제어문

조건문 – if()

1) if()

```
if(x*y > 40){
  cat("x*y의 결과는 40 이상입니다.₩n")  # ₩n 줄바꿈
  cat("x*y =", z)
}else{
  cat("x*y의 결과는 40 미만입니다. x*y =", z,"₩n")
}
```

조건문 – ifelse()

2) ifelse(조건, 참, 거짓) - 3항 연산자 기능

```
ifelse(score>=80, "우수","노력") #우수
ifelse(score<=80, "우수","노력") #노력
```

조건문 – switch / which

3) switch 문

형식) switch(비교 구문, 실행구문1, 실행구문2, 실행구문3)

```
switch("name", age=105, name="홍길동", id="hong", pwd="1234")
```

4) which 문

형식) which()의 괄호내의 조건에 해당하는 위치(인덱스)를 출력한다.

```
# 벡터에서 사용
name <- c("kim","lee","choi","park")
which(name=="choi") # [1] 3
```

조건문 – switch / which

```
# 데이터프레임에서 사용
    no <- c(1:5)
    name <-c("홍길동","이순신","강감찬","유관순","김유신")
    score <- c(85,78,89,90,74)

    exam <- data.frame(학번=no,이름=name,성적=score)
    exam
    which(exam$이름=="유관순") # [1] 4, 없으면 0
    exam[4,] # 4번째 레코드 보기
```

반복문 – for()

1) 반복문

형식) for(변수 in 값) {표현식} - 단일문{} 생략 가능

```
i <- c(1:10)
for(n in i){ # 10회 반복
    print(n * 10) # 계산식(numeric만 가능) 출력
    print(n)
}
```

반복문 – for()

```
 for(n in i){
  if(n%%2==0){
    next # 다음문장 skip -> 반복문 계속
  }else{
    print(n) # 홀수만 출력
  }
}
```

```
[1] 1
[1] 3
[1] 5
[1] 7
[1] 9
```

반복문 – for()

```
# 데이터 파일의 변수명 출력
    name <- c(names(exam))
    for(n in name){ # 변수명 출력
        print(n)
    }
```

```
[1] "학번"
[1] "이름"
[1] "성적"
```

반복문 – for()

```
score = c(85, 95, 98)
name = c('홍길동', '이순신', '강감찬')


i<- 1
for (s in score){
    cat(name[i]," -> ", s, "₩n")
    i <- i + 1
}
```

```
홍길동 -> 85
이순신 -> 95
강감찬 -> 98
```

반복문 – while()

2) 반복문 - while(조건){표현식}

```
i = 0
while(i< 10){
    i <- i + 1
    print(i) # 1~10까지 출력됨
}
```

함 수

사용자 정의 함수

- 사용자 정의함수 형식

(형식)

함수명 <- function(매개변수){ }

```
# 매개변수가 없는 함수 예
    f1 <- function( ){
      cat("매개변수가 없는 함수")
    }
    f1() # 함수 호출
```

사용자 정의 함수

```
# 매개변수가 있는 함수 예


    f2<- function(x){
        cat("x의 값 = ",x, "₩n") # ₩n 줄바꿈
        print(x) # 변수만 사용
    }
    f2(15) # 함수 호출
```

사용자 정의 함수

- 피타고라스 정의 증명- 식 : a^2+b^2=c^2

```
pythagoras <- function(s,t){
  a <- s^2 - t^2
  b <- 2*s*t
  c <- s^2 + t^2
  cat("피타고라스의 정리 : 3개의 변수 : ", a, b, c)
}

pythagoras(2,1)     # s,t는 양의 정수 -> 3 4 5
```

사용자 정의 함수

- 구구단 출력하기

```
gugudan <- function(i,j){
  for(x in i){
    cat("**", x , "단 **₩n")
    for(y in j){
      cat(x, "*", y, "=", x*y, "₩n")
    }
    cat("₩n")
  }
}

i<- c(2:9)
j<- c(1:9)

gugudan(i,j)
```

내장함수

- 기술통계량 처리 내장함수

```
min(vec) # 벡터 대상 최소값
max(vec) # 벡터 대상 최대값
range(vec) # 벡터 대상 범위 값
mean(vec) # 벡터 대상 평균값
median(vec) # 벡터 대상 사분위수
sum(vec) # 벡터 대상 합계
sort(x)  # 벡터 정렬 (단, 원래의 값을 바꾸지는 않음)
order(x)  # 벡터의 정렬된 값의 인덱스를 보여줌
rank(x)  # 벡터의 각 원소의 순위를 알려줌
sd(x) # 표준편차
summary(x)  # 데이터에 대한 기본적인 통계 정보 요약
table(x)  # 데이터 빈도수
```

내장함수

- 수학관련 내장함수

abs(x) # 절대값
sqrt(x) # 제곱근
ceiling(x), floor(), round() # 값의 올림, 내림, 반올림
factorial(x) # 팩토리얼 함수
which.min(x), which.max(x) # 벡터 내의 최소값과 최대값의 인덱스
pmin(x), pmax(x) # 여러 벡터에서의 원소 단위 최소값과 최대값
prod() # 벡터의 원소들의 곱
cumsum(), cumprod() # 벡터의 원소들의 누적합과 누적곱
cos(x), sin(x), tan(x) # 삼각함수 (also acos(x), cosh(x), acosh(x), etc)
log(x) # 자연로그(natural logarithm)
log10(x) # 10을 밑으로 하는 일반로그 함수(e^x)

내장함수

- 행렬연산 내장함수

ncol(x) # 열의 수
nrow(x) # 행의 수
t(x) # 전치행렬
cbind(…) # 열을 더할 때 이용되는 함수
rbind(…) # 행을 더할 때 이용되는 함수
diag(x) # 대각행렬
det(x) # 행렬식
apply(x, m, fun) # 행 또는 열에 함수 적용
x %*% y # 두 행렬의 곱
solve(x) # 역 행렬
svd(x) # Singular Value Decomposition
qr(x) # QR Decomposition (QR 분해)
eigen(x) # Eigenvalues(고유값)
chol(x) # choleski decomposition(Choleski 분해)

내장함수

- 집합연산 내장함수

union (x, y) # 집합 x와 y의 합집합

intersect (x, y) # 집합 x와 y의 교집합

setdiff (x, y) # x의 모든 원소 중 y에는 없는 x와 y의 차집합

setequal (x, y) # x와 y의 동일성 테스트

c %in% y # c가 집합 y의 원소인지 테스트

choose (n, k) # 크기 n의 집합에서 크기 k의 가능한 부분 집합 개수

내장함수

- 기초 통계량 관련 함수

```
getwd()
setwd("c:/workspaces/Rwork/data")
#excel에서 csv(쉼표로 분리)형식으로 저장한 파일 가져오기
excel <- read.csv("excel.csv", header=TRUE)
# head()함수이용 앞쪽 10줄 출력
head(excel,10) # q1 q2 q3 q4 q5
#colMeans()함수 이용 각 열의 평균 계산
colMeans(excel[1:5])
#q1       q2       q3       q4       q5
#2.733831 2.907960 3.621891 2.509950 3.385572

# summary()함수 이용 각 열단위 기초 통계량
summary(excel)
```