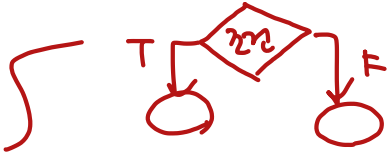
R 기초

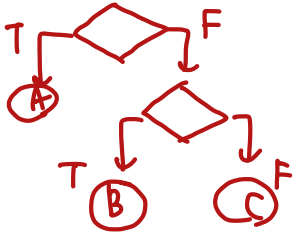
조건문과 반복문

조건문

특정조건만족 여부에 따라 별개의 작업을 진행

- R: if()와 ifelse()
  - 조건이 하나: if
  - 조건이 둘: if ~ else
  - 조건이 여러 개: if A ~ else if B (else if ...) ~ else C





- ifelse()는 조건결과가 벡터이면 각각의 결과에 대해 연산이 이루어짐

반복문

- 작업을 반복하기 위한 명령어
  - R: for문, while문, repeat문
    - for: 반복수 지정
    - while: 특정 조건을 만족하는 동안 실행
    - repeat: 내부에서 중단되지 않은 동안 계속 실행
  - 각 반복문의 구조 이해

조건문 정리

특정 조건 만족여부에 따라 별개의 작업을 진행

R: if()와 ifelse()

- 조건이 하나: if
- 조건이 둘: if ~ else
- 조건이 여러 개: if ~ else if ~ (else if ...) ~ else

주의점: if()와 ifelse()의 차이점

- 벡터 조건의 경우 if는 처음 결과만 ifelse는 각각의 결과에 대해 적용
- 논리연산에서 && vs &, || vs |와 유사

반복문 정리

- 작업을 반복하기 위한 명령어
  - R: for문, while문, repeat문
    - for(x in vector) { }
    - while(조건) {}
    - repeat{ if (조건) break}
    - { } 내에서 break를 통해 중단시킬 수 있음

R 기초

함수만들기와 기하평균추론

함수만들기

- 함수를 정의하는 방법
  - 인수의 설정과 인수의 적용범위
  - 인수의 디폴트 지정방법
  - 결과의 출력형태 지정

기하평균추론

블러누익늒,…

- **기하평균:** $\bar{x}_G = (x_1 \times \cdots \times x_n)^{1/n} = \left(\prod_{i=1}^{n} x_i\right)^{1/n} = \exp\left(\frac{1}{n}\sum_{i=1}^{n} \log(x_i)\right)$
- **기하평균추론** 데이터가 반드시 양수여야함.
  - ☑ 분포가정: $Y_1 = \log(X_1), \cdots, Y_n = \log(X_n) \sim iid\ N(\mu_Y, \sigma^2)$
  - ☑ $\mu_G = \exp\left(\mathrm{E}(\log(X))\right) = \exp(\mu_Y)$
  - ☑ 기하평균 μ_G의 신뢰구간: $(\exp(L), \exp(U))$
    - L : 로그 변환한 자료를 이용한 신뢰하한
    - U: 로그 변환한 자료를 이용한 신뢰상한
  - ☑ 자료를 로그 변환 후 t.test() 적용하고 결과를 역변환

실습

함수만들기

- 함수를 정의하는 방법: 함수이름 <- function(인수들)
  - 인수의 설정과 인수의 적용범위
  - 인수의 디폴트 지정방법: 인수 = 디폴트값
  - 결과의 출력형태 지정
  - 전역변수 지정: <<-

기하평균추론

- 기하평균: $\bar{x}_G = (x_1 \times \cdots \times x_n)^{1/n} = \left(\prod_{i=1}^{n} x_i\right)^{1/n} = \exp\left(\frac{1}{n}\sum_{i=1}^{n} \log(x_i)\right)$
  - min(x)≥0이면 exp(mean(log(x))

- 기하평균추론
  - 자료에 음수가 있는지 확인
  - 자료를 로그 변환 후 t.test() 적용하고 결과를 역변환

```
###########
## 조건문: if문
## * if (condition) {statements1} else {statements2}
## * else {statements2}는 생략가능하며 statements가 하나인 경우 {} 생략가능
## * condition의 결과가 벡터이면 첫 번째 원소만 적용
## 조건이 여러개인 경우
## * if (condition1) {
##    } else if (condition2) {
##    } else {}
## 벡터 조건문:
## * ifelse(condition, statement1, statement2)
## * condition의 결과가 벡터이면 개별 결과 적용

## * 복습: 논리연산
x <- c(F,T); y <- T
x && y; x || y ⇒ F,T
x & y; x | y  ⇒ FT, TT

x <- 1
if (x > 0) y <- "Yes" else y <- "No"
y ⇒ "Yes"

# 주의
if (x > 0) y <- "Yes"
else y <- "No"  ⇒ Error : 다른문장으로인식
y ⇒"Yes"

x <- -1:1 (x: -1,0,1)
if (x > 0) y <- x+1 else y <- x-1
y ⇒ -2 -1 X 0
Warning: the condition has length > 1 and only the first element will be used.
↳첫번째값(-1)만사용→이미 false로판정
z <- ifelse(x > 0, x+1, x-1)
z ⇒ -2 -1 2  → 개별적으로비교

ifelse(x %% 2 == 1, "odd", "even")   아니면(else)
홀짝 <- ifelse(x %% 2, "odd", "even")   같은결과, 아래가더빠름
                                        ↳1은T, 0은F로(논리연산)
홀짝 ⇒"odd" "even" "odd"
짝수에대한 홀짝도가능
##############
## 반복문

## for 루프
## * for (name in vector) statement1
## * for (name in vector) {statements1}
## * vector는 문자도 가능
```

여러개의명령어 묶어주기 :

```
계속연결되게
if (x>0) {
    y <- "Yes"
    a <- "A"
} else {
    y <- "No"
    a <- "a"
}
```

```
##  while 루프
## * while(condition) statement
## * while(condition) {statements}
## * condition이 참이면 statement(s)를 실행
## * 무한 루프에 빠질 가능성이 있음

## repeat 루프
## * repeat{statements}

## break로 중단
## * 무한 루프에 빠질 가능성

# 1~10까지 합
n <- 10
csum <- 0
for (x in 1:n)
  csum <- csum+x
csum

# 1~n까지 짝수의 합
n <- 1000000
x <- 1:n
evensum <- 0
for (value in x)
  if (value %% 2 == 0)  evensum <- evensum+value
evensum

## 반복문은 속도가 느림
sum(x[x%%2 == 0])
sum(x[!x%%2])

## 문자처리 가능
item = c("C","B","D","A","F")
for (x in c("A","B"))
  cat(x, " is ",which(x==item),"-th item.₩n")

for (x in c("A","B","K"))
  cat(x, " is ",which(x==item),"-th item.₩n")
for (x in c("A","B","K"))
  if (x %in% item)    ## %in% :vector 비교 가능(문자, 숫자 비교가능)
    cat(x, " is ",which(x==item),"-th item.₩n")

## 반복횟수가 정해지지 않은 경우: while, repeat
## while(조건): 조건이 참이면 계속수행
## repeat: 최소한 한번은 수행
```

좋지않은방법

초기값입력 X

```
# -1.0보다 작은 값이 나올때까지 표준정규난수를 발생
x <- numeric() # x <- NULL (빈껍데기)
for (i in 1:1000)    #충분히 큰 수만큼 반복
{
  난수 <-  rnorm(1)
  x <- c(x,난수)
  if (난수 < -1.0)
    break
}
x

난수 <- rnorm(1)
x <- 난수
while(난수 >= -1.0)
{
  난수 <- rnorm(1)
  x <- c(x,난수)
}
x

x <- numeric()
repeat{    일단작업부터.( (의 dowhile )
  난수 <- rnorm(1)
  x <- c(x,난수)
  if (난수 < -1.0)
    break
}
x
```

seed 정하기 : set.seed( )

<8-27

```
##############
### 함수만들기
##############
x <- rnorm(30)    # 정규난수 → ? rnorm 설명보기
mean(x)           # 평균계산
x
x[21] <- NA       # 결측값 추가
mean(x)           NA (∵ na.rm=FALSE 설정되어있음 : default)
mean(x,na.rm=T)   # 결측값 제거 후 평균계산
mean(x,trim=0.5,na.rm=T)  # 50%절사평균(trimmed mean)
median(x,na.rm=T)         # 중앙값

## * 기하평균   exp(-∞)=0, exponential 지수 : skewed data (오른쪽꼬리가길다)
x <- exp(rnorm(30))
exp(mean(log(x)))
product(x)^(1/30)   } 기하평균  데이터의개수가 많은경우 Inf가 나와버림
## *함수만들기:
### * "함수이름 <- function(인수들)" 형태로 정의
### * 인수, default 지정
### * 인수의 조건 만족여부 확인
###     - 기하평균: 자료는 0보다 커야함

minus1 <- function(x) x <- x-1   # 함수의 인수는 내부에서만 적용(지역변수)

x <- 1:5
minus5 <- function(y) x-y        # 함수의 인수가 아닌 변수는 앞에서 정의된 값 적용
minus5(5)

minusA <- function(y) a+y        # 변수 a는 앞에서 정의되지 않음
minusA(5)

geomean <- function(x)
  exp(mean(log(x)))

geomean(x)

x <- exp(rnorm(20))
x[21] <- NA
geomean(x)

geomean <- function(x)
{
  x <- x[!is.na(x)]
  if (min(x) >= 0.0)
    return(c(length(x),exp(mean(log(x)))))
  else
    cat("자료 중 0보다 작은 값이 있어 기하평균을 계산할 수 없습니다.")
}
```

표준편차와 평균이비례 → log(x)
($\frac{S}{\mu}$ 또는 $\frac{S}{\bar{x}}$ ⇒ 변동계수CV)

minus1 <- function(x) { x <- x-1 return(x) }
↳ 결과반환은 전역변수x값은 아직 바뀌지않음. 출력할게 x하나이므로 return없이 x만 써도됨.

minus1 <- function(x) { x <<- x-1 }
⇒ x도 전역변수로

점검할것
① 양수인지 (원래는 0도 안되지만 R에서 exp(-∞)=0으로계산)
② NA가아닌지

최솟값이 0이상이면 모든데이터가 0 이상이다.

ex)
geomean(x) ⇒ 29.00000 1.08097
└ 다중결과값이 실수형태와 같이 실수형태로출력됨.

}

```
geomean <- function(x)
{
  x <- x[!is.na(x)]
  if (min(x) >= 0.0)
  {
    Result <- list()
    Result$n <- length(x)
    Result$Geomean <- exp(mean(log(x)))
    Result
  }
  else
    cat("자료 중 0보다 작은 값이 있어 기하평균을 계산할 수 없습니다.")
}
```

빈리스트(null해도됨) Result를리스트형태로출력할것이므로

또는 return(Result)

geomean(x) ⇒ $n
[1]29
$Geomean
[1]1.08097

결과 <- geomean(x)
(리스트와똑같이사용가능)
결과$Geomean 또는결과[[2]]

? t.test (var.equal=FALSE가 default)

```
geomean.test <- function(x,mu=1.0,alternative="two.sided",conf.level=0.95)
{
  if (mu <= 0.0 | min(x,na.rm=T) <= 0.0)
    cat("모수설정이 잘못되거나 자료에 0보다 작거나 같은 값이 있습니다.")
  else
  {
    n <- sum(!is.na(x))
    if (n < length(x))
      x <- x[!is.na(x)]
    logx <- log(x)
    logmu <- log(mu)
    result <- t.test(logx,mu=logmu,alternative=alternative,conf.level=conf.level)
    result$null.value <- exp(result$null.value)
    result$conf.int <- exp(result$conf.int)
    result$estimate <- exp(result$estimate)
    return(result)
  }
}
```

평균

귀무가설의이름 값의이름

geomean.test(x)

```
> geomean.test(x)

        One Sample t-test

data:  logx
t = 0.34103, df = 28, p-value = 0.7356
alternative hypothesis: true mean is not equal to 1
95 percent confidence interval:
 0.6771886 1.7255096
sample estimates:
mean of x
  1.08097

>
```

```
> t.test(log(x))

        One Sample t-test

data:  log(x)
t = 0.34103, df = 28, p-value = 0.7356
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -0.3898054  0.5455224
sample estimates:
 mean of x
0.07785851
```

```
> exp(-0.3898054)
[1] 0.6771886
>
```