Matrices

> Created by Woo Hyun (Ray) Hwang

Matrices

```
y <- matrix(c(1, 2, 3, 4), nrow=2, ncol=2) #byrow=F default
#to assign each row and column
y <- matrix(nrow=2, ncol=2)
y[1, 1] <- 1
y[2, 1] <- 2
y[1, 2] <- 3
y[2, 2] <- 4
```

```
data.1 <- c(
  1.1, 1.3, 2.2,
  2.1, 2.1, 2.3
)
mat <- matrix(data.1, 2, 3, byrow=T)
mat
mat[1, ] #first row of mat
mat[, 3] #third col of mat
mat[4] #the fourth element, starting from [1,1], going down along column
mat[1, , drop=FALSE] #returns a matrix without dropping
mat[, 3, drop=F] #returns a matrix without dropping
```

Creating matrices

Vector of length np to a n*p matrix

```
n <- 100; p <- 3
x <- rnorm(n*p)
A <- matrix(x, n, p)
A[1:6, ] #shows the first 6 rows of matrix A
```

Indexing and filtering

Getting a submatrix by indexing and filtering

```
A[1:5, 1:2]
A[c(1, 3, 5), 1] #this is thought to be a vector in R
A[A[,1]>0, ] #only data with positive values in col 1 is left
A[A[,1]>0 & A[,2]>0, ] #data with positive values in col 1 and 2
```

Arithmetic operations

Multiplication of matrices: ncol A = nrow B -> `A %*% B`

Transpose of matrix : `t(A)`

Inverse of matrix : `solve(C)`

Determinant : `det(A)`

Diagonal property : `diag(A)`

```
A <- matrix(6:1, 3, 2)
A
t(A)
C <- matrix(c(16,4,1,4,4,1,1,1,1), 3, 3)
C
solve(C)
solve(C) %*% C
```

```
#Hilbert matrix: 3*3 matrix
H3 <- matrix(c(1,1/2,1/3,1/2,1/3,1/4,1/3,1/4,1/5), nrow=3)
X <- matrix(c(1,2,3,1,4,9), ncol=2)
det(H3)
diag(H3) #diagonal propoerties
trace <- function(data) {
  sum(diag(data))
}
trace(H3)
diag(diag(H3)) #creates square diagonal property matrix
t(X)
```

This can be applied to linear regression (check the PPT)

`apply()` to a matrix

`apply(matrix, 1, f)` : apply f to rows

`apply(matrix, 2, f)` : apply f to columns

```
M <- matrix(sample(1:6), 3, 2)
apply(M, 1, sum) #applies sum to the row
apply(M, 2, sum) #applies sum to the columns
n <- nrow(M); p <- ncol(M)
u <- vector(length=n)
for(i in 1:n) u[i] <- sum(M[i, ]) #apply(M, 1, sum)
u
```

Application of `apply()` : Scaling- changing the values so the mean and sd of each col is 0 and 1 respectively

```
n <- 100; set.seed(1)
X <- matrix(rnorm(n*4), n, 4)
m <- apply(X, 2, mean) # 1*4 vector
s <- apply(X, 2, sd) # 1*4 vector
```

```
X.1 <- t(apply(X, 1, "-", m)) #transpose the apply so it becomes a n*4
#if we do not transpose, it becomes a 4*n matrix
X.2 <- t(apply(X.1, 1, "/", s))
head(round(X.2, 2), 3)
#Using the scale() function
head(round(scale(X), 2), 3)
```

Example : the distribution of the 2nd smallest value among 10 unif. dist.

```
#The Monte Carlo method
n <- 1000 #Monte Carlo repetition
p <- 10
M <- matrix(runif(n*p), n, p) #1000 rows, each row is from 10 unif. dist
M.1 <- t(apply(M, 1, sort)) #sort from smallest by row, transpose
head(M.1[, 1:5], 5)
x2 <- M.1[, 2]
hist(x2, xlim=c(0, 1))
```

Avoiding decreasing dimensions

```
z <- matrix(1:8, nrow=4)
r <- z[2, ]
r
attributes(z)
attributes(r) #NULL, as r is a vector
#Use the drop=F option
r <- z[2, , drop=F]
dim(r) #now it is counted as a 1*2 matrix
z[3, 2]
"["(z, 3, 2)
#Use the as.matrix() function
u <- c(1, 2, 3)
v <- as.matrix(u)
dim(v) #3*1 matrix
```

rownames() and colnames()

Adds labels to the rows and columns.

```
n <- 10
X <- matrix(round(rnorm(n*4, 50, 10)), n, 4)
rownames(X) <- paste("S", 1:n, sep=".")
colnames(X) <- c("math", "engl", "science", "arts")
head(X)
dimnames(X) #shows the names of rows and columns
```

Class indicator

Example: 20 ppl, 3 answers (1, 2, 3), want a frequency table with gender, answer and gender*answer

```
n <- 20
gender <- sample(1:2, n, replace=T) #gender random generation
gender
answer <- sample(1:3, n, replace=T) #answer random generation
answer
#seperate gender group: if gender=1 : col.1=1 / gender=2 : col.2=1
gender.matrix <- matrix(0, n, 2)
for(i in 1:n) gender.matrix[i, gender[i]] <- 1 #[1, 2], [2, 1] so it goes
gender.matrix[1:n + n*(gender-1)] <- 1 #same result without using loop
head(gender.matrix, 5)
apply(gender.matrix, 2, sum)
#Doing the same to the answers
answer.matrix <- matrix(0, n, 3)
answer.matrix[1:n + n*(answer-1)] <- 1
apply(answer.matrix, 2, sum)
#Getting gender*answer
cross.tab <- t(gender.matrix) %*% answer.matrix
cross.tab
#Using the table() function
table(gender, answer)
```