A Minimal Book Example

John Doe

2022-06-08

차 례

1 About **5**

1.1 Usage 5

1.2 Render book 5

1.3 Preview book 6

2 Introduction **1**

2.1 R을 왜 사용해야할까? 1

2.2 이 책의 구성 3

2.3 배우지 않는 내용들 4

2.4 당부의 말 5

3 R 자료 구조의 이해 **7**

3.1 명령어의 구조와 자료 입력 7

3.2 R에서 쓰이는 자료의 유형 8

3.3 R에서 쓰이는 자료의 구조 9

3.4 factor 변수 16

3.5 NA와 NULL 17

3.6 R 내장함수를 활용한 기술통계량 산출 19

4 Parts **23**

5 Footnotes and citations **25**

5.1 Footnotes 25

5.2 Citations 25

6 Blocks **27**

6.1 Equations 27

6.2 Theorems and proofs 27

6.3 Callout blocks 28

7 Sharing your book **29**

7.1 Publishing 29

7.2 404 pages 29

7.3 Metadata for sharing 29

제 1 장

About

This is a *sample* book written in **Markdown**. You can use anything that Pandoc's Markdown supports; for example, a math equation $a^2 + b^2 = c^2$.

1.1 Usage

Each **bookdown** chapter is an .Rmd file, and each .Rmd file can contain one (and only one) chapter. A chapter *must* start with a first-level heading: # A good chapter, and can contain one (and only one) first-level heading.

Use second-level and higher headings within chapters like: ## A short section or ### An even shorter section.

The index.Rmd file is required, and is also your first book chapter. It will be the homepage when you render the book.

1.2 Render book

You can render the HTML version of this example book without changing anything:

1. Find the **Build** pane in the RStudio IDE, and
2. Click on **Build Book**, then select your output format, or select "All formats" if you'd like to use multiple formats from the same book source files.

Or build the book from the R console:

```
bookdown::render_book()
```

To render this example to PDF as a bookdown::pdf_book, you'll need to install XeLaTeX. You are recommended to install TinyTeX (which includes XeLaTeX): https://yihui.org/tinytex/.

1.3 Preview book

As you work, you may start a local server to live preview this HTML book. This preview will update as you edit the book when you save individual .Rmd files. You can start the server in a work session by using the RStudio add-in "Preview book", or from the R console:

```
bookdown::serve_book()
```

제 2 장

Introduction

이 책을 쓰게 된 계기는 간단하다. 약 6년간 서울대학교 산업인력개발 전공의 학부와 대학원 통계 수업을 진행하면서 인력개발 분야의 통계 서적의 필요성을 실감했기 때문이다. 기존의 서적들을 각기 조합하여 수업을 진행하다보니, 학생들은 물론 교수자도 힘들었던 점들이 많았다. 수많은 서적들이 제각기의 시각으로 통계 관련 이론과 실무적인 내용들을 잘 풀어내고 있지만, 우리 분야에 적합한 책을 찾기는 어려웠다. 두번째 이유는 코로나로 인해 대부분의 출장과 회의가 온라인으로 이뤄지면서, 이동시간이 절약되어 집필에 필요한 시간이 확보되었다.

이 책은 일종의 조각 모음으로 집필이 진행되었다. 매년 수업을 진행하며 조각 조각 만들어 냈던 자료들을 통합함으로서 하나의 스토리를 만들어내고자 노력했다. 개인적으로는 통계 전공자가 아니기 때문에 이 책을 쓰는데 부담이 없지 않았다. 따라서 오류를 최소화하기 위해 집필의 말미에는 각 분야별 전문가 분들께 크로스체크를 부탁하였다.

2.1 R을 왜 사용해야할까?

책의 제목에서 알 수 있듯이 기본이 되는 통계 패키지는 R을 사용하였다. 사실 교육학 분야의 많은 오래된 학자들은 SPSS가 익숙하고, 나 역시 첫 통계 공부는 SPSS로 시작했다. 관행은 계기가 없으면 바뀌기 어렵기 때문에, 학교에 부임한 이후에도 수년간은 SPSS를 이용한 수업을 진행하기도 했다. 하지만 다음과 같은 네 가지 이유로 SPSS는 나의 컴퓨터에서 사라지게 되었다.

- 첫째, SPSSS는 비싼 라이센스를 사용해야 하기 때문에 개인 연구자에 적합하지 않다. 서울대 역시 SPSS 제조사와 지리한 라이선스 가격 협상을 이어가고 있는 실정이다. 하물며 개인이 라이센스에 지갑을 열어야 하는 경우 SPSS는 좋은 선택이 아니다.

- 둘째, SPSS가 갖고 있는 직관적인 인터페이스와 간단한 분석방법은 종종 초보 연구자의 오류를 촉진한다. 모든 종류의 프로그램이 그렇듯이 이들은 분석의 적합성을 검토해주지 않는다. 자판기에 동전을 넣으면 음료수가 나오듯이 데이터를 입력하면 분석결과가 나오지만, 그것이 잘못된 선택인지 알수가 없다. 반면에 R 등의 스크립트 기반의 통계패키지는 적어도 내가 무엇을 분석하려고 하는지에 대한 기본적인 이해를 필요로 한다.

- 셋째, 복잡한 분석(이라 쓰고 삽질이라 읽는다)을 실시할 때 click to click 방식의 통계패키지는 삽질의 시간을 더 길게 만든다. 보통 우리가 마주하는 데이터는 다양한 변형(manipulation)을 요구하는데, 이때 (1) 어떠한 형태로 데이터를 변형할 것인지, (2) 변형을 위한 각 단계는 어떻게 구성해야 하는지에 대한 사전정보를 알고 있는 경우는 드물다. R과 같이 스크립트를 기반으로 데이터를 핸들링하고 분석할 수 있는 경우는 일주일간 작업한 내용의 오류를 발견했을 때, 그간 작성해놓은 코드에 일부만 수정하고 실행함으로서 간단히 오류를 고칠 수 있다. SPSS 같은 프로그램들은 데이터 변형 및 분석의 복원이 불가능하다는 점을 고려해보면 어마어마한 장점이 아닐 수 있다. 보통 이런 작업을 디버깅(debuging)이라고 하는데, 고통스러운 디버깅 작업을 10시간정도 하게 되면, R을 시작하는데 필요한 약간의 허들은 쉽게 느껴질 것이다.

- 넷째, R은 빠르게 변화하는 최신 분석기법들을 빠르게 설치, 활용할 수 있다. 종종 R을 스마트폰에 비유하곤하는데, ios나 android와 같은 플랫폼에 여러가지 어플을 설치하는 방식을 생각하면 간단하다. 플랫폼의 업데이트는 느리고 무겁지만, 각각의 어플은 가볍고, 빠르며, 쉽게 적용이 가능하다. 결국 확장성의 장점이 R이 갖고 있는 가장 큰 장점이라고 볼 수 있다.

R의 기본적인 설치, 구조 이해, 분석의 기초와 함께, 이 책에서는 통계학의 기본적인 내용들도 알기 쉽게 설명하고자 노력했다. 두 내용 모두 방대하기 때문에 보통 하나의 책에서 통계패키지의 분석 테크닉과 통계학 이론을 한꺼 번에 다루지는 않는다. 하지만 수년간 학생들을 가르치다보니 두 내용을 연결하는 책이 절실하게 필요했다. 두마리의 토끼를 과연 잡았을지는 모르겠지만, 부디 이 책이 의도한 바를 달성했기를 바란다.

2.2 이 책의 구성

이 책은 크게 3부로 구성된다.

1부. R의 기초 이해

1부는 R이라는 데이터 분석 도구tool에 대해 이해하도록 한다. 앞서 언급했던것처럼 R은 특히 복잡한 데이터 전처리에서 진가를 발휘한다. 우리가 접하는 데이터들은 전처리가 거의 필요없는 유형(내 연구 가설에 딱 맞는 형태로 직접 수집한 데이터, 예를 들어 학위논문 등을 위해 직접 수집해서 코딩까지한 자료)부터 매우 복잡한 전처리를 해야만 분석이 가능한 유형(보통 행정 및 관리를 위해 조직에 축적된 chunky한 데이터들)까지 다양하다. 1부에서는 이러한 데이터 전처리를 용이하게 하기 위해 필요한 다양한 기법들에 대해 소개할 예정이다. 1부에서 다룰 내용들은 다음과 같다.

- 1장. R설치, 작업환경 셋업과 R 환경이해하기
- 2장. R 자료 구조의 이해
- 3장. R을 활용한 데이터 전처리
- 4장. R을 활용한 데이터 시각화

2부. R을 활용한 기초 다변량 분석

2부는 연구를 위해 활용되는 다양한 다변량 분석 기법의 이론에 대해 소개한다. 1부는 R이라는 분석도구를 사용하는 테크닉에 대한 소개일 뿐이다. 다시 말해 못을 박는 망치 사용법일뿐이지 어디에다 못을 박아야 하는지, 몇개를 박아야 하는지, 얼마나 깊숙히 박아야 하는지에 대한 답을 주지는 못한다. 2부에서는 추론통계와 가설 검정에 대한 이해를 바탕으로 t검정과 ANOVA, 상관분석, 회귀분석, 로지스틱, 매개분석과 조절분석에 대해 다룰 예정이다. 이 밖에 더 많은 다변량 분석기법들이 존재하지만, 산업인력개발 분야에서 가장 빈번하게 다루는 기초적인 분석기법들을 선택하였다. 각 장별로 개념이해, 결과 해석, R을 활용한 분석코드 순서로 기술되었다.

- 5장. 추론통계와 가설검정
- 6장. t 검정과 ANOVA
- 7장. 상관분석

- 8장. 회귀분석
- 9장. 로지스틱 회귀분석
- 10장. 매개분석과 조절분석
- 11장. 구조방정식 (????)

3부. 실전데이터를 활용한 분석 사례

2.3 배우지 않는 내용들

what did you say ?

이 책에서 다루지 않는 내용은 그야말로 산더미 같이 많이 있다. 정확히 숫자로 표현 할수는 없겠지만 한 90% 정도는 책에 담지 못한 내용들일 것이다!! 아마 10년정도 후에는 99% 정도로 늘어날지 모르겠다. 하지만 이 책에 담긴 내용들을 충분히 숙지하였다면 나머지 90%의 내용은 여러분 스스로 학습할 수 있는 좋은 기본기를 갖췄다고 생각해도 무방하다. 개인적으로는 이 책이 (1) 여러분들의 통계포비아를 없애주고, (2) 새로운 개념, 기법에 대한 자기주도학습이 가능하도록 하는 일종의 밑바탕으로 기능했으면 하는 바램이다.

좀 더 구체적으로 이 책에서 제외된 내용은 다음과 같다. 우선 1부에서 R을 활용한 데이터 전처리의 맛보기만 기술하였기 때문에, 이른바 빅데이터라고 불리는 청키한 데이터를 다루는 기법까지 설명하지는 못했다. 특히 다양한 DB 등에서 데이터를 끌어와 분석에 용이한 형태로 만드는 것은 좀더 심화된 기법이 필요하다. 이와 관련해서는 관련한 기존 서적들을 충분히 참조했으면 하는 바람이다(물론 1부를 모두 이해한 후에)

2부에서도 다루지 못한 내용들이 많다. 특히 15년 전부터 사회과학분야에 거대한 유행으로 자리잡은 구조방정식structural equaation modeling, 다층선형모형hierarchical linear modeling 등 굵직한 기법들이 모두 빠져있다. 이러한 분석기법들은 각각의 대표적인 기본서들이 있으므로, 이를 참조하였으면 하는 바람이다. 이들 역시 일종의 회귀분석의 변형이기 때문에 기초를 탄탄하게 쌓았다면 이해를 확장하는데 어려움이 없을 것이라 생각한다. 이와 더불어 다양한 longitudinal data를 다루는 분석기법들도 생략되어 있다.

이밖에도 저자의 또 다른 강의인 "직업연구"나 "산업인력개발 노동시장분석론"에서 다루었던 다양한 통계기법에 대한 내용들도 빠져있다. 아마도 빠른 시간 안에 위의 두 강의에서 다루었던 내용을 별도의 책으로 출간할 수 있지 않을 까 기대한다(기대만 하고 있다….)

2.4 당부의 말

마지막으로 우연히 이 책을 접한 독자들에게 하고 싶은 말은 책의 내용을 이해하는 것을 포기하지 말라는 것이다. 고등학교 때 '수학의 정석'의 첫 챕터인 집합 부분만 까맣게 손때가 묻어있던것을 기억할 것이다. 많은 사람들이 2부의 첫 챕터에서 흥미를 잃겠지만, 포기하지 않고 여러번 완독을 한다면 분명 많은 도움이 될 것이다. 설사 이해하지 못하는 내용이 있다 하더라도 여러번 읽고, 손으로 문제를 풀어보는 버릇을 들이면 더 빠르게 이해할 수 있다.

또 간단한 소논문 등을 작성해보고 다시 책을 읽어보면 이해가 안되었던 부분들이 새롭게 보이는 날들이 있을 것이라 믿어 의심치 않는다. 또한 이 책이 정답이 아니므로, 설명이 부족한 부분들이 있다면 구글이나 유튜브 등에 키워드 검색을 통해 추가적인 설명자료와 강의등의 도움을 받길 바란다.

제 3 장

R 자료 구조의 이해

3.1 명령어의 구조와 자료 입력

R의 명령어는 일종의 언어laguage이기 때문에 나름의 문법을 갖고 있다. 처음 R의 언어를 접하는 독자들은 다소 어렵게 느껴지기때문에, 손에 익을때까지 자주 연습해볼 필요가 있다. 텅 비어있는 스크립트 창에 a라는 객체를 만드는 작업을 해보자. 이때 객체object는 다양한 형태의 자료를 담고있는 바구니라고 생각하자. a라는 객체에 2라는 데이터 하나를 삽입해보자. 명령어 구조를 보면 객체는 왼쪽, 넣을 데이터는 오른쪽에 위치시킨다. 중간의 화살표의 방향을 보면 직관적으로 이해가 가능하다. 스크립트 창에 있는 명령어를 실행시키기 위해서는 해당 명령어를 드래그 한후 "run" 버튼을 누르거나 Ctrl+Enter를 누르면 된다. 명령어로 실행시키고 싶지 않은 comment나 각주는 문장 앞에 #을 삽입하면 된다. 회색 박스안에는 스크립트창, 흰색 박스 안에는 콘솔에 나타나는 output을 보여준다. 한가지 주의해야 할 부분은 R의 실행 구조는 누적이 아니라 덮어쓰기 방식이라는 것이다. 객체 a에 다시 3이라는 데이터를 넣는다고 정의하면, 2의 데이터는 사라지게 된다. 만일 두 개 이상의 데이터를 하나의 객체에 삽입하고 싶다면 c(연결concatenate의 약자)라는 명령어를 사용하자.

```
a<-2 # a라는 객체에 2를 삽입
a #a 객체를 출력
## [1] 2
a<-3
```

```
a
## [1] 3
a<-c(3,4,5)
a
## [1] 3 4 5
```

이상의 설명을 요약하면 R의 언어는 다음과 같은 규칙이 있다

- 화살표의 방향은 데이터 또는 함수로 객체를 정의하는 것을 뜻한다(객체 <- 데이터 또는 함수)
- 문장 앞에 #을 붙이면 명령어로 실행되지 않는다(comment, 각주 등)
- 객체의 이름을 실행시키면, 객체에 담겨있는 데이터가 출력된다
- R의 명령어 실행은 덮어쓰기 방식이다.
- 다수의 데이터를 연결하기 위해서는 c를 사용한다(c(1,2,3) 등)

3.2 R에서 쓰이는 자료의 유형

본격적으로 R의 자료구조를 살펴보기 전에 R에서 쓰이는 자료의 유형에 대해서 알아보자. 연구에서 쓰이는 자료들은 다양한 유형이 있다. 키(168cm, 170cm)와 같은 수치형 자료나, 이름(홍길동, 김영희)과 같은 문자형 자료 등이 여기에 포함된다. 자료의 유형이 중요한 이유는 특정 작업은 특정한 자료의 유형에만 작동하기 때문이다. 예를 들어 덧셈, 뺄셈 등의 연산 작업은 수치형 자료에서만 작동한다. 글자의 앞 한자리만 삭제하는 것은 문자형 자료에만 작동한다. 또한 숫자를 문자형으로 인식한다면 연산 작업은 작동을 하지 않을 것이다. R에서 쓰이는 자료의 유형은 다음과 같이 요약할 수 있다.

- 수치형 값(numeric value) : 소수점을 포함하는 숫자값 (1, 2.2, pi)
- 문자형 값(character value) : 문자로 표현된 값, 큰따옴표로 표현 (“a”, “work”, “1”)
- 복소수형 값(complex value) : 실수와 허수(i)의 합으로 표현한 값(1+4i)
- 논리형 값(logical value) : 참(true) 혹은 거짓(false)으로 출력되는 논리형 값
- 정수형 값(integer value) : 수치형 자료의 특수한 형태, 정수로 표현되는 숫자 (1, 2, 10)

3.3 R에서 쓰이는 자료의 구조

이제 자료구조(data structure)에 대해 알아보자. 자료구조란 간단히 이야기해서 자료가 갖고 있는 골격, 형태를 의미한다. 사회과학에서 쓰이는 상당수의 자료는 행과 열의 구조를 갖고있는 2차원의 매트릭스 형태를 띈다. 간단히 이야기해서 엑셀의 데이터시트를 생각해보자. 행(row) 하나는 개인의 자료 set을 의미한다. 열(column)은 보통 각 개인의 특성을 나타내는 변수를 의미한다. 100명의 사례의 ID, 성별, 시험점수를 조사한 자료를 생각해보면 100 * 3의 매트릭스 형태가 될 것이다. 앞으로 설명할 자료 구조는 이처럼 자료가 갖고 있는 형태와 특성을 의미한다. SPSS나 STATA와 같은 통계 패키지에서는 엑셀 자료와 같은 매트릭스 형태(R에서는 dataframe이라 부른다)만을 사용하지만,R에서는 총 7개의 자료구조가 있다. 조금 복잡하지만 처음부터 제대로 이해해놓는 것이 중요하다.

3.3.1 스칼라 scala

구성인자element가 하나인 자료를 의미한다. 일반적으로 사회과학에서 구성인자가 하나인 데이터를 쓰는 경우는 많지 않다. 따라서 스칼라scala는 이후에 살펴볼 벡터vector의 하위구조로 생각해둘 필요가 있다. 자료를 입력할 때 문자형 자료는 큰따옴표로 정의해주는 것을 염두에 두자.

```
scalar<-1
scalar
## [1] 1
scalar<-"bts"
scalar
## [1] "bts"
```

3.3.2 벡터 vector

구성인자element가 두 개 이상인 자료를 의미한다. 따라서 스칼라는 특수한 형태의 벡터이다. 벡터를 만들때는 c() 명령어를 주로 쓴다. 쉼표로 연결해주면 무한대로 복수의 스칼라를 연결할 수 있다.

```
vector <-c(1,2,3)
vector
## [1] 1 2 3
vector <-c("v", "rm", "suga")
vector
## [1] "v"    "rm"   "suga"
```

3.3.3 매트릭스 matrix

매트릭스는 벡터를 여러 개의 row(행) 또는 column(열)으로 쌓은 자료를 의미한다. 2 by 2, 100 by 100 등의 행렬의 형태가 대표적이다. 벡터가 1차원이라면, 매트릭스는 2차원 형태의 데이터 구조를 띈다. 따라서 매트릭스부터는 생성을 위해 별도의 명령어가 필요하다.

- 매트릭스를 만들기 위한 명령어는 matrix()이다. 대체로 R의 명령어는 이렇게 직관적이다. 괄호 안에 자료에 들어갈 값을 c()를 활용해 지정해주고, 행 또는 열의 개수를 nrow= , ncol= 의 옵션으로 지정해준다.
- 1열(by column)부터 값이 부여된다. 1행(by row)부터 값을 부여하고 싶다면 by-row=TRUE의 옵션을 사용한다.
- matrix() 명령어를 찬찬히 살펴보면 R의 명령어 구조에 대한 힌트를 얻을 수 있다. 다시 말해, 부수적인 옵션들은 쉼표로 연결하는 구조이다. 당연하게도 옵션을 나열하는 순서도 변경가능하다.
- c(1:10)은 1부터 10까지의 수를 차례대로 삽입하라는 뜻이다.

```
matrix <-matrix(c(1,2,3,4,5,6), nrow=3)
matrix
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
matrix <-matrix(c(1,2,3,4,5,6), nrow=2)
matrix
##      [,1] [,2] [,3]
## [1,]    1    3    5
```

```
## [2,]    2    4    6
matrix <-matrix(c(1:20), nrow=4, ncol=5, byrow=TRUE)
matrix
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    2    3    4    5
## [2,]    6    7    8    9   10
## [3,]   11   12   13   14   15
## [4,]   16   17   18   19   20
```

매트릭스는 벡터를 행 또는 열로 쌓은 자료이기 때문에, 실제로 이러한 방식으로 데이터를 만들수도 있다. 즉, 벡터를 연결하는 방식으로 매트릭스를 만들 수 있다. 사회과학에서 쓰는 자료 구조에서 하나의 벡터는 하나의 변수(variable) 또는 하나의 케이스(case)로 이해할 수 있다.

- mat1과 mat2는 각각 1에서 3, 4에서 6의 값을 갖는 벡터이다. 이 벡터를 행 또는 열로 연결하면 매트릭스가 된다.
- 행으로 연결하기 위해서는 rbind(), 열로 연결하기 위해서는 cbind()의 명령어를 사용하면 된다. 행으로 연결한다면 몇 개의 case를 추가하는 것, 열로 연결한다면 몇 개의 변수를 추가하는 것으로 이해할 수 있다.
- c(vector1, vector2)를 사용하게 되면 1차원의 벡터로 만들어진다는 점을 유념하자.

```
mat1 <-c(1:3)
mat2 <-c(4:6)
matrix1 <-rbind(mat1, mat2)  #rbind : row을 기준으로 종으로 붙이기
matrix1
##      [,1] [,2] [,3]
## mat1    1    2    3
## mat2    4    5    6
matrix2 <-cbind(mat1, mat2)  #cbind : column을 기준으로 횡으로 붙이기
matrix2
##      mat1 mat2
## [1,]    1    4
## [2,]    2    5
```

```
## [3,]    3    6
matrix3<-c(mat1, mat2) #c()를 사용하면 벡터와 벡터를 하나의 차원으로 연결
matrix3
## [1] 1 2 3 4 5 6
```

매트릭스에서 추가로 이해해야 할 개념은 특정 요소(element)의 위치를 행과 열의 자릿수로 설명할수 있다는 것이다. "행렬"이라는 이름에서 직관적으로 이해할 수 있듯이 행렬의 원소의 위치는 [n번째 행, k번째 열]의 순서로 표기한다. 원소의 위치를 특정하는 것은 어떠한 작업과 연결될까? 예를 들어 내가 갖고 있는 데이터의 103번째 사례(행번호 103)의 3번째 변수(열번호 3번)를 수정하고 싶을 때 사용할 수 있다.

- 매트릭스의 특정 위치의 원소 추출을 위해서는 대괄호[] 를 사용한다.
- [1,2]는 1번째 행, 2번째 열에 위치를 의미한다
- 쉼표는 "전체"를 의미한다 예를 들어 [1,]는 첫번째 행과 모든 열을 의미한다. 다시 이야기하면 첫번째 행의 모든 원소를 의미한다.
- 복수의 위치를 지정하고 싶다면 만능키인 c()를 사용한다. 행 또는 열 위치에 삽입하면 된다.
- 원소를 치환하고 싶으면 equal(=)을 사용하여 간단히 정의하면 된다.

```
matrix2[1,2]
## mat2
##    4
matrix2[1,] #첫번째 row의 모든 원소를 추출
## mat1 mat2
##    1    4
matrix2[,1] #첫번째 col의 모든 원소를 추출
## [1] 1 2 3
matrix2[c(1,2),] #1,2번째 row의 모든 원소를 추출
##      mat1 mat2
## [1,]    1    4
## [2,]    2    5
matrix2[1,2]=100 # 첫번째 행, 두 번째 열의 원소를 100으로 치환한다.
matrix2
```

```
##      mat1 mat2
## [1,]    1  100
## [2,]    2    5
## [3,]    3    6
```

3.3.4 배열 array

array는 matrix를 여러 층으로 쌓은 것이다. matrix가 2차원 구조이므로, array는 3차원 구조이다. 행렬로 표현된 데이터를 켜켜히 쌓아올린다고 생각하면 된다. 통상 사회과학연구에서 자주 볼수 없는 데이터 구조이나, 시계열적인 자료나 청키한 데이터들이 array의 형태를 띈다.

- array를 생성하는 명령어는 array()이다.
- 보통 2개 이상의 매트릭스를 연결하여 만든다(c(matrix1, matrix2, ….))
- matrix와 유사하게 dimension의 구조도 옵션으로 제시해준다. dim=c()의 명령어를 사용한다.

```
matrix1<- matrix(c(1:9), nrow=3)
matrix1
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
matrix2<- matrix(c(10:18), nrow=3)
matrix3<- matrix(c(19:27), nrow=3)
matrix2
##      [,1] [,2] [,3]
## [1,]   10   13   16
## [2,]   11   14   17
## [3,]   12   15   18
matrix3
##      [,1] [,2] [,3]
## [1,]   19   22   25
```

```
## [2,]   20   23   26
## [3,]   21   24   27
array <-array(c(matrix1, matrix2, matrix3), dim=c(3,3,3))
array
## , , 1
##
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
##
## , , 2
##
##      [,1] [,2] [,3]
## [1,]   10   13   16
## [2,]   11   14   17
## [3,]   12   15   18
##
## , , 3
##
##      [,1] [,2] [,3]
## [1,]   19   22   25
## [2,]   20   23   26
## [3,]   21   24   27
```

3.3.5 데이터프레임 dataframe

지금까지 살펴본 vector, matrix, array는 모두 같은 유형의 데이터로만 구성되어 있다. 즉 문자형(character), 논리형(logic), 숫자형(numeric) 등 통일된 한종류로만 구성이 되어 있다. 우리가 일반적으로 쓰는 데이터는 문자형 변수, 숫자형 변수 등이 혼재되어 하나의 데이터셋에 담겨있다. 이러한 경우 R은 데이터 프레임(dataframe)이라는 별도의 데이터 구조를 사용한다. 앞으로 우리가 사용할 대부분의 데이터는 데이터프레임일 것이다.

간단한 형태의 데이터 프레임을 직접 만들어 보자. 저자가 좋아하는 방탄소년단의 정보를 하나의 자료로 구성해보겠다. 방탄소년단 멤버들의 이름(문자형변수), 생년(숫자형 변수), 포지션("반복"되는 문자형변수) 등 이다.

- 데이터 프레임을 만드는 명령어는 data.frame()이다.
- 통상적으로 c(원소1, 원소2…)로 벡터를 만들면 횡이 아니라 종의 방향의 벡터가 만들어진다 따라서, 각각의 벡터는 하나의 열(column)이 된다. 사회과학분야에서는 주로 변수(variable)가 된다.
- bts라는 데이터 프레임을 만들면, R studio의 오른쪽 상단의 Environment 패널에 해당 데이터 프레임이 생성된다. 더블클릭하게 되면 명령어 창에 우리게 친숙한 형태의 데이터시트가 나타난다.
- str()은 데이터 프레임의 구조(structure)를 보여준다. 3개의 변수를 가진 7개의 관측치(observation)를 가지고 있으며, 각각의 변수들을 요약해서 보여주고 있다.
- bts라는 데이터 프레임의 3개의 변수명 앞에 $ 표시가 있는것을 기억하자. $ 는 변수를 의미하는 표시로 앞으로 자주 사용하게 될 것이다.
- 변수별로 chr, num 등의 약어가 제시되는데, 이는 자료의 유형(문자형(character), 수치형(numeric) 등)을 의미한다.
- 어떠한 데이터프레임이던 분석을 시작하기 전에 반드시 str()를 사용해서 자료 구조를 확인하는 것이 좋다.
- stringAsFactors=FALSE는 문자형(string) 변수를 factor 변수로 처리하지 말라는 뜻이다. factor 변수에 대해서는 아래에서 설명할 예정이다.

```
btsname <-c("RM", "Jin", "Suga","Jhope", "Jimin", "V", "JK")
btsyear <-c(1994, 1992, 1993, 1994, 1995, 1995, 1997)
btsposition <-c("rap", "vocal", "rap", "rap", "vocal", "vocal","vocal")
bts <-data.frame(btsname, btsyear, btsposition, stringsAsFactors = FALSE)
bts
##   btsname btsyear btsposition
## 1      RM    1994         rap
## 2     Jin    1992       vocal
## 3    Suga    1993         rap
## 4   Jhope    1994         rap
## 5   Jimin    1995       vocal
```

```
## 6       V    1995       vocal
## 7      JK    1997       vocal
str(bts)
## 'data.frame':    7 obs. of  3 variables:
##  $ btsname    : chr  "RM" "Jin" "Suga" "Jhope" ...
##  $ btsyear    : num  1994 1992 1993 1994 1995 ...
##  $ btsposition: chr  "rap" "vocal" "rap" "rap" ...
```

3.4 factor 변수

bts 데이터 프레임에서 btsname변수와 btsposition 변수는 모두 문자형 변수이지만 차이점이 있다. btsposition 변수는 "rap"과 "vocal"이라는 두개의 값(value)이 반복된다. 이러한 형태의 변수를 R에서는 요인(factor)라는 특별한 데이터유형으로 취급한다. 사회과학연구에서 주로 사용하는 요인분석에서의 요인과는 구별되는 개념이다. R에서의 factor 변수는 주로 범주형 변수이다. 흔히 사용되는 변수 중에 성별("남", "여"), 학년("1학년","2학년", "3학년"), 학업성취도("상", "중", "하") 등이 factor 변수의 대표적인 예다. 범주변수를 factor로 변환하기 위해서는 다음의 사항을 기억하자.

- factor 변수로 지정하기 위해서는 factor() 명령어를 사용한다.
- factor 변수는 "값(일반 벡터)"에 "level"이라는 정보를 추가한 것이다. default로 level의 값이 부여되지만 이를 수정할 수도 있다.
- level의 순서는 알파벳 순서가 default이다.
- 경우에 따라서는 level의 순서를 바꾸고 싶을 때가 있다. 예를 들어 성별의 경우 알파벳 순서에 따라 female, male의 순서가 default이다. 이후에 그래프 등을 그릴 때 이 순서를 따르기 때문에 levels=c() 명령어를 사용해서 새롭게 지정이 가능하다.

factor() 명령어를 활용하여 btspostiion 변수를 문자형 변수에서 factor 변수로 변환을 해보자. str()를 활용하여 자료의 구조를 살펴보면 factor로 잘 변환되어 있는것을 확인할 수 있다. factor로 변환하는 순간 원래의 데이터가 숫자의 정보로 변하고(1, 2,1,1,2,2,2), level(1=rap, 2=vocal)의 정보가 추가로 생성된다. 만약에 1=vocal, 2=rap의 순서로 바꾸고 싶다면, levels=c("vocal", "rap")의 옵션을 추가하면 된다.

```
bts$btsposition <-factor(btsposition)
str(bts$btsposition)
##  Factor w/ 2 levels "rap","vocal": 1 2 1 1 2 2 2
levels(bts$btsposition)
## [1] "rap"   "vocal"
bts$btsposition <-factor(btsposition, levels=c("vocal", "rap"))
str(bts$btsposition)
##  Factor w/ 2 levels "vocal","rap": 2 1 2 2 1 1 1
summary(bts$btsposition)
## vocal   rap
##     4     3
```

factor 변수를 활용할 때 조심해야할 것들이 있다. 문자형변수를 수치형변수 +level의 정보로 축약하기 때문이다. 아래와 같이 as.numeric() 명령어를 활용해서 변수를 팩터에서 숫자형으로 변환해보면, 팩터의 원래값이 나타난다.

```
bts$btsposition <- as.numeric(bts$btsposition)
str(bts$btsposition)
##  num [1:7] 2 1 2 2 1 1 1
```

이러한 특징은 가끔 팩터형 자료를 붙이거나 자를때 문제를 일으키는 경우가 많다. 따라서 전처리 과정에서는 계속해서 문자형 변수로 두다가, 통계적 분석 과정 직전에(즉, 데이터 전처리가 모두 끝난 후에) 팩터형 변수로 바꾸는 것을 권장한다. 팩터형 변수를 다루는 것이 까다롭기 때문에 종종 별도의 패키지를 쓰곤한다. 대표적인것이 FORCAT 패키지인데, 이는 3장에서 다시 구체적으로 다루도록 하겠다.

3.5 NA와 NULL

마지막으로 R에서 결측치를 표현하는 두가지 방식에 대해 이해해보도록 하자. 어떠한 데이터든지 결측치는 흔하게 존재한다. 특히 다른 곳에서 수집된 자료를 2차 가공을 하는 경우에는 더욱 빈번하게 출현한다. R에서 벡터 또는 데이터 프레임에서 비어있는 값, 결측치를 표현하는 방식은 다음과 같다.

- NA는 not available의 약자로, 결측치를 의미한다.

- NA는 우리가 사용하는 데이터에서 흔히 볼 수 있는 결측치이기 때문에 특정 변수(벡터)의 한 요소(element)로 존재한다. 원래 있어야하는 값이 기 때문에 NA는 평균 등 통계량 산출에 영향을 미친다.
- NA를 무시하고 통계량을 계산하고 싶다면 na.rm=TRUE 옵션을 명령어 뒤에 붙이면 된다. na.rm은 NA를 제거(removing)하라는 뜻이다.
- NULL은 원래 존재하지 않는 값을 의미한다. NA와 달리 벡터 자체가 정의되지 않은 것이라는 점을 이해해야 한다. 일반적으로는 특정한 목적으로 데이터를 담지 않은 ("즉, 텅 비어있는") 객체(object)를 만들어야 할 때 사용된다.

데이터 분석시 자주 활용하게 될 NA를 중심으로 살펴보면 아래와 같다. age 변수의 첫번째 값을 결측치(NA)로 변경하면, 평균값을 산출할때 조심할 필요가 있다. na.rm=FALSE가 기본옵션(default)이기 때문에 mean() 함수를 사용하면 NA가 산출된다. 반면에 na.rm=TRUE 옵션을 사용하게 되면 결측치를 제거하고 6개의 자료의 평균을 산출해 준다.

```
#btsyear 변수를 활용(computation)해서 age 변수를 새로 만든다
bts$age <- 2021-bts$btsyear+1
bts
##   btsname btsyear btsposition age
## 1      RM    1994         rap  28
## 2     Jin    1992       vocal  30
## 3    Suga    1993         rap  29
## 4   Jhope    1994         rap  28
## 5   Jimin    1995       vocal  27
## 6       V    1995       vocal  27
## 7      JK    1997       vocal  25
bts[1,4] <-NA
bts
##   btsname btsyear btsposition age
## 1      RM    1994         rap  NA
## 2     Jin    1992       vocal  30
```

```
## 3    Suga    1993        rap  29
## 4   Jhope    1994        rap  28
## 5   Jimin    1995      vocal  27
## 6       V    1995      vocal  27
## 7      JK    1997      vocal  25
mean(bts$age)
## [1] NA
mean(bts$age, na.rm=TRUE)
## [1] 27.66667
```

3.6 R 내장함수를 활용한 기술통계량 산출

장을 마무리하기 전에 데이터의 간단한 통계량을 산출하는 방법을 알아보자. 좀 더 복잡한 분석은 다음 장에서 다룰 dplyr 패키지를 활용하는 것이 좋다. 그러나 R에 내장되어 있는 함수를 활용해서도 내가 갖고 있는 데이터 프레임의 구조를 확인하고, 간단한 통계량을 확인할 수 있다. R 내장함수중에 빈번하게 사용되는 명령어를 정리해보면 아래와 같다.

데이터 구조 및 요약

- str() : 데이터 프레임의 사례수, 변수, 변수별 자료 유형등을 제시해준다
- summary() : 데이터프레임의 각 변수별 최솟값, 최대값, 길이, 평균, 사분위 수등을 제시해준다.

기술통계량 확인

- mean() : 데이터 프레임, 또는 데이터 프레임의 특정변수(dataframe$variable)의 평균값
- median(): 데이터 프레임, 또는 데이터 프레임의 특정변수의 중앙값
- min(), max() : 데이터프레임 또는 특정변수의 최솟값, 최대값
- var(), sd() : 데이터 프레임 또는 특정변수의 분산, 표준편차
- sum() : 데이터 프레임 또는 특정변수의 합
- length() : 길이, 관측값의 갯수

테이블 또는 교차표 산출

- table(): 데이터 프레임의 각 변수별 분할표를 제시
- table(변수, 변수) : $인자를 활용해서 특정변수간의 교차표 생성

- addmargin() : 마진에 소계값을 계산
- prop.table() : 테이블의 비율 계산
- margin= : prop.table()의 옵션, 1의 값인 경우 행(row)의 비율 합을 1로, 2의 값인 경우 열(colomun)의 비율 합을 1로 계산

```
summary(bts)
##    btsname             btsyear      btsposition       age
##  Length:7           Min.   :1992   vocal:4     Min.   :25.00
##  Class :character   1st Qu.:1994   rap  :3     1st Qu.:27.00
##  Mode  :character   Median :1994               Median :28.00
##                     Mean   :1994               Mean   :27.71
##                     3rd Qu.:1995               3rd Qu.:28.50
##                     Max.   :1997               Max.   :30.00
str(bts)
## 'data.frame':    7 obs. of  4 variables:
##  $ btsname    : chr  "RM" "Jin" "Suga" "Jhope" ...
##  $ btsyear    : num  1994 1992 1993 1994 1995 ...
##  $ btsposition: Factor w/ 2 levels "vocal","rap": 2 1 2 2 1 1 1
##  $ age        : num  28 30 29 28 27 27 25
table(bts$age)
##
## 25 27 28 29 30
##  1  2  2  1  1
#소숫점 두번째에서 반올림
round(prop.table(table(bts$age)),2)
##
##   25   27   28   29   30
## 0.14 0.29 0.29 0.14 0.14
```

```
#분할표를 백분율로 계산
round(prop.table(table(bts$age)),2)*100
##
## 25 27 28 29 30
## 14 29 29 14 14
#마진에 소계값을 계산
addmargins(table(bts$age))
##
##  25  27  28  29  30 Sum
##   1   2   2   1   1   7
# 변수 * 변수의 교차표를 산출
table(bts$age, bts$btsposition)
##
##     vocal rap
##  25     1   0
##  27     2   0
##  28     0   2
##  29     0   1
##  30     1   0
# 교차표의 셀별 비율 계산(열의 합을 1로)
prop.table(table(bts$age, bts$btsposition), margin=2)
##
##         vocal       rap
##  25 0.2500000 0.0000000
##  27 0.5000000 0.0000000
##  28 0.0000000 0.6666667
##  29 0.0000000 0.3333333
##  30 0.2500000 0.0000000
```

제 4 장

Parts

You can add parts to organize one or more book chapters together. Parts can be inserted at the top of an .Rmd file, before the first-level chapter heading in that same file.

Add a numbered part: # (PART) Act one {-} (followed by # A chapter)

Add an unnumbered part: # (PART*) Act one {-} (followed by # A chapter)

Add an appendix as a special kind of un-numbered part: # (APPENDIX) Other stuff {-} (followed by # A chapter). Chapters in an appendix are prepended with letters instead of numbers.

제 5 장

Footnotes and citations

5.1 Footnotes

Footnotes are put inside the square brackets after a caret ^[]. Like this one [1].

5.2 Citations

Reference items in your bibliography file(s) using @key.

For example, we are using the **bookdown** package [Xie, 2022] (check out the last code chunk in index.Rmd to see how this citation key was added) in this sample book, which was built on top of R Markdown and **knitr** [Xie, 2015] (this citation was added manually in an external file book.bib). Note that the .bib files need to be listed in the index.Rmd with the YAML bibliography key.

The RStudio Visual Markdown Editor can also make it easier to insert citations: https://rstudio.github.io/visual-markdown-editing/#/citations

[1] This is a footnote.

제 6 장

Blocks

6.1 Equations

Here is an equation.

$$f\left(k\right) = \binom{n}{k} p^k \left(1-p\right)^{n-k} \tag{6.1}$$

You may refer to using \@ref(eq:binom), like see Equation (6.1).

6.2 Theorems and proofs

Labeled theorems can be referenced in text using \@ref(thm:tri), for example, check out this smart theorem 6.1.

Theorem 6.1. *For a right triangle, if c denotes the* length *of the hypotenuse and a and b denote the lengths of the* ***other*** *two sides, we have*

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

Read more here https://bookdown.org/yihui/bookdown/markdown-extensions-by-bookdown.html.

6.3 Callout blocks

The R Markdown Cookbook provides more help on how to use custom blocks to design your own callouts: https://bookdown.org/yihui/rmarkdown-cookbook/custom-blocks.html

제 7 장

Sharing your book

7.1 Publishing

HTML books can be published online, see: https://bookdown.org/yihui/bookdown/publishing.html

7.2 404 pages

By default, users will be directed to a 404 page if they try to access a webpage that cannot be found. If you'd like to customize your 404 page instead of using the default, you may add either a _404.Rmd or _404.md file to your project root and use code and/or Markdown syntax.

7.3 Metadata for sharing

Bookdown HTML books will provide HTML metadata for social sharing on platforms like Twitter, Facebook, and LinkedIn, using information you provide in the index.Rmd YAML. To setup, set the url for your book and the path to your cover-image file. Your book's title and description are also used.

This gitbook uses the same social sharing data across all chapters in your book- all links shared will look the same.

Specify your book's source repository on GitHub using the edit key under the configuration options in the _output.yml file, which allows users to suggest an edit by linking to a chapter's source file.

Read more about the features of this output format here:

https://pkgs.rstudio.com/bookdown/reference/gitbook.html

Or use:

```
?bookdown::gitbook
```

참고 문헌

Yihui Xie. *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition, 2015. URL http://yihui.org/knitr/. ISBN 978-1498716963.

Yihui Xie. *bookdown: Authoring Books and Technical Documents with R Markdown*, 2022. URL https://CRAN.R-project.org/package=bookdown. R package version 0.26.