HW1

1

	Homogeneous	Heterogeneous
1 dim	atomic vector	list
2 dim	matrix	data frame
n dim	array	

2

The four common types of atomic vector are logical, integer, double (sometimes called numeric), and character.

type
logical, integer, double, complex, character, list(<- dataframe), ….
type과 mode의 차이점
types ‘integer’ and ‘double’ are returned as ‘numeric’.
class
integer, numeric, list, dataframe

3

```
typeof(c('1',1:3, FALSE))
```

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

4

```
x <- 1:10
typeof(x)
```

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

```
mode(x)
```

```
## [1] "numeric"
```

5

```
x <- seq(1, 2, by=0.1)
typeof(x)
```

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

```
mode(x)
```

```
## [1] "numeric"
```

6

```
x <- c(1:4)
matrix(x, 2, 2)
```

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

```
matrix(x, 2, 2, byrow = T)
```

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

7

A list is a type of vectors.
TRUE

8

```
A <- matrix(c(1,2,3,4), 2, 2)
B <- matrix(c(1,2,2,1), 2, 2)
A * B
```

```
##      [,1] [,2]
## [1,]    1    6
## [2,]    4    4
```

```
# 단순곱셈 --> 같은 위치에 있는 항끼리 곱함.
A %*% B
```

```
##      [,1] [,2]
## [1,]    7    5
## [2,]   10    8
```

```
# 행렬 곱
```

9

```
c(1:3) + c(1:6)
```

```
## [1] 2 4 6 5 7 9
```

10

```
sum(c(F,T,T,F))
```

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

11

```
x <- c(aname = 'a', bname = 'b', cname = 'c')
x
```

```
## aname bname cname
##   "a"   "b"   "c"
```

```
names(x)
```

```
## [1] "aname" "bname" "cname"
```

```
is.vector(x)
```

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

```
attributes(x) <- list(name= 'abc vector')
is.vector(x)
```

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

```
# 벡터인지 확인해 주는 함수이다.
# cf
x <- 11:20
attr(x, 'created date') <- '2018.10.8'
attr(x, 'company') <- 'BigDate'
attr(x, 'description') <- '1부터 10까지 수열'
x
```

```
##  [1] 11 12 13 14 15 16 17 18 19 20
## attr(,"created date")
## [1] "2018.10.8"
## attr(,"company")
## [1] "BigDate"
## attr(,"description")
## [1] "1부터 10까지 수열"
```

```
x <- 11:20
x <- structure(x, 'created date'='2018.10.8', 'company' = 'BigData', 'description' = '1부터 10까지 수열')
# 하나의 항목별로 설명 문구를 조회할 경우
attr(x,'created date')
```

```
## [1] "2018.10.8"
```

```
attr(x, 'company')
```

```
## [1] "BigData"
```

```
attributes(x)$'created date'
```

```
## [1] "2018.10.8"
```

```
attributes(x)$'company'
```

```
## [1] "BigData"
```

```
attributes(x) # 설명문구만 출력생
```

```
## $`created date`
## [1] "2018.10.8"
##
## $company
## [1] "BigData"
##
## $description
## [1] "1부터 10까지 수열"
```

attr(x) # 이거는 오류 발생

12

```
check.vector <- function(x) {
  if(is.list(x) || is.atomic(x)) T
}
check.vector(x)
```

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

```
# is.vector함수는 is.list와 is.atomic함수로 만들 수 있다. 또한 attributes로 만드든 벡터는 FALSE로 반환한다.
# cf
# is.vector(vector or list ) = TRUE
# is.vector(dataframe or matrix) = FALSE
```

13

```
x <- 1:12
dim(x) <- c(3,4)
x
```

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

```
# 벡터에 차원을 정해주면 matrix가 된다.
```

14

```
df1 <- data.frame(x = 1:3, y= letters[1:3])
df2 <- data.frame(x = 1:3, y= letters[1:3], stringsAsFactors = F)
str(df1); str(df2)
```

```
## 'data.frame':    3 obs. of  2 variables:
##  $ x: int  1 2 3
##  $ y: Factor w/ 3 levels "a","b","c": 1 2 3
```

```
## 'data.frame':    3 obs. of  2 variables:
##  $ x: int  1 2 3
##  $ y: chr  "a" "b" "c"
```

```
# stringsAsFactors = T가 디폴트인데 문자를 factor로 인식
```

15

```
rm(list = ls())
(c(-2,2) >= 0) & (c(-2,2) <= 0)
```

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

```
(c(-2,2) >= 0) && (c(-2,2) <= 0)
```

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

```
(c(-2,2) >= 0) | (c(-2,2) <= 0)
```

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

```
(c(-2,2) >= 0) || (c(-2,2) <= 0)
```

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

```
# || &&는 첫번째 논리연산자만 사용한다.
# when object 'a' not found
T||a ; F&&a
```

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

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

```
# TRUE , FALSE
```

```
# 단, T&&a와 F||a -> Error: object 'a' not found
T|a -> Error: object 'a' not found
F&a -> Error: object 'a' not found
```

16

```
x <- c(-3.254, -1.887, 2.316, 2.946)
floor(x)
```

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

```
# x보다 크지 않은 정수를 반환
trunc(x)
```

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

```
# 소주점을 버림
```

17

```
round(135.789, 2) # [1] 135.79
```

```
## [1] 135.79
```

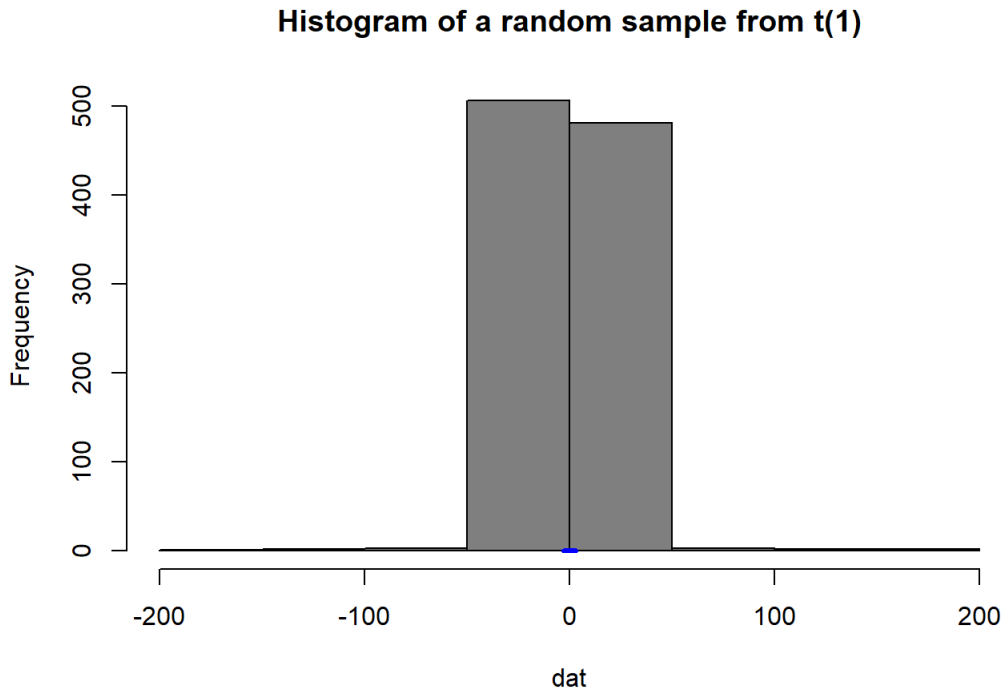
```
signif(135.789, 2) # [1] 140
```

```
## [1] 140
```

```
# 둘다 반올림이다.
```

18

```
dat <- rt(1000, df=1)
xx <- seq(-3, 3, by =0.1)
yy <- dt(xx, df=1)
hist(dat,freq = T, col = 'grey50', main = 'Histogram of a random sample from t(1)')
lines(xx, yy, col =4, lwd =3)
```




```
# freq = T 일때 y축이 빈도를 나타내고
# '' = F일때 y축은 probability densities를 나타낸다.
```

Learning-By-Yourself

Exercise 1

1

```
gender <- c(1,1,2,2,1,2,2,2,1,2)
gender <- factor(gender, levels=c(1,2), labels=c("male","female"))
gender
```

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

```
f1 <- factor(c('a', 'b')); f1; as.numeric(f1)
```

```
## [1] a b
## Levels: a b
```

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

```
f2 <- factor(c('a', 'b'), labels = c('c', 'd')); f2; as.numeric(f2)
```

```
## [1] c d
## Levels: c d
```

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

```
f3 <- factor(c('a', 'b'), levels =c('b', 'a')); f3; as.numeric(f3)
```

```
## [1] a b
## Levels: b a
```

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

```
f4 <- factor(c('a', 'b'), levels = c('b', 'a'), labels =c('c','d')); f4; as.numeric(f4)
```

```
## [1] d c
## Levels: c d
```

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

```
cbind(f1,f2,f3,f4)
```

```
##      f1 f2 f3 f4
## [1,]  1  1  2  2
## [2,]  2  2  1  1
```

```
ff <- f1
levels(ff) <- c('b', 'a')
ff
```

```
## [1] b a
## Levels: b a
```

```
# levels과 labels 모두 바뀐것을 알 수 있다.
```

```
# cf)
ff1 <- f1
levels(ff1) <- list(b='b', a='a')
ff1
```

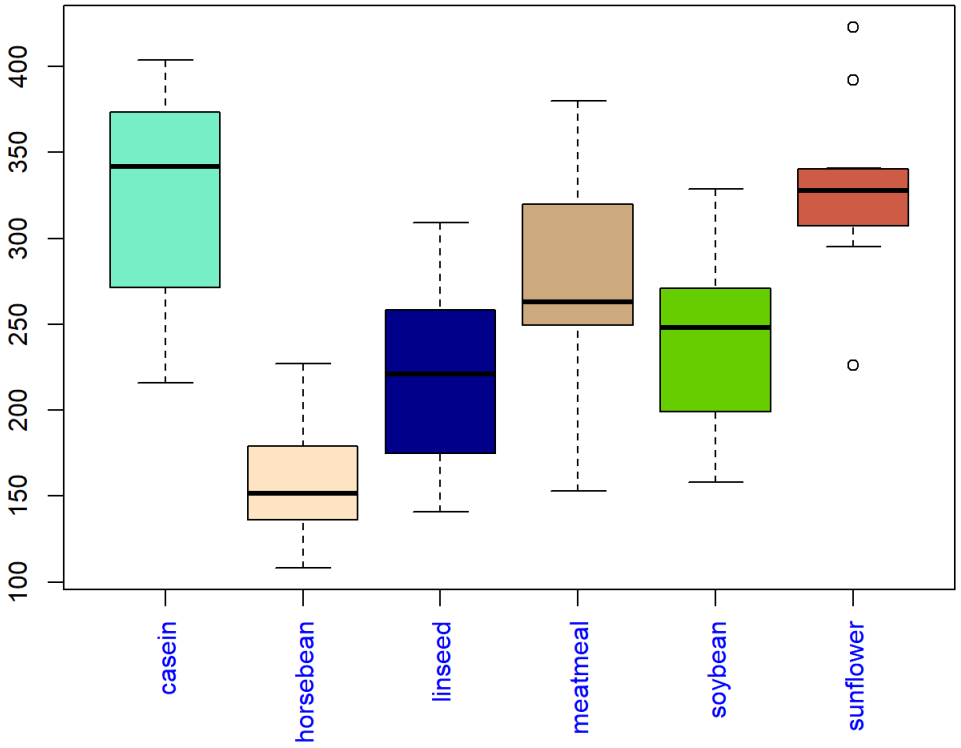
```
## [1] a b
## Levels: b a
```

```
# levels만 바뀐것을 알 수 있다.
```

```
odp = par(mar= c(5,5,1,1))
n.lev <- length(levels(chickwts$feed)); n.lev
```

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

```
boxplot(chickwts$weight ~ chickwts$feed, col= colors()[c(1:n.lev)*10], xaxt = 'n')
axis(side = 1, at= 1:6, labels = levels(chickwts$feed), col.axis= 4, las =2)
box()
```



1

```
typeof(chickwts)
```

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

```
str(chickwts)
```

```
## 'data.frame':    71 obs. of  2 variables:
##  $ weight: num  179 160 136 227 217 168 108 124 143 140 ...
##  $ feed  : Factor w/ 6 levels "casein","horsebean",..: 2 2 2 2 2 2 2 2 2 2 ...
```

```
length(levels(chickwts$feed))
```

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

```
levels(chickwts$feed)
```

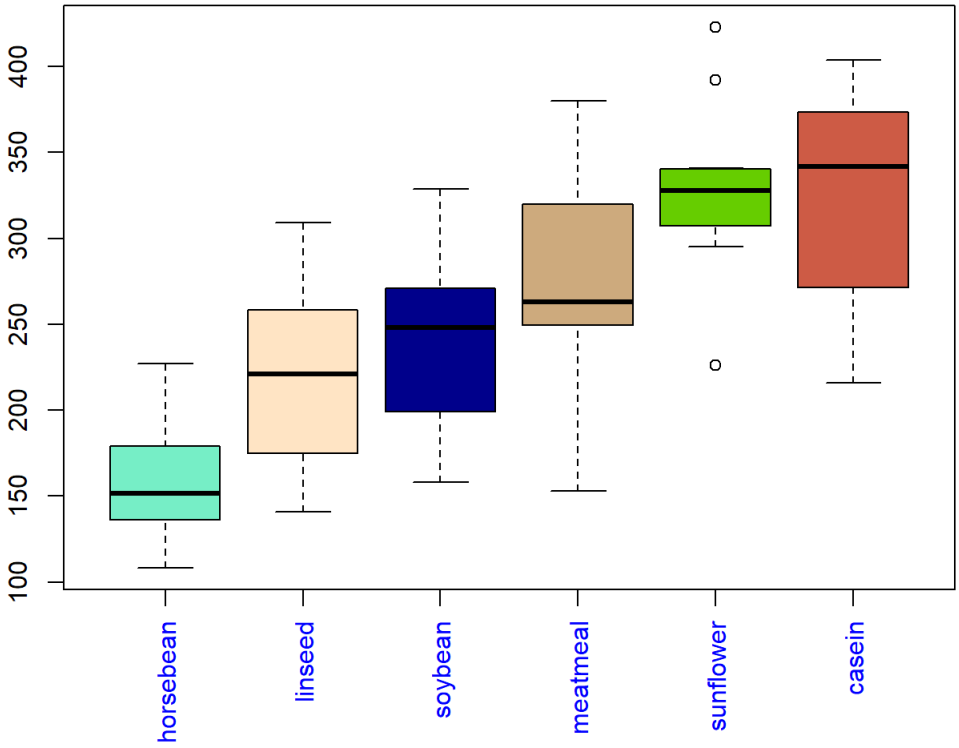
```
## [1] "casein"    "horsebean" "linseed"   "meatmeal"  "soybean"   "sunflower"
```

2

```
data("chickwts")
odp = par(mar= c(5,5,1,1))
n.lev <- length(levels(chickwts$feed)); n.lev
```

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

```
chickwts$feed <- factor(chickwts$feed, levels = names(sort(with(chickwts, tapply(weight, feed, median)))))
boxplot(chickwts$weight ~ chickwts$feed, col= colors()[c(1:n.lev)*10], xaxt = 'n')
axis(side = 1, at= 1:6, labels = levels(chickwts$feed), col.axis= 4, las =2)
box()
```




```
names(sort(with(chickwts, tapply(weight, feed, median))))
```

```
## [1] "horsebean" "linseed"   "soybean"   "meatmeal"  "sunflower" "casein"
```

Exercise 2

```
x <- rnorm(3000,1,1); y <- rnorm(3500,1,1)
mytest <- function(x, y, test = 'two-sided', alpha = 0.05) {
  xbar <- mean(x); ybar <- mean(y)
  xsd <- sd(x); ysd <- sd(y)
  n1 <- length(x); n2 <- length(y)
  ndf <- n1 + n2 - 2
  t <- (xbar-ybar)/sqrt( ((n1-1)*xsd^2 + (n2-1)*ysd^2)/(n1+n2 - 2) * (1/n1 + 1/n2) )
  if( abs(t) > qt(alpha/2, df = ndf) |
      t > qt(1-alpha, df = ndf) |
      t < qt(alpha, df = ndf)) {
    return('reject H0')
  }
  else return('accept H0')
}
mytest(x,y)
```

```
## [1] "reject H0"
```

```
t.test(x,y)
```

```
##
##  Welch Two Sample t-test
##
## data:  x and y
## t = 0.81618, df = 6340.7, p-value = 0.4144
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.02833731  0.06876559
## sample estimates:
## mean of x mean of y
## 0.9951434 0.9749292
```

```
myttest <- function( x, y, test = "two-sided", alpha = 0.05){
  n1 <- length(x)
  n2 <- length(y)
  ndf <- n1 + n2 - 2
  s2 <- ((n1 - 1) * var(x) + (n2 - 1) * var(y) ) / ndf
  tstat <- ( mean(x) - mean(y) ) / sqrt( s2 * (1/n1 + 1/n2))
  tail.area <- switch( test,
                       "two-sided" = 2 * (1 - pt( abs(tstat), ndf )),
                       "lower" = pt(tstat, ndf),
                       "upper" = 1 - pt(tstat, ndf),
                       { warning( "test must be ??two-sided??, ??lower??, or ??upper?? " ) } )
  list( tstat = tstat, df = ndf, reject = if(!is.null(tail.area)) tail.area < alpha, tail.area = tail.area
)
}
x <- rnorm(30); y <- rnorm(35) + 1;
myttest( x, y )
```

```
## $tstat
## [1] -4.128356
##
## $df
## [1] 63
##
## $reject
## [1] TRUE
##
## $tail.area
## [1] 0.0001093821
```

```
myttest(x, y , test = "a")
```

```
## Warning in myttest(x, y, test = "a"): test must be ??two-sided??, ??
## lower??, or ??upper??
```

```
## $tstat
## [1] -4.128356
##
## $df
## [1] 63
##
## $reject
## [1] FALSE
##
## $tail.area
## [1] "test must be ??two-sided??, ??lower??, or ??upper?? "
```

Exercise 3

```
rm(list = ls())
fun1 <- function(x) {
  y <- 1+ x
  return(y)
}
fun2 <- function(x) {
  y <<- 1 + x
  return(y)
}
# <<-  global environment에 변수 입력
fun1(10) #y --> Error: object 'y' not found
```

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

```
fun2(10) ; y # y = 11 로 적용
```

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

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

```
x <- 1:10
fun <- function(z) {
  res <- z + x^2
  return(res)
}
fun(10)
```

```
##  [1]  11  14  19  26  35  46  59  74  91 110
```

local variable은 함수 안에서만 적용하는 함수 global variable은 lobal Enviroment에 적용되는 변수

Exercise 4

1

```
dir(R.home('doc'))
```

```
##  [1] "AUTHORS"           "BioC_mirrors.csv" "CHANGES"
##  [4] "CHANGES.rds"       "COPYING"          "COPYRIGHTS"
##  [7] "CRAN_mirrors.csv" "FAQ"              "html"
## [10] "KEYWORDS"          "KEYWORDS.db"      "manual"
## [13] "NEWS"              "NEWS.0"           "NEWS.1"
## [16] "NEWS.2"            "NEWS.pdf"         "NEWS.rds"
## [19] "README.packages"  "README.Rterm"     "RESOURCES"
## [22] "rw-FAQ"            "THANKS"
```

```
authors <- readLines(file.path(R.home('doc'), 'AUTHORS'))[-(1:8)]
authors <- authors[1:20]
lastname <- gsub(' .*', '', authors)
```

2

```
x <- c('a', 1, 'd' ,F)
# F --> 'FALSE' 문자로 변환되는 것을 주의!
res <- grep(pattern = "[^[:alpha:]]", x= x, value = F)
if(res) cat('Not all alphabetic characters!')
```

```
## Not all alphabetic characters!
```

3

```
dat <- readLines('D:/숭실대/R&Hadoop/ullyses.txt')
words <- unlist(strsplit(dat, split = "[[:space:][:punct:]]"))
words <- tolower(words)

#words[grep(pattern = "[0-9]", x = words)]
#words[grep(pattern = "\\d", x = words)]
words <- gsub("[0-9]", "", words)
words <- words[words != ""]

wordcount <- table(words)

head(sort(wordcount, decreasing =  T), n = 20)
```

```
## words
##   the    of   and     a    to    in    he   his     i     s  that  with
## 15129  8260  7285  6581  5043  5004  4226  3333  3009  2837  2795  2562
##    it   was    on   you   for   her   him    is
##  2531  2134  2126  2084  1963  1786  1526  1462
```

Exercise 5

2

```
data("InsectSprays")
str(InsectSprays)
```

```
## 'data.frame':    72 obs. of  2 variables:
##  $ count: num  10 7 20 14 14 12 10 23 17 20 ...
##  $ spray: Factor w/ 6 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
tapply(InsectSprays$count, InsectSprays$spray, length)
```

```
##  A  B  C  D  E  F
## 12 12 12 12 12 12
```

```
data(Orange)
str(Orange)
```

```
## Classes 'nfnGroupedData', 'nfGroupedData', 'groupedData' and 'data.frame':   35 obs. of  3 variables:
##  $ Tree         : Ord.factor w/ 5 levels "3"<"1"<"5"<"2"<..: 2 2 2 2 2 2 2 4 4 4 ...
##  $ age          : num  118 484 664 1004 1231 ...
##  $ circumference: num  30 58 87 115 120 142 145 33 69 111 ...
##  - attr(*, "formula")=Class 'formula'  language circumference ~ age | Tree
##   .. ..- attr(*, ".Environment")=<environment: R_EmptyEnv>
##  - attr(*, "labels")=List of 2
##   ..$ x: chr "Time since December 31, 1968"
##   ..$ y: chr "Trunk circumference"
##  - attr(*, "units")=List of 2
##   ..$ x: chr "(days)"
##   ..$ y: chr "(mm)"
```

```
apply(Orange[,2:3], 2, mean)
```

```
##          age circumference
##     922.1429      115.8571
```

```
sapply(Orange[,2:3], mean)
```

```
##          age circumference
##     922.1429      115.8571
```

```
lapply(Orange[,2:3], mean)
```

```
## $age
## [1] 922.1429
##
## $circumference
## [1] 115.8571
```

```
by(Orange[,2:3], Orange[,1], colMeans)
```

```
## Orange[, 1]: 3
##           age circumference
##      922.1429       94.0000
## ------------------------------------------------------------
## Orange[, 1]: 1
##           age circumference
##     922.14286      99.57143
## ------------------------------------------------------------
## Orange[, 1]: 5
##           age circumference
##      922.1429      111.1429
## ------------------------------------------------------------
## Orange[, 1]: 2
##           age circumference
##      922.1429      135.2857
## ------------------------------------------------------------
## Orange[, 1]: 4
##           age circumference
##      922.1429      139.2857
```

3

```
library(stats)
(groups <- as.factor(rbinom(32, n =5, prob =0.4)))
```

```
## [1] 11 12 10 14 16
## Levels: 10 11 12 14 16
```

```
tapply(groups, groups, length)
```

```
## 10 11 12 14 16
##  1  1  1  1  1
```

```
table(groups)
```

```
## groups
## 10 11 12 14 16
##  1  1  1  1  1
```

```
str(warpbreaks)
```

```
## 'data.frame':    54 obs. of  3 variables:
##  $ breaks : num  26 30 54 25 70 52 51 26 67 18 ...
##  $ wool   : Factor w/ 2 levels "A","B": 1 1 1 1 1 1 1 1 1 1 ...
##  $ tension: Factor w/ 3 levels "L","M","H": 1 1 1 1 1 1 1 1 1 2 ...
```

```
tapply(warpbreaks$breaks, warpbreaks[,-1], sum)
```

```
##     tension
## wool   L   M   H
##    A 401 216 221
##    B 254 259 169
```

```
tapply(warpbreaks$breaks, warpbreaks[,3, drop = F], sum)
```

```
## tension
##   L   M   H
## 655 475 390
```

```r
n <- 17; fac <- factor(rep_len(1:3,n), levels = 1:5)
table(fac)
```

```
## fac
## 1 2 3 4 5
## 6 6 5 0 0
```

```r
tapply(1:n, fac, sum)
```

```
##  1  2  3  4  5
## 51 57 45 NA NA
```

```r
# 1 ~ 17의 숫자에 소속 번호 1,2,3,1,2,... 을 부여한 것이다. 그러므로 각 1,2,3 그룹의 합을 구해줌.
```

```r
tapply(1:n, fac, sum, default = 0)
```

```
##  1  2  3  4  5
## 51 57 45  0  0
```

```r
tapply(1:n, fac, sum, simplify = F)
```

```
## $`1`
## [1] 51
##
## $`2`
## [1] 57
##
## $`3`
## [1] 45
##
## $`4`
## NULL
##
## $`5`
## NULL
```

```r
tapply(1:n, fac, range)
```

```
## $`1`
## [1]  1 16
##
## $`2`
## [1]  2 17
##
## $`3`
## [1]  3 15
##
## $`4`
## NULL
##
## $`5`
## NULL
```

```r
tapply(1:n, fac, quantile)
```

```
## $`1`
##    0%   25%   50%   75%  100%
##  1.00  4.75  8.50 12.25 16.00
##
## $`2`
##    0%   25%   50%   75%  100%
##  2.00  5.75  9.50 13.25 17.00
##
## $`3`
##   0%  25%  50%  75% 100%
##    3    6    9   12   15
##
## $`4`
## NULL
##
## $`5`
## NULL
```

```
tapply(1:n, fac, length)
```

```
##  1  2  3  4  5
##  6  6  5 NA NA
```

```
tapply(1:n, fac, length, default = 0)
```

```
## 1 2 3 4 5
## 6 6 5 0 0
```

```
# == table(fac)
```

```
str(presidents)
```

```
##  Time-Series [1:120] from 1945 to 1975: NA 87 82 75 63 50 43 32 35 60 ...
```

```
tapply(presidents, cycle(presidents), mean, na.rm = T)
```

```
##        1        2        3        4
## 58.44828 56.43333 57.22222 53.07143
```

```
ind <- list(c(1,2,1), c('A', 'A', 'B'))
table(ind)
```

```
##      ind.2
## ind.1 A B
##     1 1 1
##     2 1 0
```

```
tapply(1:3, ind) # table에서 원소의 위치를 나타내는 벡터
```

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

```
tapply(1:3, ind, sum) # table에 나타나는 원소
```

```
##   A  B
## 1 1  3
## 2 2 NA
```