

한국한의학연구원, 구본초

---

## 통계 패키지 활용

2020년도 2학기 충남대학교 정보통계학과 강의 노트



---

# *Contents*

---

<b>List of Tables</b>	<b>v</b>
<b>List of Figures</b>	<b>vii</b>
<b>Course Overview</b>	<b>ix</b>
<b>1 R Markdown</b>	<b>1</b>
1.1 R Markdown의 구성 . . . . .	2
1.2 R Markdown 문서 시작하기 . . . . .	6
1.3 R Markdown 기본 문법(syntax) . . . . .	9
1.3.1 텍스트 문법 . . . . .	9
1.3.2 Block-level elements . . . . .	12
1.3.3 수식표현(math expression) . . . . .	14
1.4 R Code Chunks . . . . .	16
1.5 인라인(inline) R 코드 . . . . .	31
1.6 YAML . . . . .	32
1.7 참고문헌 인용 . . . . .	34
<b>2 제어문(Control Structure)</b>	<b>37</b>
2.1 Prerequisite . . . . .	38
2.2 프로그램 . . . . .	40
2.3 조건문(Conditionals) . . . . .	44

2.3.1	기본 구문 . . . . .	45
2.3.2	연쇄 조건문 (chained condition) . . . . .	47
2.3.3	중첩 조건문 (nested condition) . . . . .	48
2.3.4	<code>ifelse()</code> 함수 . . . . .	51
2.4	반복문 (Looping) . . . . .	53
2.4.1	<code>repeat</code> 구문 . . . . .	53
2.4.2	<code>while</code> 구문 . . . . .	57
2.4.3	<code>for</code> 구문 . . . . .	60
2.5	2.5. 함수 (function) . . . . .	64
2.6	2.6. 제어 구문 이해를 위한 몇 가지 알고리즘 . . . . .	64
3	시뮬레이션	65

---

---

## *List of Tables*

---

1.1	코드 실행 관련 첨크	17
1.2	소스 코드 출력 결과 관련 첨크	19
1.3	코드 서식 관련 첨크	25
1.4	Plot 출력 관련 첨크	27
2.1	R 예약어 종류 및 설명	38



---

## *List of Figures*

---

1.1 R markdown 세계 ( <a href="https://ulyngs.github.io/rmarkdown-workshop-2019">https://ulyngs.github.io/rmarkdown-workshop-2019</a> 에서 발췌) . . . . .	1
1.2 R markdown structure . . . . .	4
1.3 R Markdown의 최종 결과물 산출과정 ( <a href="http://applied-r.com/project-reporting-template/">http://applied-r.com/project-reporting-template/</a> ) . . . . .	5
1.4 test.html 문서 화면(저장 폴더 내 ‘test.html’을 크롬 브라우저로 실행) . . . . .	8
1.5 장난꾸러기 . . . . .	11
1.6 Chunk anatomy ( <a href="https://ulyngs.github.io/rmarkdown-workshop-2019">https://ulyngs.github.io/rmarkdown-workshop-2019</a> 에서 발췌) . . . . .	16
1.7 청크 옵션 results = ‘markup’인 경우 rmd vs. md 파일 비교 . . . . .	22
1.8 청크 옵션 results = ‘asis’인 경우 rmd vs. md 파일 비교 . . . . .	22
1.9 Taj Mahal . . . . .	29
1.10 Scatterplot of the car dataset . . . . .	30
2.1 Flow-control example ( <a href="https://homerhanumat.github.io/r-notes/flow.html">https://homerhanumat.github.io/r-notes/flow.html</a> ) . . . . .	37
2.2 if 구문 기본 flow-chart . . . . .	45
2.3 대안실행(if-else 구문) flow-chart . . . . .	47
2.4 연쇄조건(if-else if-else 구문) flow-chart . . . . .	49

2.5 중첩 조건문 flow-chart . . . . .	50
2.6 REPEAT 구문 flow-chart . . . . .	55
2.7 WHILE 구문 flow-chart . . . . .	57
2.8 FOR 구문 flow-chart . . . . .	61

## ***Course Overview***

R을 이용한 데이터 분석 시 CRAN에 등록된 패키지를 활용한다. 적절한 패키지의 활용은 데이터 분석의 효율을 증대할 뿐 아니라 분석의 재현성을 향상할 수 있다. 본 강의는 지난학기에 학습한 통계프로그래밍언어 강의 내용의 연속선상에서 진행할 예정이며, 해당 강의에서 학습한 내용들을 기반으로 데이터 분석 및 그 결과에 대한 보고서 작성, 그리고 R 생성 파일에 대한 버전 관리 방법에 대해 알아보고자 한다.

### **교과 목표**

- R Markdown의 이해와 활용
- R 프로그래밍 능력 향상 및 통계 시뮬레이션의 이해
- R을 이용한 데이터 분석 실습
- R을 이용한 기초 통계분석
- 텍스트 마이닝에 대한 이해
- Shiny, plotly 를 활용한 동적 문서 및 시각화 이해
- RStudio + Github을 이용한 버전관리 이해

## 선수과목

통계학 개론 통계 프로그래밍 언어

## 수업 방법

- 강의: 30 %
- 실험/실습: 70 %

## 평가방법

- 중간고사: 35 %
- 기말고사: 35 %
- 출석: 10 %
- 과제: 20 %

## 교재

별도의 교재 없이 본 강의 노트로 수업을 진행할 예정이며, 수업의 이해도 향상을 위해 아래 소개할 도서 및 웹 문서 등을 참고할 것을 권장함.

## 참고문헌

- R Markdown Cookbook<sup>1</sup> (Xie et al., 2020)
- bookdown: Authoring Books and Technical Documents with R Markdown<sup>2</sup> (Xie, 2016)

<sup>1</sup><https://bookdown.org/yihui/rmarkdown-cookbook/>

<sup>2</sup><https://bookdown.org/yihui/bookdown/>

- R과 knitr를 활용한 데이터 연동형 문서 만들기 (고석범, 2014)
- R for data science<sup>3</sup> (Wickham and Grolemund, 2016)
- Statistical Computing with R (Rizzo, 2019)
- R programming for data science<sup>4</sup> (Peng, 2016)
- Text mining with R<sup>5</sup> (Silge and Robinson, 2017)

---

<sup>3</sup><https://r4ds.had.co.nz/>

<sup>4</sup><https://bookdown.org/rdpeng/rprogdatascience/>

<sup>5</sup><https://www.tidytextmining.com/>

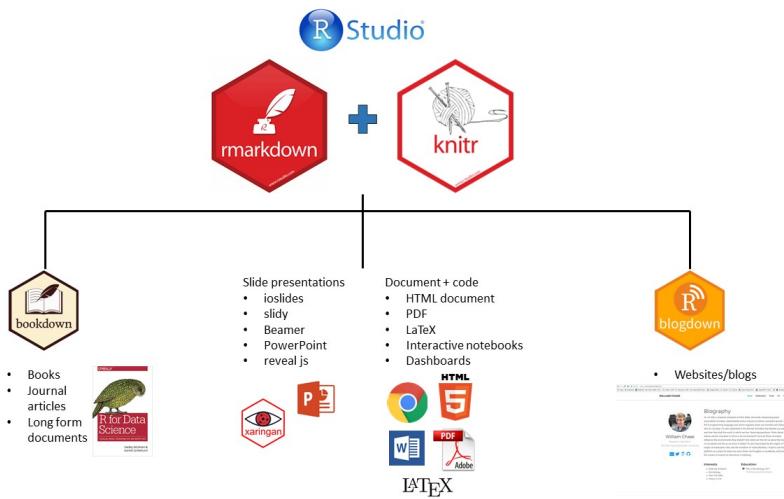


# 1

## R Markdown

### Sketch

- 동일한 문서에 코드, 결과, 텍스트가 동시에 있을 수 있을까?
- 만약 결과와 도표가 자동으로 생성된 경우 데이터가 변경 되더라도 자동으로 문서를 업데이트 할 수 있을까?
- 최종 완료한 문서가 미래에도 열 수 있을까?
- 이러한 모든 과정이 매우 쉽다면??



**FIGURE 1.1:** R markdown 세계 (<https://ulyngs.github.io/rmarkdown-workshop-2019>에서 발췌)

## 1.1 R Markdown의 구성



본 절의 내용 중 일부는 지난 학기 강의노트 1.7절과 중복되거나 재구성한 내용이 포함됨.

1. R Markdown은 R 코드와 분석 결과(표, 그림 등)을 포함한 문서 또는 컨텐츠를 제작하는 도구로 일반적으로 아래 열거한 형태로 활용함
  - 문서 또는 논문(pdf, html, docx)
  - 프리젠테이션(pdf, html, pptx)
  - 웹 또는 블로그
2. 재현가능(reproducible)한 분석 및 연구<sup>1</sup> 가능
  - 신뢰성 있는 문서 작성
  - Copy & paste를 하지 않고 효율적 작업 가능

R 마크다운 파일 = .Rmd 확장자를 가진 일반 텍스트 파일

```
---
```

```
title: "Untitled.Rmd"
```

```
date: "2020-09-11"
```

```
output: html_document
```

```
---
```

```
```{r setup, include=FALSE}
```

```
knitr::opts_chunk$set(echo = TRUE)
```

<sup>1</sup>과학적 연구의 결과물을 오픈소스로 내놓고 누구라도 검증 가능

```
```  
## R Markdown
```

Markdown은 HTML, PDF 및 MS Word 문서를 작성하기 위한 간단한 형식 지정 구문입니다.

R Markdown 사용에 대한 자세한 내용은 <<http://rmarkdown.rstudio.com>>을 참조하십시오.

\*\*Knit\*\* 버튼을 클릭하면 두 가지를 모두 포함하는 문서가 생성됩니다.

문서에 포함된 R 코드 청크의 출력 내용뿐 아니라  
다음과 같이 R 코드 청크를 포함 할 수 있습니다.

```
```{r cars}  
summary(cars)  
```
```

```
## Including Plots
```

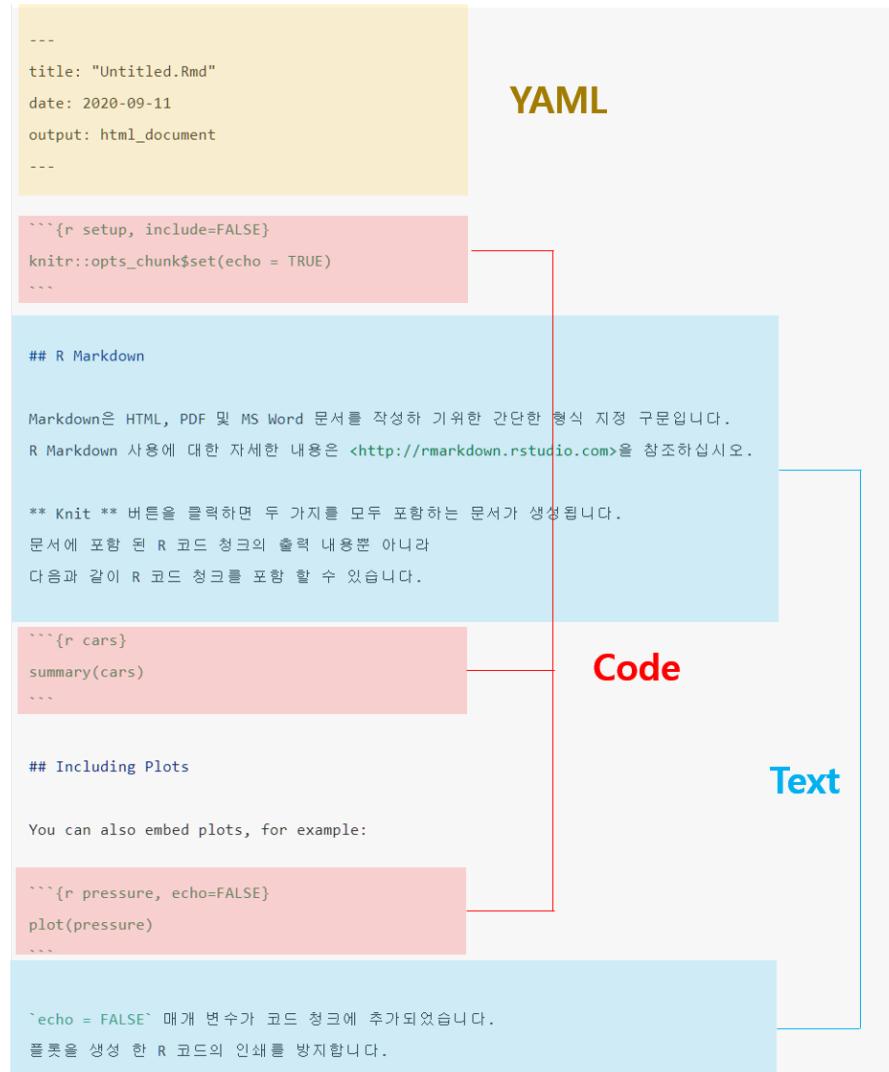
You can also embed plots, for example:

```
```{r pressure, echo=FALSE}  
plot(pressure)  
```
```

`echo = FALSE` 매개 변수가 코드 청크에 추가되었습니다.

플롯을 생성한 R 코드의 인쇄를 방지합니다.

위 R Markdown 문서는 아래 그림과 같이 **YAML**, **Markdown** 텍스트, **Code Chunk** 세 부분으로 구성됨.



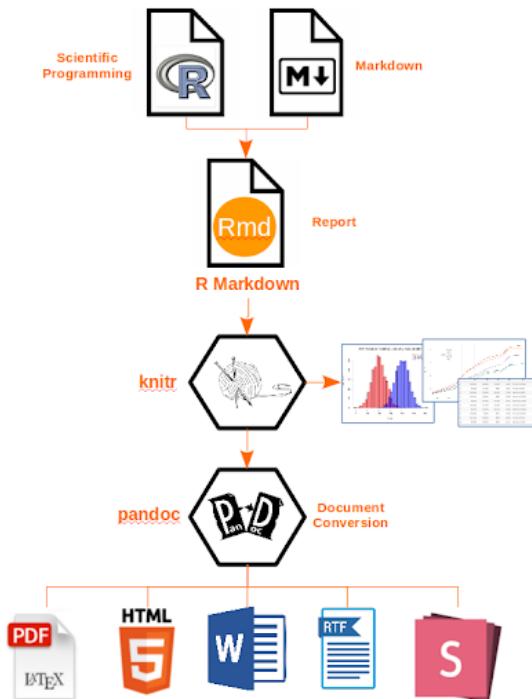
**FIGURE 1.2:** R markdown structure

**YAML (YAML Ain't Markup Language)**

- R Markdown 문서의 metadata로 문서의 맨 처음에 항상 포함(header)되어야 함.
- R Markdown 문서의 최종 출력 형태(html, pdf, docx, pptx 등), 제목, 저자, 날짜 등의 정보 등을 포함

### 최종 문서 생성 과정

- Rmd 파일을 knitr을 통해 .md 파일로 변환 후 pandoc이라는 문서 변환기를 통해 원하는 문서 포맷으로 출력



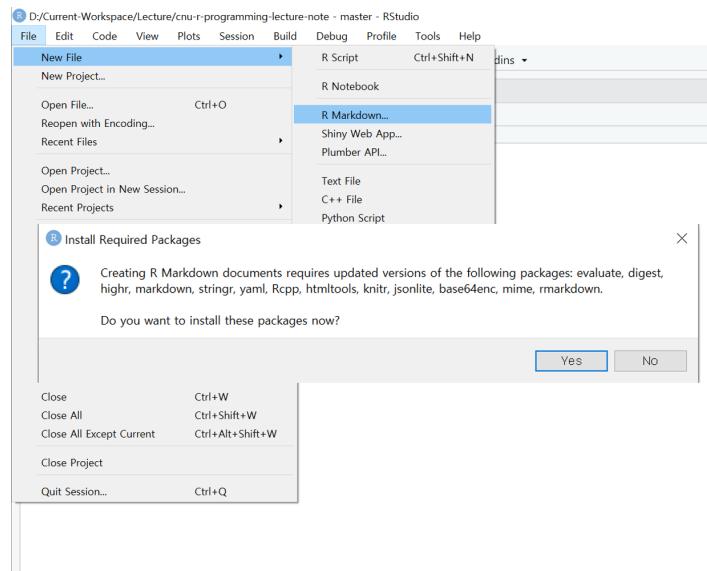
**FIGURE 1.3:** R Markdown의 최종 결과물 산출과정 (<http://applied-r.com/project-reporting-template/>)

## 1.2 R Markdown 문서 시작하기

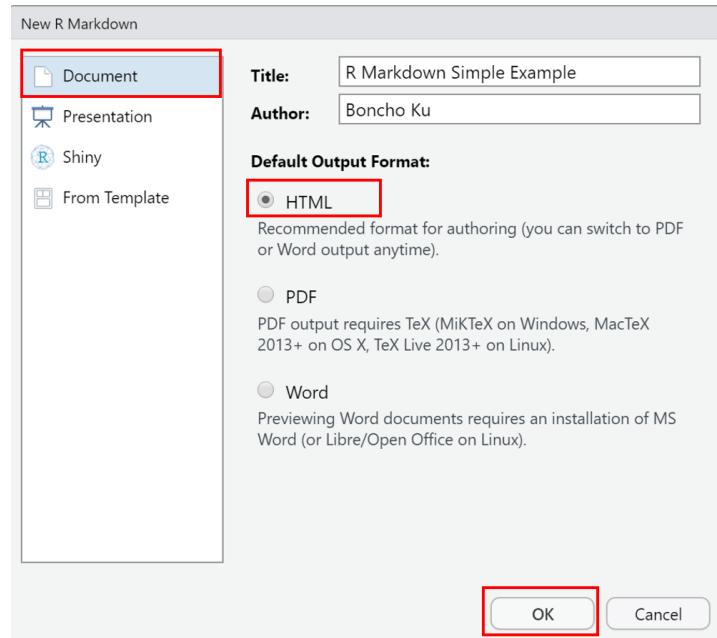
- R Markdown 문서 생성 : [File] -> [New File] -> [R Markdown..]을 선택



RStudio를 처음 설치하고 위와 같이 진행할 경우 아래와 같은 패키지 설치 여부를 묻는 팝업 창이 나타남. 패키지 설치 여부에 [Yes]를 클릭하면 R Markdown 문서 생성을 위해 필요한 패키지들이 자동으로 설치



- 설치 완료 후 R Markdown으로 생성할 최종 문서 유형 선택 질의 창이 나타남. 아래 창에서 제목 (Title)과 저자 (Author) 이름 입력 후 [OK] 버튼 클릭 (Document, html 문서 선택)



- 아래 그림과 같이 새로운 문서 창이 생성되고 test.Rmd 파일로 저장<sup>2</sup>

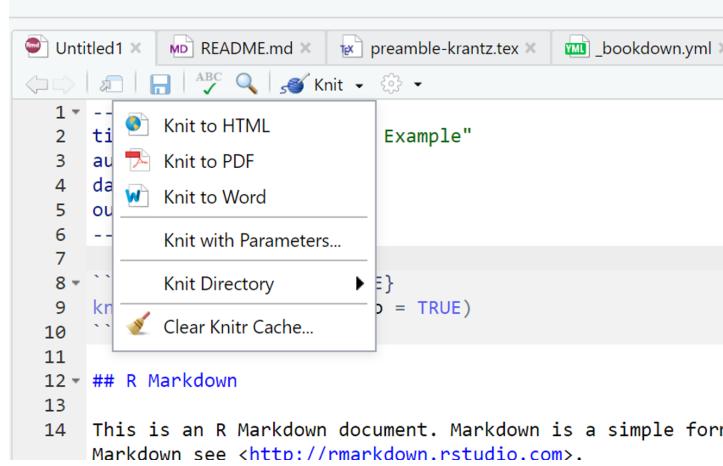
```

1: # Document
2: title: "R Markdown Simple Example"
3: author: "Boncho Ku"
4: date: "2020-3-1"
5: output: html_document
6: ...
7: ...
8: ```{r setup, include=FALSE}
9: knitr::opts_chunk$set(echo = TRUE)
10: ...
11: ...
12: ## R Markdown
13: ...
14: This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R
15: Markdown see <http://rmarkdown.rstudio.com>.
16: ...
17: When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks
18: within the document. You can embed an R code chunk like this:
19: ...
20: ...
21: ...
22: ## Including Plots
23: ...
24: You can also embed plots, for example:
25: ...
26: ...
27: ...
28: ...
29: ...
30: Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.
31: ...

```

- 문서 상단에 Knit 아이콘을 클릭 후 Knit to HTML 클릭 또는 문서 아무 곳에 커서를 위치하고 단축키 [Ctrl] + [Shift] + [K] 입력

<sup>2</sup>[RStudio 프로젝트]에서 생성한 폴더 내에 파일 저장



- knitr + R Markdown + pandoc → html 파일 생성 결과

# R Markdown Simple Example

Boncho Ku

2020 3 17

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```

##      speed          dist
##  Min.   : 4.0   Min.   : 2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   :42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00

```

## Including Plots

You can also embed plots, for example:

**FIGURE 1.4:** test.html 문서 화면(저장 폴더 내 ‘test.html’을 크롬 브라우저로 실행)

### 1.3 R Markdown 기본 문법 (syntax)

R Markdown의 기본 문법은 Rstudio 풀다운 메뉴 [Help] → [Markdown Quick Reference]에서 확인 가능

#### 1.3.1 텍스트 문법

##### 강조(emphasis)

- 텔릭체: \*italic1\*, \_italic2\_ → italic1, italic2
- 볼드(굵은)체: \*bold1\*, \_\_bold2\_\_ → bold1, bold2

##### Inline code

- ‘inline code’ → inline code

##### 아래/위 첨자(sub/superscript)

- subscript~2~ → subscript<sub>2</sub>
- superscript<sup>~2~</sup> → superscript<sup>2</sup>

##### 삭제표시(strike through)

- ~~strikethrough~~ → strikethrough

##### 생략표시(ellipsis)

- ... → ...

##### 긴/짧은 대쉬(en/emd-dash)

- 짧은 대쉬: -- → -
- 긴 대쉬: --- → —

### 특수문자 탈출 지정자

- \\*, \\_, \~, \\ → \*, \_\_, ~, \

### 하이퍼링크

- [text] (link) → 통계프로그래밍언어<sup>3</sup>

### 외부그림 삽입

- ! [image title] (path/to/image): ! [장 난 꾸 러 기] (figures/son-02.jpg)

### 강제 줄바꿈 (line breaks)

- 하나의 줄에서 공백(space) 두 개 이상 또는 백슬레시(\) 입력 후 [Enter]

```
End a line with two spaces to start
a new paragraph
```

End a line with two spaces to start a new paragraph

```
End a line with two spaces to start\
a new paragraph
```

End a line with two spaces to start  
a new paragraph

### 각주 (footnote)

- A footnote<sup>3</sup>[주석내용] → A footnote<sup>4</sup>

---

<sup>3</sup><https://zorba78.github.io/cnu-r-programming-lecture-note>

<sup>4</sup>주석내용



**FIGURE 1.5:** 장난꾸러기

### 주석 (comment)

- <!-- this is a comment that won't be shown --> →



RStudio에서 단축키 [Ctrl] + [Shift] + [C]를 통해 전체 line에 대해 주석처리 가능

### 1.3.2 Block-level elements

#### 장/절(header)

- # Header 1 (chapter, 장)
- ## Header 2 (section, 절)
- ### Header 3 (subsection, 관)

#### 목록(list)

- 비순서(unordered) 목록: -, \*, + 중 어느 하나로 입력 가능

```
- one item
* two item
  + sub-item 1
  + sub-item 2
    - subsub-item 1
    - subsub-item 2
```

- one item
- two item
  - sub-item 1
  - sub-item 2
    - \* subsub-item 1
    - \* subsub-item 2
- 순서(ordered) 목록: 비순서 목록의 기호 대신 숫자로 리스트 생성

```
1. the first item
  - sub-item 1
```

```
2. the second item  
3. the third item
```

1. the first item
  - sub-item 1
2. the second item
3. the third item
  - 같은 숫자로 적어도 순서대로 목록 생성

```
1. the first item  
- sub-item 1  
1. the second item  
1. the third item
```

1. the first item
  - sub-item 1
2. the second item
3. the third item

### 인용구(blockquote): >로 시작

```
> "There are three kinds of lies: lies, damn lies, and statistics"  
>  
> --- Benjamin Disraeli
```

"There are three kinds of lies: lies, damn lies, and statistics"

— Benjamin Disraeli

### 1.3.3 수식표현 (math expression)

- 줄 안에 수식 입력 시 \$수식표현\$ 으로 입력
- 수식 display style (보통 교과서에 정리 및 정의에 기술된 수식들) 적용 시  
\$\$ ~ \$\$ 안에 수식 입력
- 수식 표현은 LaTeX 의 수식 표현을 동일하게 준용 (<https://www.latex4technics.com/>, <https://latex.codecogs.com/legacy/eqneditor/editor.php>에서 수식 입력 명령어 학습 가능)
- LaTeX 수식 입력 코드는
- 예시

$$P(X = x) = f(x; n, p) = \binom{n}{x} p^x (1 - p)^{n-x}$$

- Inline equation: \$P(X = x) = f(x; n, p) = \{n \choose x\} p^x (1-p)^{n-x} \rightarrow P(X = x) = f(x; n, p) = \binom{n}{x} p^x (1 - p)^{n-x}\$
- Math block: \$\$P(X = x) = f(x; n, p) = \{n \choose x\} p^x (1-p)^{n-x}\$\$

$$P(X = x) = f(x; n, p) = \binom{n}{x} p^x (1 - p)^{n-x}$$

- \$ \$ 또는 \$\$ \$\$ 안에 LaTeX에서 제공하는 수식 함수 사용 가능

```
 $$\begin{array}{ccc}
 x_{11} & x_{12} & x_{13} \\

```

```
x_{21} & x_{22} & x_{23}
\end{array}$$
```

$$\begin{matrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \end{matrix}$$

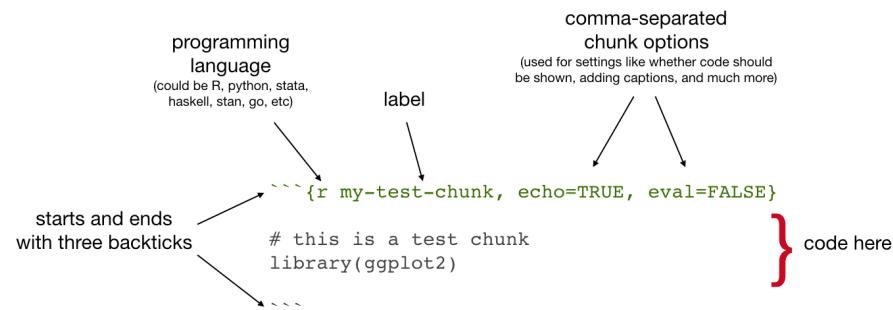
```
 $$\Theta = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}
```

$$\begin{aligned} g(X_n) &= g(\theta) + g'(\tilde{\theta})(X_n - \theta) \notag \\ \sqrt{n}[g(X_n) - g(\theta)] &= g'(\tilde{\theta}) \sqrt{n}[X_n - \theta] \end{aligned}$$

$$\begin{aligned} g(X_n) &= g(\theta) + g'(\tilde{\theta})(X_n - \theta) \\ \sqrt{n}[g(X_n) - g(\theta)] &= g'(\tilde{\theta}) \sqrt{n}[X_n - \theta] \end{aligned}$$

## 1.4 R Code Chunks

- 실제 R code가 실행되는 부분임
- Code chunk 실행 시 다양한 옵션 존재(본 강의에서는 몇 개의 옵션만 다룰 것이며, 더 자세한 내용은 <https://yihui.org/knitr/options/> 또는 R Markdown 레퍼런스 가이드<sup>5</sup> 참조)
- Code chunk는 `~~`{r}로 시작되며 r은 code 언어 이름을 나타냄.
- Code chunk는 `~~`로 종료
- R Markdown 문서 작성 시 단축키 [Ctrl] + [Alt] + [I]를 입력하면 Chunk 입력창이 자동 생성됨
- Code chunk의 옵션 조정을 통해 코드의 출력여부, 코드 출력 시 코드의 출력 형태, 코드의 결과물 출력 조정 가능



**FIGURE 1.6:** Chunk anatomy (<https://ulyngs.github.io/rmarkdown-workshop-2019>에서 발췌)

<sup>5</sup><https://rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf>

## 자주 활용하는 chunk 옵션

## 코드 실행 관련 청크

TABLE 1.1: 코드 실행 관련 청크

| Chunk 옵션 | Default | 설명                            |
|----------|---------|-------------------------------|
| eval     | TRUE    | R 실행(코드 실행 결과)에 대응하는 결과 출력 여부 |
| include  | TRUE    | 출력 문서에 코드 청크의 내용을 포함할지 여부     |

```
```{r ex01-1, eval=TRUE}
summary(iris)
hist(iris$Sepal.Length)
```

```

```
```{r ex01-2, eval=FALSE}
summary(iris)
hist(iris$Sepal.Length)
```

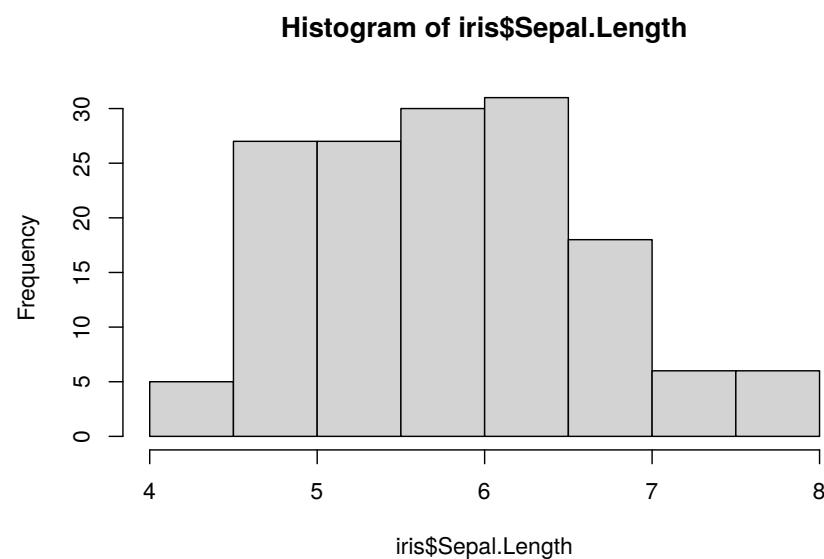
```

```
#청크 옵션 eval=TRUE
summary(iris)
```

| Sepal.Length  | Sepal.Width   | Petal.Length  | Petal.Width   |
|---------------|---------------|---------------|---------------|
| Min. :4.300   | Min. :2.000   | Min. :1.000   | Min. :0.100   |
| 1st Qu.:5.100 | 1st Qu.:2.800 | 1st Qu.:1.600 | 1st Qu.:0.300 |
| Median :5.800 | Median :3.000 | Median :4.350 | Median :1.300 |
| Mean :5.843   | Mean :3.057   | Mean :3.758   | Mean :1.199   |
| 3rd Qu.:6.400 | 3rd Qu.:3.300 | 3rd Qu.:5.100 | 3rd Qu.:1.800 |

```
Max.    :7.900  Max.    :4.400  Max.    :6.900  Max.    :2.500  
Species  
setosa    :50  
versicolor:50  
virginica :50
```

```
hist(iris$Sepal.Length)
```



```
# 청크 옵션 eval=FALSE  
summary(iris)  
hist(iris$Sepal.Length)
```

소스 코드 출력 (텍스트) 결과 관련 청크

**TABLE 1.2:** 소스 코드 출력 결과 관련 청크

| Chunk 옵션 | Default | 설명   |
|----------|---------|--|
| echo     | TRUE    | R 실행 결과에 대응하는 코드 출력 여부   |
| results  | markup  | 출력 결과 포맷 지정을 위한 옵션으로 추가적으로 3 가지 옵션 선택 가능: 'hide', 'asis', 'hold', 'markup' |
| error    | TRUE    | 코드 또는 스크립트에 구문오류 메세지 출력 여부   |
| message  | TRUE    | 코드로부터 생성된 메세지 출력 여부  |
| warning  | TRUE    | 경고 메세지 출력 여부   |

- echo: 코드 청크에 작성한 R-script 출력 여부 결정
  - echo = FALSE 이면 소스 코드 출력 없이 그림 결과만 출력

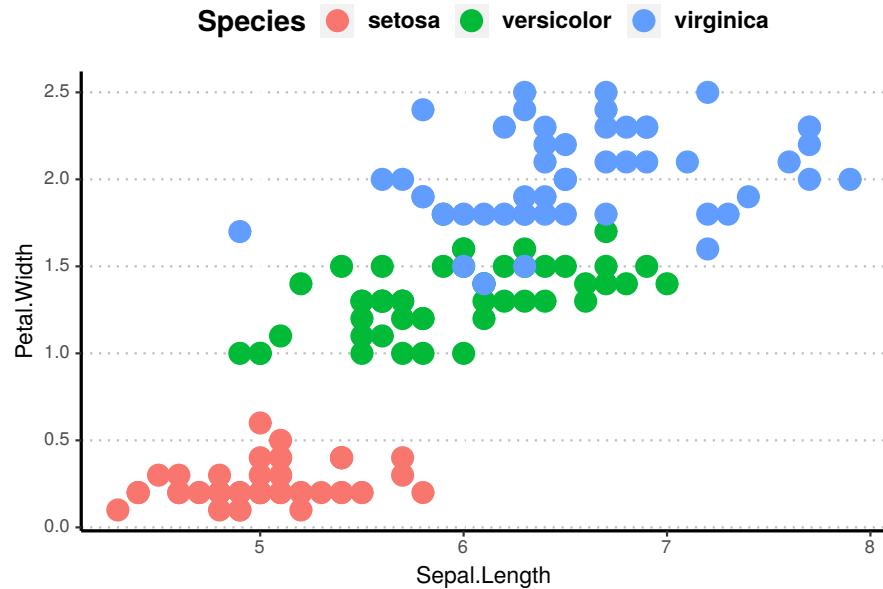
```
```{r ex01-2, echo=TRUE}
require(ggthemes) # ggtheme 패키지 불러오기
require(ggpubr) # ggpubr 패키지 불러오기
iris %>%
  ggplot(aes(x = Sepal.Length, y = Petal.Width, color = Species)) +
  geom_point(size = 5) +
  theme_pubclean() +
  theme(axis.line = element_line(size = 0.8),
        legend.title = element_text(face = "bold", size = 15),
        legend.text = element_text(face = "bold", size = 12))
```
```

```

```
```{r ex01-3, echo=FALSE}
require(ggthemes) # ggtheme 패키지 불러오기
require(ggpubr) # ggpubr 패키지 불러오기
iris %>%
  ggplot(aes(x = Sepal.Length, y = Petal.Width, color = Species)) +
  geom_point(size = 5) +
  theme_pubclean() +
  theme(axis.line = element_line(size = 0.8),
        legend.title = element_text(face = "bold", size = 15),
        legend.text = element_text(face = "bold", size = 12))

```
```

```
# echo = TRUE
require(ggthemes) # ggtheme 패키지 불러오기
require(ggpubr) # ggpubr 패키지 불러오기
iris %>%
  ggplot(aes(x = Sepal.Length, y = Petal.Width, color = Species)) +
  geom_point(size = 5) +
  theme_pubclean() +
  theme(axis.line = element_line(size = 0.8),
        legend.title = element_text(face = "bold", size = 15),
        legend.text = element_text(face = "bold", size = 12))
```



- `results`: 코드의 텍스트 출력 결과 포맷 지정
  - `markup` (default): 코드 청크 내 스크립트의 출력 형태에 따라 텍스트 출력 결과를 mark-up
  - `asis`: 변환하지 않은 원래 R 출력 결과 그대로 (as is) 출력
  - `hide`: R 스크립트로 생성된 텍스트 출력을 보여주지 않음 (warning, message 출력 예외)
  - `hold`: 코드 청크로 생성된 모든 소스 및 출력을 단일 블록으로 축소

```
# results = 'markup'인 경우 아래 텍스트를 mark-up
# (이 경우 아래 텍스트는 ```` 블럭 처리)한 결과를 md 파일로 전송
cat("I'm raw **Markdown** content.\n")
```

```
I'm raw **Markdown** content.
```

```
# results = 'asis' 인 경우 텍스트를 그대로 md 파일에 입력
cat("I'm raw **Markdown** content.\n")
```

The screenshot shows two RStudio panes. The left pane contains the Rmd file 'code-chunk.Rmd' with the following content:

```
---
title: "Untitled"
author: "Boncho Ku"
date: '2020-9-19'
output: md_document
...
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
## R Markdown
```{r, results='markup'}
# results = 'markup' 인 경우 소스 코드 블럭 내에 코드 출력
cat("I'm raw **Markdown** content.\n")
```

```

The right pane contains the resulting 'code-chunk.md' file with the output:

```
R Markdown
-----
# results = 'markup' 인 경우 소스 코드 블럭 내에 코드 출력
cat("I'm raw **Markdown** content.\n")
```

**FIGURE 1.7:** 청크 옵션 results = 'markup' 인 경우 rmd vs. md 파일 비교

I'm raw **Markdown** content.

The screenshot shows two RStudio panes. The left pane contains the Rmd file 'code-chunk.Rmd' with the following content:

```
---
title: "Untitled"
author: "Boncho Ku"
date: '2020-9-19'
output: md_document
...
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
## R Markdown
```{r, results='asis'}
# results = 'asis' 인 경우 텍스트를 그대로 최종 출력을
cat("I'm raw **Markdown** content.\n")
```

```

The right pane contains the resulting 'code-chunk.md' file with the output:

```
R Markdown
-----
# results = 'asis' 인 경우 텍스트를 그대로 최종 출력을
cat("I'm raw **Markdown** content.\n")
```

**FIGURE 1.8:** 청크 옵션 results = 'asis' 인 경우 rmd vs. md 파일 비교

```
# results = 'hide'
cat("I'm raw **Markdown** content.\n")
```

# 텍스트 결과를 출력하지 않음

```
# results = 'hold'가 아닌 경우 한 라인 별 출력 결과 생성
```

```
x <- rnorm(10)
```

```
x
```

```
[1] -0.41603691 -0.06844281  0.31028631  1.53300362 -0.76630835  0.61335592
[7] -3.31299754  0.07108634 -0.27304067  0.74256290
```

```
y <- rnorm(10, 1, 2)
y
```

```
[1] -1.9394158  2.1249310  3.4200572  4.1174784  2.8067859  1.3769160
[7]  0.9050209 -0.4775910  3.5285945  0.9991669
```

```
x + y
```

```
[1] -2.3554527  2.0564882  3.7303435  5.6504820  2.0404776  1.9902719
[7] -2.4079767 -0.4065047  3.2555538  1.7417298
```

```
# results = 'hold'인 경우 코드 부분과 출력 부분이 따로 블록 처리
x <- rnorm(10)
x
y <- rnorm(10, 1, 2)
y
x + y
```

```
[1]  1.3009318 -0.2691309 -0.3103594 -0.1729806  1.2224691 -0.9291561
[7]  1.0377210 -0.7003020  0.3886292 -0.3073199
[1]  1.6435989  1.6340404  2.9088397  0.6754476  3.2836272 -0.7267385
[7]  1.7846750  1.9813299  4.4470268 -0.7369612
[1]  2.944531   1.364909   2.598480   0.502467   4.506096  -1.655895   2.822396
[8]  1.281028   4.835656  -1.044281
```

- **error:** 코드 청크 내 스크립트에 오류에 대한 보존 여부(`stop()`)
  - 기본적으로 Rmarkdown 컴파일 시 `error`에 대한 옵션이 `FALSE`이기 때문에 스크립트(코드)에 오류가 포함되면 컴파일이 정지됨.
  - `error = TRUE` 이면 오류 메세지를 포함한 텍스트 결과를 출력

```
3x <- 3
x <- 25 # 위 행이 구문 오류를 포함하고 있기 때문에
```

```
# 오류 이후의 코드는 실행되지 않음
```

```
x
```

Error: <text>:1:2: 예상하지 못한 기호(symbol)입니다.

1: 3x

^

- **message/warning:** 텍스트 출력물 중 경고(warning, warning() 함수의 출력 결과) 메세지 출력 여부 결정

```
# message = TRUE 인 경우 함수 message 출력
testit <- function() {
  message("testing package startup messages")
  packageStartupMessage("initializing ...", appendLF = FALSE)
  Sys.sleep(1)
  packageStartupMessage(" done")
} # help(message) 예시 중 발췌

testit()
```

testing package startup messages

initializing ... done

```
# message=FALSE -> 메세지 출력하지 않음
testit()
```

```
# 경고 메세지 출력
x <- c(1, 2, "new", 4:10)
x <- as.numeric(x)
```

Warning: 강제형변환에 의해 생성된 NA입니다

코드 서식 관련 청크 옵션

**TABLE 1.3:** 코드 서식 관련 청크

| Chunk 옵션  | Default | 설명                                                 |
|-----------|---------|----------------------------------------------------|
| comment   | TRUE    | 소스 코드 실행 출력의 각 줄 앞에 붙는 표시문자 출력<br>여부: 기본 값은 '##' 임 |
| highlight | TRUE    | 구문 강조 여부                                           |
| prompt    | FALSE   | R 프롬프트 출력 여부                                       |
| tidy      | FALSE   | R 소스 코드 출력 정리 여부                                   |

- `comment`: 텍스트 출력물에 주석 표시 (default)를 함으로써 소스 코드와 출력 결과를 동시 선택과 복사를 가능 (###는 주석 표시이기 때문에 실행되지 않음)

– 주석 표시를 제거하고 싶다면 `comment = NA` 또는 `comment = ''`

```
# 디폴트 comment 사용
summary(iris)
```

```
##   Sepal.Length   Sepal.Width    Petal.Length   Petal.Width
##   Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##   1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##   Median :5.800   Median :3.000   Median :4.350   Median :1.300
##   Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##   3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##   Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##
##           Species
##           setosa     :50
##           versicolor:50
##           virginica :50
##
##
```

- **highlight:** 구문 강조 표시 여부
  - `highlight=FALSE` 일 때 소스 코드 출력 결과

```
# highlight=FALSE
```

```
iris %>%
  ggplot(aes(x = Sepal.Length, y = Petal.Width, color = Species)) +
  geom_point(size = 5) +
  theme_pubclean() +
  theme(axis.line = element_line(size = 0.8),
        legend.title = element_text(face = "bold", size = 15),
        legend.text = element_text(face = "bold", size = 12))
```

- **prompt:** R 콘솔 상 프롬프트 >, + 출력 여부

```
> # prompt = TRUE 인 경우 코드 출력 결과
> require(ggthemes) # ggtheme 패키지 불러오기
> require(ggpubr) # ggpubr 패키지 불러오기
> iris %>%
+   ggplot(aes(x = Sepal.Length, y = Petal.Width, color = Species)) +
+   geom_point(size = 5) +
+   theme_pubclean() +
+   theme(axis.line = element_line(size = 0.8),
+         legend.title = element_text(face = "bold", size = 15),
+         legend.text = element_text(face = "bold", size = 12))
```

- **tidy:** 코드를 사용자가 지정(혹은 `formatR::tidy_sorce()` 함수에 초기 값으로 지정된 코드 정리 값)한 줄 당 문자 길이 등을 반영해 코드를 정리
  - `tidy=TRUE` 인 경우 자동으로 줄 바꿈

```
> # tidy = FALSE 인 경우 코드 출력 결과
> require(ggthemes) # ggtheme 패키지 불러오기
```

```
> require(ggpubr) # ggpubr 패키지 불러오기
> iris %>% ggplot(aes(x = Sepal.Length, y = Petal.Width, color = Species)) + geom_point(size = 5) +
+   theme_pubclean() + theme(axis.line = element_line(size = 0.8), legend.title = element_text(fac
+   size = 15), legend.text = element_text(face = "bold", size = 12))
```

### 그림(plot) 출력 관련 청크 옵션

**TABLE 1.4:** Plot 출력 관련 청크

| Chunk 옵션             | Default | 설명                            |
|----------------------|---------|-------------------------------|
| fig.align            | default | 최종 문서에 plot 정렬 방식 결정 (center) |
| fig.height/fig.width | 7       | 그림 크기(단위: 인치)                 |
| fig.cap              | NULL    | 그림 캡션(문자열 입력)                 |
| dpi                  | 72      | dot per inch: 출력 그림 해상도       |

### 알아두면 좋은 청크 형태

#### Setup 청크

- 일반적으로 Rmarkdown 문서는 YAML 해더 뒤에 전역적 청크 옵션 지정과 R 패키지를 불러오는 것으로 시작
- 청크 옵션은 `knitr::opts_chunk$set(청크 옵션 지정)` 형태로 지정 가능
- 다음은 RStudio에서 Rmd 문서 생성 시 맨 처음 나오는 코드 청크 예시임

```
```{r ex01-2, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
```

- 일반적 활용 예시

```
```{r option-init, include=FALSE}
knitr::opts_chunk$set(root.dir = '../..', # 프로젝트 폴더 지정
                      eval = TRUE,
                      echo = FALSE,
                      cache = FALSE,
                      include = TRUE,
                      tidy = TRUE,
                      tidy.opts = list(blank=FALSE, width.cutoff=120), # 소스 출력길이
                      message = FALSE,
                      warning = FALSE,
                      engine = "R", # Chunks will always have R code, unless noted
                      error = TRUE,
                      fig.path="Figures/", # Set the figure options
                      fig.align = "center",
                      fig.width = 7,
                      fig.height = 7,
                      fig.keep='all', fig.retina=2)
````
```

이미지 불러오기

```
```{r, fig.cap = "Taj Mahal"}  
knitr::include_graphics("figures/taj.JPG", dpi = NA)  
```
```



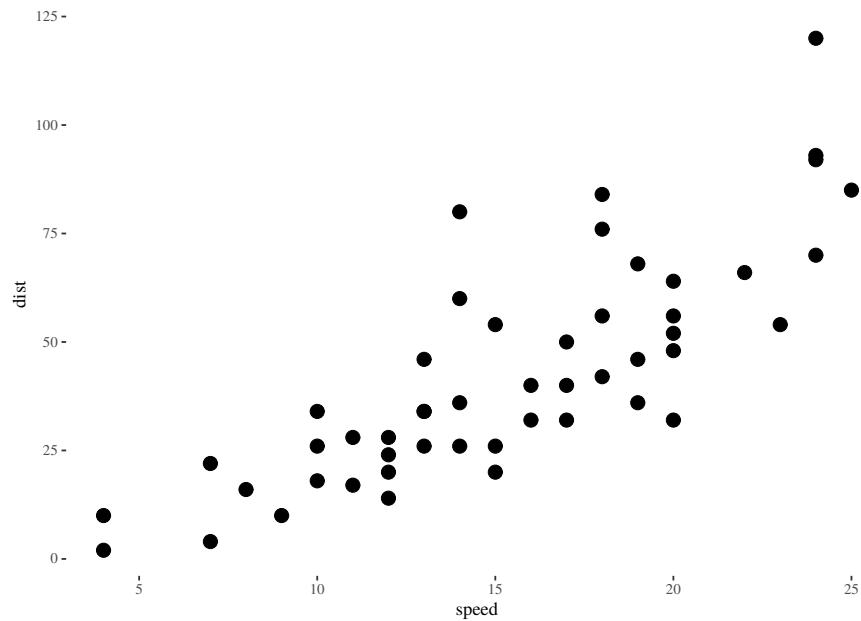
**FIGURE 1.9:** Taj Mahal

```
```{r, fig.cap = "Taj Mahal"}  
cars %>%  
  ggplot(aes(x = speed, y = dist)) +  
  geom_point(size = 5) +  
  theme_tufte(base_size = 15) # ggtheme::theme_tufte()  
```
```

R 생성 도표 포함

테이블 삽입

- 가장 간단한 테이블은 knitr::kable() 함수를 통해 생성 가능



**FIGURE 1.10:** Scatterplot of the car dataset

- `kable()` 함수는 가장 단순한 형태의 표만 생성하기 때문에 복잡한 표를 만들기에는 한계가 존재함
- 이를 보완하기 위해 다음과 같은 패키지 활용
  - `kableExtra`: HTML 또는 LaTeX 용 표 생성
    - \* [https://cran.r-project.org/web/packages/kableExtra/vignettes/awesome\\_table\\_in\\_html.html](https://cran.r-project.org/web/packages/kableExtra/vignettes/awesome_table_in_html.html)
    - \* [https://cran.r-project.org/web/packages/kableExtra/vignettes/awesome\\_table\\_in\\_pdf.pdf](https://cran.r-project.org/web/packages/kableExtra/vignettes/awesome_table_in_pdf.pdf)
  - `flextable + officer`: HTML, 워드 문서 표 작성
    - \* <https://davidgohel.github.io/flextable/>

```
```{r}
knitr::kable(head(iris))
```

```

| Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|--------------|-------------|--------------|-------------|---------|
| 5.1          | 3.5         | 1.4          | 0.2         | setosa  |
| 4.9          | 3.0         | 1.4          | 0.2         | setosa  |
| 4.7          | 3.2         | 1.3          | 0.2         | setosa  |
| 4.6          | 3.1         | 1.5          | 0.2         | setosa  |
| 5.0          | 3.6         | 1.4          | 0.2         | setosa  |
| 5.4          | 3.9         | 1.7          | 0.4         | setosa  |

## 1.5 인라인 (inline) R 코드

- 문서의 모든 숫자를 인라인 R 코드를 통해 재현가능하게 생성 가능
- 인라인 R 코드는 `r` 과 ` ` 사이에 변수 계산 스크립트를 입력해 작성 가능
- 예를 들어 `r 10 + 4` 는 14 출력
- 활용 예시

```
head(mtcars, 5)

      mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4     21.0   6 160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag 21.0   6 160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710    22.8   4 108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive 21.4   6 258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8 360 175 3.15 3.440 17.02  0  0    3    2
```

```
N <- nrow(mtcars)
```

mtcars 데이터셋에 포함된 자동차는 ‘r N ‘ 개다.

→

mtcars 데이터셋에 포함된 자동차는 32 개다.

---

## 1.6 YAML

- R Markdown 문서의 가장 처음에 정의하는 metadata
- .Rmd 파일을 .md 파일로 변환 후 최종 출력문서 생성 시 필요한 pandoc의 옵션을 설정하는 것과 같은 의미임
- 일반적으로 문서 형태 및 생성을 위해 사용하는 R package (예: bookdown, officedown, rticles 등)에 따라 YAML 구성요소가 달라짐

### 기본 문법

- /#: 주석 처리
- YAML 문서의 시작과 끝은 ---로 정의함
- 기본적으로 콜론(:)으로 구분된 태그(키): 값 쌍으로 구성됨 → key: value
  - 여기서 콜론 바로 다음에는 반드시 공백문자가 있어야 함
- 한 key의 하위 키는 리스트 형태로 표현하고, 하위 키는 두 개 이상의 스페이스로 공백을 주어 표현

```
---
```

```
key : value
  subkey1: value1
  subkey2: value2
    subsubkey1: value3
---
```

## R Markdown 기본 YAML 구조

```
---
```

```
title: "문서 제목" # 일반적으로 따옴표 사용
subtitle: "문서 부제목"
author: "문서 작성자"
date: "문서 작성일자"
output:
  - "html_document"
  - "word_document"
  - "pdf_document"
  - "md_document"
  - "isoslates_presentation"
  - "slidy_presentation"
  - "beamer_presentation"
bibliography: 참고문헌.bib # bibtex 서식 활용
```

---

- <https://bookdown.org/yihui/rmarkdown/documents.html> 에 자세한 예시 참고

## 1.7 참고문헌 인용

- 참고문헌 정보가 BibTeX 포맷으로 저장된 .bib 파일을 YAML에 선언 후 인용 가능

### Bibtex 참고문헌 입력 형태

```
@article{Shea2014,
  author = {Shea, Nicholas and Boldt, Annika},
  journal = {Trends in Cognitive Sciences},
  pages = {186--193},
  title = {{Supra-personal cognitive control}},
  volume = {18},
  year = {2014},
  doi = {10.1016/j.tics.2014.01.006},
}
```

### YAML에 bib 파일 지정

```
---
title: "Citation test"
bibliography: example.bib
output: html_document
---
```

- 참고문헌 표현 : [@citation-identifier] 또는 @citation-identifier

This...

Blah blah [@Shea2014; @Lottridge2012].  
Shea et al. says blah [-@Shea2014].  
@Shea2014 says blah.  
Blah blah [see @Shea2014, pp. 33-35; also  
@Wu2016, ch. 1].

turns into this...

Blah blah (Shea et al. 2014; Lottridge et al. 2012).  
Shea et al. says blah (2014).  
Shea et al. (2014) says blah.  
Blah blah (see Shea et al. 2014, 33–35; also Wu 2016, ch. 1).

- BibTeX 포맷은 Google Scholar에서 쉽게 획득 가능
- Citation 스타일은 YAML 헤더에 `cl: style.csl`로 변경 가능하며 Zotero<sup>6</sup>에서 .csl 파일 다운로드 가능

---

<sup>6</sup><https://www.zotero.org>

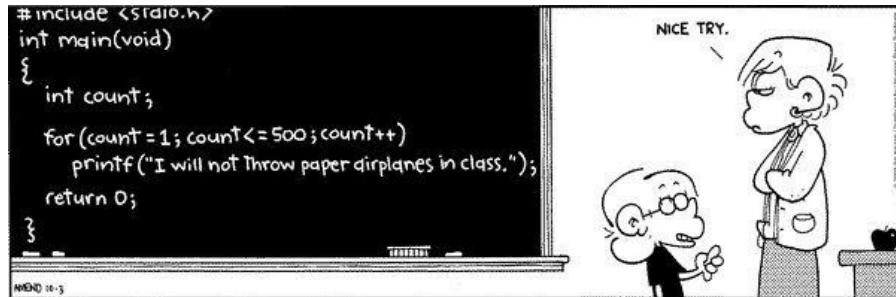


# 2

## 제어문 (Control Structure)

### Sketch

- 프로그램이 무엇이고 이를 만들기 위해 어떤 것들이 필요할까?
- 프로그램 안의 특정 구문을 주어진 조건에 맞게 실행 여부를 제어하거나 동일한 작업을 반복할 수 있을까?
- 프로그램을 통해 특정 목적을 위한 나만의 함수를 만들 수 있을까?



**FIGURE 2.1:** Flow-control example (<https://homerhanumat.github.io/r-notes/flow.html>)



참고: 본 장의 내용은 데이터과학 민주화<sup>1</sup>와 Beginning Computer Programming with R<sup>2</sup>의 내용을 기반으로 재구성함

## 2.1 Prerequisite

- 예약어 (reserved words): R에서 의미 (semantic)를 미리 정해 놓은 단어
  - 통계프로그래밍언어 강의노트<sup>3</sup> 참고

**TABLE 2.1:** R 예약어 종류 및 설명

| R 예약어                                                | 설명                       |
|------------------------------------------------------|--------------------------|
| if, else, while, function, in, next, break           | 조건, 함수, 반복문에 사용          |
| TRUE/FALSE                                           | 논리 상수(logical constants) |
| NULL                                                 | 정의되지 않은 값 혹은 값이 없음 표현    |
| Inf                                                  | 무한(infinity)             |
| NaN                                                  | 숫자가 아님(not a number)     |
| NA                                                   | 결측값(not available)       |
| NA_integer_, NA_real_,<br>NA_complex_, NA_character_ | 결측값을 처리하는 상수             |
| ...                                                  | 함수가 다른 함수에 인자를 전달하도록 지원  |

- 변수(variable): 사용자가 프로그램 처리를 위해 지정한 단어
  - 적당한 값을 저장하고 나중에 필요시 해당 값을 호출해 사용하기 위한 목적으로 사용되는 표식(label)
  - 예약어를 변수명으로 사용할 수 없음

<sup>3</sup><https://zorba78.github.io/cnu-r-programming-lecture-note/scalar.html>

- 통계프로그래밍언어 강의노트 : R 기초문법<sup>4</sup> 참고
- 고수준 언어 (**high-level language**) : 사람이 읽고 쓰기 쉬운 형태의 명령어를 컴퓨터가 읽고 처리할 수 있도록 고안된 프로그래밍 언어
  - 컴퓨터가 이해할 수 있는 언어 → 중앙처리장치 (central processing unit, CPU)가 이해하는 언어 → 기계어 (machine language)
  - 기계어는 0과 1로 구성된 이진수 (binary number) 임 (예 : 01001010010010010011101101011010110)
  - 고수준 언어의 종류 : C, C++, JAVA, 베이직, Perl, Python, R, ...
- 번역기 (**translator**) : 사람이 이해할 수 있는 표현(언어)를 기계(컴퓨터)가 이해할 수 있는 언어(기계어)로 변환
  - 인터프리터 (interpreter)
  - 컴파일러 (compiler)
- \*\*인터프리터\*\*: 코드 (스크립트) 한 줄을 즉석에서 읽고, 파싱 (프로그램을 검사하고 구문론적 구조를 분석)하고, 해석
  - R, Python, MATLAB 등은 인터프리터를 번역기로 사용
  - 인터액티브 모드 → R 프롬프트 (>) 뒤에 한 줄의 명령어를 작성하면 즉석해서 처리 후 다음 입력에 대해 준비 (prompt) 함.

안녕하세요 !!

통계패키지활용 수업에서 R을 배우고 있습니다.

처음이라 실수가 많습니다.

앞으로 잘 부탁해요 !!

Error: <text>:1:6: 예기치 않은 '!'입니다

1: 안녕하세요!

~

<sup>4</sup><https://zorba78.github.io/cnu-r-programming-lecture-note/r-basic.html>

```
print("안녕하세요!!")
print("통계패키지 활용 수업을 위해 R을 배우고 있습니다.")
print("처음이라 실수가 많습니다.")
print("앞으로 잘 부탁해요!!")
```

```
[1] "안녕하세요!!"
[1] "통계패키지 활용 수업을 위해 R을 배우고 있습니다."
[1] "처음이라 실수가 많습니다."
[1] "앞으로 잘 부탁해요!!"
```

- **컴파일러:** 완전한 프로그램을 하나의 파일에 담고 파일 안에 저장되어 있는 소스코드를 기계어로 번역 후 다음 실행할 수 있도록 변환한 기계어를 파일에 담음.
  - 보통은 .exe, .dll 파일 형태로 저장됨

## 2.2 프로그램

- **프로그램 (program):** 특정 작업(목적)을 수행할 수 있도록 작성한 일련의 R 문장(명령어)의 집합
  - 일련의 문장(명령어)들은 텍스트 편집기를 통해 작성하며, 스크립트 (script)로 명칭되는 파일로 저장 → R 스크립트 .R 확장자를 가짐

```
# Hello.R
print("안녕 R!!") #한국어
print("Hi R!!") # 영어
print("こんにちはR!!") # 일본어
print("Γ R!!") #그리스어
```

```
source("hello.R", encoding = "UTF-8")
```

```
[1] "안녕 R!!"  
[1] "Hi R!!"  
[1] "こんにちはR!!"  
[1] "Γ R!!"
```

- 예시: 텍스트 파일에서 가장 자주 나오는 단어 찾기 프로그램
  - <https://statklee.github.io/r4inf/r-intro.html#r-intro-what-is-a-program> 참고

```
require(tidyverse)  
require(stringr)  
require(ggpubr)  
require(ggthemes)  
  
text_dat <- readLines("data/text-example-01.txt")  
# 공백 또는 구둣점 문자를 기준으로 텍스트 나누기  
  
# 공백 또는 구둣점 문자 기준으로 텍스트 토큰화  
split_wd <- str_split(text_dat, pattern = "\\b|[[:punct:]]")  
split_wd <- do.call(c, split_wd)  
id <- grep("[a-zA-Z]+", split_wd) # 알파벳을 포함한 단어 인덱스  
split_wd <- split_wd[id]  
unique_wd <- unique(split_wd) # 중복을 제외한 총 사용 단어  
res_v <- vector("integer", length(unique_wd)) # 저장 벡터 생성  
  
for (i in seq_along(unique_wd)) {  
  for (j in seq_along(split_wd)) {  
    if (unique_wd[i] == split_wd[j]) {  
      res_v[i] <- res_v[i] + 1  
    }  
  }  
}
```

```

    }
}

bind_cols("word" = unique_wd, "freq" = res_v) %>%
  arrange(desc(freq))

```

```

# A tibble: 428 x 2
  word   freq
  <chr> <dbl>
1 the     57
2 in      24
3 of      23
4 to      20
5 South    17
6 a       15
7 and     14
8 Korea    13
9 was      9
10 which   9
# ... with 418 more rows

```

- 프로그램 작성을 위한 개념적 요소
  - 입력 (**input**): 외부로부터 가져온 데이터, 값 등
  - 출력 (**output**): 입력에 대한 반응 (결과 출력, 파일 저장, 음악 재생, ...)
  - 순차실행 (**sequential execution**): 스크립트 또는 코드 작성 순서에 따라 한줄씩 실행
  - 조건실행 (**conditional execution**): 특정 조건에 따라 문장 (명령)을 실행하거나 건너뜀
  - 반복실행 (**iterative execution**): 특정 명령을 반복적으로 실행

- 재사용 (resuse): 스크립트의 집합(다수 줄로 구성된 코드 또는 스크립트)에 이름을 부여하고 저장 → 사용자 지정 함수(function)
- 프로그램 오류의 종류
  - 구문오류 (syntax error): R 언어가 이해할 수 없는 문장 또는 문법으로 실행했을 때 나타나는 오류 → 가장 고치기 쉽고 즉각적으로 알려줌
  - 논리 또는 run-time 오류 (logic or run-time error): 구문은 완벽하지만 실행 순서 또는 논리적으로 연관방식에 문제가 있어서 명령어를 수행할 수 없는 경우
  - 의미론적 오류 (sementic error): 프로그램은 구문적으로 오류가 없고 실행되지만 올바른 결과를 출력하지 않는 경우 → 제일 고치기 어려움
- 가장 간단한 프로그래밍은 순차적으로 명령을 실행하되 입력 시 흐름을 잠시 중단하고 대기하는 방법 → 프롬프트 상 명령어 한 줄씩 입력

```
# 아주 간단한 프로그래밍 예제  
# readline() 함수 이용해 R한테 인사 받기  
name <- readline("What's your name?: ")  
cat("Hello, ", name, "!\\n", sep = "")
```

```
# readline() 함수를 이용해 알바비 계산  
  
x <- as.numeric(readline(prompt = "하루 아르바이트 시간을 입력하시오: "))  
y <- as.numeric(readline(prompt = "시급을 입력하시오 (단위=원): "))  
z <- as.numeric(readline(prompt = "한달 동안 총 몇 일 동안 일을 하셨나요? "))  
cat("월 급여는 ", x * y * z, " 원입니다.\\n", sep = "")
```

### 2.3 조건문 (Conditionals)

- **if** 구문을 통해 조건문 생성
- 불린 표현식 (**boolean expression**): 참 (TRUE) 또는 거짓 (FALSE) 두 값 중 하나로 값이 도출되는 표현식<sup>5</sup>
  - 비교 연산자 (**comparison operators**)
    - \* 같다, 같지 않다, 크다 등을 표현하기 위한 연산자
    - \* ==, !=, >, <, >=, <=
  - 논리 연산자 (**logical operator**)
    - \* AND (&, &&), OR (|, ||), NOT (!)

```
x <- 10; y <- 13

# x가 2의 배수이고 y가 3의 배수
# 두 조건이 모두 참이여야 참
x %% 2 == 0 & y %% 3 == 0

# x가 2의 배수이거나 y가 3의 배수
# 두 개 조건 중 하나만 참을 만족하면 참임
x %% 2 == 0 | y %% 3 == 0

# NOT (x > y)
!(x > y) # 부정에 부정은 참
```

[1] FALSE

[1] TRUE

[1] TRUE

---

<sup>5</sup>비교 및 논리 연산자(통계프로그래밍언어 2.1.4절 참고<sup>6</sup>)

### 2.3.1 기본 구문

`if` (조건) 표현식

↳ 괄호 안 조건을 만족하면 표현식을 실행하고 조건을 만족하지 않으면 실행하지 않음

```
x <- 10
if (x > 0) {
    print("x is positive")
}
```

```
x <- -5
if (x > 0) {
    print("x is positive")
}
```

[1] "x is positive"

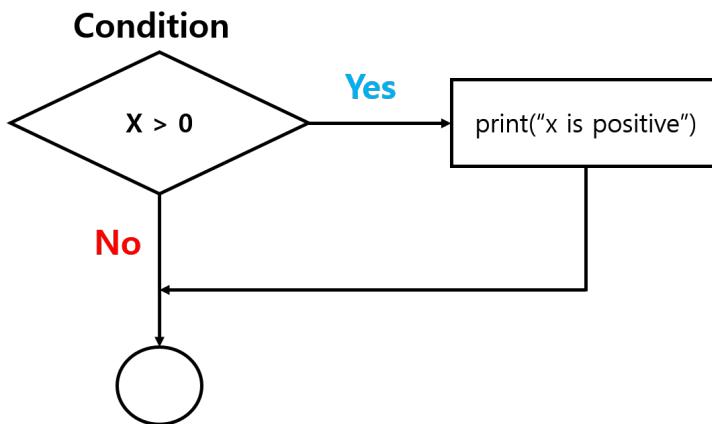


FIGURE 2.2: if 구문 기본 flow-chart

- `if` 구문의 사용 규칙

- `if` 문은 조건을 정의하는 해더 부분 ((, ))과 표현식이 위치하는 몸통 블록(body block, {표현식})으로 구성됨

- (, )에 표현되는 조건은 벡터가 아닌 단일 값으로 나타내야 함.
- {, }의 표현 또는 문장이 한 줄인 경우 블록 지정이 필요하지 않지만, 두 줄 이상인 경우 if 문의 범위를 지정해줘야 하기 때문에 꼭 중괄호 (curly bracket, {})가 사용되어야 함.

```
# 조건문 사용 예시
x <- c(TRUE, FALSE, FALSE)
y <- c(TRUE, TRUE, FALSE)
z <- "Both TRUE!!"

if (x[1] & y[1]) print(z) # x, y 첫 번째 원소만 사용
if (x && y) print(z) # 강제로 첫 번째 원소만 사용
if (x & y) print(z) # 경고 표시
```

Warning in if (x & y) print(z): length > 1 이라는 조건이 있고, 첫번째 요소만이 사용될 것입니다

```
[1] "Both TRUE!!"
[1] "Both TRUE!!"
[1] "Both TRUE!!"
```

### 대안 실행 (alternative execution)

- 두 가지 경우가 존재하고 조건에 따라 어떤 명령을 실행할지를 결정
- **if**와 **else**로 표현 가능
- 조건에 따라 실행이 분기 (branch) 되기 때문에 if-else 구문을 분기문이라고도 함
- **else**는 **if** 조건을 배제 (exclusive) 한 나머지 경우이기 때문에 조건을 따로 지정하지 않으면, **if**와 동일하게 중괄호 내에 표현되어야 함

```
x <- 9
if (x %% 2 == 0) {
  print("x is even")
} else {
  print("x is odd")
}
```

```
[1] "x is odd"
```

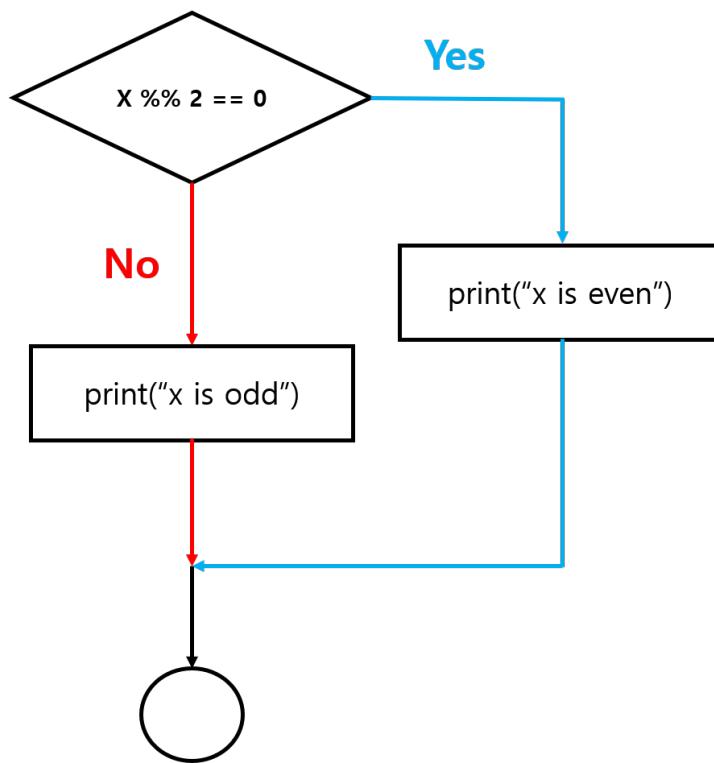


FIGURE 2.3: 대안실행(if-else 구문) flow-chart

### 2.3.2 연쇄 조건문(chained condition)

- 두 가지 이상의 분기가 존재하는 경우 조건 표현식

- 연쇄 조건문의 표현은 아래와 같음

```
if (조건1) {  
    표현식1  
    ...  
} else if (조건2) {  
    표현식2  
    ...  
} else {  
    표현식3  
    ...  
}
```

```
x <- 5; y <- 10  
if (x < y) {  
    print("x is less than y")  
} else if (x > y) {  
    print("x is greater than y")  
} else {  
    print("x is equal to y")  
}
```

```
[1] "x is less than y"
```

### 2.3.3 중첩 조건문 (nested condition)

- 하나의 조건문 내부에 하위 조건식이 존재하는 형태

```
if (조건1) {  
    표현식1  
    ...  
} else {
```

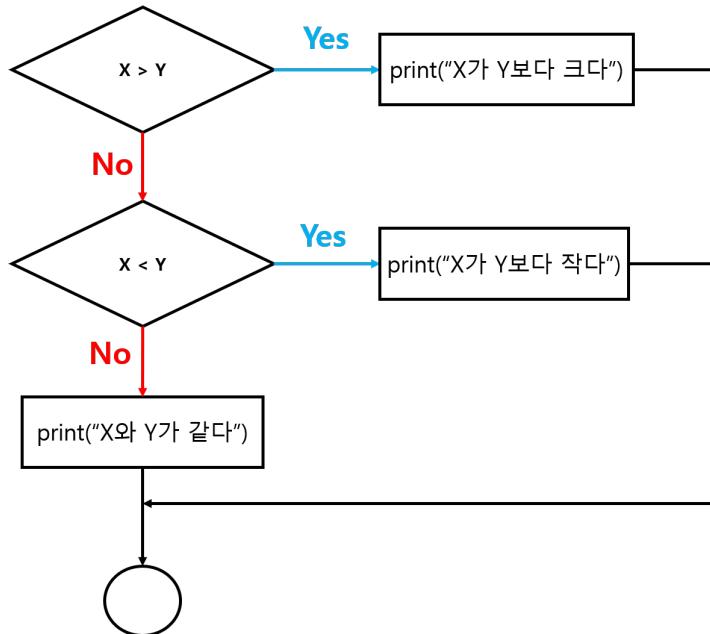


FIGURE 2.4: 연쇄조건 (if-else if-else 구문) flow-chart

```

if (조건2) {
    표현식2
    ...
} else {
    표현식3
    ...
}
  
```

```

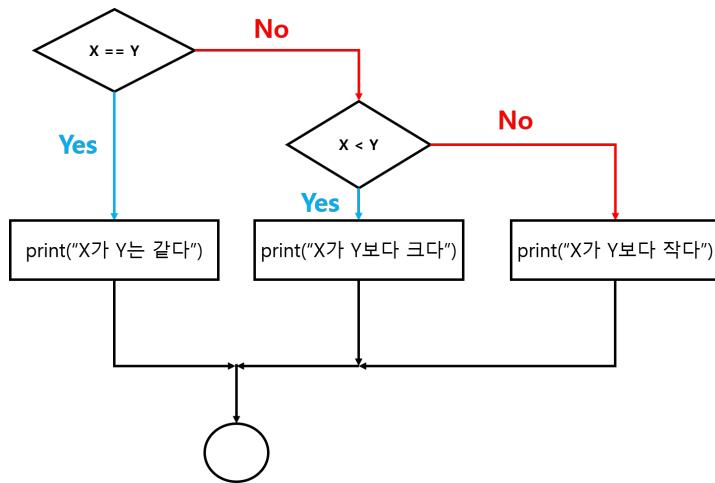
x <- 10; y <- 10
if (x == y) {
    print("x is equal to y")
} else {
    if (x > y) {
        ...
    }
}
  
```

```

print("x is greater than y")
} else {
    print("x is less than y")
}
}

```

[1] "x is equal to y"



**FIGURE 2.5:** 중첩 조건문 flow-chart



- 중첩 조건문은 코드의 가독성을 떨어뜨리기 때문에 피하는 것을 권장
- 중첩 조건문을 피하기 위한 한 가지 방법은 논리 연산자를 활용

```

# 중첩조건
x <- 58
if (x > 0) {
    if (x < 10) {
        print("x는 한 자리 양수")
    } else {
        if (x < 100) {

```

```
    print("x는 두 자리 양수")  
}  
else {  
    print("x는 세 자리 이상 양수")  
}  
}  
}
```

[1] "x는 두 자리 양수"

```
# 연쇄 조건

x <- 2020

if (x > 0 & x < 10) {
  print("x는 한 자리 양수")
} else if (x >=10 & x < 100) {
  print("x는 두 자리 양수")
} else {
  print("x는 세 자리 이상 양수")
}
```

[1] "x는 세 자리 이상 양수"

#### 2.3.4 ifelse() 함수

- `if-else` 구문을 사용하기 쉽게 구현된 R 내장 함수
  - `if-else` 구문과 다르게 조건 부분에 한 값(스칼라)이 아닌 논리형 벡터를 입력값으로 받아 조건에 따른 값(벡터)을 반환

```
# ifelse() 함수 인수  
# help(ifelse) 참고  
ifelse(  
  test, 조건에 따른 논리형 벡터
```

```

yes,  test에 정의한 조건이 참인 경우 새로운 벡터에 대입할 값
no,   test 조건이 거짓인 경우 대입할 값
)

```

- 사용 예시

```
# 평균이 23이고 표준편차가 5인 정규분포로부터 30개의 난수 추출
```

```

set.seed(12345)
bmi <- rnorm(30, 23, 5)
bmi_cat <- ifelse(bmi < 25, "normal", "overweight")
bmi_cat

```

```

[1] "overweight" "overweight" "normal"      "normal"      "overweight"
[6] "normal"       "overweight" "normal"      "normal"      "normal"
[11] "normal"       "overweight" "normal"      "overweight"  "normal"
[16] "overweight"   "normal"     "normal"      "overweight"  "normal"
[21] "overweight"   "overweight" "normal"      "normal"      "normal"
[26] "overweight"   "normal"     "overweight"  "overweight"  "normal"

```

```
# ifelse() 함수를 연쇄조건문 처럼 사용할 수 있다
```

```

bmi_cat2 <- ifelse(bmi < 18.5, "underweight",
                     ifelse(bmi < 24.9, "normal",
                           ifelse(bmi < 29.9, "overweight", "obesity")))
bmi_cat2

```

```

[1] "overweight"  "overweight"  "normal"      "normal"      "overweight"
[6] "underweight" "overweight"  "normal"      "normal"      "underweight"
[11] "normal"       "obesity"    "normal"      "overweight"  "normal"
[16] "overweight"   "normal"     "normal"      "overweight"  "normal"
[21] "overweight"   "obesity"   "normal"      "underweight" "underweight"
[26] "obesity"     "normal"     "overweight"  "overweight"  "normal"

```

## 2.4 반복문(Looping)

### Prerequisite

- 프로그램 또는 알고리즘 구현 시 특정 문장 또는 표현을 반복해야만 하는 상황이 발생
- 특히 시뮬레이션 시 반복문은 거의 필수적임
- 반복문을 통해 코딩의 효율을 극대화 할 수 있음
- 반복문은 특정 변수의 값을 갱신(update) 하기 위해 주로 사용

```
x <- x + 1 # 현재 값에 1을 더해서 x를 새로운 값으로 update
```

- 통상적으로 특정 변수의 값을 갱신하기 위해 변수 값을 초기화(initialize)

```
x <- 0 # x 변수 초기화  
x <- x + 1
```

- 몇 번 반복이라는 정의가 없는 상태에서 특정 조건이 거짓(FALSE)이 될 때 까지 계속 반복

### 2.4.1 repeat 구문

repeat 표현식

- repeat 다음에 오는 표현식을 무한 반복(infinite loop)

```
repeat print("무한 루프에 걸림...ESC 키 누르시오!!")
```

```
[1] "무한 루프에 걸림...ESC 키 누르시오!!"
...
...
```

- 특정 작업에 대해 블록을 지정(중괄호)하고 블록 안에 표현 가능
- 일반적으로 특정 조건(`if (조건) break`)을 두어 무한루프에서 탈출
- `if` 문의 조건은 언제 반복이 끝날지를 제어하는 변수로 반복변수(iteration variable)이라고도 함
- 언제까지(until) 반복(repeat) → REPEAT-UNTIL 구문으로 표현

```
repeat {
  표현식 1
  if (조건) break
  반복변수 update
}
```

```
# REPEAT-UNTIL 예시 1
# 1:100 까지 합 계산 함수
tot <- 0; i <- 1 # 사용 변수 초기화 (update 변수)
repeat {
  tot <- tot + i
  if (i >= 100) break # i는 반복 변수
  i <- i + 1
}
tot
```

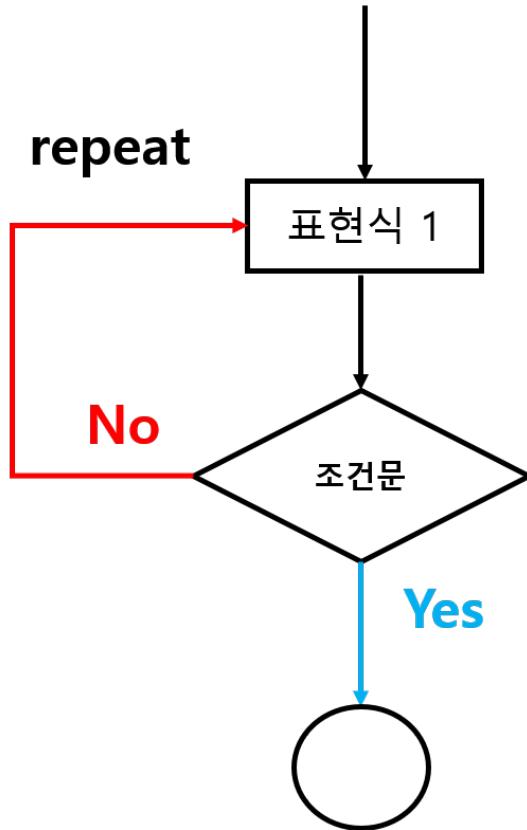


FIGURE 2.6: REPEAT 구문 flow-chart

```
# check  
sum(1:100)
```

```
[1] 5050  
[1] 5050
```

1. tot에 i를 더한 후 i 가 조건을 만족하는지 확인
2. 조건에 부합하지 않으면 다음 문장 실행 (i에 1을 증가 후 업데이트) 1.의 작업을 반복 (loop)

## 3. i가 조건에 부합하면 반복 종료

```
# REPEAT 예시 2
# 1에서 20 사이 숫자 알아맞추기 게임
n <- 20
number <- sample(1:n, size = 1)
cat("1에서 ", n, "까지 숫자 알아 맞추기", sep = "")
repeat {
  guess <- readline("어떤 숫자를 생각하시나요? (종료: q 입력) ")
  if (guess == "q") {
    cat("재미가 없나봐요.\n")
    break
  } else if (as.numeric(guess) == number) {
    cat("천재인데요?ㅋㅋㅋ")
    break
  }
  # 틀리면 계속 반복
}
```

1. guess에 readline() 으로부터 값 입력
2. guess 값이 q 이면 종료
3. guess 값이 number 와 일치하면 종료
4. 2.와 3. 조건에 부합하지 않으면 guess 값을 반복적으로 입력

어떤 숫자를 생각하시나요? (종료: q 입력) 1

어떤 숫자를 생각하시나요? (종료: q 입력) 2

어떤 숫자를 생각하시나요? (종료: q 입력) 3

천재인데요?ㅋㅋㅋ

### 2.4.2 while 구문

```
while (조건) 표현식 ...
```

- while에 지정된 조건이 참이면 계속해서 반복
- repeat는 반복이 처음부터 시작되는 반면, while 문은 조건을 먼저 평가한 후 반복이 시작됨.
- while (FALSE)인 경우 루프 본문 코드가 실행되지 않음
- while (TRUE)는 repeat 구문과 동일
- while문의 일반적 형태

```
while (조건) {  
    표현식 1  
    반복변수 update  
}
```

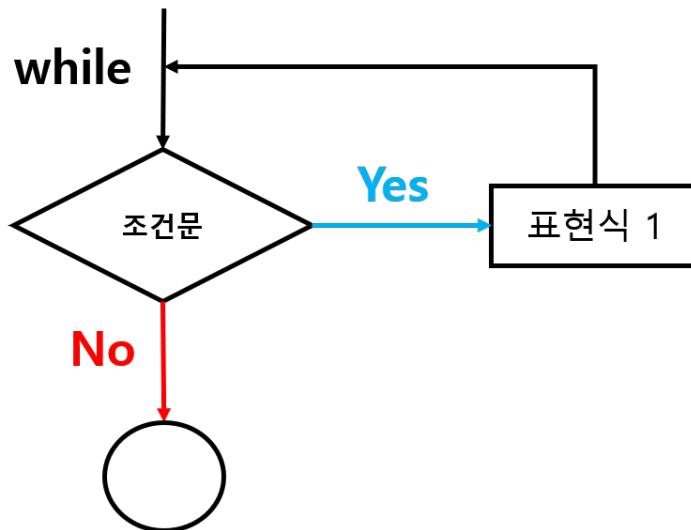


FIGURE 2.7: WHILE 구문 flow-chart

```
# WHILE 구문 예시 1
# 1:100 까지 합 계산 함수
tot <- 0; i <- 1 # 사용 변수 초기화 (update 변수)
while (i <= 100) {
  tot <- tot + i
  i <- i + 1
}
tot
```

[1] 5050

1. 초기값 i가 조건  $i \leq 100$ 인지 확인
2. 참인 경우  $tot + i$ 를 통해 tot을 업데이트 한 다음 i를 1만큼 증가
3. 만약 i에 대한 조건 평가 결과가 거짓이면 while 구문을 빠져나감

```
# while 문 조건이 TRUE 인 경우
tot <- 0; i <- 1 # 사용 변수 초기화 (update 변수)
while (TRUE) {
  tot <- tot + i
  if (i >= 100) break
  i <- i + 1
}
tot
```

[1] 5050

1. while 의 조건이 참이기 때문에 무한 반복
2. 단 i가 100과 같거나 클 경우 구문 탈출
3. 그 전 까지는 tot와 i를 갱신

```
# WHILE 구문 예시 2
# 문자열 벡터에서 특정 문자열의 인덱스를 반환

txtvec <- c("R", "package", "flow-control", "while", "if", "for", "repeat")
found <- FALSE

i <- 1

word <- readline("검색할 텍스트: ")
while (!found & i <= length(txtvec)) {
  if (txtvec[i] == word) {
    found <- TRUE
    break
  }
  cat(i, " 번째 위치에 해당 단어가 존재하지 않습니다.\n", sep="")
  i <- i + 1
}

if (found) {
  cat(i, " 번째 위치에 ", word, "를 찾았습니다.", sep = "")
} else {
  cat(word, " 단어는 해당 문자열 벡터에 존재하지 않습니다.\n", sep = "")
}
```

1. `found = FALSE, i = 1`을 초기값으로 입력
2. `readline()`으로 입력한 텍스트를 `word`에 저장
3. `found` 가 참이고 `i`가 텍스트 벡터의 길이 값과 같을 때 까지 다음 구문 반복
4. `txtvec` 각 원소와 `word` 값이 같은지 확인

`while` 입력 결과

- 1 번째 위치에 해당 단어가 존재하지 않습니다.
- 2 번째 위치에 해당 단어가 존재하지 않습니다.

3 번째 위치에 해당 단어가 존재하지 않습니다.

4 번째 위치에 `while` 를 찾았습니다.

`temp` 입력 결과

1 번째 위치에 해당 단어가 존재하지 않습니다.

2 번째 위치에 해당 단어가 존재하지 않습니다.

3 번째 위치에 해당 단어가 존재하지 않습니다.

4 번째 위치에 해당 단어가 존재하지 않습니다.

5 번째 위치에 해당 단어가 존재하지 않습니다.

6 번째 위치에 해당 단어가 존재하지 않습니다.

7 번째 위치에 해당 단어가 존재하지 않습니다.

`temp` 단어는 해당 문자열 벡터에 존재하지 않습니다.



- `repeat`, `while`과 같이 반복의 횟수가 지정되지 않는 반복구문을 불확정 반복문 (indefinite loop)이라고 함.
- 다음에 배울 `for` 구문은 위 두 반복문과는 다르게 반복의 범위를 명확히 지정하기 때문에 확정 반복문 (definite loop)라고 함.

### 2.4.3 `for` 구문

- 가장 많이 사용되는 반복구문으로 일반적인 형태는 아래와 같음

```
for (반복변수 in sequence) {
  표현식 1
  ...
}
```

- R에서 `sequence`은 특정 유형의 벡터이며, 반복변수에 `sequence`의 원소를 순차적으로 할당함

- 반복변수는 `for` 반복문 안의 표현식 1에서 사용됨

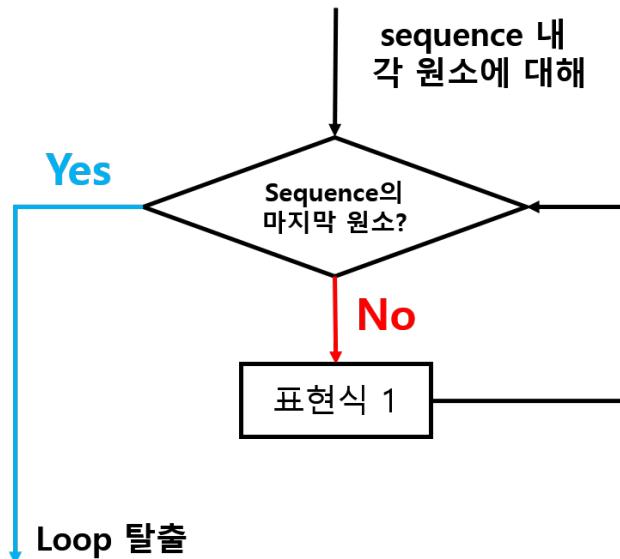


FIGURE 2.8: FOR 구문 flow-chart

```
#for 문 예시 1
student <- readxl::read_excel("data/stat-students.xlsx")
student_name <- student$이름
for (s in student_name) {
  cat(s, "즐거운 명절 보내세요^^\n")
}
```

김세민 학생!! 즐거운 명절 보내세요^^  
 김기랑 학생!! 즐거운 명절 보내세요^^  
 이종영 학생!! 즐거운 명절 보내세요^^  
 이원규 학생!! 즐거운 명절 보내세요^^  
 황보영 학생!! 즐거운 명절 보내세요^^  
 구나현 학생!! 즐거운 명절 보내세요^^  
 엄용현 학생!! 즐거운 명절 보내세요^^  
 오동원 학생!! 즐거운 명절 보내세요^^  
 윤지우 학생!! 즐거운 명절 보내세요^^

이근범 학생!! 즐거운 명절 보내세요^^  
 이승호 학생!! 즐거운 명절 보내세요^^  
 이희도 학생!! 즐거운 명절 보내세요^^  
 정보경 학생!! 즐거운 명절 보내세요^^  
 채시진 학생!! 즐거운 명절 보내세요^^  
 최호진 학생!! 즐거운 명절 보내세요^^  
 박인혜 학생!! 즐거운 명절 보내세요^^  
 박준기 학생!! 즐거운 명절 보내세요^^  
 최소미 학생!! 즐거운 명절 보내세요^^  
 백승완 학생!! 즐거운 명절 보내세요^^  
 김지원 학생!! 즐거운 명절 보내세요^^  
 신형원 학생!! 즐거운 명절 보내세요^^  
 임현정 학생!! 즐거운 명절 보내세요^^  
 정수빈 학생!! 즐거운 명절 보내세요^^  
 최예린 학생!! 즐거운 명절 보내세요^^  
 황정인 학생!! 즐거운 명절 보내세요^^

1. student\_name의 첫 번째 원소를 s에 할당
2. for 구문 안에 표현 실행
3. student\_name의 마지막 원소까지 반복

```
# 위 예시와 동일한 표현
## 인덱싱을 사용
for (i in 1:length(student_name)) {
  cat(student_name[i], "학생!! 즐거운 명절 보내세요^^\n")
}

## sequence를 만드는 함수 seq_along() 사용

for (i in seq_along(student_name)) {
```

```
cat(student_name[i], "학생!! 즐거운 명절 보내세요^^\n")
}
```

- `for` 구문 안에 `for` 문을 1개 이상 중첩 가능

```
## 2중 for 문 예시
set.seed(12345)
id <- sample(1:length(student_name), 5)
sel_student <- student_name[id]

for (i in seq_along(student_name)) {
  for (j in seq_along(sel_student)) {
    if (student_name[i] == sel_student[j]) {
      cat(sel_student[j], "님!! 당첨 축하 드립니다!!\n")
    }
  }
}
```

김기랑 님!! 당첨 축하 드립니다!!

이승호 님!! 당첨 축하 드립니다!!

채시진 님!! 당첨 축하 드립니다!!

박인혜 님!! 당첨 축하 드립니다!!

백승원 님!! 당첨 축하 드립니다!!



- 불확정 반복문 학습 시 무한루프로부터 `break`를 통해 루프에서 탈출
- 루프를 완전히 탈출하지 않고 현재 반복을 중지하고 그 다음 반복을 진행하고 싶을 경우 `next` 예약어를 사용

```
# 알파벳 e와 일치하는 경우에만 텍스트 메세지 출력
vec <- c("a", "e", "e", "i", "o", "u", "e", "z")
word <- "e"
for (i in 1:length(vec)) {
```

```
if (vec[i] != word) next  
cat(word, "가", i, "번 째 인덱스에 있네요!!\n")  
}
```

- 가 2 번 째 인덱스에 있네요!!
  - 가 3 번 째 인덱스에 있네요!!
  - 가 7 번 째 인덱스에 있네요!!
- 

## 2.5 2.5. 함수 (function)

- 함수: 특정한 목적을 위한 연산을 수행하기 위해 명명된 일련의 문장
    - 예: `sum(x) →`
- 

## 2.6 2.6. 제어 구문 이해를 위한 몇 가지 알고리즘

# 3

---

## 시뮬레이션

---

### Sketch

- 시뮬레이션.. 시뮬레이션??.. 시뮬레이션!!
- 통계학에서 시뮬레이션이 왜 필요할까?
- 중요한 기초통계 이론을 눈으로 확인할 수 있을까?



---

## **Bibliography**

---

Peng, R. D. (2016). *R programming for data science*. Learnpub.

Rizzo, M. L. (2019). *Statistical computing with R*. CRC Press.

Silge, J. and Robinson, D. (2017). *Text mining with R*. ” O'Reilly Media, Inc.”.

Wickham, H. and Grolemund, G. (2016). *R for data science: import, tidy, transform, visualize, and model data*. ” O'Reilly Media, Inc.”.

Xie, Y. (2016). *bookdown: Authoring Books and Technical Documents with R Markdown*. Chapman and Hall/CRC, Boca Raton, Florida. ISBN 978-1138700109.

Xie, Y., Dervieux, C., and Riederer, E. (2020). *R Markdown Cookbook*. Chapman and Hall/CRC, Boca Raton, Florida. ISBN 978-1000290806.

고석범 (2014). *R과 knitr를 활용한 데이터 연동형 문서 만들기*. 에이콘 출판사, 1st edition. ISBN 978-8960775510.