전산통계학 실습

R 프로그래밍구조

R 프로그래밍

- 기존의 통계학 언어 프로그래밍들은 코드 작성의 확장성이 부족하여, 사용자의 요구에 맞게 변형하기가 어려웠다.

- R은 다른 컴퓨터 프로그래밍 언어처럼 다양한 연산, 함수, 프로그래밍 등을 반영하여 개발되었다.
  - 함수, 조건문, 반복문
  - 산술, 부울 연산
  - 문자열 처리
  - 입력과 출력

함수와 조건문

```
> comparison <- function(a, b) {
+   if(a > b) {
+     return(a)
+   }
+   else {
+     return(b)
+   }
+ }
> comparison(3, 5)
[1] 5
> comparison(-2, 0)
[1] 0
> comparison <- function(a, b) {
+   if (a > b) {
+     a
+   }
+   else {
+     b
+   }
+ }
> comparison(4, -6)
[1] 4
```

- 함수의 생성
  - > name <- function(parameters) { }
    - name 이름을 가진 function을 생성하고, 입력 받을 parameters를 지정한다.
- 함수의 반환
  - > return(x)
    - 함수 결과로 하나의 인자를 반환한다.
  - return 함수를 사용하지 않고, 해당 변수를 단순 기재하여 반환할 수 있다.
- 조건문
  - 기존 언어들과 같이 if / else 조건문을 사용할 수 있다.

반복문

```
cumsum_while <- function(n) {
      sum <- 0
      now <- 1
      while (now < n+1) {
            sum <- sum + now
            now <- now + 1
      }
      return(sum)
}

cumsum_for <- function(n) {
      sum <- 0
      now <- 1
      for (i in now:n) {
            sum <- sum + i
      }
      return(sum)
}

cumsum_repeat <- function(n) {
      sum <- 0
      now <- 1
      repeat {
            if (now > n) {
                  break
            }
            sum <- sum + now
            now <- now + 1
      }
      return(sum)
}
> cumsum_while(10)
[1] 55
> cumsum_for(10)
[1] 55
> cumsum_repeat(10)
[1] 55
```

- while 반복문
  - > while (cond) { }
    - 조건이 만족되지 않을 때까지 반복
    - 조건이 만족되면 반복문 내부 실행
- for 반복문
  - > for (i in data) { }
    - 벡터, 리스트 등 여러 인자를 가진 데이터 변수로부터 내부 인자를 하나씩 가져와 대입하여 반복
    - 데이터 내부 탐색을 모두 완료하면 종료
- repeat 반복문
  - > repeat { }
    - 반복문 내부를 무한히 실행
  - > break
    - 반복문 종료
  - > next
    - 가장 가까운 반복문으로 돌아가기 (continue)

함수

```
> f <- function(...) {
+   args <- list(...)
+   for (a in args) {
+     print(a)
+   }
+ }
> f(1, 4, 5, 10)
[1] 1
[1] 4
[1] 5
[1] 10
> g <- function(a, b) {
+   print(a)
+   h <- function(a, b) {
+     print(a+b)
+   }
+   h(a, b)
+ }
> g(3, 5)
[1] 3
[1] 8
```

- 가변 길이 매개변수
  - 함수의 매개 변수를 정해지지 않은 수만큼 입력 받고 싶을 때, 가변 길이 매개변수로 선언하면 된다.
  - 입력 받은 매개변수는 리스트 형태로 할당 후 사용할 수 있다.
- 함수 내 함수 선언
  - 함수 내부에서 함수를 선언하여 사용
    - 함수 내부에서 어떤 함수를 반복적으로 사용하고 이후에 사용할 일이 없을 경우 메모리 절약 용도로 이용할 수 있다.

입력과 출력 (I/O)

```
add <- function() {
      num1 <- readline("First Num: ")
      num2 <- readline("Second Num: ")
      result <- as.numeric(num1) + as.numeric(num2)
      print(result)
}
```

```
> add()
First Num: 5
Second Num: 7
[1] 12
```

- 입력
  - > readline(“comment”)
    - 키보드로부터 한 줄 입력 받는 함수
    - 입력 받은 내용은 character 형태로 저장된다.
- 출력
  - > print(…)
    - 함수로 입력된 내용을 출력한다.

문자열 처리

```
> str1 <- "North"
> str2 <- "Pole"
> str3 <- "South Pole"
> nchar(str1)
[1] 5
> paste(str1, str2)
[1] "North Pole"
> paste(str1, str2, sep="")
[1] "NorthPole"
> paste(str1, str2, sep=".")
[1] "North.Pole"
> substr(str3, 7, 10)
[1] "Pole"
> strsplit(str3, split=" ")
[[1]]
[1] "South" "Pole"
```

- 문자열 처리
  - > nchar(x)
    - 해당 문자열의 길이 반환
  - > paste(..., sep="")
    - 여러 문자열을 이어 붙여 반환
    - sep 옵션을 이용하여 각 문자열 사이에 들어갈 문자열을 지정할 수 있다. (기본값 = 공백)
  - > substr(x, start, end)
    - 해당 문자열에서 start부터 end까지 범위에 위치한 부분 문자열을 반환
  - > strsplit(x, split="")
    - 해당 문자열에서 split 문자열을 기준으로 나눈 결과인 부분 문자열들의 리스트를 반환

산술 연산

산술 연산	설명
x + y	덧셈
x – y	뺄셈
x * y	곱셈
x / y	나눗셈
x ^ y	x의 y승
x %% y	나머지 연산
x %/% y	나눗셈 결과의 정수형
exp(x)	e의 x승
log(x, base)	밑이 base인 log함수 (기본값 = e)
log2(x), log10(x)	밑이 2, 10인 log함수
sin(x), cos(x), tan(x)	삼각함수

부울 연산

산술 연산	설명
x == y	일치 비교
x >= y, x <= y	크기 비교
x && y	AND (스칼라)
x || y	OR (스칼라)
x & y	AND (벡터)
x | y	OR (벡터)
!x	NOT

부울 사용

```
> x <- c(TRUE, FALSE, TRUE)
> y <- c(TRUE, FALSE, FALSE)
> x & y
[1]  TRUE FALSE FALSE
> x && y
[1] TRUE
> x[1] && y[1]
[1] TRUE
> x * 3
[1] 3 0 3
> x <- c(TRUE, FALSE, TRUE)
> y <- c(TRUE, FALSE, FALSE)
> x & y
[1]  TRUE FALSE FALSE
> x && y
[1] TRUE
> x[1] && y[1]
[1] TRUE
> x * 3
[1] 3 0 3
> (x * 3)[1] == 3
[1] TRUE
>
```

- 부울 연산의 사용
  - & 연산은 벡터의 각 원소마다 모두 실행되어 결과를 나타낸다.
  - && 연산은 벡터의 첫 번째 원소에만 실행되어 결과를 나타낸다.
- TRUE/FALSE
  - TRUE는 1로, FALSE는 0인 Factor와 같기 때문에 산술 연산이 가능하다.

전역변수

```
len1 <- 0
len2 <<- 0
deletemax <- function(...) {
      args <- unlist(list(...))
      maxindex <- which.max(args)
      if (maxindex == length(args)) {
            newv <- args[1:maxindex-1]
      }
      else {
            newv <- c(args[1:maxindex-1], args[maxindex+1:length(args)])
      }
      len1 <- length(newv)
      len2 <<- length(newv)
      return(newv)
}
 > len1
 [1] 0
 > len2
 [1] 0
 > deletemax(3, 4, 1, 5, 9, -2, 10)
 [1]  3  4  1  5  9 -2
 > len1
 [1] 0
 > len2
 [1] 6
```

- 전역변수 사용
  - 전역변수로 선언한 변수는 함수 내에서도 <<- 를 이용하여 수정하는 경우 수정한 내용이 반영되어 유지된다.

예시) 데이터셋 만들기

```
randomscores <- function(studentnum) {
      score <- c()
      for (i in 1:studentnum) {
            score <- c(score, sample(0:100, 1))
      }
      return(score)
}

passdata <- function(score, criteria) {
      passes <- c()
      for (s in score) {
            if (s >= criteria) {
                  passes <- c(passes, "P")
            }
            else {
                  passes <- c(passes, "F")
            }
      }
      return(passes)
}

# 3 Scores and Results of 30 students
no <- seq(1:30)
kor <- randomscores(length(no))
eng <- randomscores(length(no))
mat <- randomscores(length(no))
matpass <- passdata(mat, 60)

classdata <- data.frame("NO"=no, "KOR"=kor, "ENG"=eng, "MAT"=mat, "MAT_PASS"=matpass)
```

```
> classdata
   NO KOR ENG MAT MAT_PASS
1   1  90  75  73        P
2   2  49  96  53        F
3   3  79  50   9        F
4   4   5  59  87        P
5   5  34  11  38        F
6   6  56  17  18        F
7   7  47  19   6        F
8   8  56  36  93        P
9   9  59  64  85        P
10 10 100   8  88        P
11 11  65  26  19        F
12 12  45  34  95        P
13 13  59  55  65        P
14 14  84   9  83        P
15 15  82  57   4        F
16 16  94  90   7        F
17 17  95  21  15        F
18 18  14  32  62        P
19 19  45  61  92        P
20 20  70  39  99        P
21 21  24  55  95        P
22 22   9  99  85        P
23 23  28  95  39        F
24 24  76  99  76        P
25 25  32  91  29        F
26 26  99   0  69        P
27 27  60  23  31        F
28 28  23  73   3        F
29 29  27  71  11        F
30 30  87  96  70        P
```

- 무작위 추출 함수
  - > sample(data, size)
    - data 내부에서 size 개수만큼 무작위로 추출하여 반환

예시) 데이터셋 만들기

```
randomscores <- function(studentnum) {
      score <- c()
      for (i in 1:studentnum) {
            score <- c(score, sample(0:100, 1))
      }
      return(score)
}

passdata <- function(score, criteria) {
      passes <- c()
      for (s in score) {
            if (s >= criteria) {
                  passes <- c(passes, "P")
            }
            else {
                  passes <- c(passes, "F")
            }
      }
      return(passes)
}

# 3 Scores and Results of 30 students
no <- seq(1:30)
kor <- randomscores(length(no))
eng <- randomscores(length(no))
mat <- randomscores(length(no))
matpass <- passdata(mat, 60)

classdata <- data.frame("NO"=no, "KOR"=kor, "ENG"=eng, "MAT"=mat, "MAT_PASS"=matpass)
```

```
> classdata
   NO KOR ENG MAT MAT_PASS
1   1  90  75  73        P
2   2  49  96  53        F
3   3  79  50   9        F
4   4   5  59  87        P
5   5  34  11  38        F
6   6  56  17  18        F
7   7  47  19   6        F
8   8  56  36  93        P
9   9  59  64  85        P
10 10 100   8  88        P
11 11  65  26  19        F
12 12  45  34  95        P
13 13  59  55  65        P
14 14  84   9  83        P
15 15  82  57   4        F
16 16  94  90   7        F
17 17  95  21  15        F
18 18  14  32  62        P
19 19  45  61  92        P
20 20  70  39  99        P
21 21  24  55  95        P
22 22   9  99  85        P
23 23  28  95  39        F
24 24  76  99  76        P
25 25  32  91  29        F
26 26  99   0  69        P
27 27  60  23  31        F
28 28  23  73   3        F
29 29  27  71  11        F
30 30  87  96  70        P
```

- > randomscores(studentnum)
  - 입력된 학생 수만큼 무작위 생성된 점수들을 반환

예시) 데이터셋 만들기

```
randomscores <- function(studentnum) {
      score <- c()
      for (i in 1:studentnum) {
            score <- c(score, sample(0:100, 1))
      }
      return(score)
}

passdata <- function(score, criteria) {
      passes <- c()
      for (s in score) {
            if (s >= criteria) {
                  passes <- c(passes, "P")
            }
            else {
                  passes <- c(passes, "F")
            }
      }
      return(passes)
}

# 3 Scores and Results of 30 students
no <- seq(1:30)
kor <- randomscores(length(no))
eng <- randomscores(length(no))
mat <- randomscores(length(no))
matpass <- passdata(mat, 60)

classdata <- data.frame("NO"=no, "KOR"=kor, "ENG"=eng, "MAT"=mat, "MAT_PASS"=matpass)
```

```
> classdata
   NO KOR ENG MAT MAT_PASS
1   1  90  75  73        P
2   2  49  96  53        F
3   3  79  50   9        F
4   4   5  59  87        P
5   5  34  11  38        F
6   6  56  17  18        F
7   7  47  19   6        F
8   8  56  36  93        P
9   9  59  64  85        P
10 10 100   8  88        P
11 11  65  26  19        F
12 12  45  34  95        P
13 13  59  55  65        P
14 14  84   9  83        P
15 15  82  57   4        F
16 16  94  90   7        F
17 17  95  21  15        F
18 18  14  32  62        P
19 19  45  61  92        P
20 20  70  39  99        P
21 21  24  55  95        P
22 22   9  99  85        P
23 23  28  95  39        F
24 24  76  99  76        P
25 25  32  91  29        F
26 26  99   0  69        P
27 27  60  23  31        F
28 28  23  73   3        F
29 29  27  71  11        F
30 30  87  96  70        P
```

- > passdata(score, criteria)
  - 여러 점수들에 대하여 기준 점수 통과 여부를 반환

예시) 데이터셋 만들기

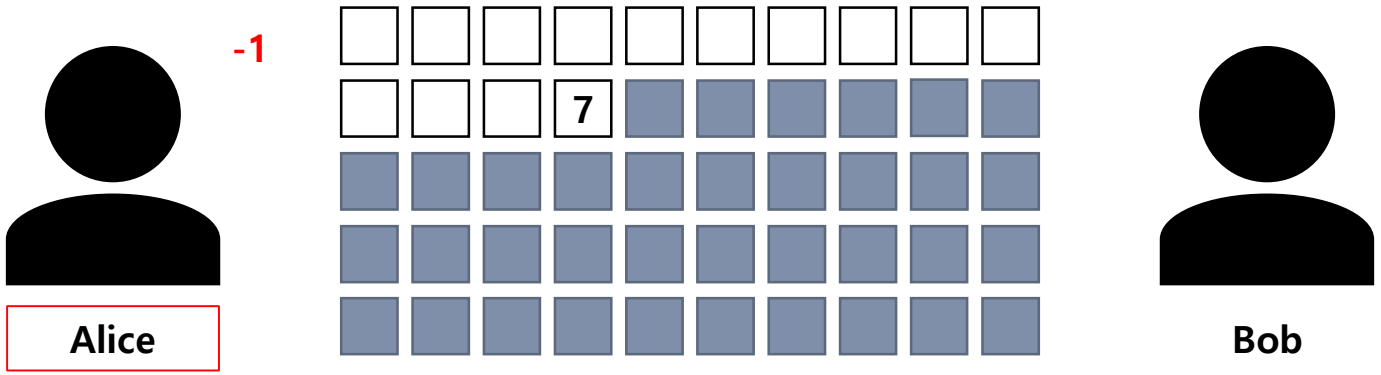
```
randomscores <- function(studentnum) {
      score <- c()
      for (i in 1:studentnum) {
            score <- c(score, sample(0:100, 1))
      }
      return(score)
}

passdata <- function(score, criteria) {
      passes <- c()
      for (s in score) {
            if (s >= criteria) {
                  passes <- c(passes, "P")
            }
            else {
                  passes <- c(passes, "F")
            }
      }
      return(passes)
}

# 3 Scores and Results of 30 students
no <- seq(1:30)
kor <- randomscores(length(no))
eng <- randomscores(length(no))
mat <- randomscores(length(no))
matpass <- passdata(mat, 60)

classdata <- data.frame("NO"=no, "KOR"=kor, "ENG"=eng, "MAT"=mat, "MAT_PASS"=matpass)
```

```
> classdata
   NO KOR ENG MAT MAT_PASS
1   1  90  75  73        P
2   2  49  96  53        F
3   3  79  50   9        F
4   4   5  59  87        P
5   5  34  11  38        F
6   6  56  17  18        F
7   7  47  19   6        F
8   8  56  36  93        P
9   9  59  64  85        P
10 10 100   8  88        P
11 11  65  26  19        F
12 12  45  34  95        P
13 13  59  55  65        P
14 14  84   9  83        P
15 15  82  57   4        F
16 16  94  90   7        F
17 17  95  21  15        F
18 18  14  32  62        P
19 19  45  61  92        P
20 20  70  39  99        P
21 21  24  55  95        P
22 22   9  99  85        P
23 23  28  95  39        F
24 24  76  99  76        P
25 25  32  91  29        F
26 26  99   0  69        P
27 27  60  23  31        F
28 28  23  73   3        F
29 29  27  71  11        F
30 30  87  96  70        P
```

- 데이터프레임 생성
  - 여러 데이터들을 생성하여 하나의 데이터셋으로 구성

과제7

- 카드 뒤집기 게임 구현을 통한 결과 도출
  - Alice와 Bob이 1~50 숫자가 적힌 카드로 뒤집기 게임을 진행한다.
  - 카드를 섞고 무작위로 카드를 뽑아서 다음과 같이 점수를 받는다.
    - 게임은 Alice부터 시작하며, 각자의 차례마다 하나의 카드를 뽑는다.
    - 둘은 서로 다른 규칙을 통해 점수를 얻는다.
    - Alice: 자신 차례에 뽑은 카드가 짝수인 경우 +2점, 홀수인 경우 -1점
    - Bob: 자신 차례에 뽑은 카드가 짝수인 경우 +0점, 홀수인 경우 +1점
  - 50개의 카드를 모두 뽑으면, 게임이 종료된다.
  - 위 카드 게임을 구현한 뒤, 총 10번 반복하여 Alice가 총 10번의 게임 동안 얻은 평균 점수가 얼마나 되는지 계산하여 출력한다.



과제 제출

- 제출 목록
  - 작성한 코드 파일(.R)
  - 결과를 출력한 터미널 캡처(이미지)를 Word 혹은 HWP에 붙여 넣어 PDF 파일로 변환
- 제출 방법
  - 목록의 파일들을 압축
  - 아래 서식으로 압축파일 이름 지정
  - 이메일 제목 또한 지정
- 제출 서식 (X=과제번호)
  - 파일 이름: 통계학실습_과제X_학번_이름.zip
  - 이메일 제목: 통계학실습_과제X_학번_이름
- 제출 이메일
  - gtsk623@gmail.com