### 

### **vector recycling**

Recycling Rule:

while carrying vector operations between two vectors; if the length of them is not equal,

then the shorter vector will be recycled in order to complete the process.

# normal case

a <- c(1, 2, 3, 4)

b <- c(10, 20, 30, 40)

a + b

# output: c(11, 22, 33, 44)

a - b

# output: c(-9, -18, -27, -36)

# vector recycling

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

d <- c(10, 20, 30, 40)

c + d

# output: c(11, 22, 33, 44, 15, 26, 37, 48)

c - d

# output: c(-9, -18, -27, -36, -5, -14, -23, -32)

### **outer multiplication**

Rules:

1. the operate that being used is %o%

2. the number of column in first matrix must equal number of row in second matrix

m <- matrix(1:8, nrow=2)

n <- matrix(8:15, nrow=4)

m %\*% n

output:

[162 226]

[200 280]

How it multiply

[1\*8+3\*9+5\*10+7\*11 = 162 1\*12+3\*13+5\*14+7\*15=226]

[2\*8+4\*9+6\*10+8\*11 = 200 2\*12+4\*13+6\*14+8\*15=280]

### **inner multiplication**

**1. matrix multiplication with vector**

If a matrix is multiplied with a vector

then vector will be promoted to either row or column matrix to make

two arguments conformable

m <- matrix(1:8, nrow=2)

v <- 1:2

v\*m

output:

[1 3 5 7]

[4 8 12 16]

How it is multiply

[1\*1=1 1\*3=3 1\*5=5 1\*7=7]

[2\*2=4 2\*4=8 2\*6=12 2\*8=16]

2. matrix multiplication with matrix

the element-wise multiplication of two matrices take place

it will be multiplied if the matrix have the same dimension

j <- matrix(1:8, nrow=2)

k <- matrix(8:15, nrow=2)

j\*k

output:

[8 30 60 98]

[18 44 78 120]

How it is multiply

1\*8=8 3\*10=30 5\*12=60 7\*14=98

2\*9=18 4\*11=44 6\*13=78 8\*15=120

### **functions**

**1. sample() -- take a random sample of elements**

Ex: dataset contains 100 records; sample() generate a random sample dataset

x <- 1:12

x # output: [1 2 3 4 5 6 7 8 9 10 11 12]

sample(x) # output: [3 7 8 4 10 11 1 6 2 9 12 5]

2. seq() -- generate a regular sequence of output base on from, to arguments; number of output specify by *length.out*

seq(0, 1, length.out = 11) # output: [0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0]

seq(0, 1, length.out = 5) # output: [0.00 0.25 0.50 0.75 1.00]

3. rep() -- generate a repeatedly sequence of output as mentioned

rep(1, 2) # output: [1 1]

rep(1:4, 2) # output: [1 2 3 4 1 2 3 4]

rep(1:2, each = 3) # output: [1 1 1 2 2 2]

4. round() -- round a provided double/ float in to n-digits after decimal base on mentioned

round(1.25567, digits = 2) # output: 1.26

round(1.25567, digits = 1) # output: 1.3

round(1.25567, digits = 0) # output: 1

**5. factorial() -- return factorial value of the input**

factorial(3) # output: 6

factorial(1:3) # output: [1 2 6]

**6. is() -- return true/ false if an object belong to a class**

is("string", "SClassExtension")

# output: FALSE

a <- FALSE

is(a, "logical") # output: TRUE

**7. mean() -- return average value of an input**

mean(1:5) # output: 3

**8 set.seed() -- set the seed for generate random number**

set.seed(3)

rnorm(3) #output: [-0.9619334 -0.2925257 0.2587882]

### **subset**

Subsetting in R is a useful indexing feature for accessing object elements.

It can be used to select and filter variables and observations.

subset() -- return subsets of vectors, matrices or data frames which meet conditions.

a <- c(1, 2, 3, 4, 5)

subset(a, a > 3) # output: [4 5]

### **bmi program**

weight <- readline(prompt = "Enter your weight (kg): ")

height <- readline(prompt = "Enter your height (m): ")

weight <- as.double(weight)

height <- as.double(height)

bmi <- round(weight / (height \* height), 0)

print(paste("your bmi: ", bmi))

### **bmi function program**

calcBMI <- function(weight, height) {

return(round(weight / (height \* height), 0))

}

weight <- readline(prompt = "Enter your weight (kg): ")

height <- readline(prompt = "Enter your height (m): ")

weight <- as.double(weight)

height <- as.double(height)

bmi = calcBMI(weight = weight, height = height)

print(paste("your bmi: ", bmi))