# Wk 4. Control / Loop statements, Advanced Function

CH 7

# Attendance Checking #1


- Please turn on your Webcam!
- To take a screenshot
- Only used for Attendance checking & Verification
- Type "Present" on the chatting room

(I will ask you to do it again at the end)

## Terminology 3


# Parametric vs. Non-Parametric

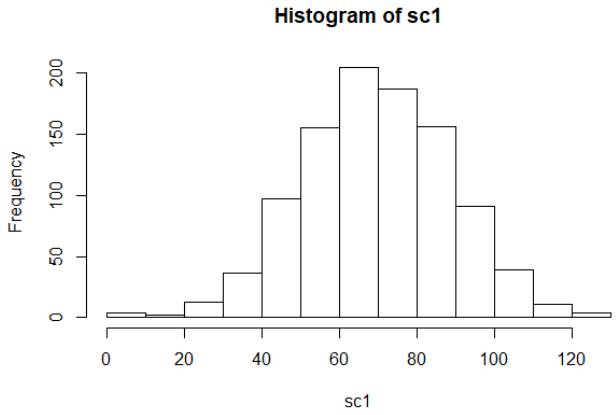
**Normal Distribution vs. Uniform Distribution**

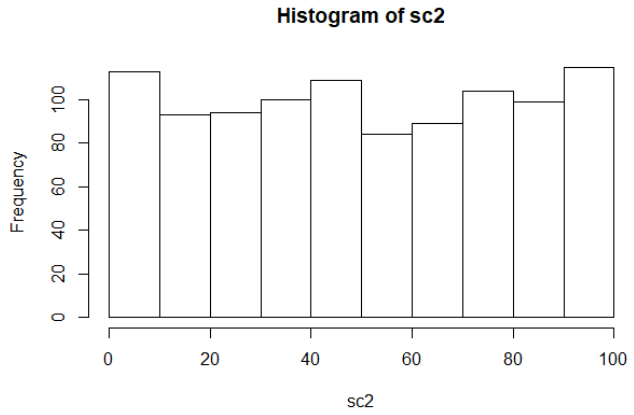
AKA: _____

**To compare 2 groups, you should first try a parametric method. If it is not applicable, you can go to a non-parametric method as an alternative.**

# Normal Distribution vs. Uniform Distribution

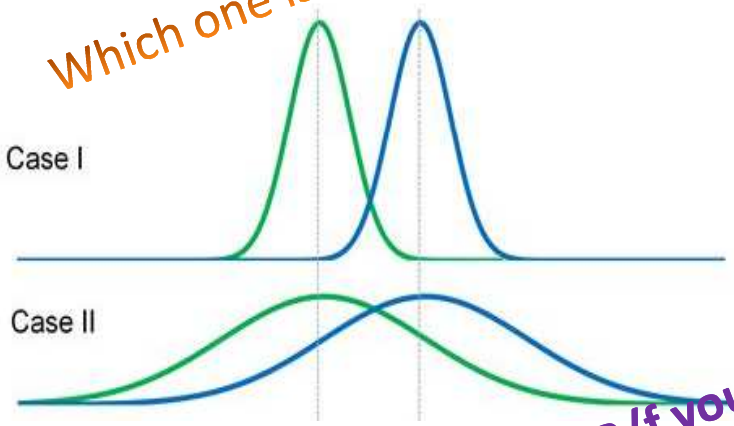

```
sc1=floor(rnorm(1000,mean=70,sd=20))
sc2=floor(runif(1000,0,100))
hist(sc1)
hist(sc2)
```



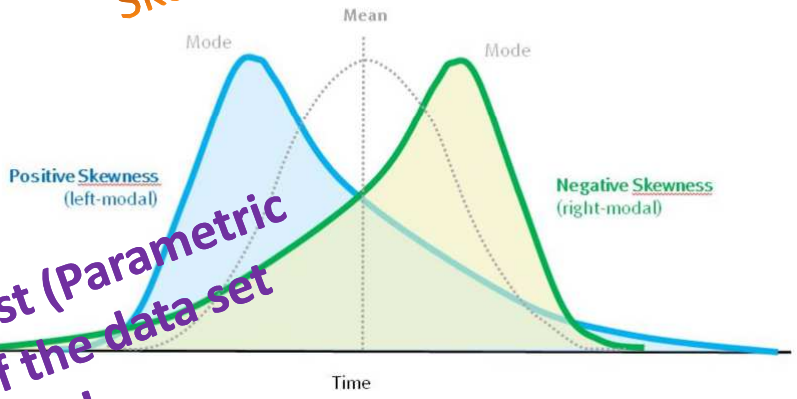
# Comparing Two groups

Which one is better result?




* ashokcharan.com

Skeweness?



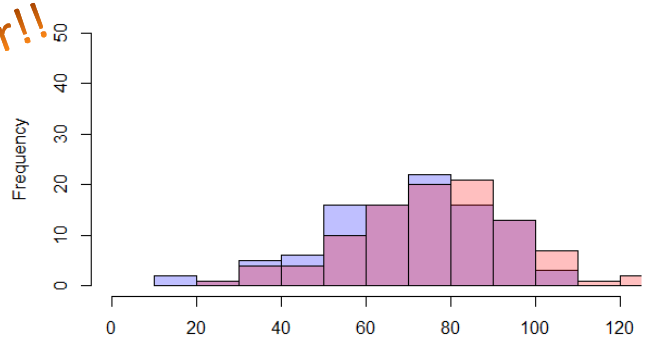

* deepai.org

**B/f you run t-Test (Parametric method), test if the data set follows Normality!**
**(And equal variance)**

# t-Test (Parametric Method)


```
n=100
grp1 <-floor(rnorm(n,mean=70,sd=20))
grp2 <-floor(rnorm(n,mean=80,sd=20))
p1 <- hist(grp1)
p2 <- hist(grp2)
plot(p1, col=rgb(0,0,1,1/4),xlim=c(0,120),ylim=c(0,50))
plot(p2, col=rgb(1,0,0,1/4),xlim=c(0,120),ylim=c(0,50),add=T)

#Let's run t-Test
t.test(grp1,grp2)
```

Hey, "n" and "sd" matter!!




Q. Is it good or bad?

```
        Welch Two Sample t-test

data:  grp1 and grp2
t = -2.4115, df = 197, p-value = 0.0168
alternative hypothesis: true difference in means is
not equal to 0
95 percent confidence interval:
 -12.070072  -1.209928
sample estimates:
mean of x mean of y
    70.55     77.19
```

Confidence Interval:
95%, 98%, 99%

Dependent vs. Independent ?

One-tail vs. Two-tail ?

# t-Test in Excel


```
# Let's run t-Test in Excel. First save the data into a csv file.
tData=cbind(grp1,grp2)
dimnames(tData)[[2]] <- c('grp1','grp2')
class(tData)
View(tData)
write.csv(tData,"tData.csv")
```

Q. BTW, if you have 3rd group?

# 01
# Control Statement

# IF-ELSE STATEMENT



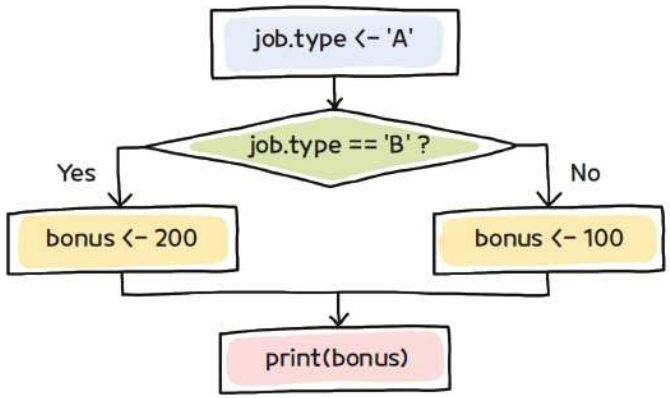



그림 7-1 조건문의 순서도

```
job.type=readline(prompt="Enter Type:")

if(job.type =='B') {
   bonus <-200
} else {
   bonus <- 100
}
print(bonus)
```

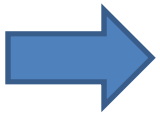
# IFELSE STATEMENT



```
ifelse(Condition, value if T, value if F)
```

man, this is like ":" operator!

```
# Using if-else #
a <- 10
b <- 20

if (a>b) {
    c <- a
} else {
    c <- b
}
print(c)
```



```
# Using ifelse #
a <- 10
b <- 20

c <- ifelse(a>b, a, b)
print(c)
```

# IF-ELSE STATEMENT: EG, RSP GAME



```
R <- 1
S <- 2
P <- 3
com <- floor(runif(1,1,4))
If(com==R) print("Rock")
else if(com==S) print("Scissor")
else print("Paper")
```

# 02
# Looping

# LOOP STATEMENT: FOR



```
for(Loop variable in range) {
    statements
}
```

*#Example 1*

```
result=0
for (i in 1:10) {
  result <- result+i
}
print(result)
```

*#Example 2*

```
scores <- c(85,74,93,100,82)
result.sum=0
for (score in scores) {
  result.sum <- result.sum + score
}
cat("Average:",result.sum/length(scores))
```

This one is cute ☺

# LOOP STATEMENT: WHILE



```
while (Loop Condition) {
    Statements
}
```

```
scores <- c(85,74,93,100,82)
result.sum=0
i=1
while(i<=length(scores)) {
  result.sum <- result.sum + score[i]
}
cat("Average:",result.sum/length(scores))
```

BTW dude, you know "break" and "next" right?

```
Let's play my RSP Game!
```

# LOOP STATEMENT: APPLY()



```
apply(Data Set, Row/Col Direction, Func)
```

- **Data Set:** Target Matrix or Data frame.
- **Row/Col :** 1 for row, 2 for column (Direction).
- **Func**: User defined function

# LOOP STATEMENT: APPLY() EXAMPLE



```
> 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 ...
```

```
apply(iris[,1:4], 1, mean)        # Row Direction
apply(iris[,1:4], 2, mean)        # Column Diretion
```

# LOOP STATEMENT: APPLY()



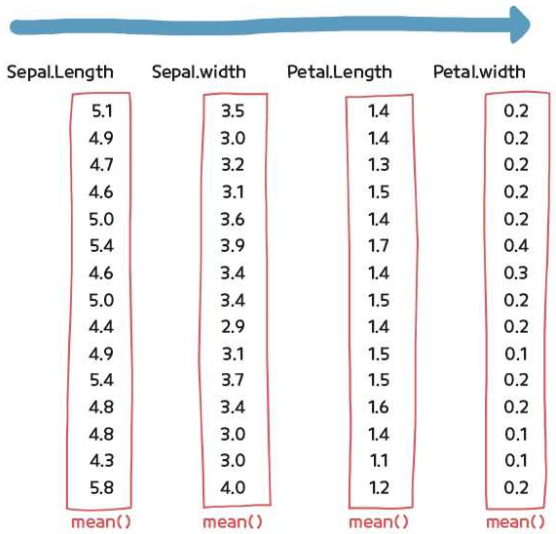



그림 7-5 apply(iris[,1:4], 2, mean)

```
> apply(iris[,1:4], 2, mean)     # Column Direction
Sepal.Length Sepal.Width Petal.Length Petal.Width
    5.843333    3.057333     3.758000    1.199333
```

# 03
# Advanced Function
## (User Defined Function)

# USER DEFINED FUNCTION



```
function_name<- function(argument(s)) {
    statements (Function Body)
    return(result)
}
```

I P O

```
rsp_name <- function(n){
 if(n==R) return("Rock ")
 else if(n==S) return("Scissor")
 else return("Paper")
}
```

```
mydiv <- function (x, y=2) {
  result <-x/y
  return(result)
}
mydiv(100,5)
Mydiv(x=100,y=5)
mydiv(100)
```

Default value

# USER DEFINED FUNCTION: LAB 2



Generate 10 random numbers (0~100) and get the Min value

# USER DEFINED FUNCTION: RETURN MORE THAN 1 RESULT



You can return more than 1 value using "**LIST**"

```
myfunc <- function(x,y) {
    val.sum <- x+y
    val.mul <- x*y
    return(list(sum=val.sum, mul=val.mul))
}

result <- myfunc(3,4)
s <- result$sum
m <- result$mul
cat('3+4 =', s, '\n')
cat('3*4 =', m, '\n')
```

List is like "Object"

person <- list(name="Sam", age=20)

# USER DEFINED FUNCTION: HOW TO SAVE AND LOAD



```
setwd('c:/Rworks')                  # myfunc.R path
source('myfunc.R')                  # load the function definition

# Use function
a <- mydiv(20,4)
b <- mydiv(30,4)
a+b
mydiv(mydiv(20,2),5)
```

Why don't you try it out?

# USER DEFINED FUNCTION: POSITION OF ELEMENT



```
score <- c(76, 84, 67, 50, 95, 60, 82, 71, 88, 84)
which(score==67)          # index of the one with 67
which(score>=85)          # indices whose score >= 85
max(score)                # max score
which.max(score)          # max score index
min(score)                # min score
which.min(score)          # min score index
```

Let's try this with runif()

## Today's Proverb

# 3. "Every cloud has a silver lining"

Echasl

M: Iystechasl, ymtesoushapsti.

Iytaslyatasptcooasous.

https://wise4edu.com/init/#s125

## Today's Proverb So Far

1. Gmshtcdat aIwstffsta.
2. Arsgnm
3. Echasl

# Local Variable vs. Global Variable

This is about SCOPE!

```
func1 <- function(){
  n <- 20  # local variable
  print(n)
}

n=10
func1()
print(n)
```

```
func2 <- function(){
  n <<- 20  # global variable
  print(n)
}

n=10
func1()
print(n)
```

# Q. Value vs. Reference?

You can pass the value of a variable to a function as an argument, which will NOT modify the variable

```
func1 <- function(a) {
   a=a+1
}
a=1
func1(a)
print(a)
```

But I want to pass it by reference, so I can update the value.

Q. Is there any way of doing that like using pointer in C?

Q. Or What if we make an object (i.e., vector, list, matrix, dataframe) and pass it into a function?

**Basically in R, passed data are "immutable"!**

# ADVANCED FUNCTION: CALL BY VALUE VS. CALL BY REFERENCE



```
void function swap(int *a, int *b) {
  int tmp=a;
  *a=*b;
  *b=tmp;
}

int a=10;
int b=20;
swap(a,b)
```

In C, you did it like this

```
val=c(10,20)

rswap <- function(vobj){
  tmp <- vobj[1]
  vobj[1] <- vobj[2]
  vobj[2] <- tmp
  return(vobj) # don't forget it!
}

rswap(val)->val
val
```

BTW you can do it using a package called "dplyr".

# ADVANCED FUNCTION: ITERATION VS. RECURSION

## You can replace a loop w/ a recursion!

```
nsum <-function(n) {#iteration
  result=0
  for(i in 1:n) {
    result=result+i
  }
  return(result)
}
```

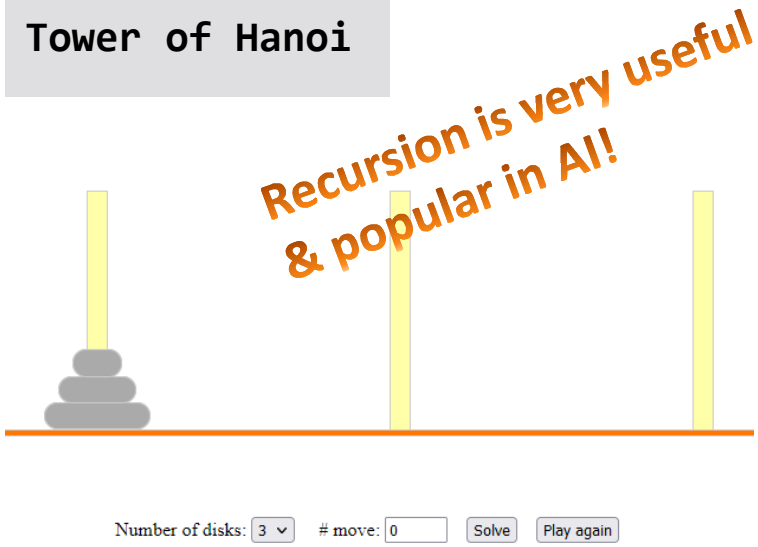
```
rnsum <-function(n) { # recursion
(induction). Using a stack

  if(n<=1) return(n)
  else return(n + rnsum(n-1))

}
```

Recursion is using Stack and can do DFS!

In this case, recursion is not a good solution. Why?

# Recursive Algorithm is Problem Solving by Induction

**Tower of Hanoi**

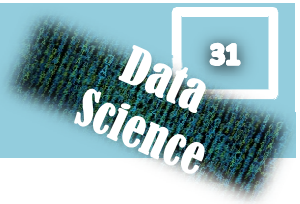



https://wise4edu.com/jsgame/hanoi/hanoi.html

```
hanoi <- function(n,src,dest,tmp) {
  if(n==1) {
    cat("move disk from",src,"to",dest,"\n")
  } else {
    hanoi(n-1,src,tmp,dest)
    cat("move disk from",src,"to",dest,"\n")
    hanoi(n-1,tmp,dest,src)
  }
}

hanoi(2,1,3,2)
hanoi(3,1,3,2)

Try hanoi(10,1,3,2)!
#How many moves do you need to make?
```

# HW #2 due by 9/30






1. Exercise 2 (Ch7)
2-a. Make “fibonacci(n)” (Are you gonna google it? ☹)
2-b. List 3 Applications of Fibonacci Sequence.
3. User Defined Function
    a. Generate 9 random numbers (200~500)
    b. Add 10000 as the $10^{th}$ number
    c. Get mean and median using **my_mean**() and **my_median**()
    d. Which one is the better measure for the central tendancy? What do you call “10000”?

    **(Make sure my functions do the right job)**
4. **[t-Test] Find an example of your interest comparing two groups of normal distribution. Run the t-Test by both R and Excel. Include the source code and the result(graphs), and REFERENCE if there is any.**
    *** Don’t copy from a source where there is everything.**

## Next Week Topics


- Manipulations of Vector & List
- Manipulations of Matrix & Data Frame (Part 1)

# To-dos for Next Class (Flipped Learning)


- **Textbook**
  - Read ch4 & 5

# Any Question?

# Attendance Checking #2

- Type “Present” on the chatting room

# Thank you for your attention!