IntroR.Results

Sung Won Kang

2016년 10월 29일

1주차: R 기초

scalar 와 벡터

1. 설치가 잘 되었을까?

```
#summary(cars)
help(apply)
?apply
?help.search
example(apply)
```

```
##
## apply> ## Compute row and column sums for a matrix:
## apply> x <- cbind(x1 = 3, x2 = c(4:1, 2:5))
##
## apply> dimnames(x)[[1]] <- letters[1:8]
##
## apply> apply(x, 2, mean, trim = .2)
## x1 x2
##  3  3
##
## apply> col.sums <- apply(x, 2, sum)
##
## apply> row.sums <- apply(x, 1, sum)
##
## apply> rbind(cbind(x, Rtot = row.sums), Ctot = c(col.sums, sum(col.sums)))
##      x1 x2 Rtot
## a     3  4    7
## b     3  3    6
## c     3  2    5
## d     3  1    4
## e     3  2    5
```

```
## f     3  3    6
## g     3  4    7
## h     3  5    8
## Ctot 24 24   48
##
## apply> stopifnot( apply(x, 2, is.vector))
##
## apply> ## Sort the columns of a matrix
## apply> apply(x, 2, sort)
##      x1 x2
## [1,]  3  1
## [2,]  3  2
## [3,]  3  2
## [4,]  3  3
## [5,]  3  3
## [6,]  3  4
## [7,]  3  4
## [8,]  3  5
##
## apply> ## keeping named dimnames
## apply> names(dimnames(x)) <- c("row", "col")
##
## apply> x3 <- array(x, dim = c(dim(x),3),
## apply+       dimnames = c(dimnames(x), list(C = paste0("cop.",1:3))))
##
## apply> identical(x,  apply( x,  2,  identity))
## [1] TRUE
##
## apply> identical(x3, apply(x3, 2:3, identity))
## [1] TRUE
##
## apply> ## Don't show:
## apply> xN <- x; dimnames(xN) <- list(row=NULL, col=NULL)
##
## apply> x2 <- x; names(dimnames(x2)) <- NULL
##
## apply> fXY <- function(u) c(X=u[1], Y=u[2])
##
## apply> ax1 <- apply(x, 1, fXY)
```

```
##
## apply> ax2 <- apply(x2,1, fXY)
##
## apply> stopifnot(identical(dimnames(ax1), list(col=c("X.x1", "Y.x2"), row=letters[1:8])),
## apply+           identical(dimnames(ax2), unname(dimnames(ax1))),
## apply+           identical( x, apply( x, 2, identity)),
## apply+           identical(xN, apply(xN, 2, identity)),
## apply+     identical(dimnames(x),
## apply+           dimnames(apply(x, 2, format))),
## apply+           identical(x3, apply(x3, 2:3, identity)),
## apply+     identical(dimnames(apply(x3, 2:1, identity)),
## apply+           dimnames(x3)[3:1]))
##
## apply> rm(xN, x2, fXY, ax1, ax2)
##
## apply> ## End(Don't show)
## apply> ##- function with extra args:
## apply> cave <- function(x, c1, c2) c(mean(x[c1]), mean(x[c2]))
##
## apply> apply(x, 1, cave,  c1 = "x1", c2 = c("x1","x2"))
##       row
##          a b   c d   e f   g h
##   [1,] 3.0 3 3.0 3 3.0 3 3.0 3
##   [2,] 3.5 3 2.5 2 2.5 3 3.5 4
##
## apply> ma <- matrix(c(1:4, 1, 6:8), nrow = 2)
##
## apply> ma
##      [,1] [,2] [,3] [,4]
## [1,]    1    3    1    7
## [2,]    2    4    6    8
##
## apply> apply(ma, 1, table)  #--> a list of length 2
## [[1]]
##
## 1 3 7
## 2 1 1
##
## [[2]]
```

```
##
## 2 4 6 8
## 1 1 1 1
##
##
## apply> apply(ma, 1, stats::quantile) # 5 x n matrix with rownames
##      [,1] [,2]
## 0%      1  2.0
## 25%     1  3.5
## 50%     2  5.0
## 75%     4  6.5
## 100%    7  8.0
##
## apply> stopifnot(dim(ma) == dim(apply(ma, 1:2, sum)))
##
## apply> ## Example with different lengths for each call
## apply> z <- array(1:24, dim = 2:4)
##
## apply> zseq <- apply(z, 1:2, function(x) seq_len(max(x)))
##
## apply> zseq         ## a 2 x 3 matrix
##      [,1]       [,2]       [,3]
## [1,] Integer,19 Integer,21 Integer,23
## [2,] Integer,20 Integer,22 Integer,24
##
## apply> typeof(zseq) ## list
## [1] "list"
##
## apply> dim(zseq) ## 2 3
## [1] 2 3
##
## apply> zseq[1,]
## [[1]]
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
##
## [[2]]
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
##
## [[3]]
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
##
##
## apply> apply(z, 3, function(x) seq_len(max(x)))
## [[1]]
## [1] 1 2 3 4 5 6
##
## [[2]]
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12
##
## [[3]]
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
##
## [[4]]
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
## [24] 24
##
##
## apply> # a list without a dim attribute
## apply>
## apply>
## apply>
```

2. 메모리 점검

- 우리는 어디에 있을까?

```
getwd()
```

```
## [1] "/Users/sungwonkang/OneDrive/work_2016/Rstat/Lesson1"
```

- 지금 메모리에 떠 있는 것은?

```
objects()
```

```
## [1] "cave"     "col.sums" "ma"       "row.sums" "x"        "x3"
## [7] "z"        "zseq"
```

```
ls()
```

```
## [1] "cave"     "col.sums" "ma"       "row.sums" "x"        "x3"
## [7] "z"        "zseq"
```

- 메모리를 비우려면 ? rm

```
list.ls=ls()
rm(list=list.ls[1])
ls()
```

```
## [1] "col.sums" "list.ls"  "ma"       "row.sums" "x"        "x3"
## [7] "z"        "zseq"
```

```
rm(list=ls())
ls()
```

```
## character(0)
```

- Tip: R은 대소문자를 구분

```
#HELP(apply)
```

3. Scalar

i.만들기

```
#Four types. Numeric,character,logic,factor
#숫자(numeric)
x1=10
x2=2.3
x3=3
#문자(string)
S1="Hello world"
S2="My name is ..."
#논리연산자(logic)
L1=TRUE
L2=FALSE
```

ii. 연산

```
x1+x2
```

```
## [1] 12.3
```

```
x1-x2
```

```
## [1] 7.7
```

```
x1/x2
```

```
## [1] 4.347826
```

```
x1*x2
```

```
## [1] 23
```

```
x1^x2
```

```
## [1] 199.5262
```

```
x1%/%x3 #몫
```

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

```
x1 %% x3 #나머지
```

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

```
x1%/%x2 #몫
```

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

```
x1 %% x2 #나머지
```

```
## [1] 0.8
```

```
nchar(S1)# 문자 수
```

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

```
paste(S1,S2) # 붙이기
```

```
## [1] "Hello world My name is ..."
```

```
L1|L2 # 둘 중 하나는 맞아요
```

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

```
L1 & L2 # 둘 다 맞아요
```

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

```
! L1 # L1은 아니에요
```

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

```
! L2
```

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

4. 벡터: 간단한 조작

i. 벡터 만들기 1: 숫자 집어넣기

```
x=c(10.4, 5.6, 3.1, 6.4, 21.7)
x
```

```
## [1] 10.4  5.6  3.1  6.4 21.7
```

```r
xn=(1:10)
xn
```

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

```r
xr=rep(c(0,2),5)
xr
```

```
##  [1] 0 2 0 2 0 2 0 2 0 2
```

ii. 벡터와 숫자(Scalar) 연산: 쉬운것부터

```r
x+2
```

```
## [1] 12.4  7.6  5.1  8.4 23.7
```

```r
x-2
```

```
## [1]  8.4  3.6  1.1  4.4 19.7
```

```r
x^3
```

```
## [1]  1124.864   175.616    29.791   262.144 10218.313
```

```r
1/x
```

```
## [1] 0.09615385 0.17857143 0.32258065 0.15625000 0.04608295
```

iii. 벡터 늘리기

```r
y=c(x,0)
y
```

```
## [1] 10.4  5.6  3.1  6.4 21.7  0.0
```

```r
y=c(x,0,y)
y
```

```
##  [1] 10.4  5.6  3.1  6.4 21.7  0.0 10.4  5.6  3.1  6.4 21.7  0.0
```

```r
y3=rep(x,10)
y3
```

```
##  [1] 10.4  5.6  3.1  6.4 21.7 10.4  5.6  3.1  6.4 21.7 10.4  5.6  3.1  6.4
## [15] 21.7 10.4  5.6  3.1  6.4 21.7 10.4  5.6  3.1  6.4 21.7 10.4  5.6  3.1
## [29]  6.4 21.7 10.4  5.6  3.1  6.4 21.7 10.4  5.6  3.1  6.4 21.7 10.4  5.6
## [43]  3.1  6.4 21.7 10.4  5.6  3.1  6.4 21.7
```

iv. 벡터 연산: 벡터와 벡터

a. 서로 다른 길이의 벡터를 더하면? 길이가 늘어남(recycle) *칠판에 써서..

```
v1=2*x+4
v1
```

```
## [1] 24.8 15.2 10.2 16.8 47.4
```

```
v2=2*x+y+1
```

```
## Warning in 2 * x + y: longer object length is not a multiple of shorter
## object length
```

```
v2
```

```
##  [1] 32.2 17.8 10.3 20.2 66.1 21.8 22.6 12.8 16.9 50.8 43.5 12.2
```

```
v3=v1+v2
```

```
## Warning in v1 + v2: longer object length is not a multiple of shorter
## object length
```

```
v3
```

```
##  [1]  57.0  33.0  20.5  37.0 113.5  46.6  37.8  23.0  33.7  98.2  68.3
## [12]  27.4
```

```
length(x)
```

```
## [1] 5
```

```
length(y)
```

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

```
length(v1)
```

```
## [1] 5
```

```
length(v2)
```

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

```
length(v3)
```

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

b. 벡터 연산 2.

```
x
```

```
## [1] 10.4  5.6  3.1  6.4 21.7
```

```
v1
```

```
## [1] 24.8 15.2 10.2 16.8 47.4
```

```
x+v1
```

```
## [1] 35.2 20.8 13.3 23.2 69.1
```

```
x-v1
```

```
## [1] -14.4  -9.6  -7.1 -10.4 -25.7
```

```
x*v1
```

```
## [1]  257.92   85.12   31.62  107.52 1028.58
```

```
x/v1
```

```
## [1] 0.4193548 0.3684211 0.3039216 0.3809524 0.4578059
```

```
x^v1
```

```
## [1] 1.668887e+25 2.357534e+11 1.027754e+05 3.498030e+13 2.229417e+63
```

길이가 다르면?

```
x/y
```

```
## Warning in x/y: longer object length is not a multiple of shorter object
## length
```

```
##  [1] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000       Inf 0.5384615
##  [8] 0.5535714 2.0645161 3.3906250 0.4792627       Inf
```

```
x^y
```

```
## Warning in x^y: longer object length is not a multiple of shorter object
## length
```

```
##  [1] 3.776998e+10 1.548292e+04 3.335963e+01 1.443949e+05 1.002712e+29
##  [6] 1.000000e+00 6.041649e+07 5.644500e+02 3.156152e+02 3.575545e+08
## [11] 1.173879e+22 1.000000e+00
```

c. 벡터연산 3. 함수

```
#로그
log(x)
```

```
## [1] 2.341806 1.722767 1.131402 1.856298 3.077312
```

```
#초월함수
exp(x)
```

```
## [1] 3.285963e+04 2.704264e+02 2.219795e+01 6.018450e+02 2.655769e+09
```

```r
#삼각함수
sin(x)
```

```
## [1] -0.82782647 -0.63126664  0.04158066  0.11654920  0.28705265
```

```r
cos(x)
```

```
## [1] -0.5609843  0.7755659 -0.9991352  0.9931849 -0.9579148
```

```r
tan(x)
```

```
## [1]  1.47566791 -0.81394328 -0.04161665  0.11734895 -0.29966407
```

```r
#지수함수
sqrt(x)
```

```
## [1] 3.224903 2.366432 1.760682 2.529822 4.658326
```

```r
#길이
length(x)
```

```
## [1] 5
```

```r
#합, 곱
sum(x)
```

```
## [1] 47.2
```

```r
prod(x)
```

```
## [1] 25073.95
```

```r
#표본통계치
max(x)
```

```
## [1] 21.7
```

```r
min(x)
```

```
## [1] 3.1
```

```r
range(x)
```

```
## [1]  3.1 21.7
```

```r
mean(x)
```

```
## [1] 9.44
```

```r
var(x)
```

```
## [1] 53.853
```

```
sd(x)
```

```
## [1] 7.33846
```

```
median(x)
```

```
## [1] 6.4
```

```
#정렬
sort(x)
```

```
## [1]  3.1  5.6  6.4 10.4 21.7
```

```
sort(x,decreasing=T)
```

```
## [1] 21.7 10.4  6.4  5.6  3.1
```

만약 두 개를 넣으면?

```
## 뱉어내거나
#초월함수
#exp(x,v1)
#길이
#length(x,v1)
#정렬
#sort(x,v1)
# 평균
#mean(x,v1)

## 뭉뜽거리거나
#합, 곱
sum(x,v1)
```

```
## [1] 161.6
```

```
#표본통계치
max(x,v1)
```

```
## [1] 47.4
```

```
min(x,v1)
```

```
## [1] 3.1
```

```
range(x,v1)
```

```
## [1]  3.1 47.4
```

```r
var(x,v1)
```

```
## [1] 107.706
```

```r
## 첫번째 것만 쓰거나
sd(x,v1)
```

```
## Warning in if (na.rm) "na.or.complete" else "everything": the condition has
## length > 1 and only the first element will be used
```

```
## [1] 7.33846
```

```r
median(x,v1)
```

```
## Warning in if (na.rm) x <- x[!is.na(x)] else if (any(is.na(x)))
## return(x[FALSE][NA]): the condition has length > 1 and only the first
## element will be used
```

```
## [1] 6.4
```

v. 벡터 만들기 2: 수열,논리,문자

w. 수열

```r
xn=(1:10)
xn
```

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

```r
## 등차수열
# 항의 수를 모르는 경우
xseq=seq(1,9,2)
xseq
```

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

```r
xseq.1=seq(from=1,to=9,by=2)
# 마지막 값을 모르는 경우
xseq.2=seq(from=1,length=5,by=2)
xseq.2
```

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

```r
# 등차를 모르는 경우
xseq.3=seq(from=1,to=9,length=5)
xseq.3
```

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

```
## 등비수열?
## 초항 2. 등비 2. 항수 10 등비수열
xseqp=2^xn
xseqp
```

```
##  [1]    2    4    8   16   32   64  128  256  512 1024
```

```
## 초항 100. 등비 2. 항수 10 등비수열
xseqp2=100*(2^(0:9))
xseqp2
```

```
##  [1]   100   200   400   800  1600  3200  6400 12800 25600 51200
```

```
## 초항 100, 등비 1/2. 항수 10 등비수열
xseqp3=100*((1/2)^(0:9))
xseqp3
```

```
##  [1] 100.0000000  50.0000000  25.0000000  12.5000000   6.2500000
##  [6]   3.1250000   1.5625000   0.7812500   0.3906250   0.1953125
```

b. 논리 : 일치. 불일치.대소관계. and/or

```
xeq=(x == 4)
xeq
```

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

```
xneq=(x != 4)
xneq
```

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

```
xge=(x >= 4)
xge
```

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

```
xgt=(x > 4)
xgt
```

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

```
xge=(x >= 4)
xge
```

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

```
xle=(x <= 4)
xle
```

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

```
xlt=(x < 4)
xlt
```

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

```
xand=(xgt & xge)
xand
```

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

```
xge
```

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

```
xor=(xgt | xlt)
xor
```

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

```
xneq
```

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

c. 문자 그리고 paste

```
xs=c("1","2","3","4")
#xsa=c(A,B,C,D)
#xsa
xsa=c("A","B","C","D")
xsa
```

```
## [1] "A" "B" "C" "D"
```

- 문자열을 붙일때

```
paste(xsa,collapse="")
```

```
## [1] "ABCD"
```

```
paste(xsa,collapse=",")
```

```
## [1] "A,B,C,D"
```

```
paste(xsa,collapse="-")
```

```
## [1] "A-B-C-D"
```

- 문자열에 1렬변호를 매길때

```
paste(xsa,(1:5),sep="")
```

```
## [1] "A1" "B2" "C3" "D4" "A5"
```

```
paste(xsa,(1:5),sep="-")
```

```
## [1] "A-1" "B-2" "C-3" "D-4" "A-5"
```

```
paste("X",(1:5),sep="-")
```

```
## [1] "X-1" "X-2" "X-3" "X-4" "X-5"
```

vi. 번호매개기(index), 고르기

a. 비어있음(missing value: NA), 숫자가 아님(NaN)

```
z=c((-10:10),NA)
z
```

```
##  [1] -10  -9  -8  -7  -6  -5  -4  -3  -2  -1   0   1   2   3   4   5   6
## [18]   7   8   9  10  NA
```

```
is.na(z)
```

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

```
z1=0/0
z1
```

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

```
is.na(z1)
```

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

```
is.nan(z)
```

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

```
is.nan(z1)
```

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

b. 벡터 중 일부만 고르기

- 조건문으로 고르기

```r
#NA 가 아닌것
z[!is.na(z)]
```

```
##  [1] -10  -9  -8  -7  -6  -5  -4  -3  -2  -1   0   1   2   3   4   5   6
## [18]   7   8   9  10
```

```r
#0 보다 큰 것
z[z>0]
```

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

```r
#0 보다 크고 NA가 아닌 것
z[z>0&!is.na(z)]
```

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

```r
z+1
```

```
##  [1] -9 -8 -7 -6 -5 -4 -3 -2 -1  0  1  2  3  4  5  6  7  8  9 10 11 NA
```

```r
(z+1)[z>0&!is.na(z)]
```

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

- 숫자로 지정

```r
# 홀수항, 짝수항
zl=length(z)
z[seq(1,zl,2)]
```

```
##  [1] -10  -8  -6  -4  -2   0   2   4   6   8  10
```

```r
z[seq(0,zl,2)]
```

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

```r
# 내맘대로
z[c(1,11,4)]
```

```
## [1] -10   0  -7
```

```r
# 내 맘대로 빼고
z[-c(1,4,14)]
```

```
##  [1] -9 -8 -6 -5 -4 -3 -2 -1  0  1  2  4  5  6  7  8  9 10 NA
```

- 문자로 지정

```r
fruit=c(5,1,10,10,20,20)
names(fruit)=c("OR","BA","AP","AP","PE","PE")
```

```
fruit[c("OR","BA")]
```

```
## OR BA
##  5  1
```

```
fruit[c(1,2)]
```

```
## OR BA
##  5  1
```

- 골라서 바꾸기

```
#NA는 0으로
z[is.na(z)]=0
z
```

```
##  [1] -10  -9  -8  -7  -6  -5  -4  -3  -2  -1   0   1   2   3   4   5   6
## [18]   7   8   9  10   0
```

```
#0 보다 작으면 100으로
z[z<0]=100
z
```

```
##  [1] 100 100 100 100 100 100 100 100 100 100   0   1   2   3   4   5   6
## [18]   7   8   9  10   0
```

vii. type 바꾸기. 범주변수

a. type 바꾸기

```
str(fruit)
```

```
##  Named num [1:6] 5 1 10 10 20 20
##  - attr(*, "names")= chr [1:6] "OR" "BA" "AP" "AP" ...
```

```
#문자열로
fruit.s=as.character(fruit)
#sum(fruit.s)
str(fruit.s)
```

```
##  chr [1:6] "5" "1" "10" "10" "20" "20"
```

```
nchar(fruit.s)
```

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

```
paste(fruit.s,collapse="-")
```

```
## [1] "5-1-10-10-20-20"
```

```
#다시숫자로
fruit.n=as.numeric(fruit.s)
str(fruit.n)
```

```
##  num [1:6] 5 1 10 10 20 20
```

```
sum(fruit.n)
```

```
## [1] 66
```

```
nchar(fruit.s)
```

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

```
paste(fruit.n,collapse="-")
```

```
## [1] "5-1-10-10-20-20"
```

- 범주변수

```
state = c("tas", "sa", "qld", "nsw", "nsw", "nt", "wa", "wa",
"qld", "vic", "nsw", "vic", "qld", "qld", "sa", "tas",
"sa", "nt", "wa", "vic", "qld", "nsw", "nsw", "wa",
"sa", "act", "nsw", "vic", "vic", "act")

incomes = c(60, 49, 40, 61, 64, 60, 59, 54, 62, 69, 70, 42, 56,
61, 61, 61, 58, 51, 48, 65, 49, 49, 41, 48, 52, 46,
59, 46, 58, 43)

statef=factor(state)
levels(statef)
```

```
## [1] "act" "nsw" "nt"  "qld" "sa"  "tas" "vic" "wa"
```

```
# 조건부 표본통계량
incmeans=tapply(incomes,statef,mean)
incster=tapply(incomes,statef,sd)
incfreq=tapply(incomes,statef,length)
table(statef)
```

```
## statef
## act nsw  nt qld  sa tas vic  wa
##   2   6   2   5   4   2   5   4
```

```
incomef=factor(cut(incomes, 4))
table(statef,incomef)
```

```
##       incomef
## statef (40,47.5] (47.5,55] (55,62.5] (62.5,70]
##    act         2         0         0         0
##    nsw         1         1         2         2
##    nt          0         1         1         0
##    qld         1         1         3         0
##    sa          0         2         2         0
##    tas         0         0         2         0
##    vic         2         0         1         2
##    wa          0         3         1         0
```

```
incomeh=factor(cut(incomes,2))
table(statef,incomef,incomeh)
```

```
## , , incomeh = (40,55]
##
##       incomef
## statef (40,47.5] (47.5,55] (55,62.5] (62.5,70]
##    act         2         0         0         0
##    nsw         1         1         0         0
##    nt          0         1         0         0
##    qld         1         1         0         0
##    sa          0         2         0         0
##    tas         0         0         0         0
##    vic         2         0         0         0
##    wa          0         3         0         0
##
## , , incomeh = (55,70]
##
##       incomef
## statef (40,47.5] (47.5,55] (55,62.5] (62.5,70]
##    act         0         0         0         0
##    nsw         0         0         2         2
##    nt          0         0         1         0
##    qld         0         0         3         0
##    sa          0         0         2         0
##    tas         0         0         2         0
##    vic         0         0         1         2
##    wa          0         0         1         0
```

- 범주 변수의 type 변환

```
as.character(statef)
```

```
##  [1] "tas" "sa"  "qld" "nsw" "nsw" "nt"  "wa"  "wa"  "qld" "vic" "nsw"
## [12] "vic" "qld" "qld" "sa"  "tas" "sa"  "nt"  "wa"  "vic" "qld" "nsw"
## [23] "nsw" "wa"  "sa"  "act" "nsw" "vic" "vic" "act"
```

```
as.numeric(statef)
```

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

```
fruit.f=as.factor(fruit)
as.numeric(fruit.f)
```

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

Array, matrix (multi-dimension)

1. Array

```
z=(1:24)
z
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
## [24] 24
```

```
str(z)
```

```
##  int [1:24] 1 2 3 4 5 6 7 8 9 10 ...
```

```
dim(z)=c(3,4,2)
z
```

```
## , , 1
##
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
##
## , , 2
##
##      [,1] [,2] [,3] [,4]
## [1,]   13   16   19   22
## [2,]   14   17   20   23
## [3,]   15   18   21   24
```

```
str(z)
```

```
##  int [1:3, 1:4, 1:2] 1 2 3 4 5 6 7 8 9 10 ...
```

```
z[1:4]
```

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

```
z[1,,]
```

```
##      [,1] [,2]
## [1,]    1   13
## [2,]    4   16
## [3,]    7   19
## [4,]   10   22
```

```
z[,1,]
```

```
##      [,1] [,2]
## [1,]    1   13
## [2,]    2   14
## [3,]    3   15
```

```
z[,,1]
```

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

2.matrix= Array with two subscripts

```
dim(z)=c(8,3)
z
```

```
##      [,1] [,2] [,3]
## [1,]    1    9   17
## [2,]    2   10   18
## [3,]    3   11   19
## [4,]    4   12   20
## [5,]    5   13   21
## [6,]    6   14   22
## [7,]    7   15   23
## [8,]    8   16   24
```

```r
str(z)
```

```
##  int [1:8, 1:3] 1 2 3 4 5 6 7 8 9 10 ...
```

```r
z[1:4]
```

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

```r
z[1,]
```

```
## [1]  1  9 17
```

```r
z[,1]
```

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

```r
A=array(incomes[1:24],c(8,3))

## 행렬계산
A+z
```

```
##      [,1] [,2] [,3]
## [1,]   61   71   75
## [2,]   51   79   69
## [3,]   43   81   67
## [4,]   65   54   85
## [5,]   69   69   70
## [6,]   66   75   71
## [7,]   66   76   64
## [8,]   62   77   72
```

```r
A-z
```

```
##      [,1] [,2] [,3]
## [1,]   59   53   41
## [2,]   47   59   33
## [3,]   37   59   29
## [4,]   57   30   45
## [5,]   59   43   28
## [6,]   54   47   27
## [7,]   52   46   18
## [8,]   46   45   24
```

```r
A*z
```

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

```
## [1,]   60  558  986
## [2,]   98  690  918
## [3,]  120  770  912
## [4,]  244  504 1300
## [5,]  320  728 1029
## [6,]  360  854 1078
## [7,]  413  915  943
## [8,]  432  976 1152
```

```
t(A)%*%z
```

```
##      [,1] [,2] [,3]
## [1,] 2047 5623 9199
## [2,] 2139 5995 9851
## [3,] 1774 5046 8318
```

```
C=(1:3)


A%*%C
```

```
##      [,1]
## [1,]  358
## [2,]  340
## [3,]  324
## [4,]  340
## [5,]  323
## [6,]  329
## [7,]  304
## [8,]  320
```

```
#inner product
t(C)%*%C
```

```
##      [,1]
## [1,]   14
```

```
#outer product
C%o%C
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    2    4    6
## [3,]    3    6    9
```

```r
# 대각행렬
DC=diag(C)
#역행렬
AA=t(A)%*%A
solve(AA)
```

```
##               [,1]          [,2]          [,3]
## [1,]  0.0016632838 -0.0003200201 -0.0014273811
## [2,] -0.0003200201  0.0006612754 -0.0004120349
## [3,] -0.0014273811 -0.0004120349  0.0020581682
```

```r
AA%*%solve(AA)
```

```
##      [,1]          [,2]          [,3]
## [1,]    1 -1.776357e-15 -7.105427e-15
## [2,]    0  1.000000e+00 -7.105427e-15
## [3,]    0 -3.552714e-15  1.000000e+00
```

```r
# 선형연립방정식
y=AA%*%C
solve(AA,y)
```

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

```r
#identity matrix
ID=diag(3)
AA%*%ID-AA
```

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

```r
#zero matrix
nr=3
nc=3
mzero=rep(0,nr*nc)
dim(mzero)=c(nr,nc)

str(z)
```

```
##  int [1:8, 1:3] 1 2 3 4 5 6 7 8 9 10 ...
```

```
str(A)
```

```
##  num [1:8, 1:3] 60 49 40 61 64 60 59 54 62 69 ...
```

```
str(AA)
```

```
##  num [1:3, 1:3] 25415 26600 22951 26600 29568 ...
```

```
str(DC)
```

```
##  num [1:3, 1:3] 1 0 0 0 2 0 0 0 3
```

```
str(ID)
```

```
##  num [1:3, 1:3] 1 0 0 0 1 0 0 0 1
```

```
str(mzero)
```

```
##  num [1:3, 1:3] 0 0 0 0 0 0 0 0 0
```

3. matrix from vector,vector from matrix

```
X1=c(1,2,3)
X2=c(4,100,20)
XX=cbind(1,X1,X2)
str(XX)
```

```
##  num [1:3, 1:3] 1 1 1 1 2 3 4 100 20
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:3] "" "X1" "X2"
```

```
as.vector(XX)
```

```
## [1]   1   1   1   1   2   3   4 100  20
```

```
c(XX)
```

```
## [1]   1   1   1   1   2   3   4 100  20
```

List and dataframe

1. list : 아무거나 다 넣는 공간

```
Lst=list(name="Fred",wife="Mary", no.children=3,child.ages=c(4,7,9))
length(Lst)
```

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

```r
names(Lst)
```

```
## [1] "name"        "wife"        "no.children" "child.ages"
```

```r
str(Lst)
```

```
## List of 4
##  $ name       : chr "Fred"
##  $ wife       : chr "Mary"
##  $ no.children: num 3
##  $ child.ages : num [1:3] 4 7 9
```

```r
Lst[[1]]
```

```
## [1] "Fred"
```

```r
Lst[1]
```

```
## $name
## [1] "Fred"
```

```r
str(Lst[[1]])
```

```
##  chr "Fred"
```

```r
str(Lst[1])
```

```
## List of 1
##  $ name: chr "Fred"
```

```r
sum(Lst[[4]])
```

```
## [1] 20
```

```r
#sum(Lst[4])
Lst2=list(name="강성원",wife="?", no.children=2,child.ages=c(2,9))
LstLong=c(Lst,Lst2)
str(LstLong)
```

```
## List of 8
##  $ name       : chr "Fred"
##  $ wife       : chr "Mary"
##  $ no.children: num 3
##  $ child.ages : num [1:3] 4 7 9
##  $ name       : chr "강성원"
##  $ wife       : chr "?"
##  $ no.children: num 2
```

```
##  $ child.ages : num [1:2] 2 9
```

2. data frame : NxK 형 저장공간

```
DA=data.frame(AA)
Stateincome=data.frame(state,incomes)


str(Stateincome)
```

```
## 'data.frame':    30 obs. of  2 variables:
##  $ state  : Factor w/ 8 levels "act","nsw","nt",..: 6 5 4 2 2 3 8 8 4 7 ...
##  $ incomes: num  60 49 40 61 64 60 59 54 62 69 ...
```

```
summary(Stateincome)
```

```
##      state      incomes
##  nsw    :6   Min.   :40.00
##  qld    :5   1st Qu.:48.25
##  vic    :5   Median :57.00
##  sa     :4   Mean   :54.73
##  wa     :4   3rd Qu.:61.00
##  act    :2   Max.   :70.00
##  (Other):4
```

```
attach(DA)
```

```
## The following objects are masked _by_ .GlobalEnv:
##
##     X1, X2
```

```
detach()
```

3. data frame과 vector

```
X1=(1:10)
X2=(101:110)
X3=(1001:1010)


D=data.frame(X1,X2,X3)
DS_C=apply(D,2,sum)
print("D/DS_C=")
```

```
## [1] "D/DS_C="
```

```
print(D/DS_C)
```

```
##              X1         X2          X3
## 1  0.0181818182 0.09573460  0.09955246
## 2  0.0018957346 0.01014421 18.21818182
## 3  0.0002983590 1.87272727  0.95071090
## 4  0.0727272727 0.09857820  0.09985082
## 5  0.0047393365 0.01044257 18.27272727
## 6  0.0005967181 1.92727273  0.95355450
## 7  0.1272727273 0.10142180  0.10014918
## 8  0.0075829384 0.01074092 18.32727273
## 9  0.0008950771 1.98181818  0.95639810
## 10 0.1818181818 0.10426540  0.10044754
```

```
# 합쳐서 1이 안되네?


print("t(t(D)/DS_C=")
```

```
## [1] "t(t(D)/DS_C="
```

```
print(t(t(D)/DS_C))
```

```
##               X1         X2         X3
##  [1,] 0.01818182 0.09573460 0.09955246
##  [2,] 0.03636364 0.09668246 0.09965191
##  [3,] 0.05454545 0.09763033 0.09975137
##  [4,] 0.07272727 0.09857820 0.09985082
##  [5,] 0.09090909 0.09952607 0.09995027
##  [6,] 0.10909091 0.10047393 0.10004973
##  [7,] 0.12727273 0.10142180 0.10014918
##  [8,] 0.14545455 0.10236967 0.10024863
##  [9,] 0.16363636 0.10331754 0.10034809
## [10,] 0.18181818 0.10426540 0.10044754
```

```
# 이제 합쳐서 1 되네?


a=c(10,100,1000)
print ("a=")
```

```
## [1] "a="
```

```
print (a)
```

```
## [1]   10  100 1000
```

```
print ("D/a=")
```

```
## [1] "D/a="
```

```
print(D/a)
```

```
##       X1     X2      X3
## 1  0.100  1.010   1.001
## 2  0.020  0.102 100.200
## 3  0.003 10.300  10.030
## 4  0.400  1.040   1.004
## 5  0.050  0.105 100.500
## 6  0.006 10.600  10.060
## 7  0.700  1.070   1.007
## 8  0.080  0.108 100.800
## 9  0.009 10.900  10.090
## 10 1.000  1.100   1.010
```

```
print("t(t(D)/a)")
```

```
## [1] "t(t(D)/a)"
```

```
print(t(t(D)/a))
```

```
##        X1   X2    X3
##  [1,] 0.1 1.01 1.001
##  [2,] 0.2 1.02 1.002
##  [3,] 0.3 1.03 1.003
##  [4,] 0.4 1.04 1.004
##  [5,] 0.5 1.05 1.005
##  [6,] 0.6 1.06 1.006
##  [7,] 0.7 1.07 1.007
##  [8,] 0.8 1.08 1.008
##  [9,] 0.9 1.09 1.009
## [10,] 1.0 1.10 1.010
```

```
divby.a=data.frame(t(t(D)/a))
print("divided by a_columnwise")
```

```
## [1] "divided by a_columnwise"
```

```
print(divby.a)
```

```
##     X1   X2    X3
```

```
## 1  0.1 1.01 1.001
## 2  0.2 1.02 1.002
## 3  0.3 1.03 1.003
## 4  0.4 1.04 1.004
## 5  0.5 1.05 1.005
## 6  0.6 1.06 1.006
## 7  0.7 1.07 1.007
## 8  0.8 1.08 1.008
## 9  0.9 1.09 1.009
## 10 1.0 1.10 1.010
```

```
DS_R=apply(D,1,sum)
Share_row=D/DS_R
print ("divison by row sum")
```

```
## [1] "divison by row sum"
```

```
print(Share_row)
```

```
##              X1         X2        X3
## 1  0.0009066183 0.09156845 0.9075249
## 2  0.0018083183 0.09222423 0.9059675
## 3  0.0027051398 0.09287647 0.9044184
## 4  0.0035971223 0.09352518 0.9028777
## 5  0.0044843049 0.09417040 0.9013453
## 6  0.0053667263 0.09481216 0.8998211
## 7  0.0062444246 0.09545049 0.8983051
## 8  0.0071174377 0.09608541 0.8967972
## 9  0.0079858030 0.09671695 0.8952972
## 10 0.0088495575 0.09734513 0.8938053
```

4. data frame에 쓰는 함수들

```
#표본통계치
colSums(D)
```

```
##    X1    X2    X3
##    55  1055 10055
```

```
rowSums(D)
```

```
##  [1] 1103 1106 1109 1112 1115 1118 1121 1124 1127 1130
```

```r
colMeans(D)
```

```
##     X1     X2     X3
##    5.5  105.5 1005.5
```

```r
rowMeans(D)
```

```
##  [1] 367.6667 368.6667 369.6667 370.6667 371.6667 372.6667 373.6667
##  [8] 374.6667 375.6667 376.6667
```

```r
dim(D)
```

```
## [1] 10  3
```

```r
#다른 표본통계치는? apply
makez=function(x) (x-mean(x))/sd(x)
cv=function(x) sd(x)/mean(x)
apply(D,1,FUN=makez)
```

```
##          [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
## X1 -0.6657503 -0.6657503 -0.6657503 -0.6657503 -0.6657503 -0.6657503
## X2 -0.4841820 -0.4841820 -0.4841820 -0.4841820 -0.4841820 -0.4841820
## X3  1.1499323  1.1499323  1.1499323  1.1499323  1.1499323  1.1499323
##          [,7]       [,8]       [,9]      [,10]
## X1 -0.6657503 -0.6657503 -0.6657503 -0.6657503
## X2 -0.4841820 -0.4841820 -0.4841820 -0.4841820
## X3  1.1499323  1.1499323  1.1499323  1.1499323
```

```r
apply(D,2,FUN=makez)
```

```
##               X1         X2         X3
##  [1,] -1.4863011 -1.4863011 -1.4863011
##  [2,] -1.1560120 -1.1560120 -1.1560120
##  [3,] -0.8257228 -0.8257228 -0.8257228
##  [4,] -0.4954337 -0.4954337 -0.4954337
##  [5,] -0.1651446 -0.1651446 -0.1651446
##  [6,]  0.1651446  0.1651446  0.1651446
##  [7,]  0.4954337  0.4954337  0.4954337
##  [8,]  0.8257228  0.8257228  0.8257228
##  [9,]  1.1560120  1.1560120  1.1560120
## [10,]  1.4863011  1.4863011  1.4863011
```

```r
#조건부 표본통계치 aggregate
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
aggregate(data.frame(iris[,1:4]),by=list(iris$Species),FUN=mean)
```

```
##      Group.1 Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     setosa        5.006       3.428        1.462       0.246
## 2 versicolor        5.936       2.770        4.260       1.326
## 3  virginica        6.588       2.974        5.552       2.026
```

```
aggregate(cbind(Sepal.Width,Sepal.Length)~Species,data=iris,FUN=mean )
```

```
##      Species Sepal.Width Sepal.Length
## 1     setosa       3.428        5.006
## 2 versicolor       2.770        5.936
## 3  virginica       2.974        6.588
```

```
aggregate(.~Species,data=iris,FUN=mean )
```

```
##      Species Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     setosa        5.006       3.428        1.462       0.246
## 2 versicolor        5.936       2.770        4.260       1.326
## 3  virginica        6.588       2.974        5.552       2.026
```

```
aggregate(.~Species,data=iris,FUN=cv)
```

```
##      Species Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     setosa   0.07041344   0.1105789   0.11878522   0.4283967
## 2 versicolor   0.08695606   0.1132846   0.11030774   0.1491348
## 3  virginica   0.09652089   0.1084387   0.09940466   0.1355627
```

```
aggregate(.~state,data=Stateincome,FUN=mean)
```

```
##   state  incomes
## 1   act 44.50000
## 2   nsw 57.33333
## 3    nt 55.50000
## 4   qld 53.60000
## 5    sa 55.00000
## 6   tas 60.50000
```

```
## 7   vic 56.00000
## 8    wa 52.25000
```

```
# 줄세우기(order)
Stateincome[order(Stateincome[,1],Stateincome[,2]),]
```

```
##    state incomes
## 30   act      43
## 26   act      46
## 23   nsw      41
## 22   nsw      49
## 27   nsw      59
## 4    nsw      61
## 5    nsw      64
## 11   nsw      70
## 18    nt      51
## 6     nt      60
## 3    qld      40
## 21   qld      49
## 13   qld      56
## 14   qld      61
## 9    qld      62
## 2     sa      49
## 25    sa      52
## 17    sa      58
## 15    sa      61
## 1    tas      60
## 16   tas      61
## 12   vic      42
## 28   vic      46
## 29   vic      58
## 20   vic      65
## 10   vic      69
## 19    wa      48
## 24    wa      48
## 8     wa      54
## 7     wa      59
```

```
Stateincome[order(Stateincome[,1],Stateincome[,2],decreasing=T),]
```

```
##    state incomes
```

```
## 7     wa      59
## 8     wa      54
## 19    wa      48
## 24    wa      48
## 10   vic      69
## 20   vic      65
## 29   vic      58
## 28   vic      46
## 12   vic      42
## 16   tas      61
## 1    tas      60
## 15    sa      61
## 17    sa      58
## 25    sa      52
## 2     sa      49
## 9    qld      62
## 14   qld      61
## 13   qld      56
## 21   qld      49
## 3    qld      40
## 6     nt      60
## 18    nt      51
## 11   nsw      70
## 5    nsw      64
## 4    nsw      61
## 27   nsw      59
## 22   nsw      49
## 23   nsw      41
## 26   act      46
## 30   act      43
```