector * 2

## [1]  2  6 10 16 26

my.vector >= 5

## [1] FALSE FALSE  TRUE  TRUE  TRUE

my.vector < 10

## [1]  TRUE  TRUE  TRUE  TRUE FALSE

my.vector >= 5 & my.vector < 10

## [1] FALSE FALSE  TRUE  TRUE FALSE

sum(my.vector)

## [1] 30

mean(my.vector)

## [1] 6

median(my.vector)

## [1] 5

min(my.vector)

## [1] 1
```

```r
max(my.vector)
```

```
## [1] 13
```

```r
summary(my.vector)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1       3       5       6       8      13
```

```r
ifelse(my.vector %% 2== 0, 'even', 'odd')
```

```
## [1] "odd"  "odd"  "odd"  "even" "odd"
```

ApplyFamily

YJLEE

2018 3 22

apply()

apply(X, MARGIN, FUN, ...)* ##### X is matrix or dataframe ##### MARGIN is a variable defining how the function is appled: 1 -> over rows, 2 -> over columns ##### FUN is the function that you want to apply to the data

```r
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```r
apply(iris[,1:4], 2, mean)
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##     5.843333     3.057333     3.758000     1.199333
```

```r
colMeans(iris[,1:4])
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##     5.843333     3.057333     3.758000     1.199333
```

```r
mean <- apply(iris[,1:4], 1, mean)
iris$mean <- mean
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species  mean
## 1          5.1         3.5          1.4         0.2  setosa 2.550
## 2          4.9         3.0          1.4         0.2  setosa 2.375
## 3          4.7         3.2          1.3         0.2  setosa 2.350
## 4          4.6         3.1          1.5         0.2  setosa 2.350
## 5          5.0         3.6          1.4         0.2  setosa 2.550
## 6          5.4         3.9          1.7         0.4  setosa 2.850
```

```
max <- apply(mtcars, 2, max)
max
```

```
##     mpg     cyl    disp      hp    drat      wt    qsec      vs      am
##  33.900   8.000 472.000 335.000   4.930   5.424  22.900   1.000   1.000
##    gear    carb
##   5.000   8.000
```

```
as.data.frame(max)
```

```
##          max
## mpg   33.900
## cyl    8.000
## disp 472.000
## hp   335.000
## drat   4.930
## wt     5.424
## qsec  22.900
## vs     1.000
## am     1.000
## gear   5.000
## carb   8.000
```

lapply() : Over components of a list; DF에 대해서는 column별로감

It applies function to dataframes, lists or vectors

It gives you back a 'list'

```
## to list
myList<- list(num = 3.14, chr = "char", logi = TRUE)
myList
```

```
## $num
## [1] 3.14
##
## $chr
## [1] "char"
##
## $logi
## [1] TRUE
```

```
lapply(myList, typeof)
```

```
## $num
## [1] "double"
##
## $chr
## [1] "character"
##
## $logi
## [1] "logical"
```

```
myList2 <- list(vec = 1:5, mat = matrix(runif(12), ncol = 4), df = iris)
myList2
```

```
## $vec
## [1] 1 2 3 4 5
##
## $mat
##           [,1]      [,2]      [,3]      [,4]
## [1,] 0.3988907 0.1253901 0.5265000 0.4766447
## [2,] 0.9388178 0.6102111 0.1778257 0.4557473
## [3,] 0.5041724 0.1531245 0.5280286 0.5244727
##
## $df
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species  mean
## 1            5.1         3.5          1.4         0.2     setosa 2.550
---
## 50           5.0         3.3          1.4         0.2     setosa 2.475
## 51           7.0         3.2          4.7         1.4 versicolor 4.075
## 52           6.4         3.2          4.5         1.5 versicolor 3.900
---
## 146          6.7         3.0          5.2         2.3  virginica 4.300
## 147          6.3         2.5          5.0         1.9  virginica 3.925
## 148          6.5         3.0          5.2         2.0  virginica 4.175
## 149          6.2         3.4          5.4         2.3  virginica 4.325
## 150          5.9         3.0          5.1         1.8  virginica 3.950
```

```
result <- lapply(myList2, length) # dataframe의 length는 column의 개 수!!!
result
```

```
## $vec
## [1] 5
##
## $mat
## [1] 12
##
## $df
## [1] 6
```

```
unlist(result) # list -> vector
```

```
## vec mat  df
##   5  12   6
```

```r
## to vector
lapply(c(1,4,9,16), sqrt)
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] 2
##
## [[3]]
## [1] 3
##
## [[4]]
## [1] 4
```

```r
## to data frame
unlist(lapply(iris[,1:4], mean))
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##     5.843333     3.057333     3.758000     1.199333
```

```r
## to data frame; can implement this task with apply()
lapply(mtcars, max)
```

```
## $mpg
## [1] 33.9
##
## $cyl
## [1] 8
##
## $disp
## [1] 472
##
## $hp
## [1] 335
##
## $drat
## [1] 4.93
##
## $wt
## [1] 5.424
##
## $qsec
## [1] 22.9
##
## $vs
## [1] 1
##
## $am
## [1] 1
##
```

```
## $gear
## [1] 5
##
## $carb
## [1] 8
```

```
a <- unlist(lapply(mtcars, max))
a[2] * 10
```

```
## cyl
##  80
```

```
as.data.frame(a)
```

```
##            a
## mpg   33.900
## cyl    8.000
## disp 472.000
## hp   335.000
## drat   4.930
## wt     5.424
## qsec  22.900
## vs     1.000
## am     1.000
## gear   5.000
## carb   8.000
```

sapply()*** ##### It applies function to dataframes, lists or vectors ##### It gives you back a vector or matrix

```
sapply(iris[,1:4], mean)
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##     5.843333     3.057333     3.758000     1.199333
```

```
sapply(iris, is.numeric) # 숫자 column만 뽑고 싶을 때 이걸 쓰면 되겠네!
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
##         TRUE         TRUE         TRUE         TRUE        FALSE
##         mean
##         TRUE
```

```
# number of NAs over columns
sapply(iris[, 1:4], function(x) {sum(is.na(x))})
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##            0            0            0            0
```

```
sapply(iris[, 1:4], function(x) {x**2})
```

```
##        Sepal.Length Sepal.Width Petal.Length Petal.Width
##   [1,]        26.01       12.25         1.96        0.04
##   [2,]        24.01        9.00         1.96        0.04
```

```
---
## [150,]        34.81        9.00        26.01        3.24
```

```
## 벡터에 적용하면 벡터 반환
sapply(c(1,3,5,7,9), function(x) {x**2})
```

```
## [1]  1  9 25 49 81
```

```
## 매트리스에 적용하면 벡터 반환
myMat <- matrix(1:12, ncol = 4)
colnames(myMat) <- c("a","b", "c", "d")
myMat
```

```
##      a b c  d
## [1,] 1 4 7 10
## [2,] 2 5 8 11
## [3,] 3 6 9 12
```

```
sapply(myMat, function(x) {x/2})
```

```
##  [1] 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0
```

```
## DF에 적용하면 Matrix로 줌
# sapply(pools, typeof)
```

```
sapply(iris[,1:4], function(x) {x/2})
```

```
##        Sepal.Length Sepal.Width Petal.Length Petal.Width
##   [1,]         2.55        1.75         0.70        0.10
##   [2,]         2.45        1.50         0.70        0.10
---
## [149,]         3.10        1.70         2.70        1.15
## [150,]         2.95        1.50         2.55        0.90
```

```
x <- sapply(iris[,1:4], function(x) {x> 3})
head(x)
```

```
##      Sepal.Length Sepal.Width Petal.Length Petal.Width
## [1,]         TRUE        TRUE        FALSE       FALSE
## [2,]         TRUE       FALSE        FALSE       FALSE
## [3,]         TRUE        TRUE        FALSE       FALSE
## [4,]         TRUE        TRUE        FALSE       FALSE
## [5,]         TRUE        TRUE        FALSE       FALSE
## [6,]         TRUE        TRUE        FALSE       FALSE
```

```
colSums(x)
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##          150           67           99             0
```

tapply()* #### tapply(X, GRP_VAR, FUN, ...) ##### apply FUN to X after grouping with GRP_VAR

```
tapply(iris$Sepal.Length, iris$Species, mean) ## calculate the means of Sepal
.Length according to Species
```

```
##     setosa versicolor  virginica
##      5.006      5.936      6.588
```

```
tapply(mtcars$mpg, mtcars$cyl, mean) # calculate the menas of mpg according t
o cyl value
```

```
##        4        6        8
## 26.66364 19.74286 15.10000
```

```
tapply(mtcars$wt, mtcars$mpg>20, mean) # calculate the means of wt for those
mpg values greater than 20(TRUE), for others (FALSE)
```

```
##    FALSE     TRUE
## 3.838833 2.418071
```

```
x <- tapply(mtcars$mpg, mtcars$cyl, function(x){x>20}) # returns a list
x
```

```
## $`4`
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
##
## $`6`
## [1]  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE
##
## $`8`
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [12] FALSE FALSE FALSE
```

```
sapply(x, sum)
```

```
##  4  6  8
## 11  3  0
```

GRP_VAR로 그룹핑해서 X의 value 하나 하나에 접근해서 FUN을 적용

aggregate()

aggregate(var1 ~ var2, data = X, FUN = func, ...)

Apply func to var1 of X after grouping by var2

Alternates to tapply

Result is data.frame

```
#aggregate(mpg ~ cyl, data = mtcars, FUN = mean)
#aggregate(Sepal.Length ~ Species, data = iris, FUN = mean)
#aggregate(mpg ~ cyl + am, mtcars, FUN = mean)
```

order() vs sort()

order() gives a vector of index of smallest element, second smallest, ..., the largest element

sort() gives a sorted vector of numbers

```
my_vector <- c(6,12,4,89,23, 35)
order(my_vector) # returns the order in index value

## [1] 3 1 2 5 6 4

my_vector[order(my_vector)] # sorts the actual values

## [1]  4  6 12 23 35 89

sort(my_vector) # sorts the actual values

## [1]  4  6 12 23 35 89

my_vector[order(my_vector,decreasing = T)]

## [1] 89 35 23 12  6  4

sort(my_vector, decreasing = T)

## [1] 89 35 23 12  6  4

## Why not sort ? - to sort dataframes according to a specific column
mtcars[order(mtcars$mpg, decreasing = T), ]

##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
```

```
## Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
## Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
```

no need to use sort(). Use order()

5_RBasics(c)

YJLEE

2018 3 26

Some useful functions

Sample()

```r
set.seed(2018)
x <- 1:20
sample(x, 10, replace = TRUE) # from x, generate 10, replace ok
```

```
##  [1]  7 10  2  4 10  7 13  3 20 11
```

```r
sample(x, 10, replace = FALSE)
```

```
##  [1]  8 13 18 12 19 10  4 20  9 15
```

```r
# random shuffling
x <- 1:10
sample(x, length(x), replace = FALSE)
```

```
##  [1]  3  6  2  1  5 10  8  9  4  7
```

```r
women_shuffle <- women[sample(1:nrow(women), 5, replace = FALSE), ] # shuffle and sample(n = 5)
head(women)
```

```
##   height weight
## 1     58    115
## 2     59    117
## 3     60    120
## 4     61    123
## 5     62    126
## 6     63    129
```

```r
women_shuffle
```

```
##    height weight
## 3      60    120
## 13     70    154
## 1      58    115
## 6      63    129
## 12     69    150
```

Split() * #### split(df, split_var, ...) ##### Split a data frame into a list of data frames with(according to) split variable

```r
lst <- split(mtcars, mtcars$cyl) # cyl은 범주형이어야겠네
typeof(lst)
```

```
## [1] "list"
```

```r
lst
```

```
## $`4`
##                 mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Datsun 710     22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Merc 240D      24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230       22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona  21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
##
## $`6`
```

```
##                  mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4       21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag   21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Hornet 4 Drive 21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Valiant         18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Merc 280        19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C       17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Ferrari Dino    19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
##
## $`8`
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
```

```
# 중요
vec <- mtcars$mpg > 20
lst2 <- split(mtcars, vec)
typeof(lst2)
```

```
## [1] "list"
```

```
lst2
```

```
## $`FALSE`
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
```

```
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
##
## $`TRUE`
##                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4        21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag    21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710       22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive   21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Merc 240D        24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230         22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Fiat 128         32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic      30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla   33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona    21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Fiat X1-9        27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2    26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa     30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Volvo 142E       21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

split_var가 df의 변수일 필요는 없음. 길이만 맞으면 됨(벡터일수도잇음 see lst2).
시험각이네

Subset()

subset(df, condition, ...)

Find a subset of dataframe with a criteria

```
subset(mtcars, mpg > 25) # DF에서 mpg가 25보다 큰 케이스만 빼내
```

```
##                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Fiat 128         32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic      30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla   33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Fiat X1-9        27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2    26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa     30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
```

```
mtcars[mtcars$mpg > 25, ] # 이렇게도 할 수 있음
```

```
##                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Fiat 128         32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic      30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla   33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Fiat X1-9        27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2    26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa     30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
```

Merge() **** 어렵

Merge(df1, df2, ...)

join two data frames into one with common variables

```
(x <- data.frame( name = c("John", "Bob", "Carol"), math = c(70,80,90)))
```

```
##    name math
## 1  John   70
## 2   Bob   80
## 3 Carol   90
```

```
(y <- data.frame( name = c("John", "Bob", "Alice"), history = c(100,55,75)))
```

```
##    name history
## 1  John     100
## 2   Bob      55
## 3 Alice      75
```

```
merge(x,y) # inner join
```

```
##   name math history
## 1  Bob   80      55
## 2 John   70     100
```

```
merge(x,y,all = T) # outer join
```

```
##    name math history
## 1   Bob   80      55
## 2 Carol   90      NA
## 3  John   70     100
## 4 Alice   NA      75
```

```
merge(x,y,all = T, by.x = "math")
```

```
##    math  name history
## 1    70  John      NA
## 2    80   Bob      NA
## 3    90 Carol      NA
## 4 Alice  <NA>      75
## 5   Bob  <NA>      55
## 6  John  <NA>     100
```

```
merge(x,y,all = T, by.y = "history")
```

```
## Warning in `[<-.factor`(`*tmp*`, ri, value = c(100, 55, 75)): invalid
## factor level, NA generated
```

```
## Warning in merge.data.frame(x, y, all = T, by.y = "history"): column name
## 'name' is duplicated in the result
```

```
##    name math  name
## 1   Bob   80  <NA>
## 2 Carol   90  <NA>
## 3  John   70  <NA>
## 4  <NA>   NA  John
## 5  <NA>   NA   Bob
## 6  <NA>   NA Alice
```

```
merge(x,y,all.x = T)
```

```
##    name math history
## 1   Bob   80      55
## 2 Carol   90      NA
## 3  John   70     100
```

```
merge(x,y,all.y = T)
```

```
##    name math history
## 1   Bob   80      55
## 2  John   70     100
## 3 Alice   NA      75
```

by.x by.y!

Which()

Find positions of elements that satisfy the condition

```
x <- c(5,1,2,6,3,17,8,9,12)
x > 10
```

```
## [1] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE
```

```
(myindex <- which( x > 10)) # returns index
```

```
## [1] 6 9
```

```
x[myindex] # returns actual value
```

```
## [1] 17 12
```

```
# returns index
which.max(x)
```

```
## [1] 6
```

```
which.min(x)
```

```
## [1] 2
```

```
# returns actual value
x[which.max(x)] # = max(x)
```

```
## [1] 17
```

```r
x[which.min(x)] # = min(x)
```

```
## [1] 1
```

```r
mtcars[which.max(mtcars$mpg),]
```

```
##                 mpg cyl disp hp drat    wt qsec vs am gear carb
## Toyota Corolla 33.9   4 71.1 65 4.22 1.835 19.9  1  1    4    1
```

```r
which.maxn(mtcars$mpg,5) # returns index
```

```
## [1] 20 18 19 28 26
```

```r
mtcars[which.maxn(mtcars$mpg,5),] ################# ******* 'doBy' package #
top 5 mpg cases
```

```
##                 mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Toyota Corolla 33.9   4 71.1  65 4.22 1.835 19.90  1  1    4    1
## Fiat 128       32.4   4 78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic    30.4   4 75.7  52 4.93 1.615 18.52  1  1    4    2
## Lotus Europa   30.4   4 95.1 113 3.77 1.513 16.90  1  1    5    2
## Fiat X1-9      27.3   4 79.0  66 4.08 1.935 18.90  1  1    4    1
```

```r
mtcars[order(mtcars$mpg, decreasing = T)[1], ] # top 1 mpg case
```

```
##                 mpg cyl disp hp drat    wt qsec vs am gear carb
## Toyota Corolla 33.9   4 71.1 65 4.22 1.835 19.9  1  1    4    1
```

```r
mtcars[order(mtcars$mpg, decreasing = T), ][1,] # same
```

```
##                 mpg cyl disp hp drat    wt qsec vs am gear carb
## Toyota Corolla 33.9   4 71.1 65 4.22 1.835 19.9  1  1    4    1
```

```r
mtcars[mtcars$mpg == max(mtcars$mpg),] # same
```

```
##                 mpg cyl disp hp drat    wt qsec vs am gear carb
## Toyota Corolla 33.9   4 71.1 65 4.22 1.835 19.9  1  1    4    1
```

cut()

makes a range-group(factor) variable; continuous to factor(비주얼빈)

```r
mtcars$wt
```

```
##  [1] 2.620 2.875 2.320 3.215 3.440 3.460 3.570 3.190 3.150 3.440 3.440
## [12] 4.070 3.730 3.780 5.250 5.424 5.345 2.200 1.615 1.835 2.465 3.520
## [23] 3.435 3.840 3.845 1.935 2.140 1.513 3.170 2.770 3.570 2.780
```

```r
(mtcars$wt_grp <- cut(mtcars$wt, breaks = c(-Inf,0,2,4,6,Inf))) # 0 < x =< 2
```

```
##  [1] (2,4] (2,4] (2,4] (2,4] (2,4] (2,4] (2,4] (2,4] (2,4] (2,4] (2,4]
## [12] (4,6] (2,4] (2,4] (4,6] (4,6] (4,6] (2,4] (0,2] (0,2] (2,4] (2,4]
## [23] (2,4] (2,4] (2,4] (0,2] (2,4] (0,2] (2,4] (2,4] (2,4] (2,4]
## Levels: (-Inf,0] (0,2] (2,4] (4,6] (6, Inf]
```

```r
mtcars[,c('wt','wt_grp')]
```

```
##                        wt wt_grp
## Mazda RX4           2.620  (2,4]
## Mazda RX4 Wag       2.875  (2,4]
## Datsun 710          2.320  (2,4]
## Hornet 4 Drive      3.215  (2,4]
## Hornet Sportabout   3.440  (2,4]
## Valiant             3.460  (2,4]
## Duster 360          3.570  (2,4]
## Merc 240D           3.190  (2,4]
## Merc 230            3.150  (2,4]
## Merc 280            3.440  (2,4]
## Merc 280C           3.440  (2,4]
## Merc 450SE          4.070  (4,6]
## Merc 450SL          3.730  (2,4]
## Merc 450SLC         3.780  (2,4]
## Cadillac Fleetwood  5.250  (4,6]
## Lincoln Continental 5.424  (4,6]
## Chrysler Imperial   5.345  (4,6]
## Fiat 128            2.200  (2,4]
## Honda Civic         1.615  (0,2]
## Toyota Corolla      1.835  (0,2]
## Toyota Corona       2.465  (2,4]
## Dodge Challenger    3.520  (2,4]
## AMC Javelin         3.435  (2,4]
## Camaro Z28          3.840  (2,4]
## Pontiac Firebird    3.845  (2,4]
## Fiat X1-9           1.935  (0,2]
## Porsche 914-2       2.140  (2,4]
## Lotus Europa        1.513  (0,2]
## Ford Pantera L      3.170  (2,4]
## Ferrari Dino        2.770  (2,4]
## Maserati Bora       3.570  (2,4]
## Volvo 142E          2.780  (2,4]
```

Quantile

to find out percentiles

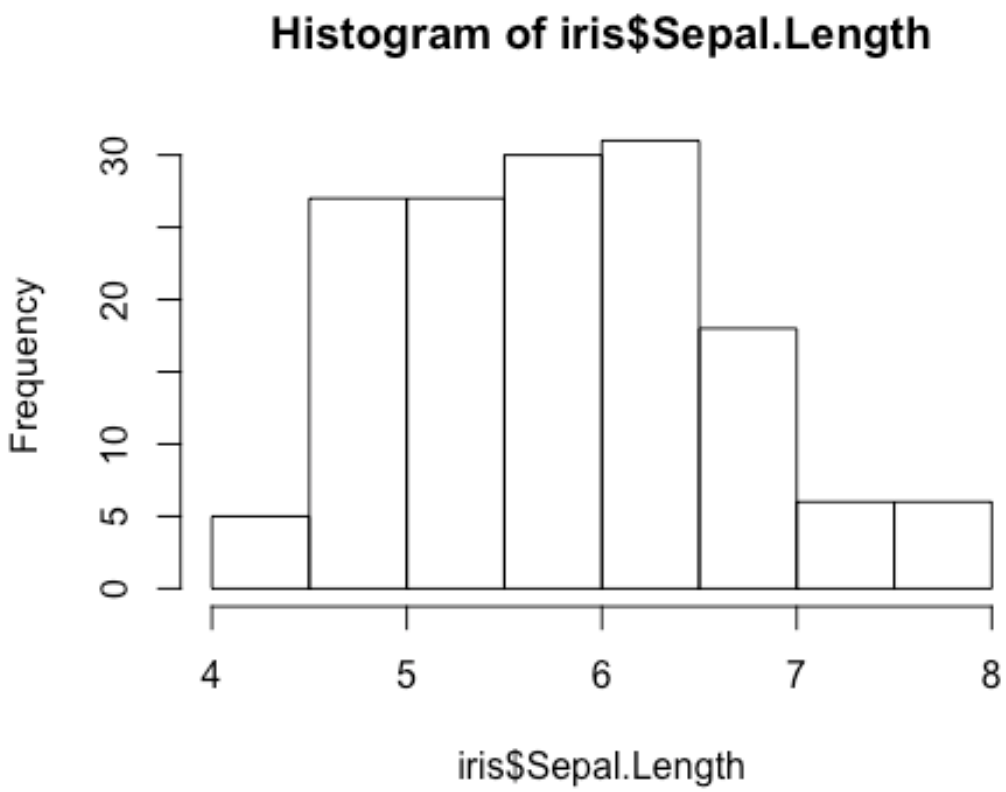
```r
quantile(iris$Sepal.Length)
```

```
##   0%  25%  50%  75% 100%
##  4.3  5.1  5.8  6.4  7.9
```

```r
quantile(iris$Sepal.Length, probs = c(0.1,0.5,0.9))
```

```
## 10% 50% 90%
## 4.8 5.8 6.9
```

```r
hist(iris$Sepal.Length)
```



Combination of quantile and cut ***** 중요

```
(cut_points <- quantile(mtcars$mpg, c(0,0.25,0.75,1)))
```

```
##     0%    25%    75%   100%
## 10.400 15.425 22.800 33.900
```

```
(mtcars$fuel_efficiency <- cut(mtcars$mpg, breaks = cut_points, include.lowes
t = T))
```

```
##  [1] (15.4,22.8] (15.4,22.8] (15.4,22.8] (15.4,22.8] (15.4,22.8]
##  [6] (15.4,22.8] [10.4,15.4] (22.8,33.9] (15.4,22.8] (15.4,22.8]
## [11] (15.4,22.8] (15.4,22.8] (15.4,22.8] [10.4,15.4] [10.4,15.4]
## [16] [10.4,15.4] [10.4,15.4] (22.8,33.9] (22.8,33.9] (22.8,33.9]
## [21] (15.4,22.8] (15.4,22.8] [10.4,15.4] [10.4,15.4] (15.4,22.8]
## [26] (22.8,33.9] (22.8,33.9] (22.8,33.9] (15.4,22.8] (15.4,22.8]
## [31] [10.4,15.4] (15.4,22.8]
## Levels: [10.4,15.4] (15.4,22.8] (22.8,33.9]
```

```
head(mtcars[,c('mpg', 'fuel_efficiency')])
```

```
##                    mpg fuel_efficiency
## Mazda RX4         21.0     (15.4,22.8]
## Mazda RX4 Wag     21.0     (15.4,22.8]
## Datsun 710        22.8     (15.4,22.8]
```

```
## Hornet 4 Drive    21.4     (15.4,22.8]
## Hornet Sportabout 18.7     (15.4,22.8]
## Valiant           18.1     (15.4,22.8]
```

```
(levels(mtcars$fuel_efficiency) <- c('low25pec', 'normal', 'high25perc'))
```

```
## [1] "low25pec"   "normal"     "high25perc"
```

```
head(mtcars[,c('mpg', 'fuel_efficiency')], 8)
```

```
##                   mpg fuel_efficiency
## Mazda RX4         21.0          normal
## Mazda RX4 Wag     21.0          normal
## Datsun 710        22.8          normal
## Hornet 4 Drive    21.4          normal
## Hornet Sportabout 18.7          normal
## Valiant           18.1          normal
## Duster 360        14.3        low25pec
## Merc 240D         24.4      high25perc
```

Frequency table

```
table(mtcars$fuel_efficiency)
```

```
##
##   low25pec     normal high25perc
##          8         17          7
```

```
table(mtcars$cyl)
```

```
##
##  4  6  8
## 11  7 14
```

```
table(mtcars$fuel_efficiency, mtcars$cyl)
```

```
##
##              4 6 8
##   low25pec   0 0 8
##   normal     4 7 6
##   high25perc 7 0 0
```

```
prop.table(table(mtcars$mpg > 20))
```

```
##
##  FALSE   TRUE
## 0.5625 0.4375
```

paste and paste0: formula만들 때, 여러 개의 csv 파일을 읽어올 때

to concatenate several values into one string

to concatenate element by element from 2 or more vectors

to smash vector elements into one string

need to use 'sep' and 'collapse' option properly

useful to generate column names and row names

paste0 equals to paste(..., sep = '')

```
paste("one", 1, "test")

## [1] "one 1 test"

(x <- seq(2, 20, 2))

##  [1]  2  4  6  8 10 12 14 16 18 20

(y <- LETTERS[1:10])

##  [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J"

paste(x,y)

##  [1] "2 A"  "4 B"  "6 C"  "8 D"  "10 E" "12 F" "14 G" "16 H" "18 I" "20 J"

paste(x,y, sep = ':')

##  [1] "2:A"  "4:B"  "6:C"  "8:D"  "10:E" "12:F" "14:G" "16:H" "18:I" "20:J"

paste('var', x) # vector

##  [1] "var 2"  "var 4"  "var 6"  "var 8"  "var 10" "var 12" "var 14"
##  [8] "var 16" "var 18" "var 20"

paste0('var', x)

##  [1] "var2"  "var4"  "var6"  "var8"  "var10" "var12" "var14" "var16"
##  [9] "var18" "var20"

paste('var', x, y, sep = '-')

##  [1] "var-2-A"  "var-4-B"  "var-6-C"  "var-8-D"  "var-10-E" "var-12-F"
##  [7] "var-14-G" "var-16-H" "var-18-I" "var-20-J"

paste(x)

##  [1] "2"  "4"  "6"  "8"  "10" "12" "14" "16" "18" "20"

paste(x, collapse = ',') # scalar
```

```
## [1] "2,4,6,8,10,12,14,16,18,20"
```

```
paste(paste0(x,y), collapse = ',')
```

```
## [1] "2A,4B,6C,8D,10E,12F,14G,16H,18I,20J"
```

6_DataPreparation

YJLEE

2018 4 12

Data Exploration

```
bmi<- read.csv(file = 'bmi_clean.csv')
```

```
class(bmi)
```

```
## [1] "data.frame"
```

```
dim(bmi)
```

```
## [1] 199  30
```

```
names(bmi)
```

```
##  [1] "Country" "Y1980"   "Y1981"   "Y1982"   "Y1983"   "Y1984"   "Y1985"
##  [8] "Y1986"   "Y1987"   "Y1988"   "Y1989"   "Y1990"   "Y1991"   "Y1992"
## [15] "Y1993"   "Y1994"   "Y1995"   "Y1996"   "Y1997"   "Y1998"   "Y1999"
## [22] "Y2000"   "Y2001"   "Y2002"   "Y2003"   "Y2004"   "Y2005"   "Y2006"
## [29] "Y2007"   "Y2008"
```

```
str(bmi)
```

```
## 'data.frame':    199 obs. of  30 variables:
##  $ Country: Factor w/ 199 levels "Afghanistan",..: 1 2 3 4 5 6 7 8 9 10 ..
.
##  $ Y1980  : num  21.5 25.2 22.3 25.7 20.9 ...
##  $ Y1981  : num  21.5 25.2 22.3 25.7 20.9 ...
##  $ Y1982  : num  21.5 25.3 22.4 25.7 20.9 ...

---
##  $ Y2007  : num  20.6 26.3 24.5 27.5 22.1 ...
##  $ Y2008  : num  20.6 26.4 24.6 27.6 22.3 ...
```

```
glimpse(bmi)
```

```
## Observations: 199
## Variables: 30
## $ Country <fct> Afghanistan, Albania, Algeria, Andorra, Angola, Antigu...
## $ Y1980   <dbl> 21.48678, 25.22533, 22.25703, 25.66652, 20.94876, 23.3...
## $ Y1981   <dbl> 21.46552, 25.23981, 22.34745, 25.70868, 20.94371, 23.3...
---
## $ Y2007   <dbl> 20.60246, 26.32753, 24.48846, 27.53363, 22.08962, 25.6...
## $ Y2008   <dbl> 20.62058, 26.44657, 24.59620, 27.63048, 22.25083, 25.7...
```

```
summary(bmi)
```

```
##                 Country        Y1980           Y1981           Y1982      
##  Afghanistan        :  1   Min.   :19.01   Min.   :19.04   Min.   :19.07  
##  Albania            :  1   1st Qu.:21.27   1st Qu.:21.31   1st Qu.:21.36  
##  Algeria            :  1   Median :23.31   Median :23.39   Median :23.46  
##  Andorra            :  1   Mean   :23.15   Mean   :23.21   Mean   :23.26  
##  Angola             :  1   3rd Qu.:24.82   3rd Qu.:24.89   3rd Qu.:24.94  
##  Antigua and Barbuda:  1   Max.   :28.12   Max.   :28.36   Max.   :28.58  
##  (Other)            :193                                                  
##      Y2003           Y2004           Y2005           Y2006      
##  Min.   :19.81   Min.   :19.79   Min.   :19.79   Min.   :19.80  
##  1st Qu.:22.37   1st Qu.:22.45   1st Qu.:22.54   1st Qu.:22.63  
##  Median :24.89   Median :25.00   Median :25.11   Median :25.24  
##  Mean   :24.61   Mean   :24.70   Mean   :24.79   Mean   :24.89  
##  3rd Qu.:26.38   3rd Qu.:26.47   3rd Qu.:26.53   3rd Qu.:26.59  
##  Max.   :32.90   Max.   :33.10   Max.   :33.30   Max.   :33.49  
##                                                                 
##      Y2007           Y2008      
##  Min.   :19.83   Min.   :19.87  
##  1st Qu.:22.73   1st Qu.:22.83  
##  Median :25.36   Median :25.50  
##  Mean   :24.99   Mean   :25.10  
##  3rd Qu.:26.66   3rd Qu.:26.82  
##  Max.   :33.69   Max.   :33.90  
##
```

```
#head(bmi)
head(bmi, n = 2)
```

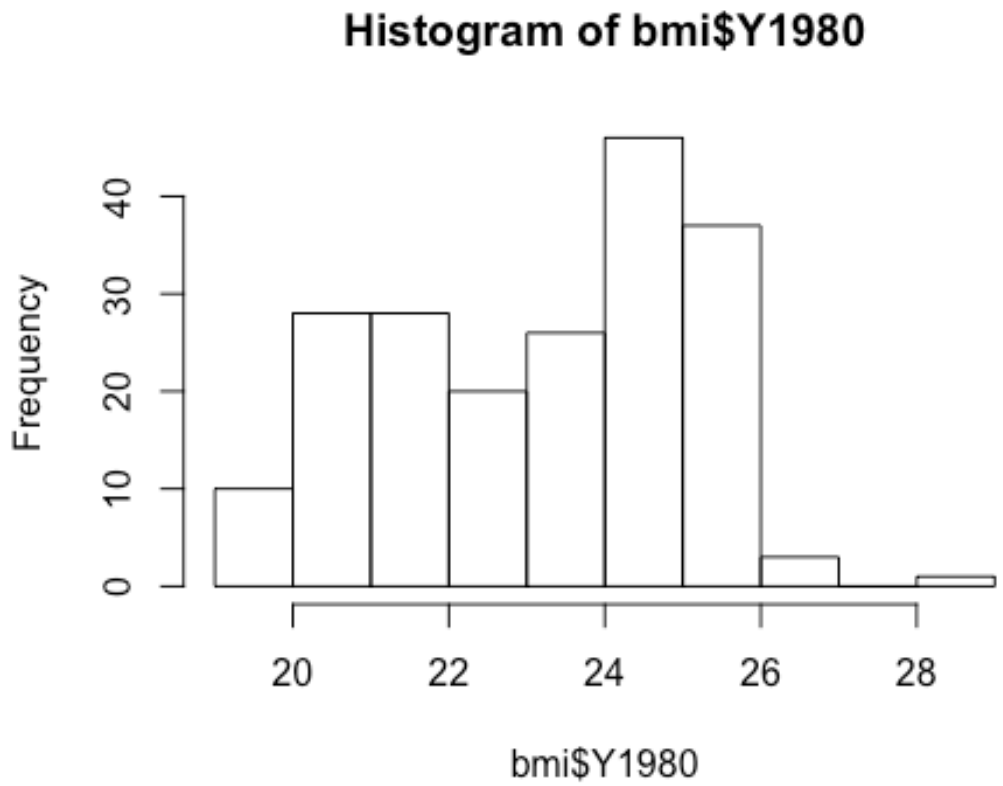
```
##       Country    Y1980    Y1981    Y1982    Y1983    Y1984    Y1985
## 1 Afghanistan 21.48678 21.46552 21.45145 21.43822 21.42734 21.41222
## 2     Albania 25.22533 25.23981 25.25636 25.27176 25.27901 25.28669
##      Y1986    Y1987    Y1988    Y1989    Y1990    Y1991    Y1992    Y1993
## 1 21.40132 21.37679 21.34018 21.29845 21.24818 21.20269 21.14238 21.06376
## 2 25.29451 25.30217 25.30450 25.31944 25.32357 25.28452 25.23077 25.21192
##      Y1994    Y1995    Y1996    Y1997    Y1998    Y1999    Y2000    Y2001
## 1 20.97987 20.91132 20.85155 20.81307 20.78591 20.75469 20.69521 20.62643
## 2 25.22115 25.25874 25.31097 25.33988 25.39116 25.46555 25.55835 25.66701
##      Y2002    Y2003    Y2004    Y2005    Y2006    Y2007    Y2008
```

```
## 1 20.59848 20.58706 20.57759 20.58084 20.58749 20.60246 20.62058
## 2 25.77167 25.87274 25.98136 26.08939 26.20867 26.32753 26.44657
```

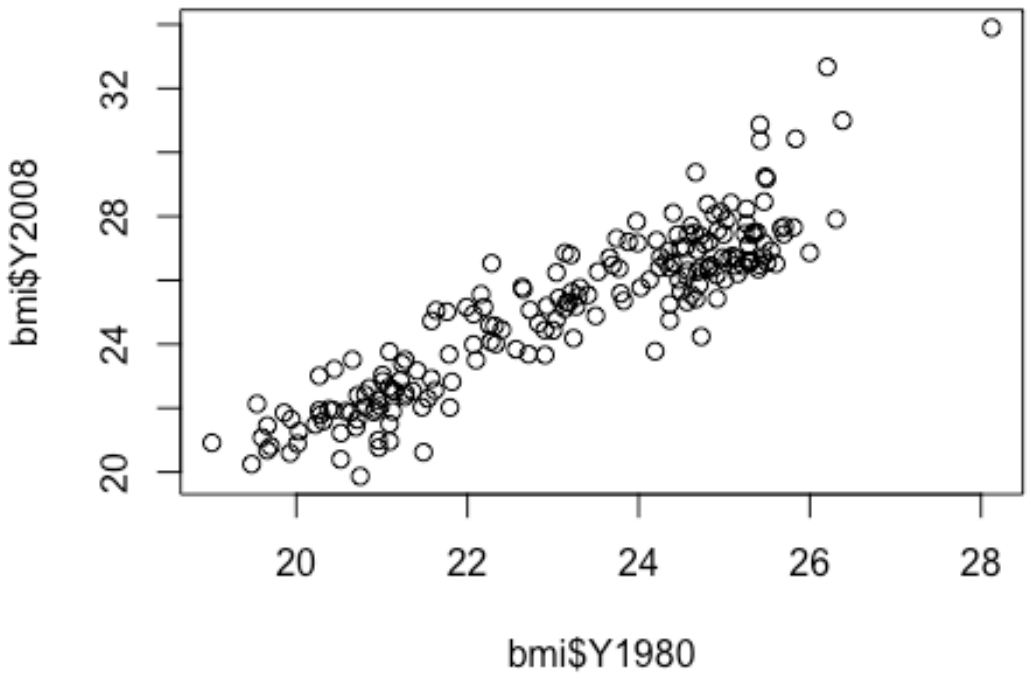
```
#tail(bmi)
tail(bmi, n = 2)
```

```
##      Country    Y1980    Y1981    Y1982    Y1983    Y1984    Y1985
## 198   Zambia 19.66295 19.69512 19.72538 19.75420 19.78070 19.80335
## 199 Zimbabwe 21.46989 21.48867 21.50738 21.52936 21.53383 21.54341
##        Y1986    Y1987    Y1988    Y1989    Y1990    Y1991    Y1992
## 198 19.82396 19.85065 19.88320 19.92451 19.96680 20.00746 20.04096
## 199 21.54859 21.54590 21.55396 21.56903 21.58005 21.59694 21.59010
##        Y1993    Y1994    Y1995    Y1996    Y1997    Y1998    Y1999
## 198 20.07781 20.09502 20.09977 20.11009 20.12375 20.13349 20.15094
## 199 21.58547 21.59029 21.58986 21.60362 21.62721 21.65496 21.68873
##        Y2000    Y2001    Y2002    Y2003    Y2004    Y2005    Y2006
## 198 20.17261 20.20266 20.24298 20.29474 20.35966 20.43398 20.51422
## 199 21.72652 21.76514 21.79645 21.82499 21.85806 21.89495 21.93371
##        Y2007    Y2008
## 198 20.59770 20.68321
## 199 21.97405 22.02660
```

```
hist(bmi$Y1980)
```



```r
plot(x = bmi$Y1980, y = bmi$Y2008)
```



1. Tidying Data

GATHER & SPREAD

```r
wide_df <- data.frame(col = c('X', 'Y'), A = c(1,4), B = c(2,5), C = c(3,6))
wide_df
```

```
##   col A B C
## 1   X 1 2 3
## 2   Y 4 5 6
```

```r
wide_df %>% gather(Alphabet, value, -col)
```

```
##   col Alphabet value
## 1   X        A     1
## 2   Y        A     4
## 3   X        B     2
## 4   Y        B     5
## 5   X        C     3
## 6   Y        C     6
```

```r
wide_df %>% gather(Alphabet, value, -col) %>% spread(Alphabet, value)
```

```
##   col A B C
## 1   X 1 2 3
## 2   Y 4 5 6
```

```
head(bmi)
```

```
##               Country    Y1980    Y1981    Y1982    Y1983    Y1984
## 1         Afghanistan 21.48678 21.46552 21.45145 21.43822 21.42734
## 2             Albania 25.22533 25.23981 25.25636 25.27176 25.27901
## 3             Algeria 22.25703 22.34745 22.43647 22.52105 22.60633
## 4             Andorra 25.66652 25.70868 25.74681 25.78250 25.81874
## 5              Angola 20.94876 20.94371 20.93754 20.93187 20.93569
## 6 Antigua and Barbuda 23.31424 23.39054 23.45883 23.53735 23.63584
##      Y2001    Y2002    Y2003    Y2004    Y2005    Y2006    Y2007    Y2008
## 1 20.62643 20.59848 20.58706 20.57759 20.58084 20.58749 20.60246 20.62058
## 2 25.66701 25.77167 25.87274 25.98136 26.08939 26.20867 26.32753 26.44657
## 3 23.86256 23.95294 24.05243 24.15957 24.27001 24.38270 24.48846 24.59620
## 4 26.92373 27.02525 27.12481 27.23107 27.32827 27.43588 27.53363 27.63048
## 5 21.43664 21.51765 21.59924 21.69218 21.80564 21.93881 22.08962 22.25083
## 6 25.05857 25.13039 25.20713 25.29898 25.39965 25.51382 25.64247 25.76602
```

```
head(bmi %>% gather(year, bmi_value, -Country))
```

```
##               Country  year bmi_value
## 1         Afghanistan Y1980  21.48678
## 2             Albania Y1980  25.22533
## 3             Algeria Y1980  22.25703
## 4             Andorra Y1980  25.66652
## 5              Angola Y1980  20.94876
## 6 Antigua and Barbuda Y1980  23.31424
```

```
head(bmi %>% gather(year, bmi_value, -Country) %>% spread(year, bmi_value))
```

```
##               Country    Y1980    Y1981    Y1982    Y1983    Y1984
## 1         Afghanistan 21.48678 21.46552 21.45145 21.43822 21.42734
## 2             Albania 25.22533 25.23981 25.25636 25.27176 25.27901
## 3             Algeria 22.25703 22.34745 22.43647 22.52105 22.60633
## 4             Andorra 25.66652 25.70868 25.74681 25.78250 25.81874
## 5              Angola 20.94876 20.94371 20.93754 20.93187 20.93569
## 6 Antigua and Barbuda 23.31424 23.39054 23.45883 23.53735 23.63584
##      Y2001    Y2002    Y2003    Y2004    Y2005    Y2006    Y2007    Y2008
## 1 20.62643 20.59848 20.58706 20.57759 20.58084 20.58749 20.60246 20.62058
## 2 25.66701 25.77167 25.87274 25.98136 26.08939 26.20867 26.32753 26.44657
## 3 23.86256 23.95294 24.05243 24.15957 24.27001 24.38270 24.48846 24.59620
## 4 26.92373 27.02525 27.12481 27.23107 27.32827 27.43588 27.53363 27.63048
## 5 21.43664 21.51765 21.59924 21.69218 21.80564 21.93881 22.08962 22.25083
## 6 25.05857 25.13039 25.20713 25.29898 25.39965 25.51382 25.64247 25.76602
```

GATHER & FILTER & SELECT

```
df <- data.frame(col = c('Jake', 'Alice', 'Tim', 'Denise'), brown = c(0,1,0,0
), blue = c(0,1,0,0),other = c(1,0,0,1))
df
```

```
##      col brown blue other
## 1   Jake     0    0     1
## 2  Alice     1    1     0
## 3    Tim     0    0     0
## 4 Denise     0    0     1
```

```
df %>% gather(eye_color, flag, -col)
```

```
##       col eye_color flag
## 1    Jake     brown    0
## 2   Alice     brown    1
## 3     Tim     brown    0
## 4  Denise     brown    0
## 5    Jake      blue    0
## 6   Alice      blue    1
## 7     Tim      blue    0
## 8  Denise      blue    0
## 9    Jake     other    1
## 10  Alice     other    0
## 11    Tim     other    0
## 12 Denise     other    1
```

```
df %>% gather(eye_color, flag, -col) %>% filter(flag == 1)
```

```
##      col eye_color flag
## 1  Alice     brown    1
## 2  Alice      blue    1
## 3   Jake     other    1
## 4 Denise     other    1
```

```
df %>% gather(eye_color, flag, -col) %>% filter(flag == 1) %>% dplyr::select(
col, eye_color)
```

```
##      col eye_color
## 1  Alice     brown
## 2  Alice      blue
## 3   Jake     other
## 4 Denise     other
```

SEPERATE & UNITE

```
treatments <- data.frame(patient = rep(c('X', 'Y'),3) ,
                         treatment = rep(c('A', 'B'), each = 3),
                         year_mo = rep(c('2010-10', '2012-08', '2014-12'), ea
ch = 2),
                         response = c(1,4,2,5,3,6))
treatments
```

```
##   patient treatment year_mo response
## 1       X         A 2010-10        1
## 2       Y         A 2010-10        4
## 3       X         A 2012-08        2
## 4       Y         B 2012-08        5
## 5       X         B 2014-12        3
## 6       Y         B 2014-12        6
```

```
head(treatments %>% separate(year_mo, c("year", "month")))
```

```
##   patient treatment year month response
## 1       X         A 2010    10        1
## 2       Y         A 2010    10        4
## 3       X         A 2012    08        2
## 4       Y         B 2012    08        5
## 5       X         B 2014    12        3
## 6       Y         B 2014    12        6
```

```
head(treatments %>% separate(year_mo, c("year", "month")) %>% unite(year_mo,
year, month))
```

```
##   patient treatment year_mo response
## 1       X         A 2010_10        1
## 2       Y         A 2010_10        4
## 3       X         A 2012_08        2
## 4       Y         B 2012_08        5
## 5       X         B 2014_12        3
## 6       Y         B 2014_12        6
```

```
bmi_cc <- read.csv(file = 'bmi_cc.csv')
head(bmi_cc)
```

```
##              Country_ISO  year  bmi_val
## 1         Afghanistan/AF Y1980 21.48678
## 2             Albania/AL Y1980 25.22533
## 3             Algeria/DZ Y1980 22.25703
## 4             Andorra/AD Y1980 25.66652
## 5              Angola/AO Y1980 20.94876
## 6 Antigua and Barbuda/AG Y1980 23.31424
```

```
head(bmi_cc %>% separate(Country_ISO, c('Country', 'ISO'), sep = "/"))
```

```
##              Country ISO  year  bmi_val
## 1         Afghanistan  AF Y1980 21.48678
## 2             Albania  AL Y1980 25.22533
## 3             Algeria  DZ Y1980 22.25703
## 4             Andorra  AD Y1980 25.66652
## 5              Angola  AO Y1980 20.94876
## 6 Antigua and Barbuda  AG Y1980 23.31424
```

```
head(bmi_cc %>% separate(Country_ISO, c('Country', 'ISO'), sep = "/") %>% uni
te(Country_ISO, Country, ISO, sep = '-'))
```

```
##              Country_ISO  year  bmi_val
## 1         Afghanistan-AF Y1980 21.48678
## 2             Albania-AL Y1980 25.22533
## 3             Algeria-DZ Y1980 22.25703
## 4             Andorra-AD Y1980 25.66652
## 5              Angola-AO Y1980 20.94876
## 6 Antigua and Barbuda-AG Y1980 23.31424
```

ADVANCED 1

```
rm(iris)

## Warning in rm(iris): object 'iris' not found

head(iris)

##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa

head(iris %>% gather(measurement, value, -Species))

##   Species  measurement value
## 1  setosa Sepal.Length   5.1
## 2  setosa Sepal.Length   4.9
## 3  setosa Sepal.Length   4.7
## 4  setosa Sepal.Length   4.6
## 5  setosa Sepal.Length   5.0
## 6  setosa Sepal.Length   5.4

head(iris %>% gather(measurement, value, -Species) %>% separate(measurement,
c("type", "measurement"), sep = "[.]")) # regular expression

##   Species  type measurement value
## 1  setosa Sepal      Length   5.1
## 2  setosa Sepal      Length   4.9
## 3  setosa Sepal      Length   4.7
## 4  setosa Sepal      Length   4.6
## 5  setosa Sepal      Length   5.0
## 6  setosa Sepal      Length   5.4
```

ADVANCED 2 : IRIS DATA

```
iris$Flower <- 1:nrow(iris)
head(iris)

##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species Flower
## 1          5.1         3.5          1.4         0.2  setosa      1
## 2          4.9         3.0          1.4         0.2  setosa      2
## 3          4.7         3.2          1.3         0.2  setosa      3
```

```
## 4          4.6         3.1          1.5         0.2  setosa      4
## 5          5.0         3.6          1.4         0.2  setosa      5
## 6          5.4         3.9          1.7         0.4  setosa      6

#### iris.wide
iris.wide<-gather(iris, Key, Value, -Species,-Flower)%>%
  separate(Key, c("Part","Measure"),"\\.")%>%
  spread(Measure, Value)
head(iris.wide)

##   Species Flower  Part Length Width
## 1  setosa      1 Petal    1.4   0.2
## 2  setosa      1 Sepal    5.1   3.5
## 3  setosa      2 Petal    1.4   0.2
## 4  setosa      2 Sepal    4.9   3.0
## 5  setosa      3 Petal    1.3   0.2
## 6  setosa      3 Sepal    4.7   3.2

#### iris.wide2
iris.wide2<-gather(iris, Key, Value, -Species, -Flower)%>%
  spread(Species, Value)%>%
  separate(Key, c("Part", "Measure"),"\\.")
head(iris.wide2)

##   Flower  Part Measure setosa versicolor virginica
## 1      1 Petal  Length    1.4         NA        NA
## 2      1 Petal   Width    0.2         NA        NA
## 3      1 Sepal  Length    5.1         NA        NA
## 4      1 Sepal   Width    3.5         NA        NA
## 5      2 Petal  Length    1.4         NA        NA
## 6      2 Petal   Width    0.2         NA        NA

col1 <- iris.wide2[201:400,5]
head(col1)

## [1] 4.7 1.4 7.0 3.2 4.5 1.5

col2 <- iris.wide2[401:600,6]
head(col2)

## [1] 6.0 2.5 6.3 3.3 5.1 1.9

iris.wide2[1:200,5] <- col1
iris.wide2[1:200,6] <- col2
iris.wide2 <- iris.wide2[1:200,]
head(iris.wide2)

##   Flower  Part Measure setosa versicolor virginica
## 1      1 Petal  Length    1.4        4.7       6.0
## 2      1 Petal   Width    0.2        1.4       2.5
## 3      1 Sepal  Length    5.1        7.0       6.3
## 4      1 Sepal   Width    3.5        3.2       3.3
```

```
## 5      2 Petal  Length    1.4        4.5       5.1
## 6      2 Petal   Width    0.2        1.5       1.9

#### iris.tidy
rm(iris)
iris.tidy <- iris %>% gather(key, "Value", -Species)%>%
  separate(key, c("Part", "Measure"), sep = "[.]")
sum(is.na(iris.tidy))

## [1] 0

head(iris.tidy)

##   Species  Part Measure Value
## 1  setosa Sepal  Length   5.1
## 2  setosa Sepal  Length   4.9
## 3  setosa Sepal  Length   4.7
## 4  setosa Sepal  Length   4.6
## 5  setosa Sepal  Length   5.0
## 6  setosa Sepal  Length   5.4

#### iris.tidy.unite
iris.tidy.unite <- unite(iris.tidy, type, Part, Measure)
head(iris.tidy.unite)

##   Species         type Value
## 1  setosa Sepal_Length   5.1
## 2  setosa Sepal_Length   4.9
## 3  setosa Sepal_Length   4.7
## 4  setosa Sepal_Length   4.6
## 5  setosa Sepal_Length   5.0
## 6  setosa Sepal_Length   5.4

sum(is.na(iris.tidy))

## [1] 0
```

2. Type check up

```
class("hello")

## [1] "character"

class(3.844)

## [1] "numeric"

class(77L)

## [1] "integer"

class(factor("yes"))

## [1] "factor"
```

```r
class(TRUE)
```

```
## [1] "logical"
```

```r
as.character(2016)
```

```
## [1] "2016"
```

```r
as.numeric(TRUE)
```

```
## [1] 1
```

```r
as.integer(99)
```

```
## [1] 99
```

```r
as.factor("something")
```

```
## [1] something
## Levels: something
```

```r
as.logical(0)
```

```
## [1] FALSE
```

3. lubridate

```r
# 날짜
my_date <- "2018 April 12"
ymd(my_date)
```

```
## [1] "2018-04-12"
```

```r
typeof(ymd(my_date))
```

```
## [1] "double"
```

```r
class(ymd(my_date))
```

```
## [1] "Date"
```

```r
my_date2 <- "April 12, 2018"
mdy(my_date2)
```

```
## [1] "2018-04-12"
```

```r
# 시간
hms("13:33:09")
```

```
## [1] "13H 33M 9S"
```

```r
class(hms("13:33:09"))
```

```
## [1] "Period"
## attr(,"package")
## [1] "lubridate"

# 날짜 + 시간
my_datetime <- "2018 4 12, 13 44 05"
ymd_hms(my_datetime)

## [1] "2018-04-12 13:44:05 UTC"
```

4. String manipulation

```
# Trim leading and trailing white space
str_trim(" this is a test ")

## [1] "this is a test"

# Pad string with zeros
str_pad("24493", width = 7, side = "left", pad = "0")

## [1] "0024493"

# Create character vector of names***
friends <- c("Sarah", "Tom", "Alice?")

# Search for string in vector***
str_detect(friends, "Alice")

## [1] FALSE FALSE  TRUE

# Replace string in vector
str_replace(friends, "Alice", "David")

## [1] "Sarah"  "Tom"    "David?"

# Make all lowercase
tolower("I AM TALKING LOUDLY!!")

## [1] "i am talking loudly!!"

# Make all uppercase
toupper("I am whispering...")

## [1] "I AM WHISPERING..."
```

4. Exercise

```
# Exercise 1
bmi_cc_clean <- head(bmi_cc %>% separate(Country_ISO, c('Country', 'ISO'), se
p = "/") %>% unite(Country_ISO, Country, ISO, sep = '-'))

str(bmi_cc_clean)

## 'data.frame':    6 obs. of  3 variables:
##  $ Country_ISO: chr  "Afghanistan-AF" "Albania-AL" "Algeria-DZ" "Andorra-A
```

```
D" ...
##  $ year       : Factor w/ 29 levels "Y1980","Y1981",..: 1 1 1 1 1 1
##  $ bmi_val    : num  21.5 25.2 22.3 25.7 20.9 ...

bmi_cc_clean$year <- str_replace(bmi_cc_clean$year, "Y", "")
bmi_cc_clean$year <- as.numeric(bmi_cc_clean$year)

# Exercise 2
students <- read.csv('students2.csv', stringsAsFactors = FALSE)
str(students)

## 'data.frame':    395 obs. of  33 variables:
##  $ X          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ school     : chr  "GP" "GP" "GP" "GP" ...
##  $ sex        : chr  "F" "F" "F" "F" ...
##  $ dob        : chr  "2000-06-05" "1999-11-25" "1998-02-02" "1997-12-20" .
..
##  $ address    : chr  "U" "U" "U" "U" ...
##  $ famsize    : chr  "GT3" "GT3" "LE3" "GT3" ...
##  $ Pstatus    : chr  "A" "T" "T" "T" ...
##  $ Medu       : int  4 1 1 4 3 4 2 4 3 3 ...
##  $ Fedu       : int  4 1 1 2 3 3 2 4 2 4 ...
##  $ Mjob       : chr  "at_home" "at_home" "at_home" "health" ...
##  $ Fjob       : chr  "teacher" "other" "other" "services" ...
##  $ reason     : chr  "course" "course" "other" "home" ...
##  $ guardian   : chr  "mother" "father" "mother" "mother" ...
##  $ traveltime : int  2 1 1 1 1 1 1 2 1 1 ...
##  $ studytime  : int  2 2 2 3 2 2 2 2 2 2 ...
##  $ failures   : int  0 0 3 0 0 0 0 0 0 0 ...
##  $ schoolsup  : chr  "yes" "no" "yes" "no" ...
##  $ famsup     : chr  "no" "yes" "no" "yes" ...
##  $ paid       : chr  "no" "no" "yes" "yes" ...
##  $ activities : chr  "no" "no" "no" "yes" ...
##  $ nursery    : chr  "yes" "no" "yes" "yes" ...
##  $ higher     : chr  "yes" "yes" "yes" "yes" ...
##  $ internet   : chr  "no" "yes" "yes" "yes" ...
##  $ romantic   : chr  "no" "no" "no" "yes" ...
##  $ famrel     : int  4 5 4 3 4 5 4 4 4 5 ...
##  $ freetime   : int  3 3 3 2 3 4 4 1 2 5 ...
##  $ goout      : int  4 3 2 2 2 2 4 4 2 1 ...
##  $ Dalc       : int  1 1 2 1 1 1 1 1 1 1 ...
##  $ Walc       : int  1 1 3 1 2 2 1 1 1 1 ...
##  $ health     : int  3 3 3 5 5 5 3 1 1 5 ...
##  $ nurse_visit: chr  "2014-04-10 14:59:54" "2015-03-12 14:59:54" "2015-09-
21 14:59:54" "2015-09-03 14:59:54" ...
##  $ absences   : int  6 4 10 2 4 10 0 6 0 0 ...
##  $ Grades     : chr  "5/6/6" "5/5/6" "7/8/10" "15/14/15" ...

students$dob <- ymd(students$dob)
class(students$dob)
```

```
## [1] "Date"
```

```
typeof(students$dob)
```

```
## [1] "double"
```

```
students$nurse_visit <- ymd_hms(students$nurse_visit)
class(students$nurse_visit)
```

```
## [1] "POSIXct" "POSIXt"
```

```
str(students)
```

```
## 'data.frame':    395 obs. of  33 variables:
##  $ X          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ school     : chr  "GP" "GP" "GP" "GP" ...
##  $ sex        : chr  "F" "F" "F" "F" ...
##  $ dob        : Date, format: "2000-06-05" "1999-11-25" ...
##  $ address    : chr  "U" "U" "U" "U" ...
##  $ famsize    : chr  "GT3" "GT3" "LE3" "GT3" ...
##  $ Pstatus    : chr  "A" "T" "T" "T" ...
##  $ Medu       : int  4 1 1 4 3 4 2 4 3 3 ...
##  $ Fedu       : int  4 1 1 2 3 3 2 4 2 4 ...
##  $ Mjob       : chr  "at_home" "at_home" "at_home" "health" ...
##  $ Fjob       : chr  "teacher" "other" "other" "services" ...
##  $ reason     : chr  "course" "course" "other" "home" ...
##  $ guardian   : chr  "mother" "father" "mother" "mother" ...
##  $ traveltime : int  2 1 1 1 1 1 1 2 1 1 ...
##  $ studytime  : int  2 2 2 3 2 2 2 2 2 2 ...
##  $ failures   : int  0 0 3 0 0 0 0 0 0 0 ...
##  $ schoolsup  : chr  "yes" "no" "yes" "no" ...
##  $ famsup     : chr  "no" "yes" "no" "yes" ...
##  $ paid       : chr  "no" "no" "yes" "yes" ...
##  $ activities : chr  "no" "no" "no" "yes" ...
##  $ nursery    : chr  "yes" "no" "yes" "yes" ...
##  $ higher     : chr  "yes" "yes" "yes" "yes" ...
##  $ internet   : chr  "no" "yes" "yes" "yes" ...
##  $ romantic   : chr  "no" "no" "no" "yes" ...
##  $ famrel     : int  4 5 4 3 4 5 4 4 4 5 ...
##  $ freetime   : int  3 3 3 2 3 4 4 1 2 5 ...
##  $ goout      : int  4 3 2 2 2 2 4 4 2 1 ...
##  $ Dalc       : int  1 1 2 1 1 1 1 1 1 1 ...
##  $ Walc       : int  1 1 3 1 2 2 1 1 1 1 ...
##  $ health     : int  3 3 3 5 5 5 3 1 1 5 ...
##  $ nurse_visit: POSIXct, format: "2014-04-10 14:59:54" "2015-03-12 14:59:5
4" ...
##  $ absences   : int  6 4 10 2 4 10 0 6 0 0 ...
##  $ Grades     : chr  "5/6/6" "5/5/6" "7/8/10" "15/14/15" ...
```

```
# Exercise 3
head(students)
```

```
##   X school sex        dob address famsize Pstatus Medu Fedu     Mjob
## 1 1     GP   F 2000-06-05       U     GT3       A    4    4  at_home
## 2 2     GP   F 1999-11-25       U     GT3       T    1    1  at_home
## 3 3     GP   F 1998-02-02       U     LE3       T    1    1  at_home
## 4 4     GP   F 1997-12-20       U     GT3       T    4    2   health
## 5 5     GP   F 1998-10-04       U     GT3       T    3    3    other
## 6 6     GP   M 1999-06-16       U     LE3       T    4    3 services
##       Fjob     reason guardian traveltime studytime failures schoolsup
## 1  teacher     course   mother          2         2        0       yes
## 2    other     course   father          1         2        0        no
## 3    other      other   mother          1         2        3       yes
## 4 services       home   mother          1         3        0        no
## 5    other       home   father          1         2        0        no
## 6    other reputation   mother          1         2        0        no
##   famsup paid activities nursery higher internet romantic famrel freetime
## 1     no   no         no     yes    yes       no       no      4        3
## 2    yes   no         no      no    yes      yes       no      5        3
## 3     no  yes         no     yes    yes      yes       no      4        3
## 4    yes  yes        yes     yes    yes      yes      yes      3        2
## 5    yes  yes         no     yes    yes       no       no      4        3
## 6    yes  yes        yes     yes    yes      yes       no      5        4
##   goout Dalc Walc health         nurse_visit absences   Grades
## 1     4    1    1      3 2014-04-10 14:59:54        6    5/6/6
## 2     3    1    1      3 2015-03-12 14:59:54        4    5/5/6
## 3     2    2    3      3 2015-09-21 14:59:54       10   7/8/10
## 4     2    1    1      5 2015-09-03 14:59:54        2 15/14/15
## 5     2    1    2      5 2015-04-07 14:59:54        4  6/10/10
## 6     2    1    2      5 2013-11-15 14:59:54       10 15/15/15
```

```
str_detect(students$dob, "1997")
```

```
##   [1] FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE
##  [12] FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
##  [23]  TRUE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE

## [375]  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE
## [386] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
students$sex <- str_replace(students$sex, "F","Female")
students$sex <- str_replace(students$sex, "M", "Male")
head(students)
```

```
##   X school    sex        dob address famsize Pstatus Medu Fedu     Mjob
## 1 1     GP Female 2000-06-05       U     GT3       A    4    4  at_home
## 2 2     GP Female 1999-11-25       U     GT3       T    1    1  at_home
## 3 3     GP Female 1998-02-02       U     LE3       T    1    1  at_home
## 4 4     GP Female 1997-12-20       U     GT3       T    4    2   health
## 5 5     GP Female 1998-10-04       U     GT3       T    3    3    other
## 6 6     GP   Male 1999-06-16       U     LE3       T    4    3 services
##       Fjob     reason guardian traveltime studytime failures schoolsup
## 1  teacher     course   mother          2         2        0       yes
```

```
## 2    other     course   father          1         2        0        no
## 3    other      other   mother          1         2        3       yes
## 4 services       home   mother          1         3        0        no
## 5    other       home   father          1         2        0        no
## 6    other reputation   mother          1         2        0        no
##   famsup paid activities nursery higher internet romantic famrel freetime
## 1     no   no         no     yes    yes       no       no      4        3
## 2    yes   no         no      no    yes      yes       no      5        3
## 3     no  yes         no     yes    yes      yes       no      4        3
## 4    yes  yes        yes     yes    yes      yes      yes      3        2
## 5    yes  yes         no     yes    yes       no       no      4        3
## 6    yes  yes        yes     yes    yes      yes       no      5        4
##   goout Dalc Walc health         nurse_visit absences   Grades
## 1     4    1    1      3 2014-04-10 14:59:54        6    5/6/6
## 2     3    1    1      3 2015-03-12 14:59:54        4    5/5/6
## 3     2    2    3      3 2015-09-21 14:59:54       10   7/8/10
## 4     2    1    1      5 2015-09-03 14:59:54        2 15/14/15
## 5     2    1    2      5 2015-04-07 14:59:54        4  6/10/10
## 6     2    1    2      5 2013-11-15 14:59:54       10 15/15/15
```

5. missing and special values

```
df <- data.frame(A = c(1, NA, 8, NA),
                 B = c(3, NA, 88, 23),
                 C = c(2, 45, 3, 1))
is.na(df)

##          A     B     C
## [1,] FALSE FALSE FALSE
## [2,]  TRUE  TRUE FALSE
## [3,] FALSE FALSE FALSE
## [4,]  TRUE FALSE FALSE

any(is.na(df))

## [1] TRUE

sum(is.na(df))

## [1] 3

colSums(is.na(df)) == 0

##     A     B     C
## FALSE FALSE  TRUE

summary(df)

##        A               B               C
##  Min.   :1.00   Min.   : 3.0   Min.   : 1.00
##  1st Qu.:2.75   1st Qu.:13.0   1st Qu.: 1.75
##  Median :4.50   Median :23.0   Median : 2.50
##  Mean   :4.50   Mean   :38.0   Mean   :12.75
```

```
##  3rd Qu.:6.25   3rd Qu.:55.5   3rd Qu.:13.50
##  Max.   :8.00   Max.   :88.0   Max.   :45.00
##  NA's   :2      NA's   :1
```

```
complete.cases(df) # Find rows with no missing values
```

```
## [1]  TRUE FALSE  TRUE FALSE
```

```
df[complete.cases(df), ] # Subset data, keeping only complete cases
```

```
##   A  B C
## 1 1  3 2
## 3 8 88 3
```

```
na.omit(df) # Another way to remove rows with NAs
```

```
##   A  B C
## 1 1  3 2
## 3 8 88 3
```

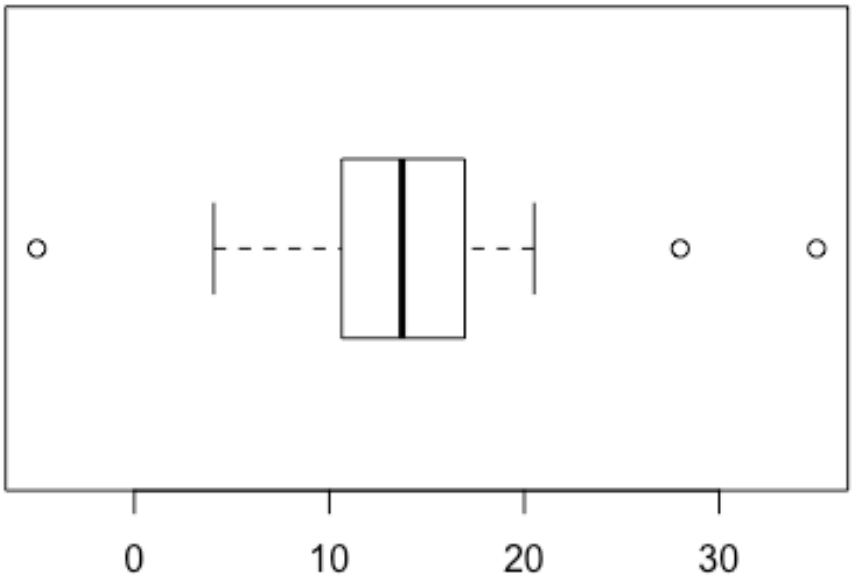
결측치가 발생 했을 때는, 이 결측치가 왜 발생했는지 파악한 후, 다른 적절한 값으로 대체하던제 지우던지 해야한다

6. outliers and obvious errors

```
# Simulate some data
par(mfrow = c(1,1))
set.seed(10)
(x <- c(rnorm(30, mean = 15, sd = 5), -5, 28, 35))
```

```
##  [1] 15.093731 14.078737  8.143347 12.004161 16.472726 16.948972  8.959619
##  [8] 13.181620  6.866637 13.717608 20.508898 18.778908 13.808832 19.937224
## [15] 18.706951 15.446736 10.225281 14.024248 19.627606 17.414893 12.018447
## [22]  4.073566 11.625670  4.404694  8.674010 13.131692 11.562223 10.639206
## [29] 14.491195 13.731097 -5.000000 28.000000 35.000000
```

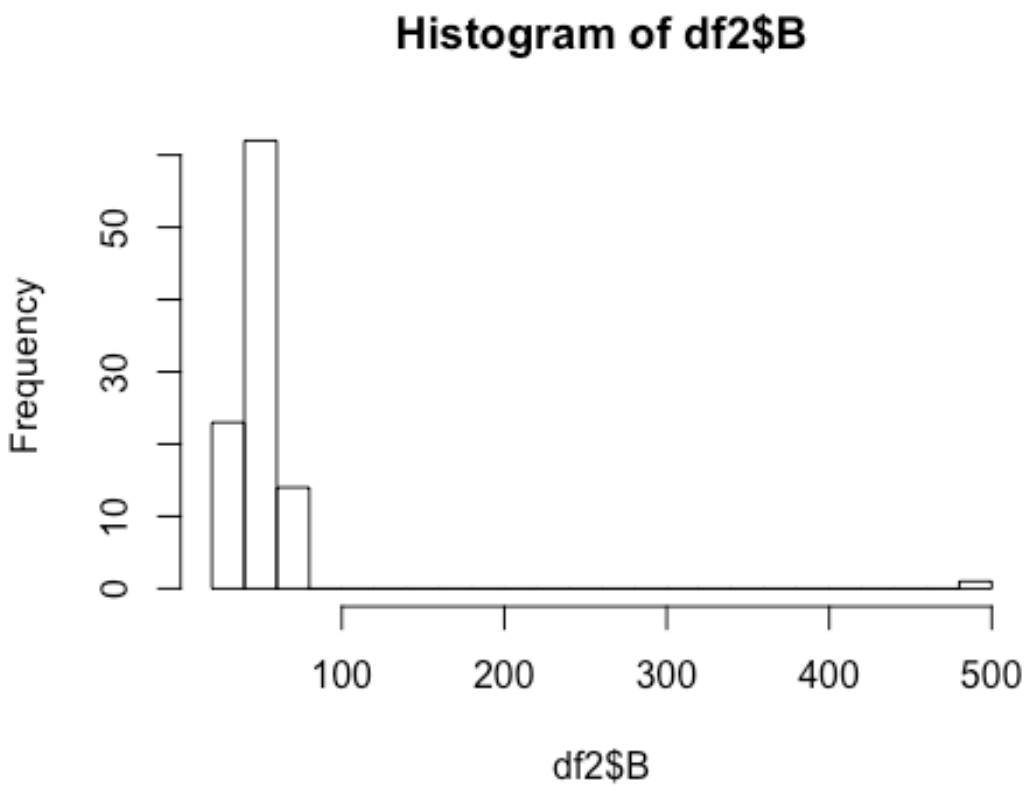
```
boxplot(x, horizontal = TRUE)
```



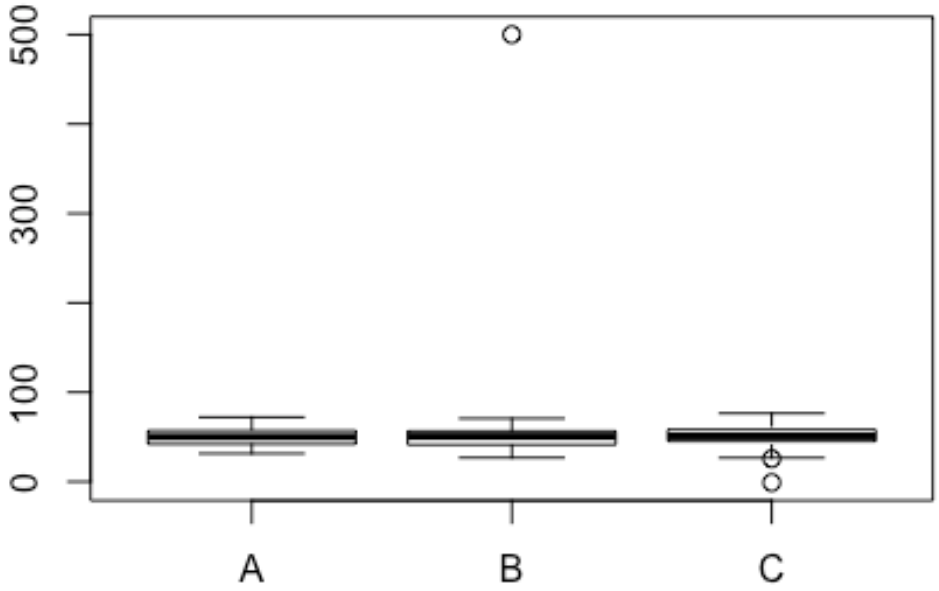

```r
df2 <- data.frame(A = rnorm(100, 50, 10),
                  B = c(rnorm(99, 50, 10), 500),
                  C = c(rnorm(99, 50, 10), -1))
summary(df2)
```

```
##        A                B                C          
##  Min.   :31.46   Min.   : 26.79   Min.   :-1.00  
##  1st Qu.:42.21   1st Qu.: 41.35   1st Qu.:45.29  
##  Median :50.20   Median : 50.67   Median :51.06  
##  Mean   :49.70   Mean   : 53.62   Mean   :50.88  
##  3rd Qu.:57.12   3rd Qu.: 56.57   3rd Qu.:58.13  
##  Max.   :72.21   Max.   :500.00   Max.   :76.44
```

```r
hist(df2$B, breaks = 20)
```


Histogram of df2$B
Frequency
0
10
30
50
100
200
300
400
500
df2$B

```
boxplot(df2)
```

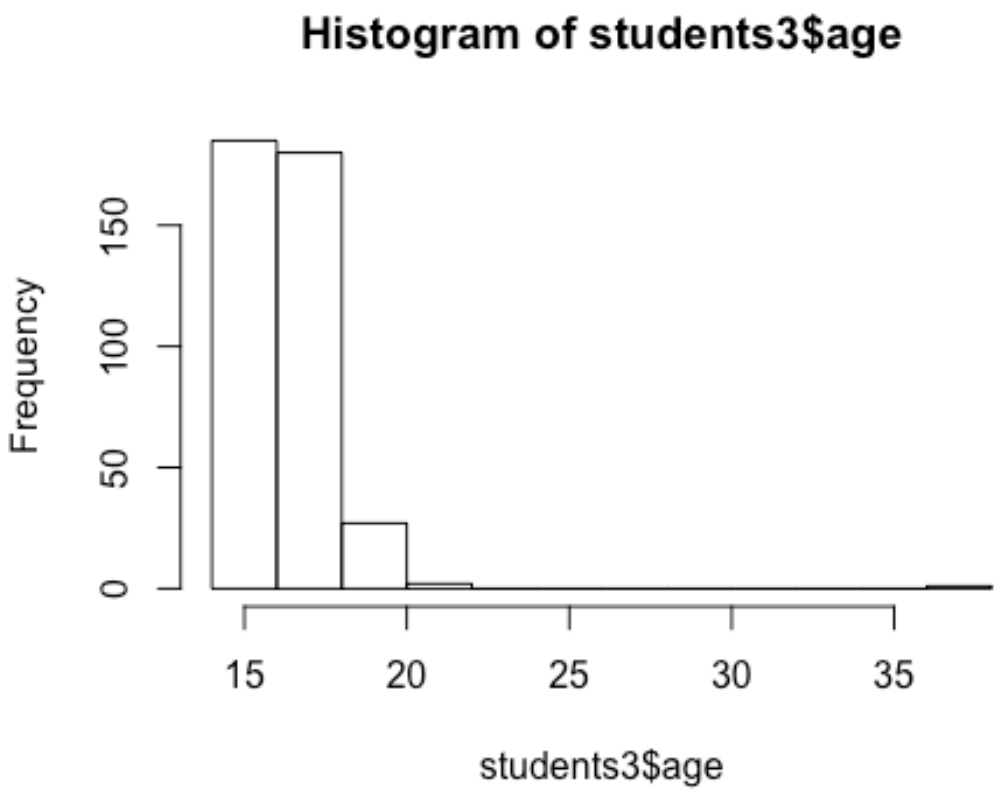


6. Exercise

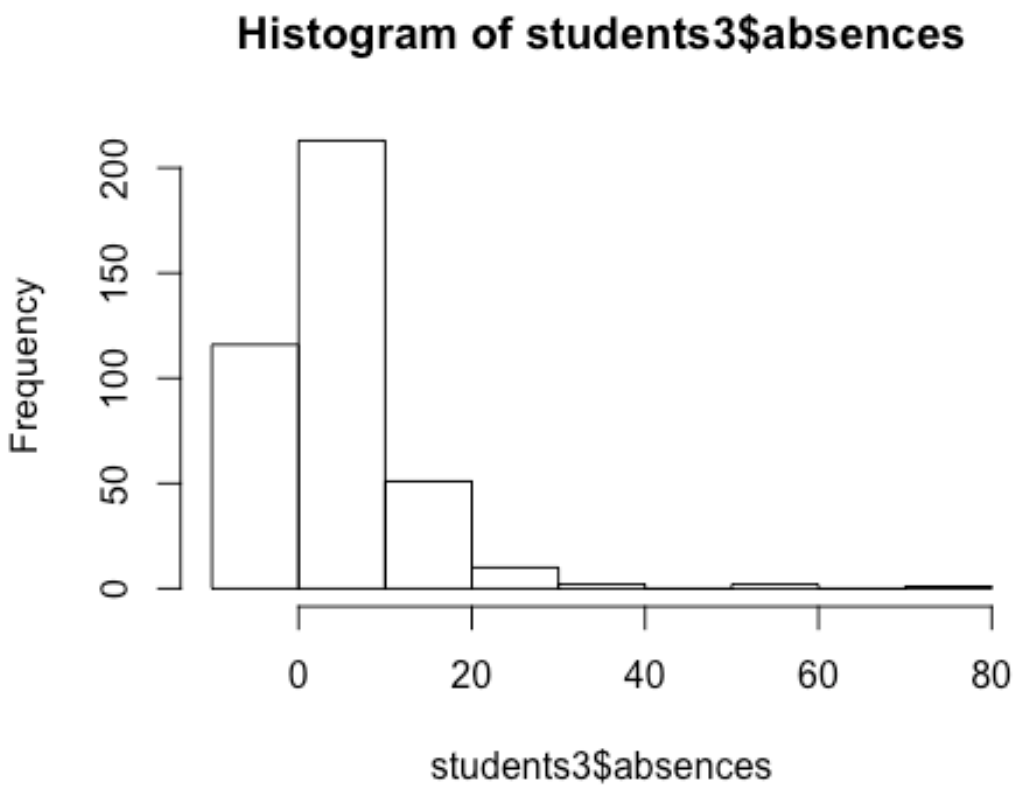
```
# Exercise
students3 <- read.csv('students3.csv', stringsAsFactors = FALSE)
summary(students3)
```

```
##     school              sex                 age          address
##  Length:395         Length:395         Min.   :15.00   Length:395
##  Class :character   Class :character   1st Qu.:16.00   Class :character
##  Mode  :character   Mode  :character   Median :17.00   Mode  :character
##                                        Mean   :16.75
##                                        3rd Qu.:18.00
##                                        Max.   :38.00
##    famsize             Pstatus              Medu            Fedu
##  Length:395         Length:395         Min.   :0.000   Min.   :0.000
##  Class :character   Class :character   1st Qu.:2.000   1st Qu.:2.000
##  Mode  :character   Mode  :character   Median :3.000   Median :2.000
##                                        Mean   :2.749   Mean   :2.522
##                                        3rd Qu.:4.000   3rd Qu.:3.000
##                                        Max.   :4.000   Max.   :4.000
##      Mjob               Fjob              reason
##  Length:395         Length:395         Length:395
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
```

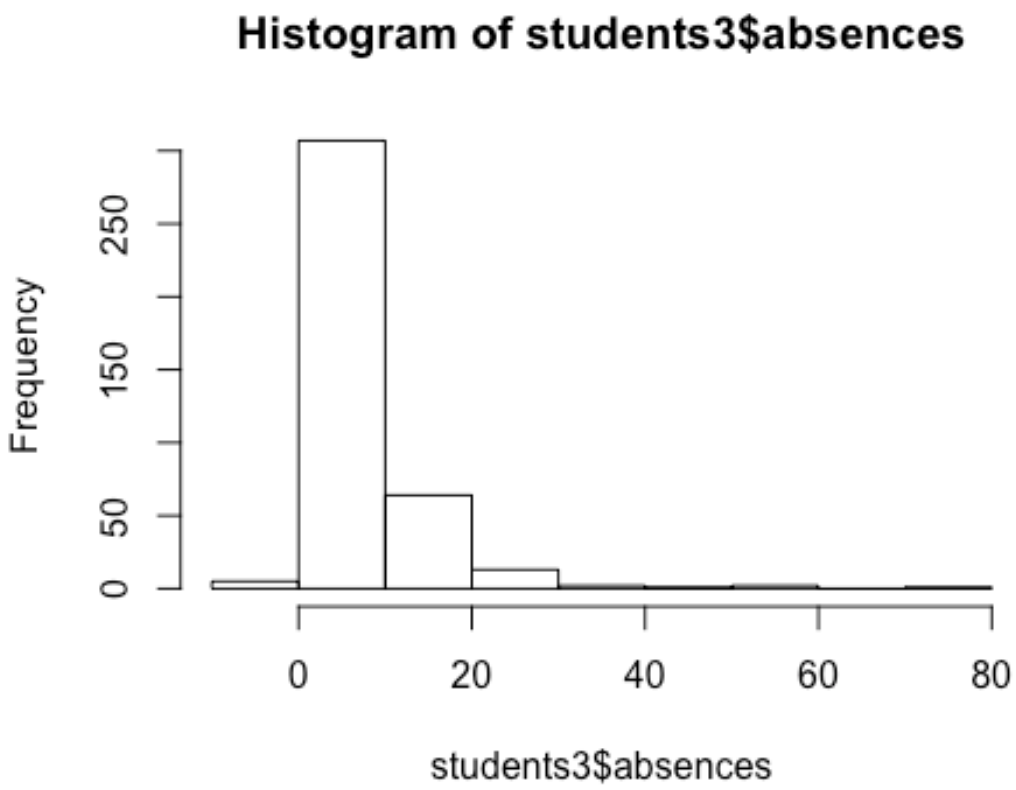
```
##
##
##
##    guardian           traveltime      studytime        failures
##  Length:395         Min.   :1.000   Min.   :1.000   Min.   :0.0000
##  Class :character   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:0.0000
##  Mode  :character   Median :1.000   Median :2.000   Median :0.0000
##                     Mean   :1.448   Mean   :2.035   Mean   :0.3342
##                     3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:0.0000
##                     Max.   :4.000   Max.   :4.000   Max.   :3.0000
##   schoolsup            famsup              paid
##  Length:395         Length:395         Length:395
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##   activities          nursery             higher
##  Length:395         Length:395         Length:395
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##    internet           romantic             famrel         freetime
##  Length:395         Length:395         Min.   :1.000   Min.   :1.000
##  Class :character   Class :character   1st Qu.:4.000   1st Qu.:3.000
##  Mode  :character   Mode  :character   Median :4.000   Median :3.000
##                                        Mean   :3.944   Mean   :3.235
##                                        3rd Qu.:5.000   3rd Qu.:4.000
##                                        Max.   :5.000   Max.   :5.000
##      goout            Dalc            Walc           health
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000
##  1st Qu.:2.000   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:3.000
##  Median :3.000   Median :1.000   Median :2.000   Median :4.000
##  Mean   :3.109   Mean   :1.481   Mean   :2.291   Mean   :3.554
##  3rd Qu.:4.000   3rd Qu.:2.000   3rd Qu.:3.000   3rd Qu.:5.000
##  Max.   :5.000   Max.   :5.000   Max.   :5.000   Max.   :5.000
##     absences         Grades
##  Min.   :-1.000   Length:395
##  1st Qu.: 0.000   Class :character
##  Median : 4.000   Mode  :character
##  Mean   : 5.691
##  3rd Qu.: 8.000
##  Max.   :75.000

hist(students3$age)
```



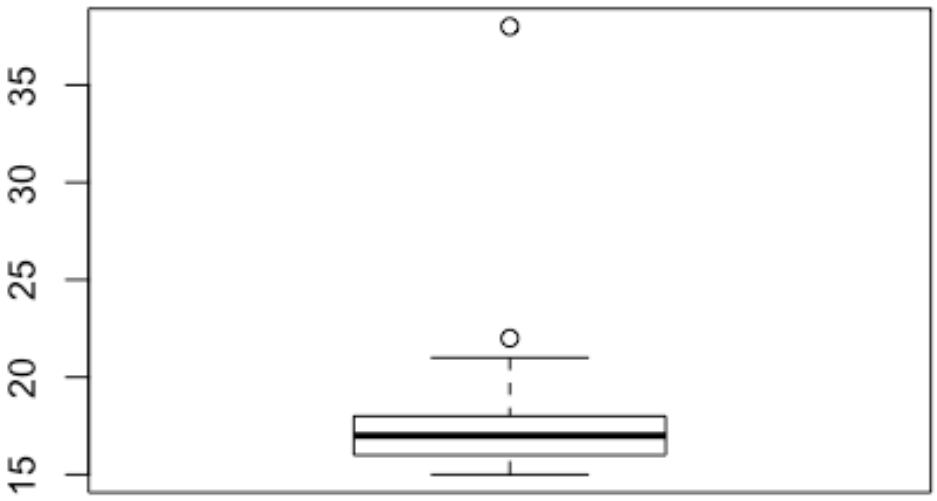

```r
hist(students3$absences)
```



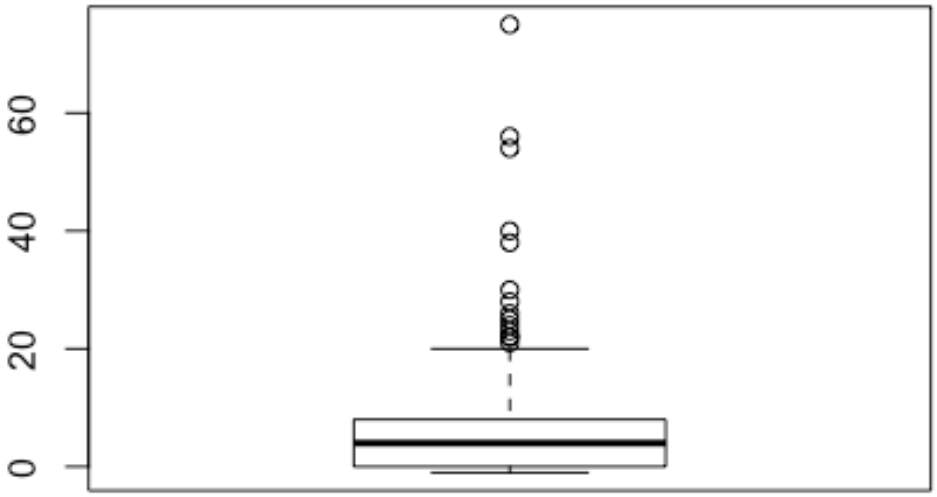

```r
hist(students3$absences, right = FALSE)
```




```r
boxplot(students3$age)
```




```
boxplot(students3$absences)
```


0
20
40
60