R-object : LJH

## **R- Object**

# 1.

x <- c(17, 16, 20, 24, 22, 15, 21, 18)  
x

## [1] 17 16 20 24 22 15 21 18

**(1)**

x[x >= 20]

## [1] 20 24 22 21

**(2)**

x[x >= 20] <- 100  
y <- x  
y

## [1] 17 16 100 100 100 15 100 18

# 2.

**(1)**

x <- matrix(c(3, -1, -1, -1, -1, -1, 4, -1, -1, -1, -1, -1, 5, -1, -1, -1, -1, -1, 6, -1, -1, -1, -1, -1, 7), nrow=5)  
x

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

**(2)**

y <- x[,-5]  
y

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

**(3)**

yinfo1 <- c(nrow(y),ncol(y))  
yinfo1

## [1] 5 4

또는

yinfo2 <- dim(y)  
yinfo2

## [1] 5 4

**(4)**

y[y==-1] <- 0  
y1 <- y  
y1

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

# 3.

rowdata <- read.table("c:/data/rowdata.txt", header = TRUE, sep=",")  
head(rowdata)

## V1 V2 V3 V4  
## 1 4 NA 3 2  
## 2 5 2 2 3  
## 3 6 3 NA NA  
## 4 1 4 1 2

**(1)**

rdata <- data.frame(rowdata)  
rdata

## V1 V2 V3 V4  
## 1 4 NA 3 2  
## 2 5 2 2 3  
## 3 6 3 NA NA  
## 4 1 4 1 2

**(2)** is.na() 함수를 사용하면 NA 값이 있는 곳에 TRUE가, 나머지는 FALSE가 할당된다.

**(3)**

no.na <- which((!is.na(rdata$V2)&(!is.na(rdata$V3))))  
no.na

## [1] 2 4

**(4)**

rdata1 <- rdata[no.na,]  
rdata1

## V1 V2 V3 V4  
## 2 5 2 2 3  
## 4 1 4 1 2

# 4.

**(1)**

temp <- list(c(TRUE, FALSE), matrix(diag(1,2), nrow=2), seq(from=0, to=1, length=100), 1, 2, 3, 4)  
temp

## [[1]]  
## [1] TRUE FALSE  
##   
## [[2]]  
## [,1] [,2]  
## [1,] 1 0  
## [2,] 0 1  
##   
## [[3]]  
## [1] 0.00000000 0.01010101 0.02020202 0.03030303 0.04040404 0.05050505  
## [7] 0.06060606 0.07070707 0.08080808 0.09090909 0.10101010 0.11111111  
## [13] 0.12121212 0.13131313 0.14141414 0.15151515 0.16161616 0.17171717  
## [19] 0.18181818 0.19191919 0.20202020 0.21212121 0.22222222 0.23232323  
## [25] 0.24242424 0.25252525 0.26262626 0.27272727 0.28282828 0.29292929  
## [31] 0.30303030 0.31313131 0.32323232 0.33333333 0.34343434 0.35353535  
## [37] 0.36363636 0.37373737 0.38383838 0.39393939 0.40404040 0.41414141  
## [43] 0.42424242 0.43434343 0.44444444 0.45454545 0.46464646 0.47474747  
## [49] 0.48484848 0.49494949 0.50505051 0.51515152 0.52525253 0.53535354  
## [55] 0.54545455 0.55555556 0.56565657 0.57575758 0.58585859 0.59595960  
## [61] 0.60606061 0.61616162 0.62626263 0.63636364 0.64646465 0.65656566  
## [67] 0.66666667 0.67676768 0.68686869 0.69696970 0.70707071 0.71717172  
## [73] 0.72727273 0.73737374 0.74747475 0.75757576 0.76767677 0.77777778  
## [79] 0.78787879 0.79797980 0.80808081 0.81818182 0.82828283 0.83838384  
## [85] 0.84848485 0.85858586 0.86868687 0.87878788 0.88888889 0.89898990  
## [91] 0.90909091 0.91919192 0.92929293 0.93939394 0.94949495 0.95959596  
## [97] 0.96969697 0.97979798 0.98989899 1.00000000  
##   
## [[4]]  
## [1] 1  
##   
## [[5]]  
## [1] 2  
##   
## [[6]]  
## [1] 3  
##   
## [[7]]  
## [1] 4

**(2)**

temp[[2]] <- NULL

**(3)**

temp

## [[1]]  
## [1] TRUE FALSE  
##   
## [[2]]  
## [1] 0.00000000 0.01010101 0.02020202 0.03030303 0.04040404 0.05050505  
## [7] 0.06060606 0.07070707 0.08080808 0.09090909 0.10101010 0.11111111  
## [13] 0.12121212 0.13131313 0.14141414 0.15151515 0.16161616 0.17171717  
## [19] 0.18181818 0.19191919 0.20202020 0.21212121 0.22222222 0.23232323  
## [25] 0.24242424 0.25252525 0.26262626 0.27272727 0.28282828 0.29292929  
## [31] 0.30303030 0.31313131 0.32323232 0.33333333 0.34343434 0.35353535  
## [37] 0.36363636 0.37373737 0.38383838 0.39393939 0.40404040 0.41414141  
## [43] 0.42424242 0.43434343 0.44444444 0.45454545 0.46464646 0.47474747  
## [49] 0.48484848 0.49494949 0.50505051 0.51515152 0.52525253 0.53535354  
## [55] 0.54545455 0.55555556 0.56565657 0.57575758 0.58585859 0.59595960  
## [61] 0.60606061 0.61616162 0.62626263 0.63636364 0.64646465 0.65656566  
## [67] 0.66666667 0.67676768 0.68686869 0.69696970 0.70707071 0.71717172  
## [73] 0.72727273 0.73737374 0.74747475 0.75757576 0.76767677 0.77777778  
## [79] 0.78787879 0.79797980 0.80808081 0.81818182 0.82828283 0.83838384  
## [85] 0.84848485 0.85858586 0.86868687 0.87878788 0.88888889 0.89898990  
## [91] 0.90909091 0.91919192 0.92929293 0.93939394 0.94949495 0.95959596  
## [97] 0.96969697 0.97979798 0.98989899 1.00000000  
##   
## [[3]]  
## [1] 1  
##   
## [[4]]  
## [1] 2  
##   
## [[5]]  
## [1] 3  
##   
## [[6]]  
## [1] 4

temp 리스트의 두 번째 원소를 제거한 결과 3번째 위치한 결과값은 1이 되었다.

**(4)**

length(temp)

## [1] 6

# 5.

**(1)**

a1 <- -1:2  
a2 <- 1:2  
a1 + a2

## [1] 0 2 2 4

길이가 다른 벡터끼리 연산을 하면, 길이가 짧은 벡터가 원소를 재사용하여 길이가 긴 벡터에 맞춰서 연산을 한다. 따라서 a1=(-1, 0, 1, 2) + a2=(1, 2) = (-1, 0, 1, 2) + (1, 2, 1, 2) = (0, 2, 2, 4) 가 된다.

**(2)**

a1 <- -(1:2)  
a2 <- 1:2  
a1 + a2

## [1] 0 0

벡터를 생성하는 연산자인 ’:’은 말그대로 하나의 연산자로 취급하기 때문에, 사칙연산과 같이 괄호를 사용하면, 해당 연산을 먼저 수행한다. 따라서 a1 <- -(1:2)의 결과 a1에는 (-1, -2)가 저장되고, (1, 2)인 a2와 더하면 (0, 0)이 된다.

**(3)**

a1 <- matrix(0,2,2)  
a2 <- c(3,4)  
a1 + a2

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

matrix(0, 2, 2)는 2행 2열의 원소가 0으로 된 행렬을 만들어 준다. 벡터와 행렬을 연산하게 되면, 행렬을 세로로 열끼리 이어서 하나의 벡터로 인식하고 계산하게 되므로 (0, 0, 0, 0) 벡터와 (3, 4) 벡터의 계산과 비슷하다. 그럼 (1)에서처럼 뒷 벡터를 재사용하여 (3, 4, 3, 4) 벡터로 연산을 마치고, 다시 원래의 행렬 형태로 출력하는 것이다.

**(4)**

a1 <- matrix(1:4,2,2)  
a1[a1>2] = 0  
a1

## [,1] [,2]  
## [1,] 1 0  
## [2,] 2 0

이것도 (3)과 마찬가지로 행렬을 벡터처럼 인식해 연산을 행한다. 그럼 a1은 (1, 2, 3, 4)에서 2보다 큰 3과 4를 0으로 만들고, (1, 2, 0, 0)의 벡터를 다시 행렬 형태로 출력해서 위와 같은 결과를 만들어낸다.

**(5)**

a1 <- 1:5  
a1[-1] - a1[length(a1)]

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

a1[-1]은 a1의 첫번째 원소를 제외한 나머지로 (2, 3, 4, 5)가 된다. a1[length(a1)]에서 length(a1)는 a1의 길이, 즉 5이므로 a1[5]와 같은 값을 같는데, a1의 5번째 원소는 5이므로 (5)가 된다. a1[-1]에서 a1[length(a1)]을 뺀 계산은 (2, 3, 4 ,5)에서 (5)를 뺀 것과 같은데, 여기서 (5)는 4번 재사용 되어서 (2, 3, 4, 5) - (5, ,5 ,5, 5)와 같은 결과가 나타난다. 따라서 (-3, -2, -1, 0)이 된다.

## **R-Programming**

# 1.

a <- function(n){  
 a1 <- 1  
 a2 <- 3  
 if (n == 1) return(a1)  
 if (n == 2) return(a2)  
 else 0.9\*a(n-1)-0.1\*a(n-2)+1  
}  
  
a(5)

## [1] 4.186

# 2.

i <- 1  
while(a(i) <= 4){  
 i <- i + 1  
}  
i

## [1] 5

# 3.

A <- matrix(runif(100), 50, 5)  
v <- rep(0,50)  
for(i in 1:50){  
 v[i] <- sum(A[i,])  
}  
v

## [1] 3.5796628 1.2967095 2.9620067 3.7220454 2.3127194 2.8523963 1.1089143  
## [8] 2.3387575 1.2096986 1.2657585 2.4047461 1.3185525 2.1872253 0.4159381  
## [15] 3.8754567 1.9891443 2.2204504 2.4315886 3.9670478 2.9311758 0.7243764  
## [22] 0.2506115 1.4870899 0.5389775 3.6290282 1.8859382 3.0599785 3.2843164  
## [29] 4.0730530 1.2659781 0.9207133 3.5059521 1.7201738 2.6514141 2.4441014  
## [36] 4.3649420 3.3181825 3.0610886 2.9542243 3.8498242 1.7226303 2.3938965  
## [43] 2.8984300 3.0572408 4.1509150 3.5237071 0.8435431 0.9485153 1.2956202  
## [50] 3.8746670

# 4.

tmp = rep(0, 10)  
a <- 10:1  
idx = 1  
for ( j in a)  
{  
 if (j<5)  
 {  
 tmp[idx] <- a[j]  
 idx <- idx + 1  
 }  
}  
tmp

## [1] 7 8 9 10 0 0 0 0 0 0

# 5.

set.seed(123)  
x <- matrix(sample(1:50, 5000 , replace = TRUE), 1000, 5)  
sid <- sample(1:10, 1000, replace = TRUE)  
head(sid)

## [1] 3 10 8 7 3 1

# 6.

m.mat <- matrix(0, 10, 5)  
for(j in 1:10){  
 m.mat[j,] <- colMeans(x[which(sid==j),])  
}  
m.mat

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 24.24444 23.67778 25.66667 25.51111 26.01111  
## [2,] 29.46067 26.87640 25.24719 26.68539 26.59551  
## [3,] 24.25000 26.04630 25.57407 24.02778 25.97222  
## [4,] 25.51042 25.58333 23.61458 27.42708 24.68750  
## [5,] 25.08065 25.54839 23.63710 24.80645 24.18548  
## [6,] 25.01031 24.42268 25.77320 25.94845 23.41237  
## [7,] 28.39362 27.03191 24.14894 25.23404 25.43617  
## [8,] 25.14583 25.36458 25.70833 25.47917 25.28125  
## [9,] 22.91429 24.25714 29.37143 26.05714 23.80000  
## [10,] 24.47525 25.15842 25.75248 26.46535 24.57426

# 6.

idist <- matrix(0, 1000, 10)  
for(i in 1:1000){  
 for(j in 1:10){  
 idist[i,j] <- sum(x[i,]\*m.mat[j,])/sqrt(sum(x[i,]^2)\*sum(m.mat[j,]^2))  
 }  
}  
head(idist)

## [,1] [,2] [,3] [,4] [,5] [,6] [,7]  
## [1,] 0.9717143 0.9809046 0.9753201 0.9755334 0.9776579 0.9673570 0.9820011  
## [2,] 0.9216976 0.9321902 0.9156904 0.9363645 0.9275336 0.9181183 0.9301661  
## [3,] 0.9112487 0.9121534 0.9022280 0.9082746 0.9041044 0.8959785 0.9074352  
## [4,] 0.9770340 0.9880001 0.9817498 0.9824392 0.9851594 0.9763489 0.9897233  
## [5,] 0.8673173 0.8963128 0.8831076 0.8774824 0.8895853 0.8803160 0.9028624  
## [6,] 0.8549035 0.8317140 0.8514123 0.8559508 0.8493635 0.8645070 0.8310039  
## [,8] [,9] [,10]  
## [1,] 0.9732687 0.9515631 0.9692688  
## [2,] 0.9205497 0.8934125 0.9209433  
## [3,] 0.9032901 0.8787610 0.8994462  
## [4,] 0.9804622 0.9613148 0.9770210  
## [5,] 0.8811101 0.8619376 0.8745354  
## [6,] 0.8555868 0.8857079 0.8662289

# 7.

ivec <- rep(0, 1000)  
for(i in 1:1000){  
 ivec[i] <- which.min(idist[i,])  
}  
head(ivec)

## [1] 9 9 9 9 9 7

# 8.

set.seed(1)  
a = list()  
for (i in 1:1000){  
 x = rpois(1,4)+1  
 x = min(x,10)   
 a[[i]] = sample(1:10, x)  
}

**(1)**

freq\_game <- rep(0,1000)  
for(i in 1:1000){  
 freq\_game[i] <- length(a[[i]])  
}  
table(freq\_game)

## freq\_game  
## 1 2 3 4 5 6 7 8 9 10   
## 15 84 146 195 207 130 112 62 28 21

간혹 경기에 출전한 사람이 1명뿐인 경기들이 있다. 이 경우는 경기가 이루어지지 않으므로 점수 부여를 하지 않겠다.

**(2)**

score <- rep(0,10)  
a1 <- a[which(freq\_game == 2 | freq\_game == 3)]  
for(i in 1:length(which(freq\_game == 2 | freq\_game == 3))){  
 p <- a1[[i]][1]  
 score[p] <- score[p] + 1  
}  
  
a2 <- a[which(freq\_game > 3 & freq\_game <= 6)]  
for(i in 1:length(which(freq\_game > 3 & freq\_game <= 6))){  
 p <- a2[[i]][1]  
 q <- a2[[i]][2]  
 score[p] <- score[p] + 2  
 score[q] <- score[q] + 1  
}  
  
a3 <- a[which(freq\_game >= 7)]  
for(i in 1:length(which(freq\_game >= 7))){  
 p <- a3[[i]][1]  
 q <- a3[[i]][2]  
 r <- a3[[i]][3]  
 score[p] <- score[p] + 3  
 score[q] <- score[q] + 2  
 score[r] <- score[r] + 1  
}  
score

## [1] 316 299 292 321 339 276 292 334 349 346

which.max(score)

## [1] 9

경기 점수를 합산한 결과 9번 선수가 가장 높은 점수를 획득했다.

# 9.

set.seed(1)  
m1 = 10  
m2 = 5  
num = 0

**(1)**

for(i in 1:4){  
 c <- rbinom(1,1,1/2)  
 if(c == 0){  
 m1 <- m1 - 1  
 m2 <- m2 + 1  
 }else{  
 m1 <- m1 + 1  
 m2 <- m2 - 1  
 }  
 num <- num + 1  
}  
m1

## [1] 10

0이 2번, 1이 2번으로 여전히 10원을 갖고 있다.

**(2)**

set.seed(1)  
while(TRUE){  
 c <- rbinom(1,1,1/2)  
 if(c == 0){  
 m1 <- m1 - 1  
 m2 <- m2 + 1  
 }else{  
 m1 <- m1 + 1  
 m2 <- m2 - 1  
 }  
 num <- num + 1  
 if(m1 == 0 | m2 == 0){  
 if(m1 == 0) print("B양 승")  
 else print("A군 승")  
 break  
 }  
}

## [1] "A군 승"

num

## [1] 195

A군이 195번만에 게임에서 이긴다.

**(3)**

win <- rep(0,2)  
for(k in 1:200){  
 set.seed(k)  
 m1 = 10  
 m2 = 5  
 num = 0  
 while(TRUE){  
 c <- rbinom(1,1,1/2)  
 if(c == 0){  
 m1 <- m1 - 1  
 m2 <- m2 + 1  
 }else{  
 m1 <- m1 + 1  
 m2 <- m2 - 1  
 }  
 num <- num + 1  
 if(m1 == 0 | m2 == 0){  
 if(m1 == 0) win[2] <- win[2] + 1  
 else win[1] <- win[1] + 1  
 break  
 }  
 }  
}  
win

## [1] 129 71

A군은 200번의 실험 중 129번의 시험에서 이긴다.

# 10.

Awin <- rep(0,4)  
for (i in 2:5){  
 for(k in 1:200){  
 set.seed(k)  
 m1 = 10  
 m2 = 5 \* i  
 num = 0  
 while(TRUE){  
 c <- rbinom(1,1,1/2)  
 if(c == 0){  
 m1 <- m1 - 1  
 m2 <- m2 + 1  
 }else{  
 m1 <- m1 + 1  
 m2 <- m2 - 1  
 }  
 num <- num + 1  
 if(m1 == 0 | m2 == 0){  
 if(m1 == 0) Awin[i-1] <- Awin[i-1]  
 else Awin[i-1] <- Awin[i-1] + 1  
 break  
 }  
 }  
 }  
 print(Awin[i-1] / 200)  
}

## [1] 0.495  
## [1] 0.39  
## [1] 0.335  
## [1] 0.3

A군이 승리하는 비율은 B양이 10원일 때는 49.5%, 15원 일 때는 39%, 20원일 때는 33.5%, 25원일 때는 30%이다.