```
##################################################################
### Computational Policy and Project Analysis - Lecture 04 ########
### Subject: Functions and Operations                      ########
### Developed by. KKIM                                     ########
##################################################################

################### Functions & Key Concepts ##########################

### User-defined function
cube <- function(n) {
  return(n*n*n)
}
cube(10)

cube(1:5)

#
is.even.number <- function(n) {
  n %% 2 == 0
}
is.even.number(10)
is.even.number(5)
is.even.number(c(1,2,5,6,7,9,15))
numbers_vector <- c(1,3,4,2,6,8,7,5)

#
myFunction <- function(n) {
  # n+2
  n+1
  a = n+3
  # return(a)
  n+4
}
myFunction(2)
b <- myFunction(2)

#
diff.max.min <- function(...) {
  a <- c(...)
  largest <- max(a)
  smallest <- min(a)
  largest - smallest
}
diff.max.min(6,5,6,23,4,25)
diff.max.min(-55, 100, 23, -7)

################### Conditional Statements ##########################

### Conditional Statements
## If
medium <- "LinkedIn"
num_views <- 14
if (medium == "LinkedIn") {
  print("Showing LinkedIn information")
}
if (num_views > 15) {
  print("You're popular!")
}

## If-else
# Variables related to your last day of recordings
medium <- "LinkedIn"
num_views <- 14
if (medium == "LinkedIn") {
  print("Showing LinkedIn information")
} else {
  print("Unknown medium")
}
if (num_views > 15) {
  print("You're popular!")
} else {
  print("Try to be more visible!")
}

################### Iterative Statements ###########################

## Iterative Statement
cities <- c("New York", "Paris","London", "Tokyo",
            "Rio de Janeiro", "Cape Town")
for(city in cities){
  print(city)
}

### Conditional & Iterative Statement
data(mtcars)
i<-1
for (i in 1:nrow(mtcars)){
  if (mtcars$am[i]==1){
    mtcars$transmission[i] <- "manual"
  } else{
    mtcars$transmission[i] <- "automatic"
  }
}
head(mtcars)

### Quiz
# Write down a R code that creates a new variable called “engine” that returns
# “v-shaped” if the engine shape is v-shaped, and “straight”
# if the engine shape is straight.
for (i in 1:nrow(mtcars)){
  if (mtcars$vs[i]==1){
    mtcars$engine[i] <- "V-shaped"
  } else{
    mtcars$engine[i] <- "Straight"
  }
  print(i)
}
head(mtcars)

################### Vectorized Operations ###########################

### Vectorized Operation
my.vector <- c(1, 3, 5, 8, 13)
my.vector * 2
my.vector >= 5
my.vector < 10
my.vector >= 5 & my.vector < 10

sum(my.vector)
mean(my.vector)
median(my.vector)
min(my.vector)
max(my.vector)
summary(my.vector)

ifelse(my.vector %% 2 == 0, 'even', 'odd')

numbers_vector <- c(1,3,4,2,6,8,7,5)
numbers_even_odd <-
  ifelse(numbers_vector %% 2 == 0,
         'even', 'odd')
numbers_even_odd
# table(numbers_even_odd)

### Vectorized Operations
# classic operation vs. vectorized operation
# version.1
score.set <-
```

```
  data.frame(total.score=c(91, 98, 92, 88, 81, 80, 48))

for (i in 1:nrow(score.set)){
  if(score.set$total.score[i]>60){
    score.set$grade[i] <- "Pass"
  } else{
    score.set$grade[i] <- "Fail"
  }
  # print(i)
}
score.set$grade.1 <-
  ifelse(score.set$total.score > 60,
         "Pass", "Fail")
score.set

# version.2
score.set <- data.frame(total.score=c(91, 98, 92, 88, 81, 80, 48))
for (i in 1:nrow(score.set)) if(score.set$total.score[i]>60) score.set$grade[i] <- "Pass" else
score.set$grade[i] <- "Fail"
score.set$grade.1 <- ifelse(score.set$total.score > 60, "Pass", "Fail")
score.set

score.set <- data.frame(total.score=c(91, 98, 92, 88, 81, 80, 48))
for (i in 1:nrow(score.set)){
  score.set$mama[i] <- "Yes"
  score.set$haha[i] <- "No"
}

score.set <- data.frame(total.score=c(91, 98, 92, 88, 81, 80, 48))
for (i in 1:nrow(score.set)) score.set$mama[i] <- "Yes" score.set$haha[i] <- "No"
for (i in 1:nrow(score.set)) score.set$mama[i] <- "Yes"; score.set$haha[i] <- "No"

### Vectorized Operations
data(mtcars)
head(mtcars)

avg_mpg <- mean(mtcars$mpg)
new_var <- ifelse(mtcars$mpg >= avg_mpg,
                  'good', 'bad')
## adding new variable to mtcars
mtcars$fuel_efficiency <- new_var
head(mtcars)

##
data(mtcars)
head(mtcars)

avg_mpg <- mean(mtcars$mpg)
new_var <- ifelse(mtcars$mpg >= avg_mpg, 'good', 'bad')
mtcars$fuel_efficiency <- new_var
head(mtcars)

################### Apply Functions ###########################

### apply
myMat <- matrix(1:12, ncol = 4)
myMat
apply(myMat, 1, mean)
apply(myMat, 2, mean)

set.seed(2018)
myMat <- matrix(runif(12), ncol = 4)
myMat
apply(myMat, 1, mean)
apply(myMat, 2, mean)

### apply
data(iris)
head(iris)
```

```
apply(iris, 2, mean)
apply(iris[, 1:4], 2, mean)
apply(iris[, 1:4], 1, mean)
colMeans(iris[, 1:4])

### lapply
myList <- list(num = 3.14,
               chr = "char",
               logi = TRUE)
length(myList)
myList
lapply(myList, typeof)

### lapply
myList2 <- list(vec = 1:5,
                mat = matrix(runif(12),
                             ncol = 4),
                df = iris)
myList2
length(1:5)
length(myList2)
length(iris)
result <- lapply(myList2, length)
result

### lapply
lapply(c(1,4,9,16), sqrt)

names(mtcars)
lapply(mtcars, max)
unlist(lapply(mtcars, max))

### Quiz
data(mtcars)
mySample <- list(vec = 1:5,
                 mat = matrix(1:12,
                              ncol = 4),
                 df = head(mtcars))
# 1) Using apply() function, write down R code that measures mySample’ s
# third element’ s column average.
apply(mySample[[3]],2,mean)

# 2) Create a list that returns the number of elements of mySample
lapply(mySample, length)
unlist(lapply(mySample, length))

lapply(mySample, nrow)
lapply(mySample, sum)
lapply(mySample, mean)
# mySample[[3]]$new<-"c"
# lapply(mySample, mean)

# 3) In mySample’ s third list element, create a new variable called  ‘avg_hp’ ,
# which returns  ‘High’  if hp is greater than and equal to the average hp and
#  ‘Low’  if otherwise (Hint: ifelse()).
mySample[[3]]$avg_hp <-
  ifelse(mySample[[3]]$hp >= mean(mySample[[3]]$hp),
         "High","Low")
mySample

### sapply
data(iris)
head(iris)

sapply(iris[, 1:4], mean)
# lapply(iris[, 1:4], mean)

sapply(iris, is.numeric)
# lapply(iris, is.numeric)
```

```
sapply(c(1,3,5,7,9),
       function(x) {x**2})

### sapply
data(iris)
head(iris)
x <- sapply(iris[,1:4],
            function(x) { x> 3})
# x <- sapply(iris[,1:4], function(x) x> 3)
head(x)
colSums(x)

### sapply
pools <- read.csv(file="R file/R file_LEC04/pools.csv")
head(pools)
sapply(pools, typeof)

### tapply
head(iris)
table(iris$Species)

tapply(iris$Sepal.Length,
       iris$Species,
       mean)

### tapply
head(mtcars)
# greater cylnder, poor fuel efficiency
unique(mtcars$cyl)
tapply(mtcars$mpg,
       mtcars$cyl,
       mean)
# high fuel efficiency, lighter weight
tapply(mtcars$wt,
       mtcars$mpg>20,
       mean)

### tapply
x <- tapply(mtcars$mpg,
            mtcars$cyl,
            function(x){x>20})
x
sapply(x, sum)
lapply(x, sum)

################### Aggregate ###########################
### aggregate

data(mtcars)
head(mtcars)

aggregate(mpg ~ cyl, data = mtcars, FUN = mean)

tapply(mtcars$mpg, mtcars$cyl, mean)

load(file="R file/R file_LEC04/iris_new.RData")

# aggregate multiple groups
aggregate(Sepal.Length ~ Species + avg_sl,
          data = iris_new,
          FUN = mean)

# aggregate multiple values
aggregate(cbind(Sepal.Length,Sepal.Width) ~ Species + avg_sl,
          data = iris_new,
          FUN = mean)

################### Pipe Operator (dplyr) ###########################
```

```
### dplyr
data(mtcars)

mtcars %>% head

library(dplyr)

mtcars %>% head

head(women)
women %>% head

women %<>% head

library(magrittr)
women %<>% head
```