

데이터 입출력

학습 내용

R에서 제공하는 함수와 패키지를 이용하여 데이터를 처리하기 위해서는 필요한 데이터를 생성하거나 파일에 보관된 데이터를 불러와야 한다.

이 장에서는 간단한 데이터를 키보드로 입력하여 생성하는 방법과 다양한 양식으로 파일에 보관된 데이터를 불러오는 방법을 알아본다. 또한, R 스크립트로 처리된 결과를 콘솔에 출력하거나 파일에 저장하는 방법에 대해서도 알아본다.

학습 목표

- 키보드로 10개 이하의 데이터를 입력받아서 벡터 객체를 생성할 수 있다.
- 텍스트 파일과 엑셀 파일 계열의 파일을 R 스크립트로 불러올 수 있다.
- 파일에 저장된 데이터를 sep와 header 속성을 지정하여 불러올 수 있다.
- 변수에 저장된 데이터를 테이블 형식으로 파일에 저장할 수 있다.

Chapter 03의 구성

1. 데이터 불러오기
2. 데이터 저장하기

1. 데이터 불러오기

데이터 분석을 위해서는 가장 먼저 분석에 필요한 데이터들을 준비해야 한다. 간단한 데이터는 키보드로 입력해서 생성할 수 있지만, 방대한 데이터는 파일 또는 웹사이트로부터 가져올 수 있다. 또한, 실시간으로 분산된 데이터를 수집(data crawling)하기 위해서는 별도의 시스템이 준비되어 있어야 한다.

1.1 키보드 입력

키보드로 직접 데이터를 입력받는 방법에는 scan() 함수를 이용하는 방법과 edit() 함수를 이용하는 방법이 있다.

(1) scan() 함수 이용

실습 키보드로 숫자 입력하기

```
> num <- scan() # 키보드로부터 숫자 입력하기
1: 1
2: 2
3: 3
4: 4
5: 5
6:
Read 5 items
> num # 입력된 벡터 원소 보기
[1] 1 2 3 4 5
> sum(num) # 입력된 자료의 합계 구하기
[1] 15
>
```

해설 scan() 함수를 실행하면 콘솔 창에 입력 데이터의 순서를 나타내는 프롬프트(④: "1: ")가 표시된다. 프롬프트에서 임의의 숫자를 입력하고 (Enter) 키를 누른다. 더는 입력할 데이터가 없을 때는 그냥 (Enter) 키를 누르면 입력된 데이터의 개수를 표시한다. scan() 함수를 변수 num에 할당했기 때문에 입력된 데이터는 입력된 순서대로 벡터 변수 num에 저장된다.

실습 키보드로 문자 입력하기

```
> name <- scan(what = character())
1: 홍길동
2: 이순신
3: 강감찬
4: 유관순
5:
Read 4 items
> name
[1] "홍길동" "이순신" "강감찬" "유관순"
>
```

해설 키보드로 문자를 입력하기 위해서는 scan() 함수에서 'what = character()' 인수를 사용하여 콘솔 창에서 임의의 문자를 입력할 수 있다. scan() 함수를 변수 name에 할당했기 때문에 순서대로 입력한 문자는 name 벡터에 저장된다.

(2) edit() 함수를 이용한 입력

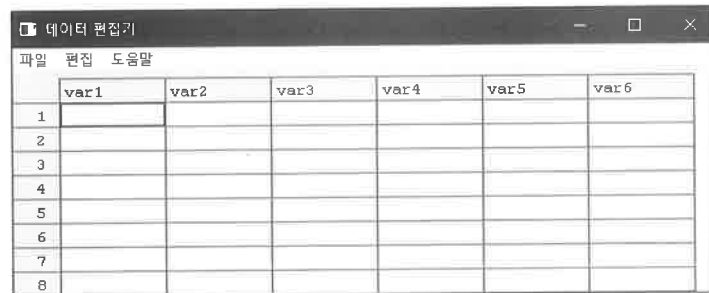
edit() 함수를 이용하면 데이터 입력을 돕기 위해 표 형식의 데이터 편집기를 제공한다.

실습 편집기를 이용한 데이터프레임 만들기

```
> df = data.frame()
> df = edit(df)
> df
```

	학번	이름	국어	영어	수학
1	1	홍길동	80	80	80
2	2	이순신	95	90	95
3	3	강감찬	95	95	100
4	4	유관순	85	85	85
5	5	김유신	95	90	95

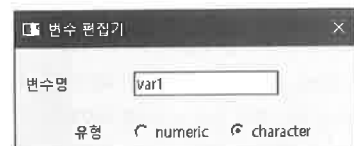
해설 코드를 행 단위로 실행하여, 두 번째 행을 실행하면 다음 그림과 같은 데이터 편집기 창이 열린다.



데이터 편집기 창은 '파일', '편집', '도움말' 메뉴를 가진다. 표의 열 제목은 var1, var2, var3, var4, var5, var6이다.

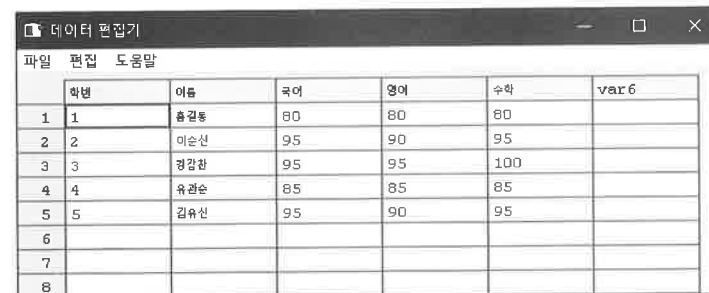
	var1	var2	var3	var4	var5	var6
1						
2						
3						
4						
5						
6						
7						
8						

데이터 편집기 창에서 데이터를 입력한다. 열 제목은 해당 열 제목을 클릭하면 다음 그림과 같은 변수 편집기 창이 열린다. 변수명을 입력하고 변수 편집기 창을 닫으면 입력된 변수명이 데이터 편집기 창에 반영된다.



변수 편집기 창은 '변수명' 입력란과 '유형' 선택(numeric, character)을 가진다. 현재 'var1'이 입력되어 있다.

데이터 편집기 창에서 다음 그림에서처럼 데이터를 입력한다. 데이터를 모두 입력했으면 데이터 편집기 창을 닫거나 데이터 편집기 창의 [파일]-[닫기] 메뉴를 선택하면 입력된 데이터가 데이터프레임에 전달된다.



데이터 편집기 창은 '파일', '편집', '도움말' 메뉴를 가진다. 표의 열 제목은 학번, 이름, 국어, 영어, 수학, var6이다.

	학번	이름	국어	영어	수학	var6
1	1	홍길동	80	80	80	
2	2	이순신	95	90	95	
3	3	강감찬	95	95	100	
4	4	유관순	85	85	85	
5	5	김유신	95	90	95	
6						
7						
8						

1.2 로컬 파일 가져오기

파일에 저장된 데이터를 불러오기 위해서 R에서는 다양한 함수를 제공하고 있다. 사용자는 파일에 저장된 데이터의 형태에 따라서 해당 함수를 이용하여 파일의 자료를 가져올 수 있다.

(1) read.table() 함수 이용

테이블(칼럼이 모여서 레코드 구성) 형태로 작성되어 있으며, 칼럼이 공백, 탭, 콜론(:), 세미콜론(;), 콤마(,) 등으로 구분된 자료 파일을 불러올 수 있는 함수이다.

만약 구분자가 공백이거나 탭이면 sep 속성을 생략할 수 있다. 또한, 칼럼명이 있는 경우 header 속성을 'header = TRUE'로 지정한다.

명식 read.table(file = "경로명/파일명", sep = "칼럼구분자", header = "T|F")

실습 칼럼명이 없는 파일 불러오기

```
> getwd() # 현재 작업 디렉터리의 경로 확인
[1] "C:/Rwork/Part-I"
> setwd("C:/Rwork/Part-I/") # 작업 디렉터리 설정
> student <- read.table(file = "student.txt") # 테이블 형식의 파일 읽기
> student
```

	V1	V2	V3	V4
1	101	hong	175	65
2	201	lee	185	85
3	301	kim	173	60
4	401	park	180	70

```
> names(student) <- c("번호", "이름", "키", "몸무게") # 칼럼명 변경
> student # 변경된 칼럼명이 적용된 데이터 출력
```

	번호	이름	키	몸무게
1	101	hong	175	65
2	201	lee	185	85
3	301	kim	173	60
4	401	park	180	70

해설 데이터 파일에 칼럼명이 없는 경우에는 'V1', 'V2', 'V3', 'V4' 형태로 기본 칼럼명이 지정된다. names() 함수를 이용하여 사용자가 원하는 칼럼명을 지정한다.

실습 칼럼명이 있는 파일 불러오기

```
> student1 <- read.table(file = "student1.txt", header = T)
> student1
```

	번호	이름	키	몸무게
1	101	hong	175	65
2	201	lee	185	85

```
3 301 kim 173 60
4 401 park 180 70
>
```

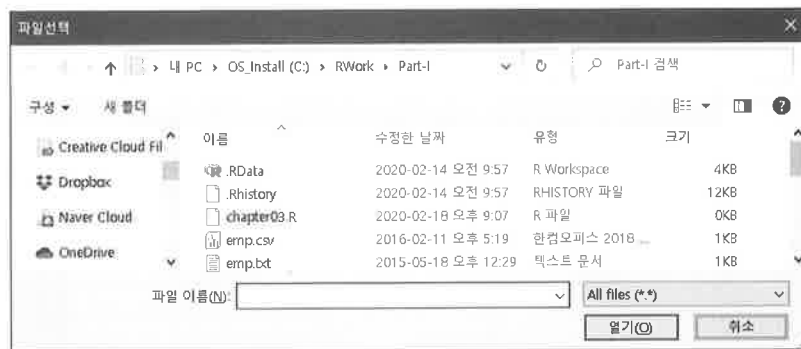
해설 칼럼명 있는 파일 데이터를 불러오기 위해서는 'header = TRUE' 또는 'header = T' 속성을 이용한다.

실습 탐색기를 통해서 파일 선택하기

탐색기를 통해서 불러올 파일을 선택하기 위해서는 file 속성 대신에 file.choose() 함수를 사용한다.

```
> student1 <- read.table(file.choose(), header = TRUE)
>
```

해설 file.choose() 함수를 이용하면 파일을 선택할 수 있도록 다음 그림처럼 파일선택 대화상자가 열린다. 파일선택 대화상자에서 불러올 파일을 선택하여 선택된 파일의 데이터를 가져올 수 있다.



실습 구분자가 있는 경우

칼럼의 구분이 세미콜론으로 구성된 경우는 sep = ";" 형식으로 지정하고, 칼럼의 구분이 탭으로 구성된 경우는 sep = "\t" 형식으로 sep 속성의 값을 지정하여 파일을 불러온다.

```
> student2 <- read.table(file = "student2.txt", sep = ";", header = TRUE)
```

```
> student2 <- read.table(file = "student2.txt", sep = "\t", header = TRUE)
```

해설 read.table() 함수 이용하여 파일을 불러오는 경우 칼럼의 구분자는 sep 속성으로 지정한다.

실습 결측치를 처리하여 파일 불러오기

파일에 특정 문자열을 NA로 처리하여 파일을 불러올 수 있다.

```
> student3 <- read.table(file = "student3.txt", header = TRUE, na.strings = "-")
> student3
  번호 이름 키 몸무게
1 101 hong 175 65
2 201 lee 185 85
3 301 kim 173 NA
4 401 park NA 70
>
```

← 결측치를 NA로 처리

해설 "student3.txt" 파일의 데이터 중 '-' 문자를 결측치로 처리하기 위해서 na.strings = "-" 속성을 사용하면 '-' 문자를 NA로 변경하여 파일의 데이터를 불러올 수 있다.

(2) read.csv() 함수 이용

엑셀(Excel)에서는 작업한 파일을 R에서 처리할 수 있도록 CSV 형식으로 변환하여 저장할 수 있다. CSV(Comma Separated Value) 파일 형식은 콤마(,)를 기준으로 각 칼럼을 구분하여 저장한 데이터 형식을 말한다.

read.csv() 함수는 구분자인 ","가 sep의 기본값이며, "header = TRUE"가 기본값이다. 따라서 칼럼명이 있는 경우에 header 속성은 생략할 수 있다. 형식에서 [] 표시는 생략 가능한 속성을 의미한다.

형식 read.csv(file = "경로명/파일명" [, sep = ","] [, header = TRUE])

실습 CSV 파일 형식 불러오기

```
> # NA 처리를 위해 na.strings 속성 사용
> student4 <- read.csv(file = "student4.txt", sep = ",", na.strings = "-")
> student4
  번호 이름 키 몸무게
1 101 hong 175 65
2 201 lee 185 85
3 301 kim 173 NA
4 401 park NA 70
>
```

해설 CSV 파일 형식은 콤마(,)가 칼럼의 구분자이기 때문에 sep 속성은 생략할 수 있다. file.choose() 속성을 사용하면 파일선택 대화상자를 통해 읽을 파일을 선택할 수도 있다.

(3) read.excel() 함수 이용

엑셀 파일(*.xlsx)을 직접 R에서 불러올 수 있는 함수이다. 이 함수는 기본 함수가 아니므로 먼저 'readxl' 패키지를 설치해야 한다. 엑셀 파일은 일반 파일과는 다르게 시트(sheet) 단위로 자료를 저장한다. 따라서 엑셀 파일을 읽을 때는 시트명과 셀(cell)의 범위를 지정해서 읽는다.

실습 readxl 패키지 설치와 로드

```
> install.packages("readxl") # xlsx 패키지 설치
... 생략 ...
> library(readxl)
>
```

해설 엑셀 파일을 읽기 위해서 "readxl" 패키지를 설치하고 메모리로 로드한다. 해당 패키지에서는 엑셀 파일을 읽어올 수 있는 read_excel() 함수를 제공한다.

실습 엑셀 파일 읽어오기

read_excel() 함수를 이용하여 엑셀 파일의 자료를 읽어올 수 있다. read_excel() 함수의 형식은 다음과 같다.

형식 read_excel(path, sheet, col_names, range, na, skip, ...)

read_excel() 함수의 주요 속성은 다음과 같다.

- **path**: xls 또는xlsx 파일의 경로와 파일명
- **sheet**: xlsx의 시트명
- **col_names**: 첫 줄에 칼럼명이 있는 경우 TRUE
- **range**: xlsx의 시트에서 읽어올 셀 범위
- **na**: 빈 셀을 누락된 데이터로 보고 결측치 처리
- **skip**: 읽기전에 건너될 최소 행의 수를 지정

```
> setwd("C:/Rwork/Part-1") # 읽어올 엑셀 파일의 경로
> st_excel <- read_excel(path = "studentexcel.xlsx", sheet = "student") # 엑셀 파일 읽기
> st_excel
# A tibble: 5 x 4
  학번   이름   성적   평가
<dbl> <chr> <dbl> <chr>
1 101 홍길동   80    B
2 102 이순신   95   A+
3 103 강감찬   78   C+
4 104 유관순   85   B+
5 105 김유신   65   D+
> str(st_excel)
Classes 'tbl_df', 'tbl' and 'data.frame':
5 obs. of 4 variables:
 $ 학번: num 101 102 103 104 105
 $ 이름: chr "홍길동" "이순신" "강감찬" "유관순" ...
 $ 성적: num 80 95 78 85 65
 $ 평가: chr "B" "A+" "C+" "B+" ...
>
```

해설 read_excel() 함수의 첫 번째 속성인 'path = "studentexcel.xlsx"'는 읽어올 엑셀 파일을 지정하고, 두 번째 속성인 'sheet = "student"'는 읽어올 엑셀 파일에 포함된 시트 탭의 이름을 지정한다. 따라서 "C:/Rwork/Part-1/studentexcel.xlsx" 엑셀 파일 내의 "student" 시트의 자료를 읽어 온다.

1.3 인터넷에서 파일 가져오기

인터넷에서 제공하는 CSV 파일 형식의 데이터를 해당 사이트에서 직접 R 스크립트로 가져와서 데이터를 가공 처리한 후 분석에 활용할 수 있다.

깃허브 사이트(<https://vincentarelbundock.github.io>)에서 제공하는 CSV 파일 자료를 가져와 시각화하기까지의 과정을 실습으로 살펴본다.

실습 인터넷에서 파일을 가져와 시각화하기

> 단계 1: 깃허브에서 URL을 사용하여 타이타닉(titanic) 자료 가져오기

```
> titanic <-
+ read.csv("https://vincentarelbundock.github.io/Rdatasets/csv/COUNT/titanic.csv")
> titanic
   X   class   age   sex survived
1  1 1st class adults  man      yes
2  2 1st class adults  man      yes
3  3 1st class adults  man      yes
... 중간 생략 ...
199 199 1st class adults women      yes
200 200 1st class adults women      yes
[ reached 'max' / getOption("max.print") -- omitted 1116 rows ]
>
```

해설 read.csv() 함수의 인수로 깃허브 사이트에서 가져올 CSV 파일의 URL을 전달하여 함수를 실행한다. 타이타닉(titanic) 자료가 저장된 변수를 실행하면 5개의 칼럼으로 구성된 csv 파일이 내용을 확인할 수 있다.

> 단계 2: 인터넷(깃허브)에서 가져온 자료의 차원 정보와 자료구조 보기 및 범주의 빈도수 확인

```
> dim(titanic) # 1316(행) 5(열)
[1] 1316 5
> str(titanic)
'data.frame': 1316 obs. of 5 variables:
 $ X      : int 1 2 3 4 5 6 7 8 9 10 ...
 $ class  : chr "1st class" "1st class" "1st class" "1st class" ...
 $ age    : chr "adults" "adults" "adults" "adults" ...
 $ sex    : chr "man" "man" "man" "man" ...
 $ survived: chr "yes" "yes" "yes" "yes" ...
>
> table(titanic$age)
adults child
1207 109
> table(titanic$sex)
man women
869 447
> table(titanic$survived)
no yes
817 499
>
```

해설 타이타닉 자료는 1,316개의 관측치와 5개 변수의 칼럼으로 구성된다. 5개 변수의 칼럼을 살펴보면 X는 정수형(int)이고, 나머지 4개는 문자형(chr)이다. 문자형은 일반적으로 일정한 범주(category)를 갖는다. 나이(age)는 성인(adults)과 어린이(child)의 범주를 가지고, 성별(sex)은 남자(man)와 여자(women)의 범주를 가지며, 생존 여부(survived)는 사망(no), 생존(yes)의 범주를 갖는다. 이러한 범주형 변수는 교차 분할표를 토대로 교차분석에 이용될 수 있다.

더 알아보기 데이터프레임에서 문자형 칼럼

R 3.x 버전에서 데이터프레임의 칼럼으로 사용되는 자료형이 문자형(chr)인 경우 요인형(Factor)으로 자동 변환되지만, R 4.0 버전에서는 문자형을 그대로 유지한다.

• R 3.x 버전에서 titanic 자료구조 예

```
> str(titanic)
'data.frame': 1316 obs. of 5 variables:
 $ X      : int 1 2 3 4 5 6 7 8 9 10 ...
 $ class  : Factor w/ 3 levels "1st class", "2nd class"...: 1 1 1 1 1 1 1 1 1 1 ...
 $ age    : Factor w/ 2 levels "adults", "child": 1 1 1 1 1 1 1 1 1 1 ...
 $ sex    : Factor w/ 2 levels "man", "woman": 1 1 1 1 1 1 1 1 1 1 ...
 $ survived: Factor w/ 2 levels "no", "yes": 2 2 2 2 2 2 2 2 2 2 ...
>
```

• R 4.0 버전에서 titanic 자료구조 예

```
> str(titanic)
'data.frame': 1316 obs. of 5 variables:
 $ X      : int 1 2 3 4 5 6 7 8 9 10 ...
 $ class  : chr "1st class" "1st class" "1st class" "1st class" ...
 $ age    : chr "adults" "adults" "adults" "adults" ...
 $ sex    : chr "man" "man" "man" "man" ...
 $ survived: chr "yes" "yes" "yes" "yes" ...
>
```

> 단계 3: 관측치 살펴보기

```
> head(titanic)
  X   class age sex survived
1 1 1st class adults man      yes
2 2 1st class adults man      yes
3 3 1st class adults man      yes
4 4 1st class adults man      yes
5 5 1st class adults man      yes
6 6 1st class adults man      yes
> tail(titanic)
      X   class age sex survived
1311 1311 3rd class child women      no
1312 1312 3rd class child women      no
1313 1313 3rd class child women      no
1314 1314 3rd class child women      no
1315 1315 3rd class child women      no
1316 1316 3rd class child women      no
>
```

해설 1,000개 이상의 관측치를 갖는 자료를 한꺼번에 확인할 수 없어서 head() 함수를 이용하여 관측치의 앞부분 6개와 tail() 함수를 사용하여 관측치의 뒷부분 6개를 대략적으로 확인하여 자료의 특징을 살펴본다.

사이트에서 가져온 자료를 대상으로 두 개의 범주형 변수를 이용하여 교차 분할표를 작성하고, 이를 시각화하는 방법에 대해서 알아본다.

> 단계 4: 교차 분할표 작성하기

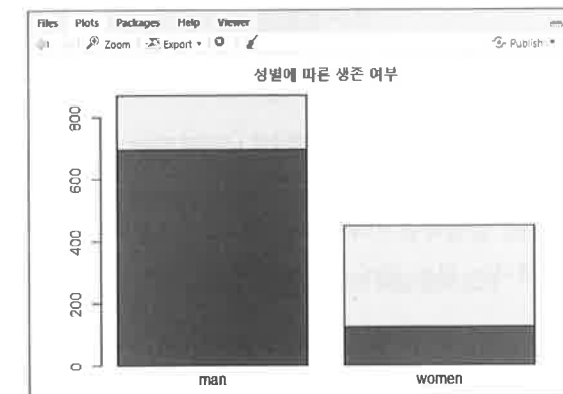
```
> tab <- table(titanic$survived, titanic$sex) # 성별에 따른 생존 여부
> tab
      man women
no  694    123
yes 175    324
>
```

해설 성별에 따른 생존 여부를 분석하기 위해 교차 분할표를 작성한다. table() 함수에서 첫 번째 인수는 교차 분할표에서 행으로 나타나고, 두 번째 인수는 열로 나타난다. 남자(man)를 기준으로 생존은 175명, 사망은 694명이며, 여자(women)는 생존이 324명, 사망은 123명으로 집계된다. 따라서 교차 분할표를 통해서 분석된 결과는 남자의 생존율(20%)에 비해서 여자의 생존율(72%)이 현저하게 높게 나타난 것으로 분석된다.

- 남자의 생존율: $175 / (694 + 175) = 0.2013809$
- 여자의 생존율: $324 / (324 + 123) = 0.7248322$

> 단계 5: 범주의 시각화 - 막대 차트 그리기

```
> barplot(tab, col = rainbow(2), main = "성별에 따른 생존 여부") # 막대 차트
>
```



해설 교차 분할표를 통해서 분석된 결과이지만, 막대 차트를 통해서 성별에 따른 생존 여부를 시각화할 경우 변수의 특성을 분석하는데 훨씬 더 가독성이 높은 것으로 나타난다. 왼쪽의 남자(man)는 오른쪽의 여자(women)에 비해서 사망률이 현저하게 높게 나타난다. 차트에 관한 자세한 내용은 "5장 데이터 시각화"에서 알아본다.

2. 데이터 저장하기

데이터를 처리한 결과를 콘솔에 출력하거나 처리가 완료된 결과물을 특정 파일에 영구적으로 저장하는 방법에 대해서 알아본다.

2.1 화면(콘솔) 출력

처리결과가 저장된 변수를 화면(콘솔)에 출력하는 R 함수는 cat() 함수와 print() 함수가 있다.

실습 cat() 함수 이용 변수 출력하기

cat() 함수는 출력할 문자열과 변수를 함께 결합하여 콘솔에 출력해 준다.

```
> x <- 10
> y <- 20
> z <- x * y
> cat("x * y의 결과는 ", z, "입니다.\n") # "\n"은 줄 바꿈을 위한 제어문자
x * y의 결과는 200 입니다.
> cat("x * y = ", z)
x * y = 200
>
```

해설 cat() 함수를 이용하여 문자열과 변수의 값을 출력하는 예제이다.

실습 print() 함수 이용 변수 출력하기

print() 함수는 변수 또는 수식만을 대상으로 콘솔에 출력해 준다.

```
> print(z) # 변수 또는 수식만 출력 가능
[1] 200
>
```

해설 print() 함수는 문자열을 함께 사용할 수 없다. 변수의 값 또는 수식의 결과만을 출력할 수 있다.

2.2 파일 저장

R 스크립트를 이용하여 처리된 결과를 특정 파일로 저장하는 방법에 대해서 알아본다.

(1) sink() 함수 이용

결과를 저장할 경로와 파일명을 지정하여 sink() 함수를 실행하면 이후에 작업한 모든 내용이 지정된 파일에 저장된다. sink() 함수의 기능을 종료하기 위해서는 인수 없이 sink() 함수를 한 번 더 실행하면 된다.

실습 sink() 함수를 사용한 파일 저장

```
> setwd("C:/Rwork/Part-I") # 작업 디렉터리 설정
> library(RSADBE) # 패키지를 메모리에 로드
> data(Severity_Counts) # Severity_Counts 데이터 셋 가져오기
> sink("severity.txt") # 저장할 파일 Open
> severity <- Severity_Counts # 데이터 셋을 변수에 저장
> severity # 변수 출력: 콘솔에 출력되지 않고, 파일에 저장
> sink() # Open된 파일 Close
>
```

해설 sink() 함수에 의해서 파일에 저장된 결과는 setwd() 함수에 의해 지정된 경로("C:/Rwork/Part-I")에 sink() 함수에 지정한 파일("severity.txt")에 저장된다. 메모장을 이용하여 텍스트 파일을 열면 Severity_Counts 데이터 셋의 내

용이 저장된 것을 확인할 수 있다. Severity_Counts 데이터 셋의 상세한 내용은 "2장 데이터의 유형과 구조"에서 Severity_Counts 데이터 셋에 관한 설명을 참고하자

Bugs.BR	Bugs.AR	NT.Bugs.BR	NT.Bugs.AR	Major.BR	Major.AR	Critical.BR	Critical.AR	H.Priority.BR	H.Priority.AR
11605	374	10119	17						
1135	35	432	10						
459	3								

(2) write.table() 함수 이용

write.table() 함수는 처리된 결과(변수)를 테이블 형식으로 파일에 저장할 수 있는 함수이다. 행 번호를 제거하는 'row.names' 속성과 따옴표를 제거하는 'quote' 속성을 사용할 수 있다.

실습 write.table() 함수를 이용한 파일 저장하기

```
> # 단계 1: titanic 자료 확인
> titanic # titanic 데이터 생략...

> # 단계 2: 파일 저장 위치 지정
> setwd("C:/Rwork/output") # 작업 디렉터리 지정

> # 단계 3: titanic.txt 파일에 저장
> write.table(titanic, "titanic.txt", row.names = FALSE)
>
```

해설 write.table() 함수를 이용하여 titanic 데이터프레임을 테이블 형식으로 파일에 저장한다. 'row.names = F'는 행의 이름을 제거한다. 파일 생성 결과는 파일 탐색기를 이용하여 생성된 파일을 확인하여 다음 실습 예의 결과와 같은지 확인한다.

실습 write.table() 함수로 저장한 파일 불러오기

write.table() 함수에 의해서 저장된 데이터는 다음과 같이 read.table() 함수를 이용하여 텍스트 파일을 불러올 수 있다.

```
> titanic_df <- read.table(file = "titanic.txt", sep = ",", header = T)
> titanic_df
  X  class  age  sex  survived
1  1 1st class adults  man    yes
2  2 1st class adults  man    yes
3  3 1st class adults  man    yes
4  4 1st class adults  man    yes
5  5 1st class adults  man    yes
... 중간 생략 ...
198 198 1st class adults  women    yes
```

```
199 199 1st class adults women yes
200 200 1st class adults women yes
[ reached 'max' / getOption("max.print") -- omitted 1116 rows ]
>
```

해설 write.table() 함수에 의해서 파일에 저장된 데이터는 read.table() 함수를 이용하여 데이터프레임 형식으로 가져올 수 있다. 예제를 실행한 결과가 앞 예제에서 확인한 titanic 데이터프레임의 값과 같은지 확인해 보자.

(3) write.csv() 함수 이용

write.csv() 함수는 데이터프레임 형식의 데이터를 CSV 형식으로 파일에 저장한다.

실습 write.csv() 함수를 이용한 파일 저장하기

```
> setwd("C:/Rwork/Part-I")
> st.df <- st_excel # 이전 예에서 생성된 데이터프레임
  학번  이름 성적 평가
1 101 홍길동 80 B
2 102 이순신 95 A+
3 103 강감찬 78 C+
4 104 유관순 85 B+
5 105 김유신 65 D+
> # 행 번호와 따옴표 제거하여 stdf.csv 파일로 저장
> write.csv(st.df, "stdf.csv", row.names = F, quote = F)
>
```

해설 R에서 처리한 데이터프레임을 write.csv() 함수를 이용하여 CSV 형식으로 파일에 저장한다.

더 알아보기 엑셀에서 텍스트 파일 대상 csv 파일 만들기

메모장과 같은 프로그램을 이용하여 작성된 텍스트(text) 파일을 엑셀의 CSV 파일로 만드는 절차는 다음과 같다.

- [단계 1] 엑셀에서 "text.txt" 파일 열기
- [단계 2] 텍스트 마법사 3단계 중 1단계: 구분 기호로 분리됨
- [단계 3] 텍스트 마법사 3단계 중 2단계: 구분기호 -> 탭
- [단계 4] 텍스트 마법사 3단계 중 3단계: 일반 -> [마침]
- [단계 5] 첫 행에 변수(칼럼)명 추가: A B C D E

(4) write.xlsx() 함수 이용

R 스크립트에서 처리된 결과(변수)를 엑셀 파일로 저장할 수 있다. 엑셀 파일 형식으로 저장하기 위해서는 "writexl" 패키지를 설치해야 한다.

실습 writexl 패키지 설치와 로드

```
> install.packages("writexl") # writexl 패키지 설치
*** 생각 ***
> library(writexl) # writexl 패키지 로드
>
```

해설 처리된 결과를 엑셀 파일로 저장하기 위해서 "writexl" 패키지를 설치하고 메모리로 로드한다. 해당 패키지에서는 엑셀 파일로 저장할 수 있는 write.xlsx() 함수를 제공한다.

실습 write.xlsx() 함수 이용 엑셀 파일 저장하기

write.xlsx() 함수를 이용하여 엑셀 파일로 자료를 저장할 수 있다. write.xlsx() 함수의 형식은 다음과 같다.

형식 write.xlsx(x, path, col_names = TRUE, format_headers = TRUE ...)

write.xlsx() 함수의 주요 속성은 다음과 같다.

- x: xlsx의 시트가 될 데이터프레임
- path: 파일 경로와 파일명
- col_names = TRUE: file 첫줄에 칼럼명 표시
- format_headers = TRUE: 굵은 글씨체와 가운데 정렬로 xlsx 파일에서 칼럼명 표시

```
> setwd("C:/Rwork/output") # 엑셀 파일을 저장할 경로
> write.xlsx(x = st.df, path = "st_excel.xlsx", col_names = TRUE) # 파일 저장
>
```

해설 R에서 처리된 결과가 있는 st.df 변수의 자료를 "st_excel.xlsx" 엑셀 파일로 저장한다. "col_names = TRUE" 속성에 의해서 엑셀 파일의 첫 줄에 칼럼명이 포함되어 저장된다. 생성된 파일은 "C:/Rwork/output" 폴더에서 확인할 수 있다.

1. 본문에서 작성한 titanic 변수를 다음과 같은 단계를 통해서 "titanic.csv" 파일로 저장한 후 파일을 불러오시오.

[단계 1] "C:/Rwork/output" 폴더에 "titanic.csv"로 저장한다.

● 힌트: write.csv() 함수 사용

[단계 2] "tanic.csv" 파일을 titanicData 변수로 가져와서 결과를 확인하고, titanicData의 관측치와 칼럼수를 확인한다.

● 힌트: str() 함수 사용

[단계 3] 1번, 3번 칼럼을 제외한 나머지 칼럼을 대상으로 상위 6개의 관측치를 확인한다.

● 힌트: write.csv() 함수 사용

```
class sex survived
1 1st class man      yes
2 1st class man      yes
3 1st class man      yes
4 1st class man      yes
5 1st class man      yes
6 1st class man      yes
>
```

2. R에서 제공하는 CO2 데이터 셋을 대상으로 다음과 같은 단계로 파일에 저장하시오.

[단계 1] Treatment 칼럼 값이 'nonchilled'인 경우 'CO2_df1.csv' 파일로 행 번호를 제외하고 저장한다.

[단계 2] Treatment 칼럼 값이 'chilled'인 경우 'CO2_df2.csv' 파일로 행 번호를 제외하고 저장한다.

학습 내용

대부분의 프로그래밍 언어는 조건문과 반복문과 같은 제어문을 이용하여 프로그래밍 로직(logic)을 작성할 수 있다.

R은 Java나 Python 프로그래밍 언어처럼 객체지향언어이다. 연산자와 제어문을 이용하여 프로그래밍할 수 있다.

이장에서는 연산자의 유형과 제어문의 구조에 대해서 살펴보고, 조건문과 반복문을 이용하여 함수를 정의하는 방법과 R의 주요 내장함수에 대해서 영역별로 알아본다.

학습 목표

- if() 함수를 이용하여 논리적인 true와 false 값을 구분하여 출력할 수 있다.
- for() 함수를 이용하여 조건에 만족하는 동안 지정된 문장을 반복적으로 수행할 수 있도록 프로그램을 작성할 수 있다.
- 제어문을 이용하여 매개변수를 갖는 사용자 정의함수를 생성할 수 있다.
- 기초 통계량을 구하는 R의 내장함수를 이용하여 벡터 객체의 통계량을 구할 수 있다.

Chapter 04의 구성

1. 연산자
2. 조건문
3. 반복문
4. 함수 정의
5. 주요 내장함수

1. 연산자

R에서 제공되는 연산자(operator)는 산술연산자와 관계연산자 그리고 논리연산자를 제공한다. [표 4.1]은 각 연산자에 대한 연산 기호와 기능 설명이다.

[표 4.1] R의 연산자

구분	연산자	기능 설명
산술연산자	+, -, *, /, %%, ^, **	사칙연산, 나머지 계산, 제곱 계산
관계연산자	==, !=, >, <, >=, <=	동등비교, 크기 비교
논리연산자	&, , !, xor()	논리곱, 논리합, 부정, 배타적 논리합

1.1 산술연산자

산술연산자는 덧셈과 뺄셈, 곱셈 나눗셈 기능의 사칙연산자(+, -, *, /)와 나머지를 계산하는 연산자(%%) 그리고 거듭제곱(^ 또는 **)을 계산하는 연산자로 구성된다.

실습 산술연산자 사용

```
> num1 <- 100          # 피연산자 1
> num2 <- 20           # 피연산자 2
> result <- num1 + num2 # 덧셈
> result
[1] 120
> result <- num1 - num2 # 뺄셈
> result
[1] 80
> result <- num1 * num2 # 곱셈
> result
[1] 2000
> result <- num1 / num2 # 나눗셈
> result
[1] 5
>
> result <- num1 %% num2 # 나머지 계산
> result
[1] 0
>
> result <- num1 ^ 2     # 제곱 계산(num1 ** 2)
> result
[1] 10000
> result <- num1 ^ num2 # 100의 20승(10020)
> result
[1] 1e+40
```

해설 마지막 행의 출력 결과인 "1e+40"은 " 1×10^{40} "으로 " 1×10^40 "과 같은 결과를 나타낸다. 즉, 1 뒤에 40개의 0이 있다는 표현으로 아주 큰 값을 표기해야 하는 경우 이와 같은 지수 표현 형식으로 콘솔 창에 표시된다.

1.2 관계연산자

관계연산자를 이용한 관계식의 결과가 참이면 TRUE, 거짓이면 FALSE 값을 반환하는 연산자이다. 동등비교와 크기 비교로 분류할 수 있다.

실습 관계연산자 사용

```
> # (1) 동등비교
> boolean <- num1 == num2 # 두 변수의 값이 같은지 비교
> boolean
[1] FALSE
> boolean <- num1 != num2 # 두 변수의 값이 다른지 비교
> boolean
[1] TRUE
>
> # (2) 크기 비교
> boolean <- num1 > num2 # num1 값이 큰지 비교
> boolean
[1] TRUE
> boolean <- num1 >= num2 # num1 값이 크거나 같은지 비교
> boolean
[1] TRUE
> boolean <- num1 < num2 # num2 이 큰지 비교
> boolean
[1] FALSE
> boolean <- num1 <= num2 # num2 이 크거나 같은지 비교
> boolean
[1] FALSE
>
```

해설 관계연산자의 결과는 조건이 참이면 TRUE, 거짓이면 FALSE의 상수값을 반환한다.

1.3 논리연산자

산술연산자와 관계연산자를 이용한 논리식이 참이면 TRUE, 거짓이면 FALSE 값을 반환한다.

실습 논리연산자 사용

```
> logical <- num1 >= 50 & num2 <= 10 # 두 관계식이 같은지 판단
> logical
[1] FALSE
> logical <- num1 >= 50 | num2 <= 10 # 두 관계식 중 하나라도 같은지 판단
> logical
[1] TRUE
>
> logical <- num1 >= 50 # 관계식 판단
> logical
[1] TRUE
```

```
> logical <- !(num1 >= 50)      # 괄호 안의 관계식 판단 결과에 대한 부정
> logical                      # FALSE
[1] FALSE
>
> x <- TRUE; y <- FALSE
> xor(x, y)                    # [1] TRUE
[1] TRUE
>
```

해설 배타적 논리합을 연산하는 xor() 함수는 두 논리적인 값이 상반되는 경우 TRUE를 반환하고, 같으면 FALSE를 반환한다.

2. 조건문

R은 Java나 Python과 같은 객체지향 프로그래밍 언어처럼 조건문이나 반복문을 이용하여 논리적인 문장의 흐름을 표현할 수 있다. 조건문은 특정 조건에 따라서 실행되는 문이 결정되는 것으로 R에서는 if(), ifelse(), switch(), which() 함수를 이용하여 조건문을 작성할 수 있다.

2.1 if() 함수

if() 함수는 비교판단문을 작성할 수 있다. if() 함수의 인수로 조건식을 작성하고, 조건이 참일 때와 거짓일 때 각각 블록 단위로 처리할 문장을 작성한다. if() 함수의 형식은 다음과 같다.

형식 if(조건식){ 참인 경우 처리문 } else { 거짓인 경우 처리문 }

실습 if() 함수 사용하기

```
> x <- 50; y <- 4; z <- x * y
> if(x * y >= 40) {
+   cat("x * y의 결과는 40 이상입니다.\n") # '\n'은 줄바꿈
+   cat("x * y =", z)
+ } else {
+   cat("x * y의 결과는 40 미만입니다. x * y =", z, "\n")
+ }
x * y의 결과는 40 이상입니다.
x * y = 200
>
```

해설 "x * y"의 연산 결과가 40 이상인 경우와 그렇지 않은 경우를 if() 함수로 작성한 예제이다. {}를 사용하여 if와 else의 블록을 지정한다.

실습 if() 함수 사용으로 입력된 점수의 학점 구하기

```
> score <- scan()              # 점수 입력받기
1: 85
2:
```

```
Read 1 item
> score                        # 입력된 점수 확인
[1] 85
> result <- "노력"             # 결과 초기값 지정
> if(score >= 80) {            # 입력 점수가 80 이상이면
+   result <- "우수"           # 결과 변경
+ }
> cat("당신의 학점은", result, score) # 결과 확인
당신의 학점은 우수 85
>
```

해설 점수가 80 이상이면 "우수", 아니면 초기값인 "노력"을 출력하는 if() 함수 사용 예이다.

실습 if ~ else if 형식으로 학점 구하기

```
> score <- scan()
1: 90
2:
Read 1 item
> if(score >= 90) {
+   result = "A학점"
+ } else if(score >= 80) {
+   result = "B학점"
+ } else if(score >= 70) {
+   result = "C학점"
+ } else if(score >= 60) {
+   result = "D학점"
+ } else {
+   result = "F학점"
+ }
> cat("당신의 학점은", result) # 문자열과 변수의 값을 함께 출력
당신의 학점은 A학점
> print(result)                # print() 함수는 변수의 값 또는 수식의 결과만 출력
[1] "A학점"
>
```

해설 if() 함수의 첫 번째 조건이 참이면 90점 이상이고, else와 함께하는 if() 함수의 두 번째 조건이 참이면 80점 이상, 세 번째 조건이 참이면 70점 이상, 네 번째 조건이 참이면 60점 이상 그리고 모든 조건에 만족하지 않는다면 60점 미만을 의미한다.

2.2 ifelse() 함수

ifelse() 함수는 3항 연산자와 유사하다. 조건식이 참인 경우와 거짓인 경우 처리할 문장을 각각 작성한 후 조건식 결과에 따라서 처리문이 실행된다. ifelse() 함수의 형식은 다음과 같다.

형식 ifelse(조건식, 참인 경우 처리문, 거짓인 경우 처리문)

실습 ifelse() 함수 사용하기

```
> score <- scan()
1: 90
2:
Read 1 item
> ifelse(score >= 80, "우수", "노력") # "우수" 선택
[1] "우수"
> ifelse(score <= 80, "우수", "노력") # "노력" 선택
[1] "노력"
>
```

해설 ifelse() 함수는 첫 번째 인수인 조건식이 참이면 두 번째 인수 값을 선택하고, 거짓이면 세 번째 인수 값을 선택한다. 예에서는 입력된 값이 90으로 첫 번째 ifelse() 함수는 "우수"를 출력하고, 두 번째 ifelse() 함수는 "노력"을 출력한다.

실습 ifelse() 함수 응용하기

```
> excel <- read.csv("C:/Rwork/Part-I/excel.csv", header = T)
> q1 <- excel$q1 # q1 변수값 추출
> q1 # q1 변수값 확인
[1] 2 1 2 3 3 4 3 4 4 4 4 3 3 3 1 3 3 3 2 3 3 4 2 2 3 4 4 4 3 3 1
[32] 3 3 3 4 3 4 2 3 2 2 4 3 3 2 2 4 1 4 3 4 4 3 4 3 2 2 2 3 4 4
[63] 2 3 3 3 3 4 1 2 3 3 3 4 2 2 1 3 3 2 2 2 3 2 3 3 4 3 2 2 2 2 2
[94] 4 1 4 4 4 2 2 3 3 2 3 3 2 4 2 4 2 2 4 3 4 2 2 3 3 3 2 1 3 4 3
[125] 3 2 3 3 3 2 2 4 1 3 3 4 3 2 3 1 2 3 5 4 3 3 2 4 3 2 3 3 1 3 2
[156] 4 3 3 2 4 2 2 2 2 3 3 3 2 2 4 1 3 3 4 3 2 3 1 2 3 5 4 3 3 2 4
[187] 3 2 3 3 1 3 2 4 2 4 2 2 4 3 4 2 2 3 3 3 2 1 3 4 3 3 2 3 3 1 3
[218] 3 4 3 2 3 1 2 3 5 4 3 3 2 4 3 2 3 3 1 3 2 4 3 3 2 4 2 2 2 2 3
[249] 1 3 2 4 2 4 2 2 4 3 4 2 2 3 3 3 2 1 3 4 3 3 2 4 3 2 3 3 1 3 2
[280] 4 3 3 2 4 2 2 2 2 3 1 3 2 4 2 4 2 2 4 3 4 2 2 3 3 3 2 1 3 3 2
[311] 4 3 2 3 3 1 3 2 4 3 3 2 4 2 2 2 3 1 3 2 4 2 4 2 2 4 3 4 2 2
[342] 3 3 3 2 1 2 3 3 1 3 2 4 3 3 2 4 2 2 2 2 3 1 3 2 4 2 4 2 2 4 3
[373] 4 2 2 3 3 3 2 1 3 4 3 3 2 4 3 2 3 3 1 3 2 4 3 3 2 4 2 2 2 2
> ifelse(q1 >= 3, sqrt(q1), q1) # q1 값이 3보다 큰 경우 sqrt() 함수 적용
[1] 2.000000 1.000000 2.000000 1.732051 1.732051 2.000000 1.732051
[8] 2.000000 2.000000 2.000000 2.000000 1.732051 1.732051 1.732051
[15] 1.000000 1.732051 1.732051 1.732051 2.000000 1.732051 1.732051
[22] 2.000000 2.000000 2.000000 1.732051 2.000000 2.000000 2.000000
[29] 1.732051 1.732051 1.000000 1.732051 1.732051 1.732051 2.000000
... 이하 생략 ...
>
```

해설 read.csv() 함수를 이용하여 지정된 CSV 파일을 읽어, q1 칼럼의 값이 3 이상이면 칼럼값에 sqrt() 함수를 적용하여 출력하고, 그렇지 않으면 q1 칼럼값을 그대로 출력한다. 이처럼 ifelse() 함수는 벡터를 입력받아서 조건을 만족하는 값만 벡터로 출력할 수 있다.

실습 ifelse() 함수에서 논리연산자 사용하기

```
> ifelse(q1 >= 2 & q1 <= 4, q1 ^ 2, q1) # 1과 5만 출력, 나머지(2~4)는 제곱승 적용
[1] 4 1 4 9 9 16 9 16 16 16 16 9 9 9 1 9 9 9 4 9 9
[22] 16 4 4 9 16 16 16 9 9 1 9 9 9 16 9 16 4 9 4 4 16
[43] 9 9 4 4 16 1 16 9 16 16 9 16 9 4 4 4 4 9 16 16 4
... 중간 생략 ...
[358] 4 4 4 4 9 1 9 4 16 4 16 4 4 16 9 16 4 4 9 9 9
[379] 4 1 9 16 9 9 4 16 9 4 9 9 1 9 4 16 9 9 4 16 4
[400] 4 4 4
>
```

해설 q1 칼럼의 값이 2~4에 포함되면 q1 칼럼의 값에 제곱승을 적용하고, 그렇지 않으면 q1 칼럼의 값을 그대로 출력한다.

2.3 switch() 함수

switch() 함수는 비교 문장의 내용에 따라서 여러 개의 실행 문장 중 하나를 선택할 수 있도록 프로그램을 작성할 수 있다. if ~ else 문과는 다르게 값을 이용하여 실행 문장이 결정된다는 차이점이 있다. switch() 함수의 형식은 다음과 같다.

형식 switch(비교문, 실행문1[, 실행문2, 실행문3, ...])

switch 문의 형식에서 "비교문" 위치에 있는 변수의 이름이 "실행문" 위치에 있는 변수의 이름과 일치할 때 일치하는 변수에 할당된 값을 출력한다.

다음의 간단한 예를 살펴보자.

switch 문의 형식에서 비교문 위치에 상수값으로 문자열 "name"이 있다. 주어진 4개의 실행문에서 같은 이름의 변수("name")를 찾아서 변수명이 일치하는 변수에 할당된 값인 "홍길동"을 반환한다.

```
> switch("name", id = "hong", pwd = "1234", age = 105, name = "홍길동")
[1] "홍길동"
>
```

실습 switch() 함수를 사용하여 사원명으로 급여정보 보기

```
> empname <- scan(what = "")
1: hong
2:
Read 1 item
> empname
```

```
[1] "hong"
> switch(empname,
+   hong = 250,
+   lee = 350,
+   kim = 200,
+   kang = 400
+ )
[1] 250
>
```

해설 키보드에서 문자열 "hong"을 입력하여 변수 empname에 저장한 뒤에 입력된 값을 확인하고, switch() 함수에서 변수 empname에 입력된 값과 switch 문에 기술된 4개의 문장에서 변수의 이름을 비교하여 같은 경우 해당 변수(hong)가 선택되어 그 값인 250이 출력된다.

2.4 which() 함수

which() 함수는 벡터 객체를 대상으로 특정 데이터를 검색하는데 사용되는 함수이다. which() 함수의 인수로 사용되는 조건식에 만족하는 경우, 즉 조건식의 결과가 참인 벡터 원소의 위치(인덱스)가 출력되며, 조건식의 결과가 거짓이면 0이 출력된다.

형식 which(조건)

실습 벡터에서 which() 함수 사용: index 값을 반환

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

해설 which() 함수는 주어진 벡터 name을 대상으로 벡터 내에서 데이터 "choi"의 위치(인덱스)를 반환한다.

실습 데이터프레임에서 which() 함수 사용

```
> # 단계 1: 벡터 생성과 데이터프레임 생성
> 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
```

	학번	이름	성적
1	1	홍길동	85
2	2	이순신	78
3	3	강감찬	89
4	4	유관순	90
5	5	김유신	74

해설 벡터 no, name, score를 정의하고, 이를 이용하여 데이터프레임을 생성하여 생성된 데이터프레임을 확인한다.

> # 단계 2: 일치하는 이름의 위치(인덱스) 반환

```
> which(exam$이름 == "유관순")
[1] 4
> exam[4, ] # 4번째 레코드 보기
  학번   이름   성적
4    4 유관순   90
>
```

해설 exam 데이터프레임을 대상으로 "유관순" 원소를 찾는 예이다. "유관순" 원소는 exam에서 4번째 해당하는 원소이므로 첨자 4가 출력된다. 만약 exam 데이터프레임에 해당 원소가 없으면 0이 출력된다. 끝으로 exam의 4번째 행을 출력하여 "유관순"의 세부정보를 확인한다. which() 함수는 크기가 큰 데이터프레임이나 테이블 구조의 자료를 대상으로 특정 정보를 검색하는 데 유용하게 사용된다.

3. 반복문

조건에 따라서 특정 실행문을 지정된 횟수만큼 반복적으로 수행할 수 있는 문을 의미한다. R에서는 for() 함수와 while() 함수를 이용하여 반복문을 작성할 수 있다.

3.1 for() 함수

for() 함수는 지정한 횟수만큼 실행문을 반복 수행하는 함수이며, for() 함수의 형식은 다음과 같다. 형식에서 in 뒤쪽에 있는 값을 in 앞쪽에 있는 변수에 순서대로 값을 넘기면서 반복을 수행한다.

형식 for(변수 in 변수) { 실행문 }

실습 for() 함수 사용 기본

```
> i <- c(1:10)
> for(n in i) { # 10회 반복, 단일 문인 경우 {}는 생략 가능
+   print(n * 10) # print() 함수는 변수의 값 또는
+   print(n)     # 계산식의 연산 결과를 출력
+ }
[1] 10
[1] 1
[1] 20
[1] 2
[1] 30
[1] 3
... 중간 생략 ...
[1] 90
[1] 9
[1] 100
[1] 10
>
```

해설 i 벡터의 원소 개수만큼 반복된다. 즉, i 벡터의 첫 번째 원소를 in 앞에 있는 변수 n에 순서대로 넘기면서 {} 블록의 내용을 10회 반복한다. 반복할 문장이 하나 뿐일 때는 {}를 생략할 수 있다.

실습 짝수 값만 출력하기

```
> i <- c(1:10)
> for(n in i)
+   if(n %% 2 == 0) print(n) # 짝수만 출력
[1] 2
[1] 4
[1] 6
[1] 8
[1] 10
>
```

해설 %% 연산자는 나눗셈 연산의 한 종류로 나머지를 계산한다. 따라서 if() 함수는 변수 n의 값을 2로 나눈 나머지를 계산하여 0과 같으면 짝수이므로 해당 숫자만 출력한다.

실습 짝수이면 넘기고, 홀수 값만 출력하기

```
> i <- c(1:10)
> for(n in i) {
+   if( n %% 2 == 0) {
+     next # 다음 문장으로 skip, 반복문 계속(continue 키워드와 동일)
+   } else {
+     print(n) # 홀수일 때만 출력
+   }
+ }
```

해설 %% 연산자를 이용하여 짝수 또는 홀수를 판단하여 반복문을 수행하는 과정에서 원소를 출력하는 예문이다. for() 함수의 반복 범위에서 문장을 실행하지 않고 계속 반복할 때는 next 문을 사용한다.

실습 변수의 칼럼명 출력하기

```
> name <- c(names(exam)) # 데이터프레임 exam에서 칼럼명 추출
> for(n in name) {      # 벡터 name에서 각 칼럼명 추출
+   print(n)            # 칼럼명 출력
+ }
```

해설 데이터프레임 객체 exam에서 칼럼명을 추출하여, name 변수에 벡터 형태로 저장한 뒤에 for() 함수를 이용하여 exam 객체의 칼럼명을 출력하는 예이다.

실습 벡터 데이터 사용하기

```
> score <- c(85, 95, 98)
> name <- c("홍길동", "이순신", "강감찬")
>
> i <- 1 # 첨자로 사용하는 변수
> for(s in score) {
+   cat(name[i], " -> ", s, "\n")
+   i <- i + 1 # 카운터 변수: 첨자 변경
+ }
```

해설 변수 i를 name 벡터의 첨자로 사용하기 위해서 1로 초기화한 뒤에, for() 함수의 반복 범위 내에서 1씩 증가하는 카운터 변수로 사용한다. 즉 변수 i의 값이 1씩 증가한다는 의미는 벡터 변수 name의 데이터를 첨자를 이용하여 출력한다.

3.2 while() 함수

while() 함수는 for() 함수와 동일한 방식으로 수행된다. 따라서 while() 함수를 이용하여 for() 함수로 구현된 문장을 구현할 수 있다. for() 함수와 차이점으로 for() 함수는 반복 회수를 결정하는 변수를 사용하는 대신에 while() 함수는 사용자가 블록 내에서 증감식을 이용하여 반복 회수를 지정해야 한다.

형식 while(조건) { 실행문 }

실습 while() 함수 사용하기

```
> i = 0
> while(i < 10) {
+   i <- i + 1
+   print(i)
+ }
```

해설 변수 i를 기준으로 변수 i의 값이 10 미만일 때까지 반복하면서 변수 i의 값을 출력하는 예이다.

4. 함수 정의

함수는 코드의 집합을 의미한다. 따라서 사용자가 직접 함수 내에 필요한 코드를 작성하여 필요한 경우 함수를 호출하여 사용할 수 있다. 이와 같은 형태로 작성된 함수를 사용자 정의 함수라고 한다. 사용자 정의 함수의 형식은 다음과 같다.

형식 함수명 <- function(매개변수) { 실행문 }

4.1 사용자 정의 함수

사용자가 직접 함수를 정의하는 형식과 외부에서 값을 받아서 처리할 수 있도록 매개변수를 이용하여 함수를 정의하는 방법에 대해서 알아본다.

실습 매개변수가 없는 사용자 함수 정의하기

```
> f1 <- function(){
+   cat("매개변수가 없는 함수")
+ }
>
> f1()      # 사용자 정의 함수 호출
매개변수가 없는 함수
>
```

해설 매개변수가 없는 함수는 "함수명()" 형태로 함수명에 빈 괄호를 붙여 호출한다. 정의된 함수를 호출하지 않으면 함수의 내용은 실행되지 않는다.

실습 결과를 반환하는 사용자 함수 정의하기

```
> f3 <- function(x, y) {
+   add <- x + y
+   return(add)
+ }
>
> add <- f3(10, 20)
> add
[1] 30
>
```

해설 함수의 결과를 반환하는 사용자 정의 함수 f3를 정의하고, 두 개의 실인수를 이용하여 호출하면 두 수의 덧셈 결과를 변수 add로 반환받아서 출력한다. return() 함수는 사용자 정의 함수 내에 있는 값을 함수를 호출하는 곳으로 반환하는 역할을 제공한다.

4.2 기술통계량을 계산하는 함수 정의

추론통계의 기초 정보를 제공하는 요약통계량, 빈도수 등의 기술통계량을 계산하는 함수를 정의해 본다.

실습 기본 함수를 사용하여 요약통계량과 빈도수 구하기

```
> # 단계 1: 파일 불러오기
> setwd("C:/Rwork/Part-I")
> test <- read.csv("test.csv", header = TRUE)
> head(test)
  A B C D E
1 2 4 4 2 2
2 1 2 2 2 2
3 2 3 4 3 3
4 3 5 5 3 3
5 3 2 4 4 4
6 4 3 3 4 2
>
```

단계 2: 요약통계량 구하기

```
> summary(test)      # 변수(A, B, C, D, E)별 요약통계량
      A          B          C          D          E
Min.   :1.000 Min.   :1.000 Min.   :1.000 Min.   :1.00 Min.   :1.000
1st Qu.:2.000 1st Qu.:2.000 1st Qu.:3.000 1st Qu.:2.00 1st Qu.:3.000
Median :3.000 Median :3.000 Median :4.000 Median :2.00 Median :4.000
Mean   :2.734 Mean   :2.908 Mean   :3.622 Mean   :2.51 Mean   :3.386
3rd Qu.:3.000 3rd Qu.:4.000 3rd Qu.:4.000 3rd Qu.:3.00 3rd Qu.:4.000
Max.   :5.000 Max.   :5.000 Max.   :5.000 Max.   :4.00 Max.   :5.000
>
```

단계 3: 특정 변수의 빈도수 구하기

```
> table(test$A)      # 변수 A를 대상으로 빈도수 구하기: 5점 척도(만족도 조사)

 1  2  3  4  5
30 13 15 6 8  3
>
```

해설 요약통계량과 빈도수를 구하기 위해서 summary()와 table() 함수를 적용한 예제이다. 만약 사용자 정의 함수로 정의하면, 한 번의 함수 호출을 통해서 각 칼럼별로 요약통계량과 빈도수를 구할 수 있다.

단계 4: 각 칼럼 단위의 빈도수와 최대값, 최소값 계산을 위한 사용자 함수 정의

```
> data_pro <- function(x) {
+   for(idx in 1:length(x)) {      # 칼럼 수 만큼 반복
+     cat(idx, "번째 칼럼의 빈도 분석 결과")
+     print(table(x[idx]))         # 칼럼별 빈도수 출력
+     cat("\n")
+   }
+
+   for(idx in 1:length(x)) {
+     f <- table(x[idx])
+   }
+ }
```

```

+   cat(idx, "번째 칼럼의 최대값/최소값\n")
+   cat("max = ", max(f), "min = ", min(f), "\n")
+ }
+ }
>
> data_pro(test)
1 번째 칼럼의 빈도분석 결과
 1  2  3  4  5
30 133 156 80  3

... 중간 생략 ...

5 번째 칼럼의 빈도분석 결과
 1  2  3  4  5
8 81 107 160 46

1 번째 칼럼의 최대값/최소값
max = 156 min = 3
2 번째 칼럼의 최대값/최소값
max = 150 min = 7
3 번째 칼럼의 최대값/최소값
max = 176 min = 3
4 번째 칼럼의 최대값/최소값
max = 178 min = 30
5 번째 칼럼의 최대값/최소값
max = 160 min = 8
>

```

해설 각 칼럼 단위로 빈도수와 최대값 그리고 최소값을 계산하는 사용자 정의 함수 data_pro를 작성하고, 한 번의 함수 호출로 다수의 칼럼에 대한 통계량을 계산할 수 있다.

실습 분산과 표준편차를 구하는 사용자 함수 정의

표본분산은 x 변량을 대상으로 "변량의 차의 제곱의 합 / (변량의 개수 - 1)" 수식으로 나타내며, R 코드는 다음과 같다. 표준편차는 sqrt() 함수를 이용하여 표본분산에 루트를 적용하면 된다.

- 표본분산 식: $\text{var} \leftarrow \text{sum}((x - \text{산술평균})^2) / (n - 1)$
- 표준편차 식: $\text{sqrt}(\text{var})$

```

> x <- c(7, 5, 12, 9, 15, 6)           # x 변량 생성
>
> var_sd <- function(x) {
+   var <- sum(x - mean(x) / 2) / (length(x) - 1) # 표본분산
+   sd <- sqrt(var)                               # 표준편차
+   cat("표본분산: ", var, "\n")
+   cat("표본표준편차: ", sd)
+ }
>

```

```

> var_sd(x)                               # 사용자 함수 호출
표본분산: 5.4
표본표준편차: 2.32379
>

```

해설 var() 함수와 sd() 함수를 이용하여 표본분산과 표준표준편차를 구할 수 있는 사용자 함수를 정의하였다.

4.3 피타고라스와 구구단 함수 정의

프로그래밍 예제에서 자주 다루어지는 피타고라스식을 증명하는 함수와 구구단을 출력하는 함수를 정의해 본다.

피타고라스식은 다음과 같다.

$$a^2 + b^2 = c^2$$

실습 피타고라스식 정의 함수 만들기

```

> pytha <- function(s, t) {
+   a <- s^2 - t^2
+   b <- 2 * s * t
+   c <- s^2 + t^2
+   cat("피타고라스 정리: 3개의 변수: ", a, b, c)
+ }
>
> pytha(2, 1)      # s, t 인수는 양의 정수를 갖는다.
피타고라스 정리: 3개의 변수: 3 4 5
>

```

해설 피타고라스식 [(빗변)의 제곱 = (밑변)의 제곱 + (높이)의 제곱]에 함수의 결과를 적용하면 $3^2 + 4^2 = 5^2$ 이 된다.

실습 구구단 출력 함수 만들기

```

> gugu <- 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)      # 단수 지정
> j <- c(1:9)      # 단수와 곱해지는 수 지정

```

```
> gugu(i, j)      # 구구단 출력 함수 호출로 구구단 보기
** 2 단 **
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
... 중간 생략 ...
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
>
```

해설 바깥쪽에 있는 for() 함수의 i는 단수를 의미하고, 안쪽 for() 함수의 j는 단수와 곱해지는 수이다. i가 1회 수행될 때 j는 1에서 9까지 9회 수행된다. 즉 i = 2일때 j는 1에서 9까지 1씩 증가하여 9회 반복되기 때문에 2단이 출력된다.

4.4 결측치 포함 자료의 평균 계산 함수 정의

결측치 데이터(NA)를 포함하는 데이터를 대상으로 평균을 구하기 위해서는 먼저 결측치를 처리한 후 평균과 관련 함수를 이용하여 평균을 계산해야 한다.

실습 결측치를 포함하는 자료를 대상으로 평균 구하기

```
> # 단계 1: 결측치(NA)를 포함하는 데이터 생성
> data <- c(10, 20, 5, 4, 40, 7, NA, 6, 3, NA, 2, NA) # sum: 97
>

> # 단계 2: 결측치 데이터를 처리하는 함수 정의
> na <- function(x) {
+   # 1차: NA 제거
+   print(x)
+   print(mean(x, na.rm = T))
+
+   # 2차: NA를 0으로 대체
+   data = ifelse(!is.na(x), x, 0)
+   print(data)
+   print(mean(data))
+
+   # 3차: NA를 평균으로 대체
+   data2 = ifelse(!is.na(x), x, round(mean(x, na.rm = TRUE), 2))
+   print(data2)
+   print(mean(data2))
+ }
>
```

> # 단계 3: 결측치 처리를 위한 사용자 함수 호출

```
> na(data) # 함수 호출
[1] 10 20 5 4 40 7 NA 6 3 NA 2 NA
[1] 10.77778
[1] 10 20 5 4 40 7 0 6 3 0 2 0
[1] 8.083333
[1] 10.00 20.00 5.00 4.00 40.00 7.00 10.78 6.00 3.00 10.78 2.00 10.78
[1] 10.77833
>
```

해설 결측치가 포함된 데이터를 대상으로 mean() 함수를 이용하여 평균을 구하는 예제이다. 단계 2에서 “1차: NA 제거”는 무조건 결측치를 제거하여 평균을 구하는 예문이고, “2차: NA를 0으로 대체”는 결측치를 0으로 대체하여 평균을 구하는 예이며, “3차: NA를 평균으로 대체”는 결측치를 전체 변량의 평균으로 대체하여 평균을 구하는 예이다. 2차와 3차 예에서처럼 결측치를 무조건 제거하지 않고 0이나 평균으로 대체해야 데이터 손실을 예방할 수 있다.

4.5 몬테카를로 시뮬레이션 함수 정의

몬테카를로 시뮬레이션은 현실적으로 불가능한 문제의 해답을 얻기 위해서 난수의 확률 분포를 이용하는 모의시험으로 근사적 해를 구하는 기법을 의미한다.

실습 동전 앞면과 뒷면에 대한 난수 확률분포의 기대확률 모의시험

동전을 던지는 경우 나올 수 있는 기대확률은 앞면과 뒷면 각각 0.5(1/2)에 해당한다. 일정한 시행 횟수 이하이면 기대확률이 나타나지 않지만, 시행 횟수를 무수히 반복하면 동전의 앞면과 뒷면의 기대확률은 0.5에 가까워진다.

> # 단계 1: 동전 앞면과 뒷면의 난수 확률분포 함수 정의

```
> coin <- function(n) {
+   r <- runif(n, min = 0, max = 1)
+   result <- numeric()
+   for(i in 1:n) {
+     if(r[i] <= 0.5)
+       result[i] <- 0 # 앞면
+     else
+       result[i] <- 1 # 뒷면
+   }
+   return(result)
+ }
>
```

> # 단계 2: 동전 던지기 횟수가 10회인 경우 앞면(0)과 뒷면(1)이 나오는 vector 값

```
> coin(10)
[1] "1" "0" "1" "1" "1" "0" "0" "1" "0" "1"
>
```


> # 단계 3: 몬테카를로 시뮬레이션 함수 정의

```
> montaCoin <- function(n) {
+   cnt <- 0
+   for(i in 1:n) {
+     cnt <- cnt + coin(1) # n 시행 횟수만큼 동전 함수 호출
+   }
+   # 동전 앞면과 뒷면의 누적 결과를 시행 횟수(n)로 나눈다.
+   result <- cnt / n
+   return(result)
+ }
>
```

> # 단계 4: 몬테카를로 시뮬레이션 함수 호출

```
> montaCoin(10)
[1] 0.6
> montaCoin(30)
[1] 0.5666667
> montaCoin(100)
[1] 0.4
> montaCoin(1000)
[1] 0.497
> montaCoin(10000)
[1] 0.5001
>
```

해설 시행 횟수(표본 수)가 무한히 클수록 동전의 앞면과 뒷면이 나올 기대확률(0.5)에 근사적 해가 구해지는 것을 확인할 수 있다.

5. 주요 내장함수

R을 설치하면 기본적으로 사용할 수 있는 함수를 내장함수라고 한다. 예를 들면 평균을 구하는 mean() 함수, 합계를 구하는 sum() 함수, 요약통계량을 제공하는 summary() 함수 등이 내장함수이다. 이 절에서는 R에서 제공하는 다양한 내장함수를 영역별로 정리하여 제공한다.

5.1 기술통계량 처리 관련 내장함수

전체 자료를 대표하는 대표값, 중심에 얼마나 떨어져있는가를 나타내는 산포도 등을 나타내는 기술통계량 처리 관련 내장함수는 [표 4.2]와 같다.

[표 4.2] 기술통계량 처리 관련 내장함수

함수	의미
min(vec)	벡터 대상 최소값을 구하는 함수
max(vec)	벡터 대상 최대값을 구하는 함수
range(vec)	벡터 대상 범위값을 구하는 함수(최소값 ~ 최대값)
mean(vec)	벡터 대상 평균값을 구하는 함수
median(vec)	벡터 대상 중위수를 구하는 함수(중앙값)
sum(vec)	벡터 대상 합계를 구하는 함수
sort(x)	벡터 데이터 정렬 함수
order(x)	벡터의 정렬된 값의 색인(index)을 보여주는 함수
rank(x)	벡터의 각 원소의 순위를 제공하는 함수
sd(x)	표준편차를 구하는 함수
summary(x)	x에 대한 기초 통계량을 함수
table(x)	x에 대한 빈도수를 구하는 함수
sample(x, y)	x 범위에서 y 만큼 sample 데이터를 생성하는 함수

실습 행/칼럼 단위의 합계와 평균 구하기

> # 단계 1: 데이터 셋 불러오기

```
> library(RSADBE) # 패키지를 메모리에 로드
> data("Bug_Metrics_Software") # RSADBE 패키지에서 제공하는 데이터 셋 불러오기
> Bug_Metrics_Software[, 1] # 전체 자료 보기
      Bugs
Software Bugs NT.Bugs Major Critical H.Priority
JDT      11605   10119   1135    432     459
PDE       5803    4191    362    100     96
Equinox   325    1393    156     71     14
Lucene   1714    1714     0      0      0
Mylyn   14577    6806    592    235    8804
>
```

> # 단계 2: 행 단위 합계와 평균 구하기

```
> rowSums(Bug_Metrics_Software[, 1]) # 소프트웨어 별 버그 수 합계
      JDT      PDE    Equinox    Lucene    Mylyn
23750 10552   1959    3428   31014
> rowMeans(Bug_Metrics_Software[, 1]) # 소프트웨어 별 버그 수 평균
      JDT      PDE    Equinox    Lucene    Mylyn
4750.0 2110.4   391.8    685.6  6202.8
>
```

> # 단계 3: 열 단위 합계와 평균 구하기

```
> colSums(Bug_Metrics_Software[, 1]) # 버그 종류별 버그 수 합계
  Bugs NT.Bugs Major Critical H.Priority
34024  24223  2245    838    9373
> colMeans(Bug_Metrics_Software[, 1]) # 버그 종류별 버그 수 평균
  Bugs NT.Bugs Major Critical H.Priority
6804.8  4844.6  449.0   167.6   1874.6
>
```

해설 R의 기본 함수를 이용하여 행/열 단위 합계와 평균을 간편하게 계산할 수 있다.

실습 기술통계량 관련 내장함수 사용하기

```
> seq(-2, 2, by = .2) # -2 ~ 2 범위에서 0.2씩 증가
[1] -2.0 -1.8 -1.6 -1.4 -1.2 -1.0 -0.8 -0.6 -0.4 -0.2 0.0 0.2 0.4 0.6 0.8 1.0
[17] 1.2 1.4 1.6 1.8 2.0
> vec <- 1:10
> min(vec) # 최소값
[1] 1
> max(vec) # 최대값
[1] 10
> range(vec) # 범위
[1] 1 10
> mean(vec) # 평균
[1] 5.5
> median(vec) # 중위수
[1] 5.5
> sum(vec) # 합계
[1] 55
> sd(rnorm(10)) # 정규분포 자료 10개(무작위 추출)를 대상으로 표준편차 구하기
[1] 0.9939842
> table(vec) # 벡터 자료 대상으로 빈도수 구하기
vec
 1  2  3  4  5  6  7  8  9 10
 1  1  1  1  1  1  1  1  1  1
>
```

해설 기술통계량 관련 내장함수를 이용하여 여러 가지 통계량을 구하는 예이다.

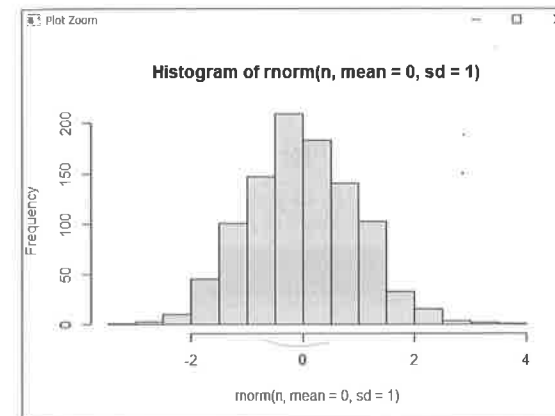
난수(random)를 발생하는 함수와 확률 분포의 관계를 알아보자.

실습 정규분포(연속형)의 난수 생성하기

정규분포(연속형)의 난수 생성은 rnorm() 함수를 이용한다. 함수의 형식은 다음과 같다.

형식 rnorm(n, mean, sd) # 평균과 표준편차 이용

```
> n <- 1000
> rnorm(n, mean = 0, sd = 1) # 표준정규분포를 갖는 난수 생성하기
[1] 2.0992866337 -0.0577741624 0.3869033329 1.5786375651
[5] -0.3917067707 0.0896869155 1.3770505773 0.0148738842
[9] -0.7201502551 0.3894034118 0.9144231007 0.8314149473
[13] -0.0873186022 1.7835455925 -0.2081214007 1.2262260663
... 중간 생략 ...
[993] -0.2816706710 -0.9784765451 -0.3252483589 1.3864058229
[997] -0.3388966929 -1.1026911852 -0.1126539547 -1.1457405634
> hist(rnorm(n, mean = 0, sd = 1)) # 표준정규분포 - 히스토그램
>
```



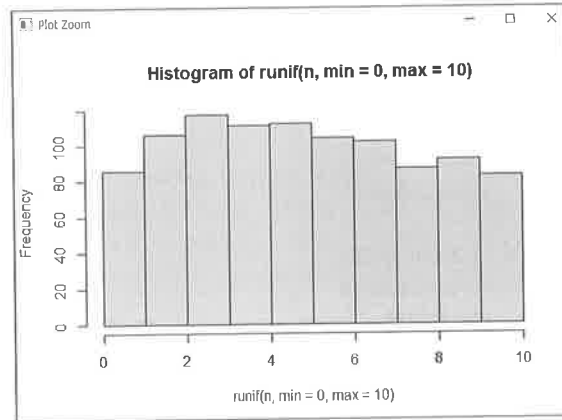
해설 평균 = 0, 표준편차 = 1로 고정한 표준정규분포로 히스토그램을 그리면 평균을 중심으로 좌우 균등한 종 모양의 이상적인 분포가 그려진다. rnorm() 함수는 정규분포를 따르는 난수를 생성해 준다.

실습 균등분포(연속형)의 난수 생성하기

균등분포(연속형)의 난수 생성은 runif() 함수를 이용한다. 함수의 형식은 다음과 같다.

형식 runif(n, min, max) # 최소값, 최대값 이용

```
> n <- 1000
> runif(n, min = 0, max = 10) # 정규분포를 갖는 난수 생성하기
[1] 1.32381588 0.34807969 7.91260693 2.52325204 9.52873275 5.34417174
[7] 5.41701012 3.85365522 8.51306140 2.82420243 1.15500967 3.35054572
[13] 4.38053326 3.22312720 6.19401668 5.95217042 6.76686368 0.04038205
... 중간 생략 ...
[991] 0.68656039 6.71186347 3.27295398 4.17877703 6.20817850 3.53303677
[997] 6.23805072 1.11995040 3.06846721 6.31266644
> hist(runif(n, min = 0, max = 10)) # 표준정규분포 - 히스토그램
>
```



해설 $0 < n < 10$ 사이의 범위에서 난수를 발생한다. 난수를 히스토그램으로 그려보면 계급에 따라서 빈도가 균등한 분포로 그려진다.

실습 이항분포(이산형)의 난수 생성하기

이항분포는 정규분포와 같은 이상적인 분포형을 갖지만, 연속형이 아닌 이산변량을 갖는다. 이항분포의 난수를 생성하기 위해서는 `rbinom()` 함수를 사용한다. 함수의 형식은 다음과 같다.

형식 `rbinom(n, size, prob)` # 독립적 n 회 반복

```
> n <- 20          # 독립시행 횟수
>                  # 0 또는 1의 이산변량 대상 0.5 확률로 n개 선정
> rbinom(n, 1, prob = 1 / 2)
[1] 1 0 1 0 0 1 1 1 1 0 0 1 0 0 0 1 1 1 0 1
> rbinom(n, 2, 0.5) # 0, 1, 2의 이산변량 대상 0.5 확률로 n개 선정
[1] 1 0 1 2 2 1 1 2 2 2 2 1 0 0 0 0 2 0 1
> rbinom(n, 10, 0.5) # 0 ~ 10의 이산변량 대상 0.5 확률로 n개 선정
[1] 6 7 1 5 4 3 3 8 4 2 3 4 4 4 4 2 5 6 4 5
> n <- 1000        # 독립시행 횟수
>                  # 0 ~ 5 사이의 이산변량 대상 1/6의 확률로 n개 선정
> rbinom(n, 5, prob = 1 / 6)
[1] 0 0 0 0 0 1 2 0 1 0 3 0 1 1 0 1 0 0 2 1 1 0 2 0 1 1 0 0 2 3 1 1 2 2
[35] 0 1 1 0 0 0 0 1 0 1 1 2 1 1 1 2 0 0 1 1 0 2 0 0 2 1 0 1 0 0 1 0 1 1
[69] 0 3 2 0 2 0 1 0 0 1 1 1 0 0 1 0 1 1 0 0 0 0 0 2 0 0 0 0 0 0 2 1 0 0
... 중간 생략 ...
[953] 1 1 1 0 1 0 1 1 0 1 1 0 0 2 2 0 1 2 1 0 1 2 1 0 0 1 1 0 0 1 1 0 2 0
[987] 1 0 1 2 0 2 1 2 0 1 0 1 3 1
>
```

해설 독립적인 반복 회수와 변량의 크기 그리고 확률을 적용하여 이산형의 난수로 발생시킨다. 여기서 이산형은 정수형을 의미한다.

실습 종자값으로 동일한 난수 생성하기

난수 관련 함수는 매번 실행할 경우 임의의 난수를 생성하는데, 종자(seed)값을 지정하면 동일한 난수

를 발생시킬 수 있다. 종자값 지정은 위해서는 `seed()` 함수를 사용한다. 함수의 형식은 다음과 같다.

형식 `set.seed(임의의 정수)` # 임의의 정수를 종자값으로 하여 동일한 난수 생성

```
> rnorm(5, mean = 0, sd = 1) # 매번 실행하는 경우 임의의 난수 생성
[1] 1.8667723 -1.3998326 -0.8496292 0.3184496 0.9035913
> set.seed(123)             # 종자값을 123으로 설정
> rnorm(5, mean = 0, sd = 1) # 종자값을 이용한 난수 생성
[1] -0.56047565 -0.23017749 1.55870831 0.07050839 0.12928774
> set.seed(123)             # 종자값을 123으로 설정
> rnorm(5, mean = 0, sd = 1) # 이전 실행으로 생성된 난수와 같은 결과
[1] -0.56047565 -0.23017749 1.55870831 0.07050839 0.12928774
> set.seed(345)             # 종자값을 345로 설정, 같은 종자값이면 같은 난수 생성
> rnorm(5, mean = 0, sd = 1)
[1] -0.78490816 -0.27951436 -0.16145790 -0.29059656 -0.06753159
>
```

해설 첫 번째 행에서 종자값을 지정하지 않고 `rnorm()` 함수를 실행하면 임의의 난수 5개가 생성된다. 이 경우 실행할 때마다 난수의 결과가 달라진다. 한편 두 번째와 세 번째 난수 생성에서는 같은 종자값을 이용하기에 같은 난수 5개를 생성한다. 네 번째 난수 생성은 종자값을 345로 변경하여 또 다른 난수 5개를 생성하게 된다. 즉, 종자값이 같으면 동일한 난수가 생성된다는 의미이다.

5.2 수학 관련 내장함수

절댓값, 제곱근, 사인, 코사인, 탄젠트 등 수학 관련 내장함수는 [표 4.3]과 같다.

[표 4.3] 수학 관련 내장함수

함수	의미
<code>abs(x)</code>	절댓값을 구하는 함수
<code>sqrt(x)</code>	제곱근을 구하는 함수
<code>ceiling(x)</code> , <code>floor()</code> , <code>round()</code>	값의 올림, 내림, 반올림을 구하는 함수
<code>factorial(x)</code>	계승(factorial)값을 구하는 함수
<code>which.min(x)</code> , <code>which.max(x)</code>	벡터 내 최소값과 최대값의 인덱스를 구하는 함수
<code>pmin(x)</code> , <code>pmax(x)</code>	여러 벡터에서의 원소 단위 최소값과 최대값을 구하는 함수
<code>prod()</code>	벡터의 원소들의 곱을 구하는 함수
<code>cumsum()</code> , <code>cumprod()</code>	벡터의 원소들의 누적합과 누적곱을 구하는 함수
<code>cos(x)</code> , <code>sin(x)</code> , <code>tan(x)</code>	삼각함수(also <code>acos(x)</code> , <code>cosh(x)</code> , <code>acosh(x)</code> 등)
<code>log(x)</code>	자연로그(natural logarithm)
<code>log10(x)</code>	10을 밑으로 하는 일반로그 함수(ex)
<code>exp(x)</code>	지수함수(exponential function)

실습 수학 관련 내장함수 사용하기

```
> vec <- 1:10
> prod(vec)      # 벡터의 곱(1 * 2 * 3 * ... * 10)
[1] 3628800
> factorial(5)   # 계승(1 * 2 * 3 * 4 * 5) = 120
[1] 120
> abs(-5)       # 절대값(5)
[1] 5
> sqrt(16)      # 제곱근(4)
[1] 4
> vec
[1] 1 2 3 4 5 6 7 8 9 10
> cumsum(vec)    # 벡터 값에 대한 누적 합
[1] 1 3 6 10 15 21 28 36 45 55
> log(10)        # 10의 자연로그
[1] 2.302585
> log10(10)      # 10의 일반로그
[1] 1
>
```

해설 수학 관련 내장함수를 이용하여 통계량을 구하는 예이다. 특히 로그 관련 함수는 10에 대한 자연로그($\log_e(10)$)와 일반로그($\log_{10}(10)$)를 계산하는 함수이다. 자연로그는 밑수가 무리수 $e(e = 2.718281828)$ 이고, 일반로그는 밑수가 10이다.

5.3 행렬연산 관련 내장함수

행렬 계산을 효율적으로 하기 위해서 R에서는 행렬 분해(matrix decomposition)에 관련된 함수를 제공한다. 여기서 행렬 분해는 행렬을 특정한 구조를 가진 다른 행렬의 곱으로 나타내는 것을 의미하며, 관련 함수는 `eigen()`, `svd()`, `qr()`, `chol()` 등으로 [표 4.4]와 같다.

[표 4.4] 행렬연산 관련 내장함수

함수	의미
<code>ncol(x)</code>	x의 열(칼럼) 수를 구하는 함수
<code>nrow(x)</code>	x의 행 수를 구하는 함수
<code>t(x)</code>	x 대상의 전치행렬을 구하는 함수
<code>cbind(...)</code>	열을 추가할 때 이용되는 함수
<code>rbind(...)</code>	행을 추가할 때 이용되는 함수
<code>diag(x)</code>	x의 대각행렬을 구하는 함수
<code>det(x)</code>	x의 행렬식을 구하는 함수
<code>apply(x, m, fun)</code>	행 또는 열에 지정된 함수를 적용하는 함수
<code>solve(x)</code>	x의 역행렬을 구하는 함수

<code>eigen(x)</code>	정방행렬을 대상으로 고유값을 분해하는 함수
<code>svd(x)</code>	m x n 행렬을 대상으로 특이값을 분해하는 함수
<code>x %*% y</code>	두 행렬의 곱을 구하는 수식

실습 행렬연산 내장함수 사용하기

```
> # 행렬연산을 위한 x,y 행렬 생성
> x <- matrix(1:9, nrow = 3, ncol = 3, byrow = T) # 3 x 3 정방행렬
> y <- matrix(1:3, nrow = 3)                    # 3x1 행렬
> ncol(x)                                         # 열 수 반환
[1] 3
> nrow(x)                                         # 행 수 반환
[1] 3
> t(x)                                            # x의 전치행렬 반환
      [,1] [,2] [,3]
[1,]  1  4  7
[2,]  2  5  8
[3,]  3  6  9
> cbind(x, 1:3)                                  # x에 열 추가
      [,1] [,2] [,3] [,4]
[1,]  1  2  3  1
[2,]  4  5  6  2
[3,]  7  8  9  3
> rbind(x, 10:12)                                # x에 행 추가
      [,1] [,2] [,3]
[1,]  1  2  3
[2,]  4  5  6
[3,]  7  8  9
[4,] 10 11 12
> diag(x)                                         # 정방행렬 x에서 대각선 값 반환
[1] 1 5 9
# 6.661338e-16
> det(x)
[1] 6.661338e-16
# x의 행 단위 합계
> apply(x, 1, sum)
[1] 6 15 24
# x의 열 단위 평균
> apply(x, 2, mean)
[1] 4 5 6
# 정방행렬 x에서 d, u, v 행렬로 특이값 분해
> svd(x)
$d
[1] 1.684810e+01 1.068370e+00 4.418425e-16
$u
      [,1]      [,2]      [,3]
[1,] -0.2148372  0.8872307  0.4082483
[2,] -0.5205874  0.2496440 -0.8164966
[3,] -0.8263375 -0.3879428  0.4082483
```

```
$v
      [,1]      [,2]      [,3]
[1,] -0.4796712 -0.77669099 -0.4082483
[2,] -0.5723678 -0.07568647  0.8164966
[3,] -0.6650644  0.62531805 -0.4082483
> eigen(x)      # 정방행렬 x에서 고유값 분해
eigen() decomposition
$values
[1] 1.611684e+01 -1.116844e+00 -1.303678e-15

$vectors
      [,1]      [,2]      [,3]
[1,] -0.2319707 -0.78583024  0.4082483
[2,] -0.5253221 -0.08675134 -0.8164966
[3,] -0.8186735  0.61232756  0.4082483
> x %*% y      # x, y의 행과 열을 곱하고 더하는 행렬 곱을 반환
      [,1]
[1,] 14
[2,] 32
[3,] 50
```

해설 행렬 관련 내장함수를 이용하여 행렬연산을 수행하는 예이다

5.4 집합연산 관련 내장함수

집합을 대상으로 합집합, 교집합, 차집합 등의 집합연산을 수행하는 R의 내장함수는 [표 4.5]와 같다.

[표 4.5] 집합연산 관련 내장함수

함수	의미
union (x, y)	집합 x와 y의 합집합
setequal (x, y)	x와 y의 동일성 검사
intersect (x, y)	집합 x와 y의 교집합
setdiff (x, y)	x의 모든 원소 중 y에는 없는 x와 y의 차집합
c %in% y	c가 집합 y의 원소인지 검사

실습 집합연산 관련 내장함수 사용하기

```
> # 집합연산을 위한 벡터 설정
> x <- c(1, 3, 5, 7, 9)
> y <- c(3, 7)
>
> union(x, y)      # x, y 벡터에 관한 합집합
[1] 1 3 5 7 9
```

```
> setequal(x, y)   # x, y 벡터에 관한 동일성 검사
[1] FALSE
> intersect(x, y)  # x, y 벡터에 관한 교집합
[1] 3 7
> setdiff(x, y)    # x, y 벡터에 관한 차집합
[1] 1 5 9
> setdiff(y, x)    # y, x 벡터에 관한 차집합
numeric(0)
> 5 %in% y         # 5가 y의 원소인지 검사
[1] FALSE
>
```

해설 집합 관련 내장함수를 이용하여 통계량을 구하는 예이다

4장 연습문제

1. 다음 조건에 맞게 client 데이터프레임을 생성하고, 데이터를 처리하시오.

```
> # Vector 데이터 준비
> name <- c("유관순", "홍길동", "이순신", "신사임당")
> gender <- c("F", "M", "M", "F")
> price <- c(50, 65, 45, 75)
```

조건 1 | 3개의 벡터 객체를 이용하여 client 데이터프레임을 생성하시오.

조건 2 | price 변수의 값이 65만 원 이상이며 문자열 "Best", 65만 원 미만이면 문자열 "Normal" 을 변수 result에 추가하시오.

☞ 힌트: ifelse() 함수 사용

조건 3 | result 변수를 대상으로 빈도수를 구하시오.

2. 다음의 벡터 EMP는 '입사연도이름급여'순으로 사원의 정보가 기록된 데이터이다. 벡터 EMP를 이용하여 다음과 같은 출력 결과가 나타나도록 함수를 정의하시오.

```
> # Vector 데이터 준비
> EMP <- c("2014홍길동220", "2002이순신300", "2010유관순260")
```

☞ 출력결과: 전체 급여 평균 : 260
평균 이상 급여 수령자
이순신 => 300
유관순 => 260

☞ 힌트: 사용할 함수

stringr 패키지 : str_extract(), str_replace() 함수,
숫자변환 함수 : as.numeric() 함수
한글 문자 인식 정규표현식 패턴 : [가-힣]

사용자 함수 정의

```
emp_pay <- function(x) {
  # 함수 내용 작성
}
```

함수 호출

```
emp_pay(EMP)
```

3. 함수 $y = f(x)$ 에서 x 의 값이 a 에서 b 까지 변할 때 $\Delta x = b - a$ 를 x 의 증분이라고 하며, $\Delta y = f(b) - f(a)$ 를 y 의 증분으로 표현합니다. 여기서, 평균변화율은 다음식과 같습니다

$$\text{평균변화율} = \Delta y / \Delta x = (f(b) - f(a)) / (b - a)$$

조건 | 함수 $f(x) = x^3 + 4$ 에서 x 의 값이 1에서 3까지 변할 때 평균변화율(mean ratio of change)을 구하는 함수를 작성하시오.

4. RSADBE 패키지에서 제공되는 Bug_Metrics_Software 데이터 셋을 대상으로 소프트웨어 발표 후의 행 단위 합계와 열 단위 평균을 구하고, 칼럼 단위로 요약통계량을 구하시오.

```
library('RSADBE')
data('Bug_Metrics_Software')
```