**### 조건문**

if (cond) {

# cond가 참일 때 실행할 문장

} else {

# cond가 거짓일 때 실행할 문장

}

if( 조건1 ) {

# 표현식 1

} else if (조건2) {

# 표현식 2

} else {

# 표현식 3

team\_A <- 2

team\_B <- 2

if (team\_A > team\_B){

print ("Team A won")

} else if (team\_A < team\_B){

print ("Team B won")

} else {

"Team A & B tied"

}

**## 하나의 논리값에 대한 판단**

x1 <- c(4)

if (x1 %% 2 == 0) {

y1 = c("Even Number")

print(y1)

} else {

y1 = c("Odd Number")

print(y1)

}

**## 두개 이상의 논리값에 대한 판단 (엑셀if와 똑같음)**

**### TYPE1**

ifelse(조건,참,거짓)

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

ifelse(x %% 2 == 0, "even", "odd")

**### TYPE2**

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

y <- c(2,1,4,5)

ifelse(x<y, x, y)

ifelse(sum(x-y) > 0, "positive", ifelse(sum(x-y) < 0 , "negative", "zero"))

* 크게 보면 ifelse(조건, 참, 거짓)의 형식이다.

**## TYPE2와 연관있는 연습문제**

**# 문제 2\_1**

temp <-c(5,20,-6,37,24,13)

# 문제 2번의 값을 ifelse 로 바꿔서 값을 변경하시오

ifelse(temp <= 0, "freezing",

ifelse(temp <= 10, "cold",

ifelse(temp <= 20, "cool",

ifelse(temp <= 30, "warm", "hot"))))

**# 문제 3**

# - ifelse 를 사용해서 iris의 Sepal.Length가 6보타 크면 1 작으면 0 변수 생성하시오

# - new라는 변수 추가하고 new가 1인 Sepal.Width의 합을 구하시오

data(iris)

iris$new <-ifelse(iris$Sepal.Length > 6, 1, 0)

sum(iris[iris$new == "1", "Sepal.Width"])

* new라는 변수를 추가하라고 조건이 주어지면 어떻게 기재하는 것이 가장 좋을지 고민필요 (다른 모든 문제 포함)

**# 루프란?**

# 루프는 작업을 반복하는 R의 방법으로

# 시뮬레이션 프로그래밍에 유용한 도구입니다.

**#expand.grid**

# n개의 벡터에 있는 요소의 모든 조합을 작성

# ex) 두 주사위의 모든 조합

die <- 1:6

rolls <- expand.grid(die,die)

# 모든 주사위의 합은?

rolls$value <- rolls$Var1 + rolls$Var2

* Var1과 Var2는 자동으로 기재된다.

# 확률 N 개의 독립적 인 랜덤 사건의 모든 발생은

각각 임의의 이벤트가 발생하는 확률의 곱과 같다.

prob <- c("1" = 1/8, "2" = 1/8, "3" = 1/8, "4" = 1/8, "5" = 1/8, "6" = 3/8)

prob

rolls$prob <- rolls$prob1 \* rolls$prob2

rolls$prob1 <- prob[rolls$Var1]

rolls$prob2 <- prob[rolls$Var2]

* Var1과 Var2를 Prob로 묶어서 prob1, prob2라는 변수를 생성하고, prob라는 변수는 prob1과 prob2라는 확률의 곱을 나타낸 값이다.

## **주사위의 예상 기대값은?** 각 눈**의 값**에 각 확률을 곱한(\*) **값의** 합(+)

sum(rolls$value \* rolls$prob)

#1번 문제

# wheel 과 같이 총 7개의 경우의 수가 있다.

# 각 확률은 prob와 같고 총 3번의 시도를 했을 경우에 0.001보다 높은 경우의 수 개수는?

# (3번 추출하며 각각 독립이다)

# (DD, BBB, 7) 과 (DD , 7 , BBB)는 다른 경우의 수다

wheel <- c("DD", "7", "BBB", "BB", "B", "C", "0")

rolls <- expand.grid(wheel, wheel, wheel)

head(rolls)

rolls$prob1 = prob[rolls$Var1]

rolls$prob2 = prob[rolls$Var2]

rolls$prob3 = prob[rolls$Var3]

rolls$prob = rolls$prob1 \* rolls$prob2 \* rolls$prob3

nrow(rolls[rolls$prob > 0.001,]) ## 내가 기재한 정답

sum(rolls$prob > 0.001) ## 또다른 계산 방식

**nrow**는 dataframe의 행의 수를 출력하는 **함수**입니다.

**## 2번 문제**

**#2번 문제**

**# 동전을 3번 던질 떄 확률은 0.3과 0.7이다**

**# 첫번째에 앞이나오고 그리고(&) 세번째에 뒤가 나올 확률을 구하시오**

coin <- c("앞","뒤")

prob <- c("앞" = 0.3, "뒤" = 0.7)

rolls2 <- expand.grid(coin, coin, coin)

rolls2$prob1 = prob[rolls2$Var1]

rolls2$prob2 = prob[rolls2$Var2]

rolls2$prob3 = prob[rolls2$Var3]

rolls2$prob = rolls2$prob1 \* rolls2$prob2 \* rolls2$prob3

sum(rolls2[rolls2$Var1 == "앞" & rolls2$Var2 == "뒤", "prob"])

**## for 루프**

For (초기식;조건식;증감식)

{

　　함수식 / printf 내용 등등..

}

for (word in c("My", "second", "for", "loop")) {

print(word)

} ## 그대로 출력

for (value in c("My", "third", "for", "loop")) {

value

} ## 그대로 적재

**### for문을 통한 계산 TYPE1**

chars <- rep(0,4)

words <- c("My", "fourth", "for", "loop")

for (i in 1:4) {

chars[i] <- words[i]

}

chars

**### for문을 통한 계산 TYPE2**

for (i in 1:nrow(rolls)) {

symbols <- sum(rolls[i, 1], rolls[i, 2], rolls[i, 3])

rolls$new[i] <- symbols

**}**

**## 연습문제 1**

**# for문을 사용해서 1부터 100까지의 누적합을 구하시오**

sum <- 0

for(i in 1:100){

sum <- sum + i

}

Sum

**## 연습문제 2**

**# for문을 사용해서 위의 주사위 20번 던진 누적 합을 구하시오.**

sum2 <- 0

sample(1:6,1)

for (i in 1:20) {

sum2 <- sum2 + sample(1:6,1)

}

sum2

* 누적합을 구하는 문제는 수식이 정해져있다. 완벽하게 이해를 못했으면 수식을 인지해놓는 것도 방법

**# while 루프**

while (조건식) {

실행문

**}**

while문은 조건식이 true일 경우에 계속해서 반복하는 문법입니다. 조건식에는 비교 또는 논리 연산식이 줄로 오는데 조건식이 false가 되면 반복을 멈추고 while문을 종료합니다.

* **While문은 항상 i를 제시해줘야한다.**

**#예시**

i <- 1

while (i < 6) {

print(i)

i = i+1

**}**

**## 1~10까지 누적합 구하기**

z <- 0

i <- 1

while( i <= 10) {

z = z + i

cat("cummulative summation",z, "\n")

i = i + 1

}

cat("cummulative summation",z, "\n") 는 paste0 과 같은 개념이다.

## **repeat문**

k <- 1

repeat {

k <- k+3

if (k > 5) break

}

\* k를 따로 클릭해줘야 답이 나오고 k <- k + 3 에서 앞의 k가 5 이상이 되면 멈춘다는 개념인데, k <- k + 3를 누적함수개념으로 대입하면 2번째에 7이 되면서 끝이 난다.

* repeat는 다음것까지 출력을 해준다. (시작하면서 조건이 만족하면 나오는 개념)
* while은 그 전까지 출력을 해준다. (위에서부터 조건을 만족하면서 시작하는 개념)

**###### 함수 만들기 #######**

**# 1. 함수 생성 및 실행하기**

myfunction <- function(){

print("Hi Hello")

}

myfunction()

**# 2. 인수 값 전달 함수**

make\_sum <- function(x,y){

x+y

}

make\_sum(3,4)

**# 3. 기본 값 지정하기**

pp <- function(x,y=6){

x^y

}

pp(2)

pp(4,2)

* y값이 기본 값으로 고정이다.

**# 4. 함수에서 특정 값 반환 return**

make\_sum <- function(x,y){

return(x+y)

}

make\_sum(3,4)

* 내가 Output을 지정하는게 return()이다.

dt <- function(x,y){

add <- x+y

mul <- x\*y

c(add = add, mul = mul)

}

dt(3,5)

add는 비어있는 변수를 의미한다.

**# 5. 인수의 개수가 가변적인 상황**

my\_function <- function(x,...){

print(x)

summary(...)

}

z<-1:20

zz <- my\_function("hi",z)

* (x,...) 인풋의 몇 개가 될지는 모르는데 어쨌든 개수를 가지고 와서 나는 다 사용할꺼야 라는 개념
* Summary는 함수
* x….를 가지고 와서 summary를 구하고 print(x)를 출력하겠다는 의미
* z를 function(x,...)에 대입을 하고 my\_function(“hi”, z)는 print(“hi”)를 할 것이고, 1:20에 대한 summary를 구한다. 그게 바로 zz 변수이기도 하다.
* 값을 저장할 때 summary (z)에 대해서는 저장이 되지만 hi 부분은 저장이 아닌 출력만 된다.
* 사실 return 값은 summary인데 print() 부분은 return 값이 나올 때 마다 반복해서 출력이 되는 것이다.
* Print는 중간에 값을 확인하고 싶을 때 많이 사용된다.

**###### 다양한 apply #######**

**apply(X, MARGIN, FUN)**

**X :** 배열 또는 행렬

**MARGIN :** 함수를 적용하는 방향. 1은 행 방향, 2는 열 방향

* c(1, 2)는 행과 열 방향 모두를 의미

**FUN :** 적용할 함수

* 회사 시험문제로도 나왔던 내용 (특정 값에 대해서 summary 하는 것을 apply로 풀기)

**## apply 예시 1**

d <- matrix(1:9, ncol=3)

d

apply(d, 1, sum)

**## apply 예시 2**

head(iris)

apply(iris[,1:4],2,sum)

* iris 데이터의 1:4행 기준으로 열(컬럼)의 합계를 계산

**## 행, 열의 합 평균은 빈번하게 사용되므로 알면 좋은 함수들**

* **rowSums(x, na.rm=FALSE)**

x : 배열 또는 숫자를 저장한 데이터 프레임

na.rm=FALSE : NA를 제외할지 여부

* **colSums(X, na.rm=False)**
* **행/열들 기준으로 합을 구해줘 (제한적인 부분은 SUM만 가능하다)**

Apply() 함수는 mean, sum, function 등 모든 함수 사용이 가능하다.

**# 반환 값은 행 방향에 저장된 값의 평균이다.**

* rowSums(iris[, 1:4])
* colSums(iris[, 1:4])

**### lapply( )**

# 벡터, 리스트 또는 표현식에 함수를 적용하여 그 결과를 리스트로 반환

# 벡터, 리스트, 표현식, 데이터 프레임 등에 함수를 적용하고 그 결과를 리스트로 반환

lapply(X, FUN, …..)

X : 벡터, 리스트, 표현식 또는 데이터 프레임

FUN : 적용할 함수

... : 추가 인자. 이 인자들은 FUN에 전달된다.

## **lapply 예시**

(result <- lapply(1:3, function(x) { x\*2 })) ## 1:3은 벡터

https://thebook.io/006723/ch04/04/02/ (설명참조)

unlist(result) # 리스트를 다시 해제할 수도 있음

1:3이 function(x)의 x에 적용이 된다.

lapply 그 값의 길이만큼 출력이 된다. (3개의 리스트, 값은 각각 2,4,6)

## lapply가 list 자체를 인자(x)로 받아들일 수 있는 예시

(x <- list(a=1:3, b=4:6))

lapply(x, mean)

# lapply가 data.frame를 인자로 받을 수 있다.

## 리스트형 그대로를 분석하지는 않는다. 그래서 리스트를 데이터 프레임으로 만들기 위해서 부순다(해체한다).

* 그러기 위해서 unlist, do.call 함수를 많이쓴다.

**# unlist : 리스트 구조를 벡터로 변환한다.**

unlist(x, recursive=FALSE, recursive=FALSE)

* x : R 객체. 보통 리스트 또는 벡터
* recursive=FALSE : x에 포함된 리스트 역시 재귀적으로 변환할지 여부
* use.names=TRUE : 리스트 내 값의 이름을 보존할지 여부

**# do.call : 함수를 리스트로 주어진 인자에 적용하여 결과를 반환한다. (강제로 변환한다)**

do.call(what, args)

* **what : 호출할 함수 (예: cbind, rbind 등)**
* **args : 함수에 전달할 인자의 리스트**

[,1] = matrix

**## unlist로 백터 변환 후 데이터프레임 만들기**

d <- as.data.frame(matrix(unlist(lapply(iris[, 1:4], mean)),ncol=4, byrow=TRUE))

names(d) <- names(iris[, 1:4])

1. lapply(iris[, 1:4], mean : lapply로 결과를 도출하면 list 형태로 나옴
2. unlist : list를 unlist하여 vector 형태로 변환한다.
3. matrix ncol=4, byrow=TRUE : vector 형태를 matrix 형태로 변환한다.
4. as.data.frame : matrix 형태를 다시 data.frame 형태로 변환한다.
5. 그리고, (iris[, 1:4])의 컬럼명들을 위에 할당한 변수 d에 다시 적용한다.

**## do.call을 활용한 데이터 프레임 만들기**

data.frame(do.call(cbind, lapply(iris[, 1:4], mean)))

lapply(iris[, 1:4], mean) 으로 나온 리스트를 do.call을 이용하여 리스트들을 cbind로 하여 열로 합친다. 그리고 마지막에는 data.frame으로 묶어준다.

**#### unlist의 문제점!**

x <- list(data.frame(name="foo", value=1),data.frame(name="bar", value=2))

unlist(x)

do.call(rbind,x)

* 구조가 많으면 펼쳤을 때 전부 무너진다 (name, value)

Name, value 밑으로 데이터가 쌓이지 않지만, do.call(rbind,x) 을 사용하면 가능하다.

**## sapply( )**

# lapply와 유사하지만 결과를 벡터, 행렬 또는 배열로 반환

sapply(X, FUN, ...)

X : 벡터, 리스트, 표현식 또는 데이터 프레임

FUN : 적용할 함수

... : 추가 인자. 이 인자들은 FUN에 전달된다.

**## 예시**

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

[1] "numeric" 이런식으로 결과값이 나온다.

**## sapply( )에서 반환한 벡터는 as.data.frame( )을 사용해 데이터 프레임으로 변환**

x <- sapply(iris[, 1:4], mean)

as.data.frame(x)

as.data.frame(t(x))

sapply( )에서 반환한 벡터는 as.data.frame( )을 사용해 데이터 프레임으로 변환할 수 있다. 이때 t(x)를 사용해 벡터의 행과 열을 바꿔주지 않으면 기대한 것과 다른 모양의 데이터 프레임을 얻게 된다.

**## 변수 구조 확인할 시 자주 사용**

sapply(iris, class)

sapply(X, FUN, ...) 기본 형식 적용 (class함수 사용한 것임)

str(iris)

> sapply(iris, class)

Sepal.Length Sepal.Width Petal.Length Petal.Width Species

"numeric" "numeric" "numeric" "numeric" "factor"

다수의 컬럼을 포함하는 데이터 프레임을 처리하다 보면 종종 각 컬럼의 데이터 타입을 알아내고 싶을 때가 있다. 예를 들면, 어떤 컬럼에 숫자가 저장되어 있는지를 판단하는 경우다. 이때 sapply( )를 유용하게 사용할 수 있다. 다음은 아이리스 데이터에서 각 컬럼의 데이터 타입을 구하는 예다.

**## sapply( )에 인자로 주어진 함수의 출력이 길이가 1보다 큰 벡터라면 sapply( )는 행렬을 반환**

y <- sapply(iris[, 1:4], function(x) { x > 3 })

TRUE, FALSE로 값을 반환

**## tapply( )**

# 벡터에 있는 데이터를 특정 기준에 따라 그룹으로 묶은 뒤 각 그룹마다 주어진 함수를 적용하고 그 결과를 반환

# 반환 값은 배열이다.

tapply(X, INDEX, FUN, ...)

X : 벡터 (데이터)

INDEX : 데이터를 그룹으로 묶을 색인. 팩터를 지정해야 하며 팩터가 아닌 타입이 지정되면

팩터로 형 변환된다.

FUN : 각 그룹마다 적용할 함수

... : 추가 인자. 이 인자들은 FUN에 전달된다.

**# 1부터 10까지의 숫자가 있고 이들이 모두 한 그룹에 속해 있을 때, 이 그룹에 속한 데이터의 합**

tapply(1:10, rep(1, 10), sum)

위 코드에서 rep(1, 10)은 1을 10회 반복하는 것을 의미한다. 따라서 숫자 1, 2, 3, …, 10에 대해 동일한 소속 번호 1, 1, 1, …, 1을 부여한 것이다. 그러므로 그룹 1에 속한 데이터의 합은 55(=1+2+3+…+10)다.

**# 홀짝에 따라서**

tapply(1:10, 1:10 %% 2 == 1, sum)

INDEX에 홀수와 짝수별로 다른 팩터 값이 주어지도록 %% 2를 사용했다. 수행 결과 짝수의 합이 30(=2+4+6+8+10), 홀수의 합이 25(=1+3+5+7+9)로 구해졌다.

**## 아이리스 데이터에서 Species별 Sepal.Length의 평균을 구해보자.**

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

**### mapply( )**

mapply(FUN, ...)

FUN : 실행할 함수

... : 적용할 인자

# sapply의 확장된 버전으로, 여러 개의 벡터 또는 리스트를 인자로 받아 함수에 각 데이터의 첫째 요소들을 적용한 결과,

# 둘째 요소들을 적용한 결과, 셋째 요소들을 적용한 결과 등을 반환

# sapply( )와 유사하지만 다수의 인자를 함수에 넘긴다는 점에서 차이

**rnorm(10, 0, 1)**

정규분포식에서 랜덤으로 평균이 0이고 분산이 1인 데이터를 10개 뽑아와

**mapply(rnorm,**

**c(1, 2, 3),**

**c(0, 10, 100),**

**c(1, 1, 1))**

* Mapply를 이용하여 rnorm을 적용할꺼야
* 값이 3개의 인자가 들어가 있는데

rnorm 함수를 사용하는데, 첫째 인자(생성할 난수의 개수)는 1,2,3이고, 둘째 인자(평균)은 0,10,100, 셋째 인자(표준편차)는 1,1,1로 연속 실행하라

# 1은 rnorm(n=1, mean=0, sd=1),

# 2는 rnorm(n=2, mean=10, sd=1),

# 3은 rnorm(n=3, mean=100, sd=1)에 해당한다.

**# mapply( )에는 iris의 모든 행이 나열되어 인자**

mapply(mean, iris[, 1:4])

사실 이해를 못한 부분임 (인터넷 서칭 및 선생님 질문할 것)