

한국한의학연구원, 구본초

통계 프로그래밍 언어

2020년도 1학기 충남대학교 정보통계학과 강의 노트



Contents

| | |
|-----------------------------------|-------------|
| List of Tables | ix |
| List of Figures | xi |
| Course Overview | xiii |
| I Get Started | 1 |
| 1 Introduction | 3 |
| 1.1 R 설치하기 | 4 |
| 1.2 R 시작 및 작동 체크 | 14 |
| 1.3 R script 편집기 사용 | 18 |
| 1.4 RStudio | 21 |
| 1.4.1 RStudio 설치하기 | 21 |
| 1.4.2 RStudio IDE 화면 구성 | 24 |
| 1.4.3 RStudio 환경 설정 | 30 |
| 1.4.4 RStudio 프로젝트 | 39 |
| 1.5 R 패키지 | 42 |
| 1.5.1 R 패키지 경로 확인 및 변경 | 43 |
| 1.5.2 R 패키지 설치하기 | 45 |
| 1.5.3 R 패키지 불러오기 | 46 |
| 1.6 R 기초 문법 | 47 |

| | |
|------------------------------------|-----------|
| 1.7 R Markdown (맛보기) | 51 |
| 2 R 객체(R object) | 61 |
| 2.1 스칼라(scalar) | 62 |
| 2.1.1 선언 | 63 |
| 2.1.2 숫자형 | 64 |
| 2.1.3 문자형 | 66 |
| 2.1.4 논리형 스칼라 | 67 |
| 2.1.5 결측값(missing value) | 71 |
| 2.1.6 NULL 값 | 72 |
| 2.1.7 무한대/무한소/숫자아님 | 73 |
| 2.2 벡터(vector) | 74 |
| 2.2.1 벡터의 특징 | 74 |
| 2.2.2 벡터의 연산 | 79 |
| 2.2.3 벡터의 색인(indexing) | 87 |
| 2.2.4 벡터 관련 함수 | 90 |
| 2.3 리스트(list) | 100 |
| 2.3.1 리스트 생성 | 101 |
| 2.3.2 리스트 색인 | 104 |
| 2.3.3 리스트에 원소 추가/제거 | 108 |
| 2.3.4 리스트의 결합 | 111 |
| 2.4 행렬(matrix) | 113 |
| 2.4.1 행렬의 연산 | 117 |
| 2.4.2 행렬의 색인 | 129 |
| 2.4.3 행과 열 추가 및 제거 | 135 |
| 2.4.4 행렬 관련 함수 | 137 |
| 2.4.5 벡터와 행렬의 차이점 | 142 |

Contents v

| | | |
|-------|----------------------------------|-----|
| 2.4.6 | 의도치 않은 차원축소 피하기 | 142 |
| 2.5 | 배열(array) | 144 |
| 2.5.1 | 배열의 생성 및 색인 | 144 |
| 2.5.2 | 배열의 확장 예제 | 147 |
| 2.6 | 요인(factor)과 테이블(table) | 151 |
| 2.6.1 | 요인(factor) | 152 |
| 2.6.2 | 테이블(table) | 159 |
| 2.7 | 데이터 프레임(data frame) | 171 |
| 2.7.1 | 데이터 프레임 생성 | 171 |
| 2.7.2 | 데이터 프레임 접근 및 필터링 | 179 |
| 2.7.3 | 데이터 프레임 관련 함수 | 186 |
| 2.7.4 | *apply() 계열 함수 | 195 |
| 2.8 | 객체의 유형 판별 및 변환 | 208 |
| 2.9 | Homework #2 | 212 |
| 2.10 | Homework #3 | 214 |
| 3 | 문자열 처리와 정규표현식 | 221 |
| 3.1 | 유용한 문자열 관련 함수 | 224 |
| 3.1.1 | nchar() | 224 |
| 3.1.2 | paste(), paste0() | 225 |
| 3.1.3 | sprintf() | 228 |
| 3.1.4 | substr() | 231 |
| 3.1.5 | tolower(), toupper() | 232 |
| 3.2 | 정규표현식 기본 함수 | 232 |
| 3.2.1 | grep(), grep1() | 232 |
| 3.2.2 | gregexpr(), gregexpr() | 235 |
| 3.2.3 | sub(), gsub() | 238 |

| | |
|--|------------|
| 3.2.4 <code>regexec()</code> | 240 |
| 3.2.5 <code>strsplit()</code> | 245 |
| 3.3 정규 표현식 (regular expression) | 246 |
| 3.3.1 기본 메타 문자 | 247 |
| 3.3.2 문자 집합 | 254 |
| 3.3.3 문자 클래스 | 256 |
| 3.3.4 정규 표현식 예시 | 258 |
| 4 데이터 핸들링 (Data handling) | 265 |
| 4.1 Prerequisites | 267 |
| 4.1.1 외부데이터 불러오기 및 저장 | 267 |
| 4.2 Tidyverse | 278 |
| 4.3 <code>readr</code> 패키지 | 280 |
| 4.3.1 Excel 파일 입출력 | 286 |
| 4.3.2 <code>tibble</code> 패키지 | 288 |
| 4.4 <code>dplyr</code> 패키지 | 293 |
| 4.4.1 파이프 연산자: <code>%>%</code> | 295 |
| 4.4.2 <code>filter()</code> | 298 |
| 4.4.3 <code>arrange()</code> | 303 |
| 4.4.4 <code>select()</code> | 305 |
| 4.4.5 <code>mutate()</code> | 313 |
| 4.4.6 <code>transmute()</code> | 316 |
| 4.4.7 <code>summarise()</code> | 317 |
| 4.4.8 <code>group_by()</code> | 318 |
| 4.4.9 <code>dplyr</code> 관련 유용한 함수 | 321 |
| 4.4.10 부가 기능 | 331 |
| 4.4.11 데이터 연결 | 343 |

| | |
|--|------------|
| <i>Contents</i> | vii |
| 4.4.12 확장 예제: Gapminder | 356 |
| 4.5 데이터 변환 | 366 |
| 4.5.1 Tidy data | 367 |
| 4.5.2 Long format | 372 |
| 4.5.3 Wide format | 385 |
| 4.5.4 Separate and unite | 391 |
| 4.6 Homework #4 | 400 |
| 5 데이터 시각화 | 405 |
| 5.1 R 기본 그래프 함수 | 407 |
| 5.2 고수준 그래프 함수 | 409 |
| 5.2.1 <code>plot()</code> 함수 | 409 |
| 5.2.2 주요 고수준 그래픽 함수 | 426 |



List of Tables

| | |
|--|-----|
| 0.1 강의 계획표 | xvi |
| 1.1 R help 관련 명령어 리스트 | 17 |
| 2.1 R언어의 기본 수치 연산자 | 64 |
| 2.2 R언어의 논리형 연산자 | 68 |
| 2.3 R언어의 비교 연산자 | 68 |
| 2.4 리스트 데이터 접근 방법 | 105 |
| 2.5 스프레드시트 기본 형태 예시 | 171 |
| 2.6 R 객체 타입 판별 및 변환 함수 | 209 |
| 3.1 정규표현식 메타 문자: 기본 | 247 |
| 3.2 정규표현식 메타 문자: 문자집합 | 254 |
| 3.3 정규표현식 주요 문자 클래스 | 256 |
| 3.4 정규표현식: POSIX 문자 클래스 | 257 |
| 4.1 dplyr 패키지 함수와 R base 패키지 함수 비교 | 294 |
| 4.2 mpg 데이터셋 설명(코드북) | 300 |
| 4.3 flights 데이터셋 코드북 | 345 |
| 4.4 airports 데이터셋 코드북 | 347 |
| 4.5 planes 데이터셋 코드북 | 348 |
| 4.6 weather 데이터셋 코드북 | 349 |

| | |
|--|-----|
| 4.7 dplyr join 함수와 merge() 함수 비교 | 356 |
| 4.8 gapminder-exercise.xlsx 설명 | 357 |
| 4.9 Tidy data 예시 데이터 1 | 368 |
| 4.10 Tidy data: 예시 데이터 1과 동일 내용, 다른 레이아웃 | 369 |
| 4.11 Tidy data: 예시 데이터 1 구조 변환 | 369 |
| 4.12 tidyr 패키지 내장 데이터 who 코드 설명 | 383 |

List of Figures

| | | |
|------|---|----|
| 1.1 | Windows에서 R 실행화면(콘솔 창, SDI 모드) | 13 |
| 1.2 | 정규분포 100개의 히스토그램 | 17 |
| 1.3 | cars 데이터셋의 speed와 dist 간 2차원 산점도: speed는 자동차 속도(mph)이고 dist는 해당 속도에서 브레이크를 밟았을 때 멈출 때 까지 걸린 거리(ft)를 나타냄. | 20 |
| 1.4 | RStudio 화면 구성 : 우하단 그림은 http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html 에서 발췌 | 24 |
| 1.5 | RStudio 콘솔창에서 명령어 실행 후 출력결과 화면 | 25 |
| 1.6 | RStudio 스크립트 새로 열기 | 26 |
| 1.7 | RStudio Environment 창 객체 상세 정보 및 스프레드 시트 출력 결과 | 27 |
| 1.8 | R General option 팝업 창 | 31 |
| 1.9 | R Markdown의 최종 결과물 산출과정 (http://r-project-reporting-template/) | 52 |
| 1.10 | test.html 문서 화면(저장 폴더 내 ‘test.html‘을 크롬 브라우저로 실행) | 55 |

| | |
|---|-----|
| 2.1 R 데이터 탑입 구조 다이어그램: [R, Python 분석과 프로그래밍 (by R Friend)](http://rfriend.tistory.com/)에서 발췌 후 수정 | 63 |
| 2.2 https://www.geeksforgeeks.org/matlab-rgb-image-representation/ 에서 발췌 | 147 |
| 4.1 Data 분석의 과정. @wickham-2016r에서 발췌 | 266 |
| 4.2 filter() 함수 다이어그램 | 298 |
| 4.3 가능한 모든 boolean 연산 종류: x는 좌변, y는 우변을 의미하고 음영은 연산 이후 선택된 부분을 나타냄. | 299 |
| 4.4 arrange() 함수 다이어그램 | 303 |
| 4.5 select() 함수 다이어그램 | 306 |
| 4.6 mutate() 함수 다이어그램 | 313 |
| 4.7 summarise() 함수 다이어그램 | 317 |
| 4.8 NYC flight 2013 데이터 관계도(https://r4ds.had.co.nz/ 에서 발췌) | 350 |
| 4.9 데이터의 구성 요소 | 368 |
| 5.1 Anscombe's quartet: https://goo.gl/Ugv3Cz 에서 스크립트 발췌 | 406 |
| 5.2 R 그래프영역 | 408 |

Course Overview



본 문서는 2020년도 1학기 정보통계학과에서 개설한 “통계 프로그래밍 언어” 강의를 위해 개발한 강의 노트이고 주 단위로 업데이트될 예정임. 본 강의 노트는 <https://zorba78.github.io/cnu-r-programming-lecture-note/> 에서 확인할 수 있고, 해당 페이지에서 pdf 파일 다운로드가 가능함. 본 문서는 Yihui Xie가 개발한 **bookdown** 패키지 (Xie, 2016)를 활용하여 생성한 문서이고 Google Chrome 또는 Firefox 브라우저에 최적화 됨. 아울러 충남대학교 정보통계학과 이상인 교수님의 2019년도 2학기 “통계패키지활용” 강의 노트와 동국대학교 ICT빅데이터학부 김진석 교수님의 R 프로그래밍 및 실습¹ 강의 자료 내용과 구성을 참고하여 작성함. 재택 수업 시 학생들이 사용하고 있는 컴퓨터의 인터넷 접속이 원활하다는 가정 하에서 강의를 진행할 예정이기 때문에 수강 시 온라인 상태 유지가 필수임.

강의소개

R은 뉴질랜드 오클랜드 대학의 Robert Gentleman 과 Ross Ihaka 가 AT&T 벨 연구소에서 개발한 S 언어를 기반으로 개발한 GNU 환경의 통계 계산 및 프로그래밍 언어이다. 현재 R 소프트웨어는 통계학 뿐 아니라 데이터 과학을 포함한 의학, 생물학 등 다양한 분야에서 활용되고 있으며 특히 통계 소프트웨어 개발과 데이터 분석에 많이 활용되고 있다. 본 강의는 데이터 분석을 위한 R의

기초 문법과 통계학 입문에서 학습한 몇 가지 중요한 통계적 이론에 대한 시뮬레이션 방법을 다룬다. 아울러 R package를 활용한 데이터 핸들링 및 시각화 그리고 Rmarkdown을 활용한 재현가능(reproducible)한 문서 작성법에 대해 학습하고자 한다.

교과 목표

- R 기초 문법 습득
- R package를 활용한 데이터 핸들링 및 자료 시각화
- R 시뮬레이션을 통한 통계학 기초 이론 확인
- R을 이용한 데이터 분석 실습
- R markdown을 이용한 재현가능(reproducible)한 보고서 작성 방법
습득

선수과목

통계학 개론

수업 방법

- 강의: 50 %
- 실험/실습: 50%

평가방법

- 중간고사: 40 %
- 기말고사: 40 %
- 출석: 10 %
- 과제: 10 %

수업 규정

- 3번 지각은 1번 결석으로 처리
- 특별한 사유 없이 수업 중간에 이탈한 경우 결석으로 처리
- 특별한 사유로 인해 결석 또는 지각을 할 경우 사유를 증빙할 수 있는 서류 제출 시 출석으로 인정
- 출결 미달, 중간 또는 기말고사 미 응시인 경우 F 학점으로 처리
- 수업 중 휴대폰 및 각종 모바일 기기 사용 금지

교재 및 참고문헌

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

참고 자료

- 빅데이터 분석 도구 R 프로그래밍 ([매트로프](#), 2012)
- 실리콘밸리 데이터과학자가 알려주는 따라하며 배우는 데이터 과학 ([권재명](#), 2017)
- R을 이용한 데이터 처리&분석 ([서민구](#), 2014)
- R 그래픽스 ([유충현 et al.](#), 2005)
- ggplot2: elegant graphics for data analysis² ([Wickham](#), 2016)
- R for data science³ ([Wickham and Grolemund](#), 2016)
- Statistical Computing with R ([Rizzo](#), 2019)

²<https://ggplot2-book.org/>

³<https://r4ds.had.co.nz/>

강의 계획

TABLE 0.1: 강의 계획표

| 주차 | 강의 내용 | 과제 |
|---------|--|------|
| Week 1 | R 소개, R/R Studio 설치, R 패키지 설치, 과제 1 R 맛보기 및 markdown 문서 만들기 | |
| Week 2 | R 자료형: 스칼라, 벡터, 리스트 | |
| Week 3 | R 자료형: 행렬 및 배열 | 과제 2 |
| Week 4 | R 자료형: 팩터, 테이블, 데이터 프레임 | |
| Week 5 | R 자료형: 문자열과 정규 표현식 | 과제 3 |
| Week 6 | 데이터 프레임 가공 및 시각화 I | |
| Week 7 | 데이터 프레임 가공 및 시각화 II | 과제 4 |
| Week 8 | 중간고사 | |
| Week 9 | 데이터 프레임 가공 및 시각화 III | |
| Week 10 | R 프로그래밍: 조건문, 반복문, 함수 | 과제 5 |
| Week 11 | 통계시뮬레이션 I: 표본분포 및 중심극한정리 | |
| Week 12 | 통계시뮬레이션 2: 신뢰구간과 가설검정 | 과제 6 |
| Week 13 | R을 이용한 기초통계 분석 | |
| Week 14 | R markdown 활용 | 과제 7 |
| Week 15 | 기말고사 | |

Part I

Get Started



1

Introduction

1. R 프로그램

- 데이터 분석을 위한 자료 전처리, 통계 및 시각화를 지원하는 컴퓨터 언어 및 환경
- 1980년 AT&T 벨 연구소의 John Chambers가 개발한 S 언어를 기반으로 1995년 뉴질랜드 Auckland 대학의 통계학과 교수 Robert Gentleman과 Ross Ihaka 가 개발
- GNU¹ 기반의 오픈 소스
- 통계학, 전산학, 생물학, 의학 등 거의 모든 학문분야에서 분석도구로 활용되고 있고, 최근 data science 분야에서 널리 활용

2. R 언어의 특징

- 무료 소프트웨어
- CRAN (Comprehensive R Archive Network)²에서 배포
- 특정 vendor가 아닌 전 세계 연구자들이 개발한 알고리즘 및 최신 함수 활용 가능 (packaging system)
- 범용적으로 사용되는 거의 대부분의 운영체제 (Windows, Mac, Linux)에서 작동 가능

¹https://en.wikipedia.org/wiki/GNU_Project

²<http://cran.r-project.org/web/view>

- 방대한 개발 및 사용 생태계 형성
- 강력한 그래픽 기능



유용한 웹 사이트: R과 관련한 거의 모든 문제는 Googling (구글을 이용한 검색)을 통해 해결 가능(검색주제 + “in R” or “in R software”)하고 많은 해답들이 아래 열거한 웹 페이지에 게시되어 있음.

- R 프로그래밍에 대한 Q&A: Stack Overflow³
- R 관련 웹 문서 모음: Rpubs⁴
- R package에 대한 raw source code 제공: Github⁵
- R을 이용한 통계 분석: Statistical tools for high-throughput data analysis (STHDA)⁶

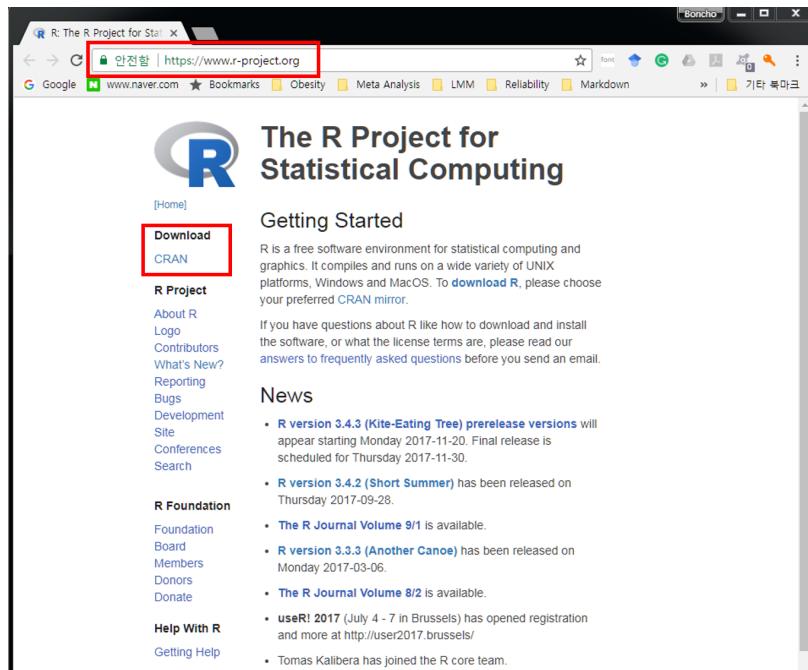
1.1 R 설치하기

R 다운로드 사이트: <https://www.r-project.org> 또는 <https://cran.r-project.org>

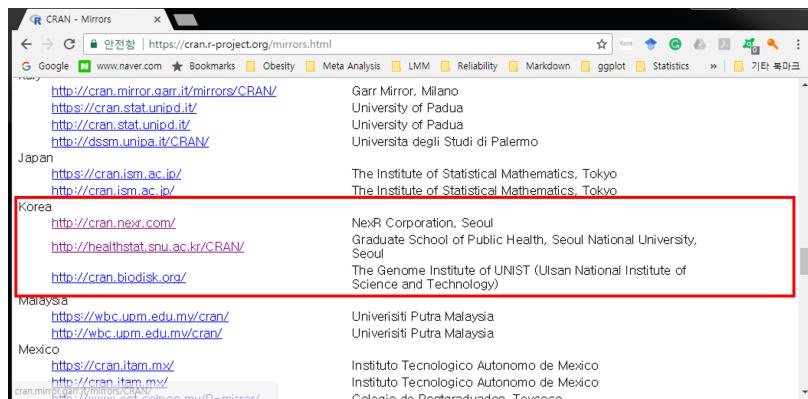
1. 웹 브라우저 (i.e. Explore, Chrome, Firefox 등)의 주소 입력창에 <https://www.r-project.org>
2. 좌측 R Logo 하단 Download 아래 CRAN 클릭

1.1 R 설치하기

5



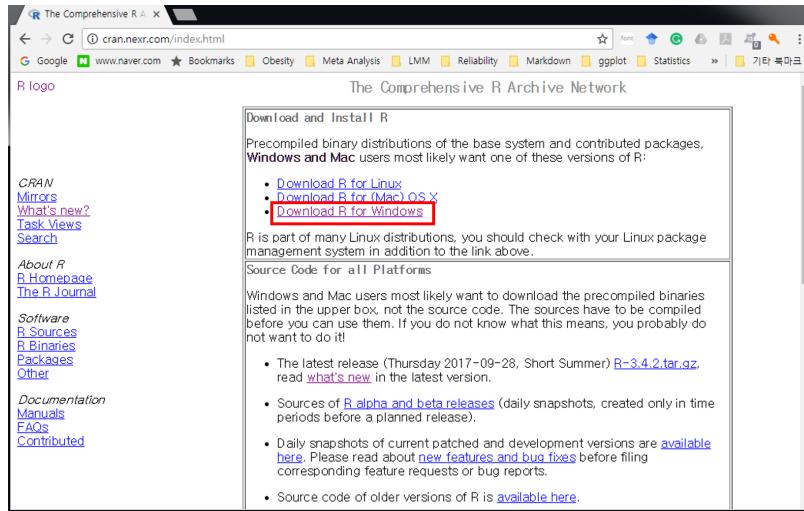
3. 클릭 후 연결한 페이지를 스크롤 후 Korea 아래 링크⁷ 클릭



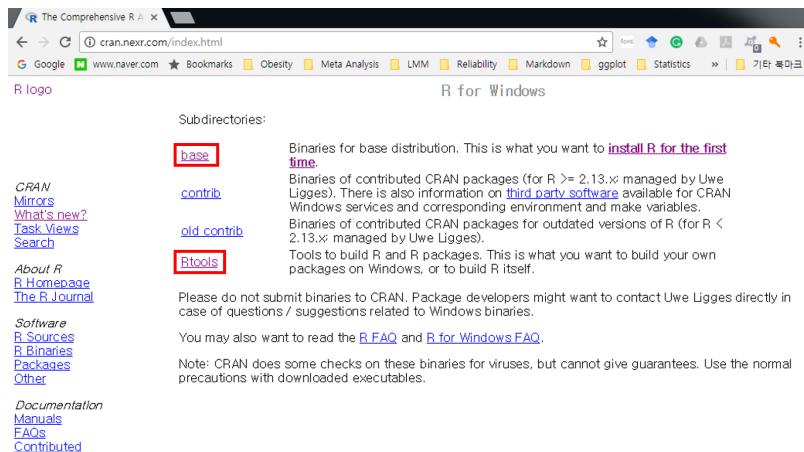
4. 클릭 후 세 가지 운영체제(Linux, Mac OS X, Windows)에 따른 R 버전 선택 가능⁸

⁷ 해당 링크들은 접속 시점에 따라 변경될 수 있음

⁸ 본 노트는 Windows 버전 설치만 다룸



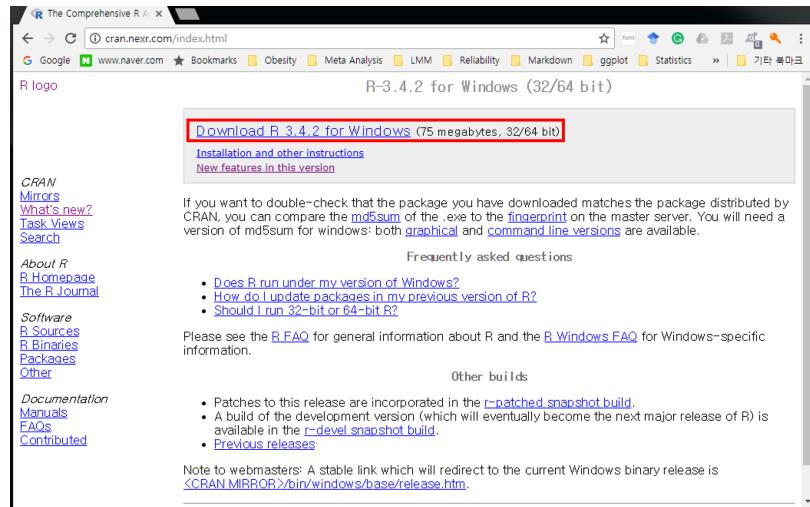
5. Downloads R for Windows 링크 클릭하면 다음과 같은 화면으로 이동



다음 하위폴더에 대한 간략 설명

- **base**: R 실행 프로그램
- **contrib**: R package의 바이너리 파일
- **Rtools**: R package 개발 및 배포를 위한 프로그램

6. 위 화면에서 **base** 링크 클릭 후 아래 화면에서 **Downloads R 3.x.x for Windows** 를 클릭 후 설치 파일을 임의의 디렉토리에 저장 및 실행

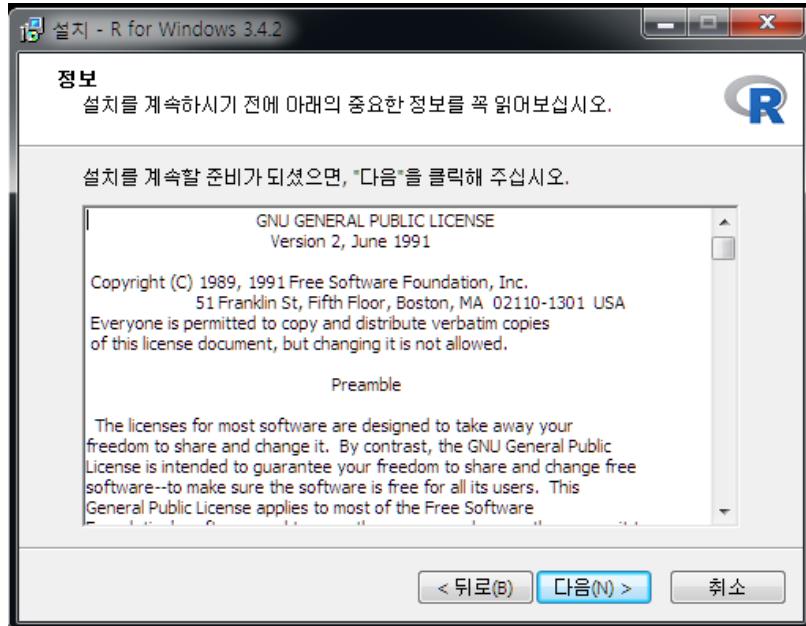


7. 다운로드한 파일을 실행하면 아래와 같은 대화창이 나타남

- 한국어 선택 → 환영 화면에서 [다음(N)>] 클릭

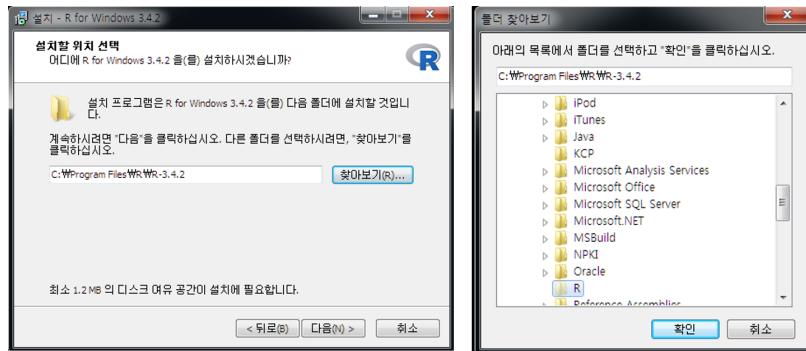


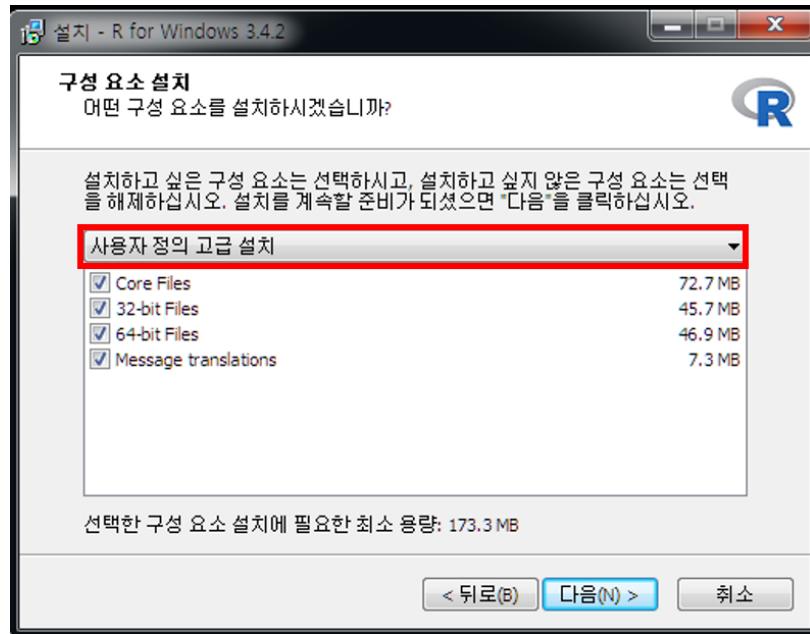
8. GNU 라이센스에 대한 설명 및 동의 여부([다음(N)>]) 클릭



9. 설치 디렉토리 설정 및 구성요소 설치 여부

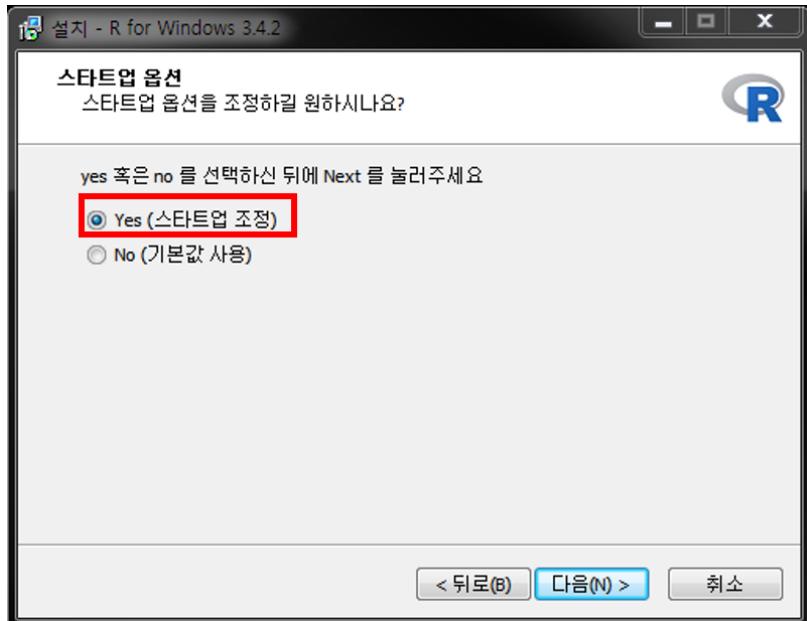
- 원하는 디렉토리 설정 (예: C:\R\R-3.x.x)
 - 기본 프로그램 (“Core Files”), 32 또는 64 bit 용 설치 파일, R console
- 한글 번역 모두 체크 뒤 [다음(N)>] 클릭





10. R 스타트업 옵션 지정

- 기본값("No" check-button)으로도 설치 진행 가능
- 본 문서에서는 스타트업 옵션 변경으로 진행

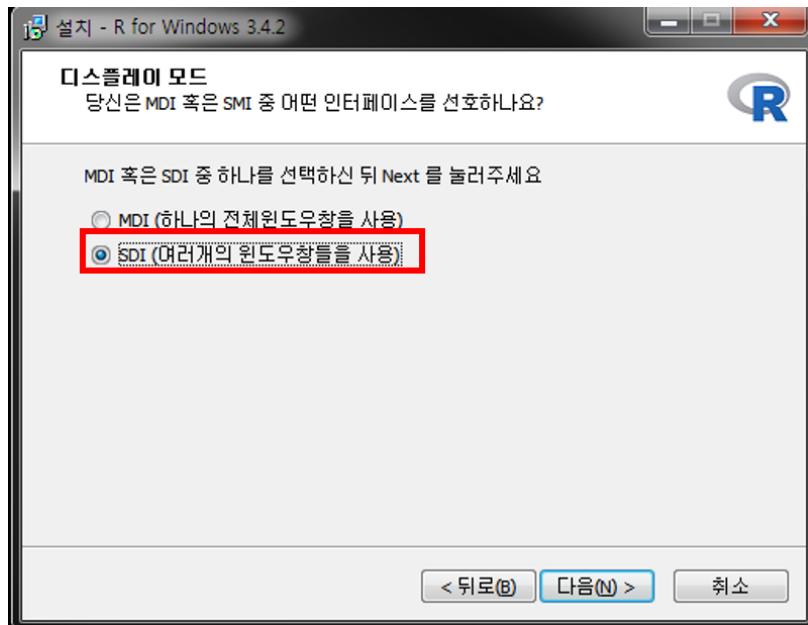


11. 화면표시방식(디스플레이) 모드 설정 변경

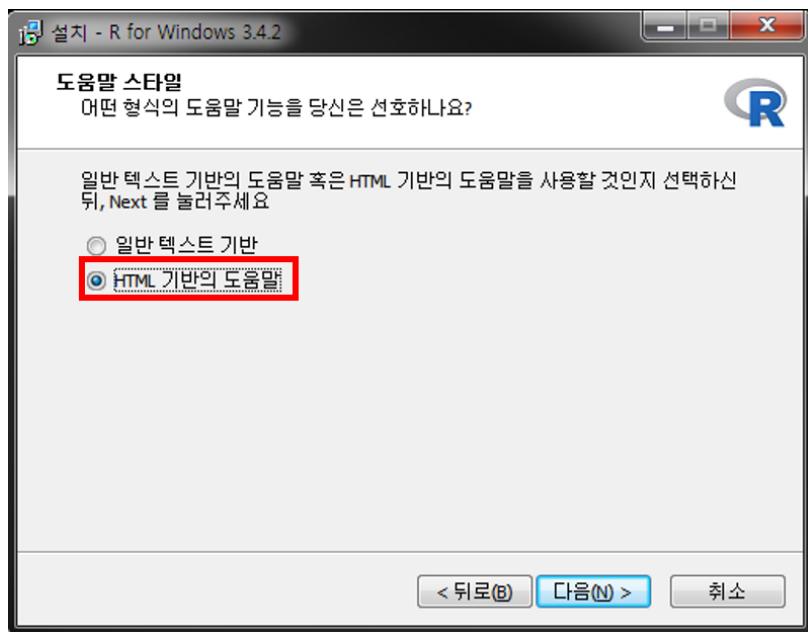
- MDI: 한 윈도우 내에서 script 편집창, 출력, 도움말 창 사용
- SDI: 다중 창에서 각각 script 편집창, 출력, 도움말 등을 독립적으로 열기

1.1 R 설치하기

11

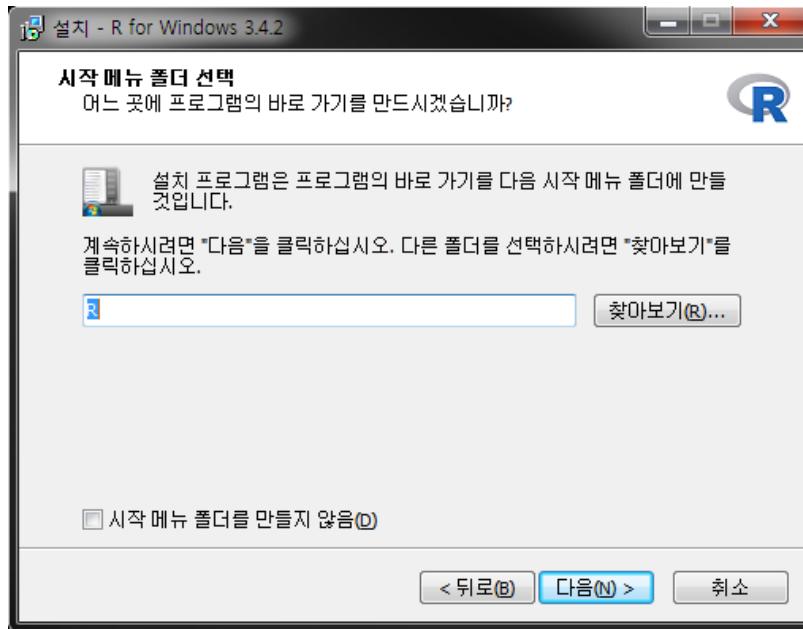


12. 도움말 형식에서 HTML 도움말 기반 선택



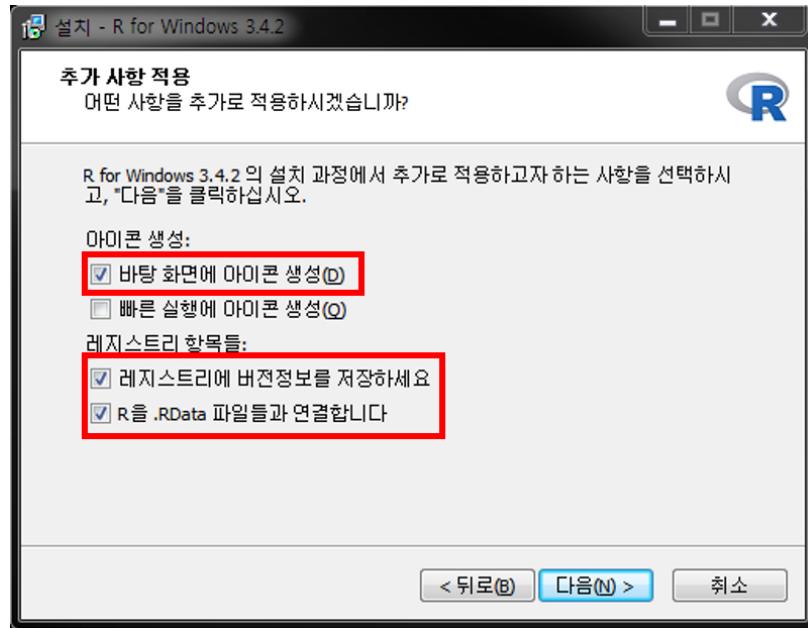
13. 시작메뉴 폴더 선택

- “바로가기”를 생성할 시작 메뉴 폴더 지정 후 [다음(N)>] 클릭 후 설치 진행
- 하단 “시작메뉴 폴더 만들지 않음” 체크박스 표시 시 시작메뉴에 “바로가기” 아이콘이 생성되지 않음(실행에 전혀 지장 없음)



14. 추가 옵션 지정 : 바탕화면 아이콘 생성 등 추가적 작업 옵션 체크 후 [다음(N)>] 클릭 → 설치 진행

- 설치된 R 버전 정보 레지스트리 저장 여부
- .Rdata 확장자를 R 실행파일과 자동 연계



15. 설치 완료 후 바탕화면의 R 아이콘을 더블클릭하면 Rgui가 실행

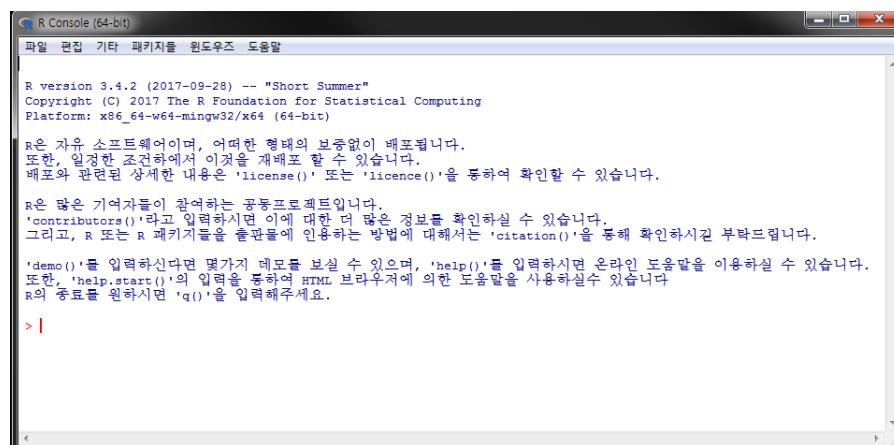


FIGURE 1.1: Windows에서 R 실행화면(콘솔 창, SDI 모드)

1.2 R 시작 및 작동 체크



실습: 설치된 R을 실행 후 보이는 R 콘솔(console) 창에서 명령어를 실행하고 결과 확인

Figure 1.1 에서 > 기호는 R의 명령 프롬프트(command prompt) 임

- → 컴퓨터가 사용자 명령을 기다리고 있다는 기호
- 1. 현재 R session⁹ 정보(R 설치 버전, locale, 로딩 packages) 출력

```
# R의 설치 버전 및 현재 설정된 locale(언어, 시간대) 및 로딩된 R package 정보 출력
sessionInfo()
```

```
R version 3.6.3 (2020-02-29)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 18363)
```

```
Matrix products: default
```

```
locale:
[1] LC_COLLATE=Korean_Korea.949  LC_CTYPE=Korean_Korea.949
[3] LC_MONETARY=Korean_Korea.949 LC_NUMERIC=C
[5] LC_TIME=Korean_Korea.949
```

```
attached base packages:
[1] stats      graphics   grDevices utils      datasets  methods   base
```

```
other attached packages:
```

⁹현재 실행되고 있는 R의 작업공간

```
[1] knitr_1.28
```

```
loaded via a namespace (and not attached):  
[1] compiler_3.6.3  magrittr_1.5    bookdown_0.18.1 htmltools_0.4.0  
[5] tools_3.6.3     yaml_2.2.1      Rcpp_1.0.4       stringi_1.4.6  
[9] rmarkdown_2.1    stringr_1.4.0   digest_0.6.25   xfun_0.12  
[13] rlang_0.4.5     evaluate_0.14
```

2. 문자열 출력

```
#문자열 출력  
print("Hello R") #문자열  
  
[1] "Hello R"
```

```
# 기호는 주석의 시작을 의미하고 실제로 실행되지 않음 같은 행에서 # 뒤 내용의  
코드 역시 실행되지 않음
```

3. a라는 변수에 숫자 9, b라는 변수에 숫자 7를 할당 후 출력

```
# 수치형 값(scalar)을 변수에 할당(assign)  
# 여러 명령어를 한줄에 입력할 때에는 세미콜론(;)으로 구분  
a = 9; b = 7  
a
```

```
[1] 9
```

```
b
```

```
[1] 7
```

4. 변수 a와 b의 사칙연산

```
a+b; a-b; a*b; a/b
```

```
[1] 16
```

```
[1] 2
```

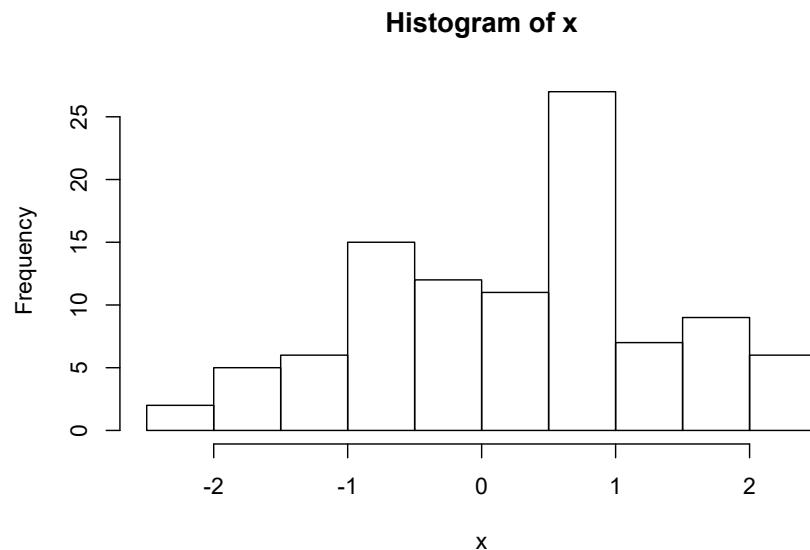
```
[1] 63
```

```
[1] 1.285714
```

5. R 그래픽 맛보기 : 정규분포로부터 난수 100개 생성 후 생성된 데이터에 대한 히스토그램 작성

```
# 난수 생성 시 같은 매번 달라지기 때문에 seed를 주어 일정값이 생성되도록 고정
# "="과 "<->"는 모두 동일한 기능을 가진 할당 연산자임
# 평균이 0이고 분산이 1인 정규분포에서 난수 100개 생성
set.seed(12345) # random seed 지정
x <- rnorm(100) # 난수 생성
hist(x) # 히스토그램
```

 R 명령어 또는 전체 프로그램 소스 실행 시 매우 빈번히 오류가 나타나는데, 이를 해결할 수 있는 가장 좋은 방법은 앞에서 언급한 Google을 이용한 검색 또는 R 설치 시 자체적으로 내장되어 있는 도움말을 참고하는 것이 가장 효율적임.

**FIGURE 1.2:** 정규분포 100개의 히스토그램**TABLE 1.1:** R help 관련 명령어 리스트

| 도움말 보기 명령어 | 설명 | 사용법 |
|-----------------------|--|------------------------|
| ‘help’ 또는 ‘?’ | 도움말 시스템 호출 | ‘help(함수명)’ |
| ‘help.search’ 또는 ‘??’ | 주어진 문자열을 포함한 문서 검색 | ‘help.search(pattern)’ |
| ‘example’ | topic의 도움말 페이지에 있는 examples section 실행 | ‘example(함수명)’ |
| ‘vignette’ | topic의 pdf 또는 html 레퍼런스 메뉴얼 불러오기 | ‘vignette(패키지명 또는 패턴)’ |

Vignette 의 활용

– vignette()에서 제공하는 문서는 데이터를 기반으로 사용하고자 하는 패키지의 실

제 활용 예시를 작성한 문서이기 때문에 초보자들이 R 패키지 활용에 대한 접근성을 높혀줌.

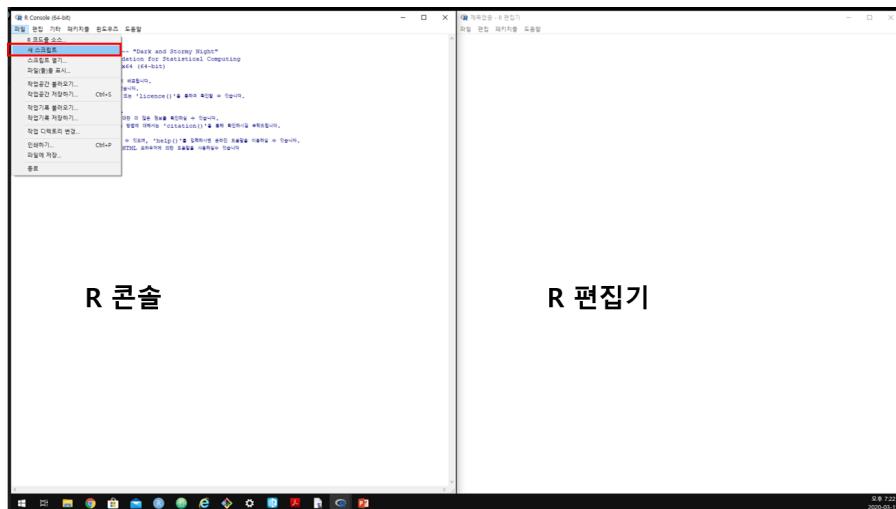
- `browseVignettes()` 명령어를 통해 vignette을 제공하는 R 패키지 및 해당 vignette 문서 확인 가능

1.3 R script 편집기 사용



실습: R 설치 후 Rgui에서 제공하는 편집기(R editor)에 명령어를 입력하고 실행

설치된 R을 실행 후 상단 pull-down 메뉴에서 [File] → [새 스크립트]를 선택하면 아래 그림과 같이 편집창(R 인스톨 시 SDI 옵션 기준)이 나타남



편집기 창에 다음 명령어 입력

```
# R에 내장된 cars 데이터셋 불러오기 cars dataset에 포함된 변수들의 기초통계량
# 출력 2차원 산점도

data(cars)
help(cars) # cars 데이터셋에 대한 설명 help 창에 출력
head(cars) # cars 데이터셋 처음 6개 행 데이터 출력
summary(cars) # cars 데이터셋 요약
plot(cars) # 변수가 2개인 경우 산점도 출력
```

- 편집창에서 한 줄을 실행시키려면 명령어가 입력된 줄에서 [Ctrl] + [R] 입력
- 편집창에 입력한 모든 명령어를 실행시키려면 모든 줄을 선택(마우스 또는 [Shift] + ↓)

```
speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10

      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
```

- R은 명령어를 입력하고 실행결과를 확인하는 대화형(interpreter) 방식
- 콘솔창에서 ↑/↓를 누르면 이전/이후 실행 명령 기록 확인 가능

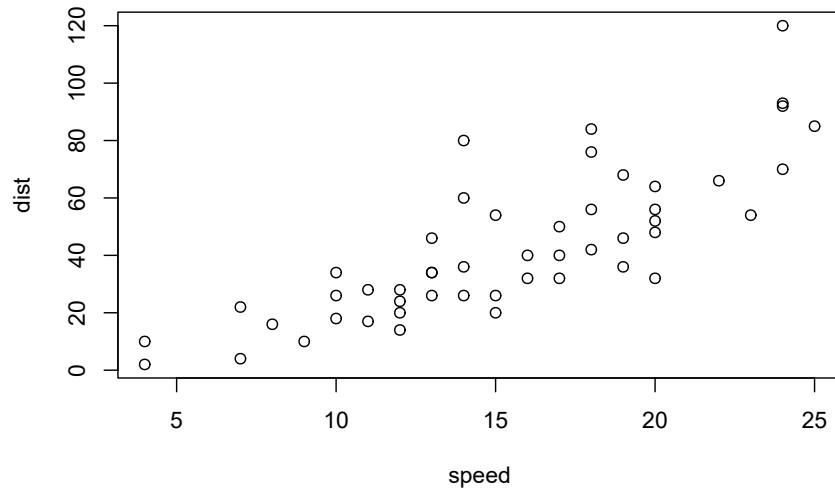


FIGURE 1.3: cars 데이터셋의 speed와 dist 간 2차원 산점도: speed는 자동차 속도(mph)이고 dist는 해당 속도에서 브레이크를 밟았을 때 멈출 때 까지 걸린 거리(ft)를 나타냄.

- 여러 줄 이상 R 명령어라든가 반복적, 장기간 작업을 수행해야 할 경우 R 명령어로 구성된 스크립트 작성 후 일괄 실행하는 것이 일반적
- 여러 다중 명령 코딩 시 콘솔창에 직접 입력하는 것은 비효율적이므로 스크립트 에디터를 사용
- 위 예시처럼 R 에디터 사용할 수 있으나 가독성 및 코딩 효율이 떨어짐
- 과거 많이 사용됐던 R 에디터: WinEdt¹⁰, Tinn-R¹¹, Vim¹²
- 현재 가장 범용적 R 에디터: Rstudio

¹⁰<http://www.winedt.com>

¹¹<https://sourceforge.net/projects/tinn-r/>

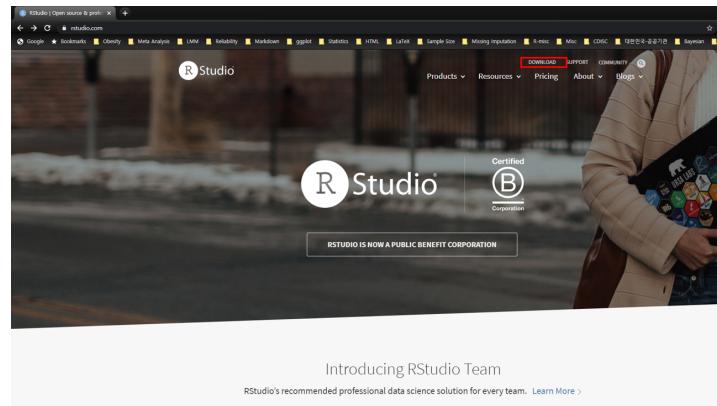
¹²http://www.vim.org/scripts/script.php?script_id=2628

1.4 RStudio

- RStudio¹³: R 통합 분석/개발 환경(integrated development environment, IDE)으로 현재 가장 대중적으로 사용되고 있는 R 사용 환경
- 명령 곤솔 외 파일 편집, 데이터 객체, 명령 기록(.history), 그래프 등에 쉽게 접근 가능
- RStudio 독자적인 개발 환경 제공: Rmarkdown, Rnotebook, Shiny Web Application 등 다양한 R 환경을 제공
- 버전관리(git, subversion)를 통해 project 관리 가능
- 무료 및 유료 소프트웨어 제공

1.4.1 RStudio 설치하기

1. 웹 브라우저를 통해 <https://rstudio.com> 접속 후 상단 DOWNLOAD¹⁴ 링크 클릭



2. Desktop 또는 Server 버전 중 택일

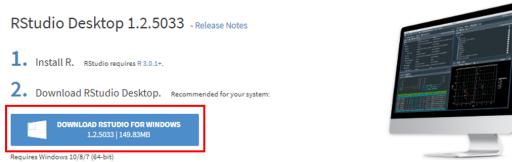
¹³<https://rstudio.com/>

¹⁴<https://rstudio.com/products/rstudio/download/>

- 서버용 설치를 위해서는 Server 클릭 → 소규모 자료 분석용으로는 불필요
- 여기서는 Desktop 버전 선택 후 다음 링크로 이동

The screenshot shows the RStudio download page. At the top, there's a navigation bar with links for Products, Resources, Pricing, About, and Blogs. Below the navigation is a large blue header with the text "Download RStudio". Underneath the header, there's a section titled "Choose Your Version" which describes RStudio as a set of integrated tools for R. It lists four options: RStudio Desktop (Free), RStudio Desktop (Commercial License \$995/year), RStudio Server (Free), and RStudio Server Pro (\$4,975/year). Each option has a "DOWNLOAD" or "BUY" button. The "DOWNLOAD" button for RStudio Desktop (Free) is highlighted with a red box. To the right of the version options, there's a sidebar for the RStudio Team, which includes a bio and a "LEARN MORE" link. Below the main section, there's a comparison chart showing features like "Integrated Tools for R", "Priority Support", and "Access via Web Browser" across the different RStudio products.

3. 운영체제에 맞는 Rstudio installer 다운로드(여기서는 Windows 버전 다운로드)



All Installers

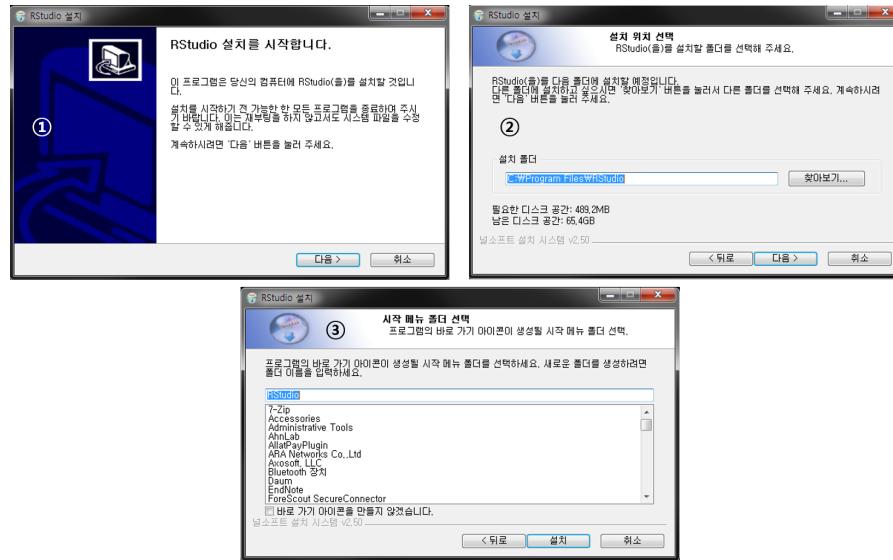
Linux users may need to import RStudio's public code-signing key prior to installation, depending on the operating system's security policy.
RStudio 1.2 requires a 64-bit operating system. If you are on a 32 bit system, you can use an older version of RStudio.

| OS | Download | Size | SHA-256 |
|---------------------|---|-----------|----------|
| Windows 10/8/7 | RStudio-1.2.5033.exe | 149.83 MB | 77d08c1b |
| macOS 10.12+ | RStudio-1.2.5033.dmg | 128.89 MB | b07v9075 |
| Ubuntu 14/Debian 8 | rstudio-1.2.5033-amd64.deb | 94.18 MB | 05cc6a22 |
| Ubuntu 16 | rstudio-1.2.5033-amd64.deb | 104.14 MB | a1591ed7 |
| Ubuntu 18/Debian 10 | rstudio-1.2.5033-amd64.deb | 108.21 MB | 05ea4295 |
| Fedor 18/Red Hat 7 | rstudio-1.2.5033-x86_64.rpm | 120.23 MB | 35e14bd8 |
| Fedor 28/Red Hat 8 | rstudio-1.2.5033-x86_64.rpm | 120.87 MB | a52b1d0d |

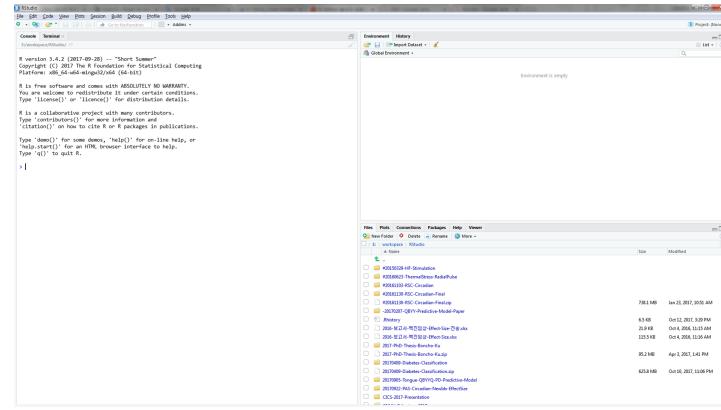
4. RStudio installer 다운로드 시 파일이 저장된 폴더에서 보통

RStudio-xx.xx.xxx.exe 형식의 파일명 확인

- 더블 클릭 후 실행
- [다음>] 몇 번 클릭 후 설치 종료



5. 바탕화면 혹은 시작 프로그램에 새로 설치된 RStudio 아이콘 클릭 후 아래와 같은 프로그램 창이 나타나면 설치 성공



1.4.2 RStudio IDE 화면 구성

RStudio는 아래 그림과 같이 4개 창으로 구성¹⁵

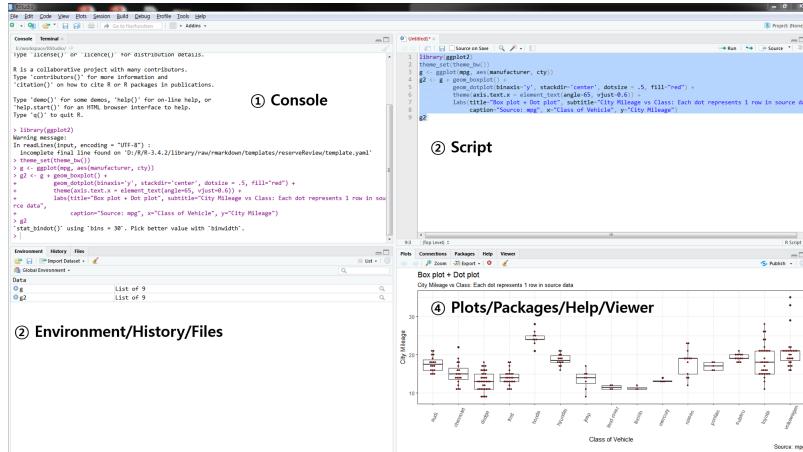
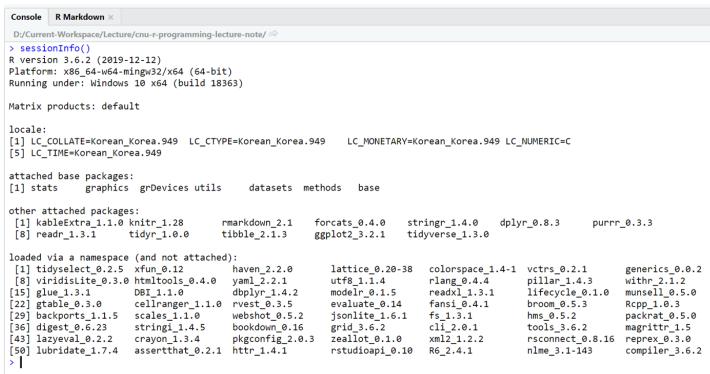


FIGURE 1.4: RStudio 화면구성: 우하단 그림은 <http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html>에서 발췌

¹⁵ 각 창의 위치는 세팅 구성에 따라 달라질 수 있음. 창 구성 방법은 RStudio 환경 옵션 설정에서 설명함.

1. 콘솔(console)

- R 명령어 실행 공간 (RGui, 정확하게는 R 설치 디렉토리에서 “~/R/R.x.x/bin/x64/Rterm.exe” 가 구동되고 있는 공간)
- R script 또는 콘솔 창에서 작성한 명령어(프로그램) 실행 및 그 결과 출력
- 경고, 에러/로그 등의 메세지 확인



```

Console | R Markdown | D:/Current Workspace/Lecture/cnu-r-programming-lecture-note/ ⓘ
> sessionInfo()
R version 3.6.2 (2019-12-12)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 10364 (build 18363)

Matrix products: default

locale:
[1] LC_COLLATE=korean_Korea.949  LC_CTYPE=korean_Korea.949   LC_MONETARY=korean_Korea.949 LC_NUMERIC=korean_Korea.949
[5] LC_TIME=korean_Korea.949

attached base packages:
[1] stats      graphics    grDevices utils      datasets   methods     base

other attached packages:
[1] kableExtra_1.1.0 knitr_1.28    rmarkdown_2.1   forcats_0.4.0  stringr_1.4.0  dplyr_0.8.3   purrr_0.3.3
[8] readr_1.3.1    tidyverse_1.3.0 tibble_2.1.3   ggplot2_3.2.1 tidyverse_1.3.0

loaded via a namespace (and not attached):
[1] tidyselect_0.2.5 xfun_0.12      haven_2.2.0    lattice_0.20-38 colorspace_1.4-1  vctrs_0.2.1   generics_0.0.2
[8] viridislite_0.3.0 htmltools_0.4.0 yaml_2.2.1     utf8_1.1.4     rlang_0.4.4    pillar_1.4.3   withr_1.1.2
[15] glue_1.3.2     DBI_1.1.0     dbplyr_1.4.2   modelr_0.1.5  readxl_1.3.1  lifecycle_0.1.0 munspell_0.5.0
[22] grid_3.6.2     curl_4.3.1    scales_1.1.0   assertthat_0.1.4  rlang_0.4.1  broom_0.5.3   magrit_1.5
[29]槐花包_1.1.5  scales_1.1.0   webshot_0.5.2  jsonlite_1.6.1  fansi_0.3.1  here_0.5.2   packrat_0.5.0
[36] digest_0.6.23  stringr_1.4.5  bookdown_0.16  grid_3.6.2    cli_1.2.0.1  tools_3.6.2   magritr_1.5
[43] lazyeval_0.2.2 crayon_1.3.4    pkgconfig_2.0.3  zealot_0.1.0  xrl_1.2.2   rconnect_0.8.16 reprex_0.3.0
[50] lubridate_1.7.4 assertthat_0.2.1  httr_1.4.1    R6_2.4.1     nime_3.1-143 compiler_3.6.2
> |

```

FIGURE 1.5: RStudio 콘솔창에서 명령어 실행 후 출력결과 화면

2. 스크립트(script) (Figure 1.6)

- R 명령어 입력 공간으로 일괄처리 (batch processing) 가능
- 새로운 스크립트 창 열기
 - 아래 그림과 같이 pull-down 메뉴 좌측 상단 아이콘 클릭 후 [R script] 선택
 - [File] → [New File] → [R Script] 선택
 - 단축 키: [Ctrl] + [Shift] + [N]
- 일괄 명령어 처리를 위한 RStudio 제공 단축 키
 - [Ctrl] + [Enter]: 선택한 블럭 내 명령어 실행
 - [Alt] + [Enter]: 선택 없이 커서가 위치한 라인의 명령어 실행
- R 스크립트 이외 R Markdown, R Notebook, Shiny web application 등 새 문서의 목적에 따라 다양한 종류의 소스 파일 생성 가능

- 저장된 R 스크립트 파일은 파일명.R로 저장됨
- 파일 실행 방법
 - 실행하고자 하는 파일을 읽은 후 ([File] → [Open File] + 파일명 선택 또는 파일명.R 더블 클릭) 입력된 모든 라인을 선택한 뒤 [Ctrl] + [Enter]
 - 파일 읽은 후 [Ctrl] + [Shift] + [S] (현재 열려있는 *.R 파일에 대해) 또는 [Ctrl] + [Shift] + [Enter]

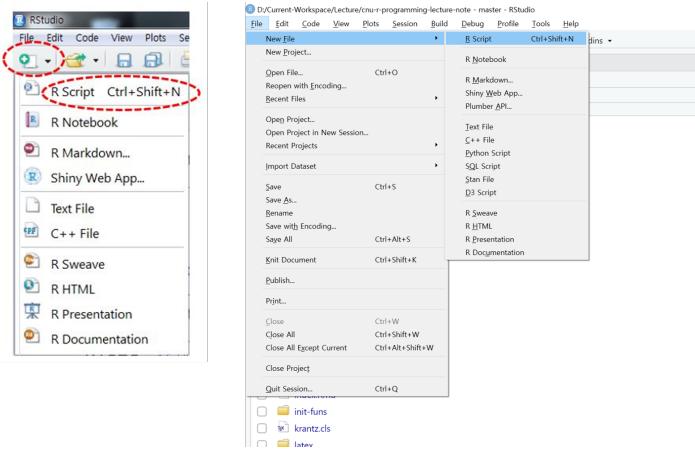


FIGURE 1.6: RStudio 스크립트 새로 열기

i RStudio는 코딩 및 소스 작성의 효율성을 위해 여러 가지 단축 키를 제공하고 있음. 단축키는 아래 그림과 같이 pull down 메뉴 [Tools] 또는 [Help]에서 [Keyboard shortcut help] 또는 [Alt] + [Shift] + [K] 단축키를 통해 확인할 수 있음. 또는 Rstudio cheatsheet에서 단축키에 대한 정보를 제공하는데 pull down 메뉴 [Help] → [Cheatsheets] → [RStudio IDE Cheat Sheet]을 선택하면 각 아이콘 및 메뉴 기능에 대한 개괄적 설명 확인 가능함.

3. 환경/명령기록(Environment/History) (Figure 1.7)

- Environment:** 현재 R 작업환경에 저장되어 있는 객체의 특성 및 값 등을 요약 제시

- 좌측 아래 화살표 버튼 클릭: 해당 객체의 상세 정보 확인
- 우측 사각형 버튼 또는 객체(데이터셋명) 클릭: 객체가 데이터셋(데이터프레임)인 경우 스프레드 시트 형태로 데이터셋 확인

The figure consists of four screenshots of the RStudio Environment tab, each showing a different way to inspect objects:

- Top Left:** Shows the standard Environment view with objects: cars, mpg, and tab.
- Top Right:** Shows the same environment with the 'cars' object selected (highlighted by a red box), displaying its details: 50 obs. of 2 variables, speed: num 4 4 7 7 8 9 10 10 10 11 ..., dist: num 2 10 4 22 16 10 18 26 34 17 ...
- Bottom Left:** Shows the same environment with the 'cars' object selected (highlighted by a red box), displaying its details: 50 obs. of 2 variables, speed: num 4 4 7 7 8 9 10 10 10 11 ..., dist: num 2 10 4 22 16 10 18 26 34 17 ...
- Bottom Right:** Shows the 'cars' dataset as a spread sheet with columns 'speed' and 'dist'. The data rows are numbered 1 to 10, with values corresponding to the first 10 observations of the 'cars' dataset.

FIGURE 1.7: RStudio Environment 창 객체 상세 정보 및 스프레드 시트 출력 결과

- History: R 콘솔에서 실행된 명령어(스크립트)들의 이력 확인

The figure shows the RStudio History tab with the following R script history:

```

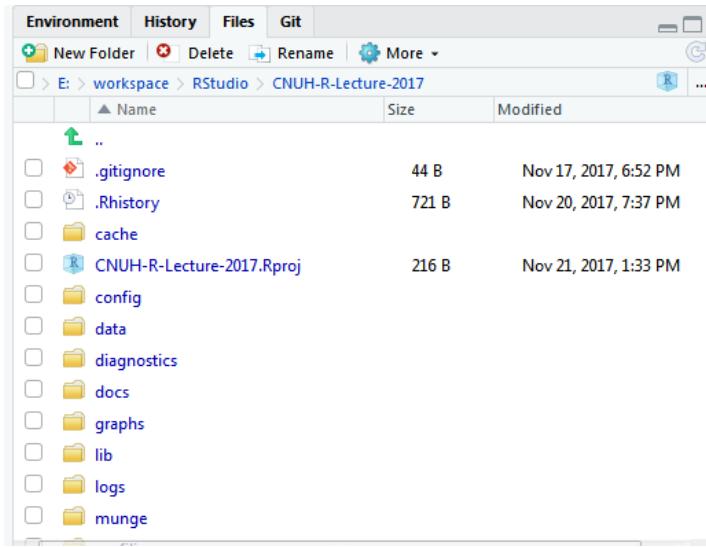
Environment History Connections Git
To Console To Source ⚡ 🔍

ReadExcel <- function(filename) {
  require(XLConnect)
  require(plyr)
  WB <- loadWorkbook(filename)
  SheetName <- getSheets(WB)
  DF1 <- llply(SheetName, function(name) readWorksheet(WB, sheet=name))
  names(DF1) <- SheetName
  return(DF1)
}
ReadExcel <- function(filename) {
  require(XLConnect)
  require(plyr)
  WB <- loadWorkbook(filename)
  SheetName <- getSheets(WB)
  DF1 <- llply(SheetName, function(name) readWorksheet(WB, sheet=name))
  names(DF1) <- SheetName
  return(DF1)
}

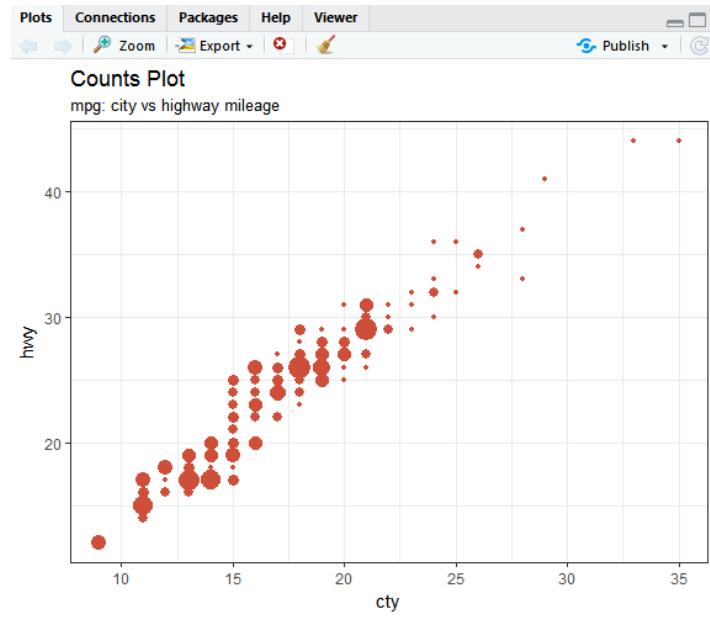
```

4. File/Plots/Packages/Help/Viewer

- File: Windows 파일 탐색기와 유사한 기능 제공
 - 파일 및 폴더 생성, 삭제/파일 및 폴더명 수정, 그리고 작업경로 설정



- Plots: 생성한 그래프 출력
 - 작업 중 생성한 그래프 이력이 Plots 창에 저장: ← 이전, → 최근
 - Zoom: 클릭 시 해당 그래프의 팝업창이 생성되고 팝업창의 크기 조정을 통해 그래프의 축소/확대 가능
 - Export: 선택한 그래프를 이미지 파일 (.png, .jpeg, .pdf 등)로 저장할 수 있고, 클립보드로 복사 가능

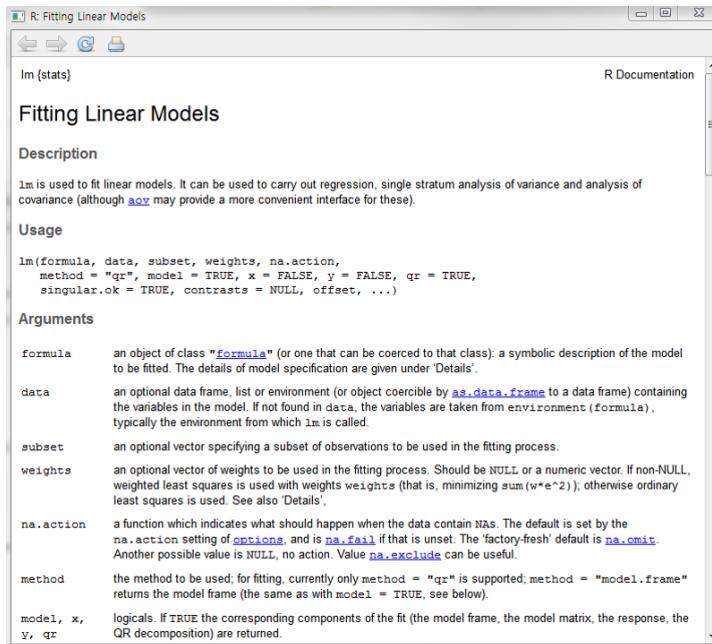


- **Packages:** 현재 컴퓨터에 설치된 R 패키지 목록 출력
 - 신규 설치 및 업데이트 가능

| Name | Description | Version |
|-----------------------|--|---------|
| System Library | | |
| A3 | Accurate, Adaptable, and Accessible Error Metrics for Predictive Models | 1.0.0 |
| abbyyR | Access to Abbyy Optical Character Recognition (OCR) API | 0.5.1 |
| abc | Tools for Approximate Bayesian Computation (ABC) | 2.1 |
| abc.data | Data Only: Tools for Approximate Bayesian Computation (ABC) | 1.0 |
| ABC.RAP | Array Based CpG Region Analysis Pipeline | 0.9.0 |
| ABCAnalysis | Computed ABC Analysis | 1.2.1 |
| abcdefBA | ABCDE_FBA: A-Biologist-Can-Do-Everything of Flux Balance Analysis with this package. | 0.4 |
| ABCOptim | Implementation of Artificial Bee Colony (ABC) Optimization | 0.15.0 |
| ABCp2 | Approximate Bayesian Computational Model for Estimating P2 | 1.2 |
| abcfr | Approximate Bayesian Computation via Random Forests | 1.7 |
| abctools | Tools for ABC Analyses | 1.1.1 |
| abd | The Analysis of Biological Data | 0.2.0 |

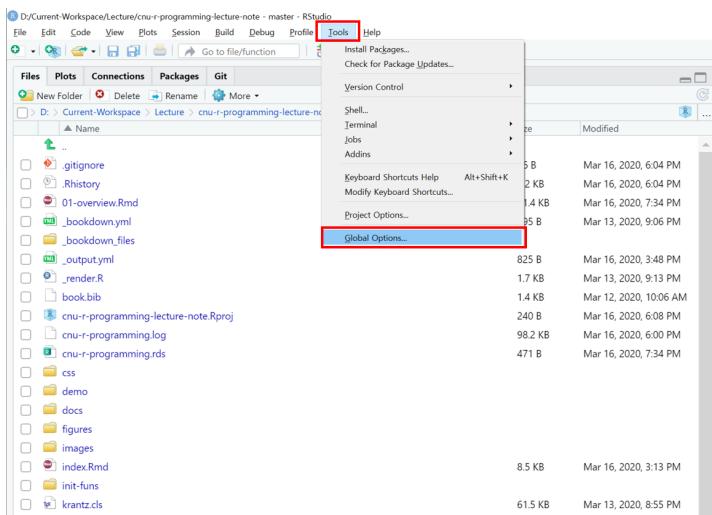
- **Help:** `help(topic)` 입력 시 도움말 창이 출력되는 공간

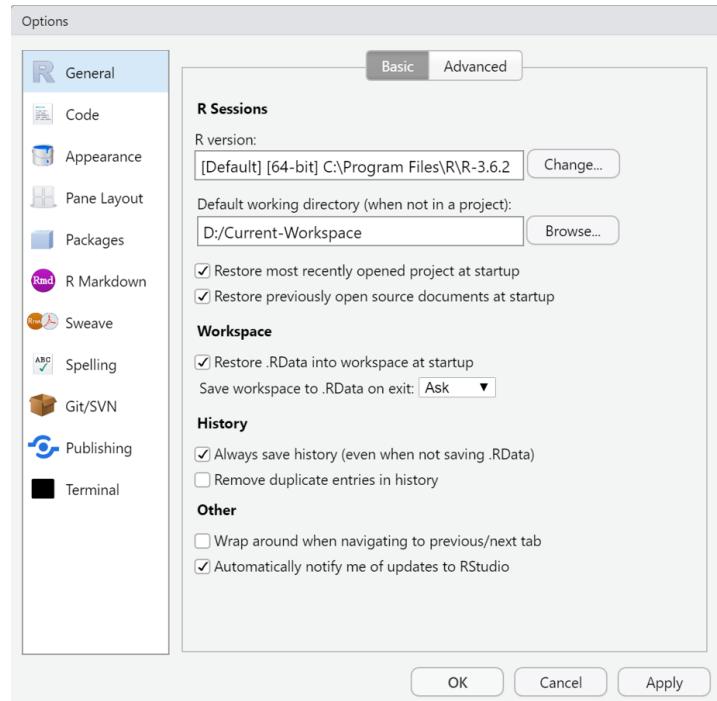
```
help(lm)
```



1.4.3 RStudio 환경 설정

Pull-down 메뉴에서 [Tools] → [Global Options...]를 선택



General: RStudio 운용 관련 전반적 설정 세팅**FIGURE 1.8:** R General option 팝업 창

- **R version:** 만약 컴퓨터에 두 개 이상 다른 R 버전이 설치되어 있는 경우 [Change] 클릭 후 설정 변경 가능
- **Default Working directory:** 작업 디렉토리 지정 ([Browse] 클릭 후 임의 폴더 설정 가능)
- **Restore most recently opened project at startup:** RStudio 실행 시 가장 최근에 작업한 프로젝트로 이동
- **Restore previously open source documents at startup:** RStudio 실행 시 현재 프로젝트에서 가장 최근에 작업한 소스코드 문서를 함께 열어줌.

- **Restore .RData into workspace at startup:** 작업 디렉토리에 존재하는 .RData 파일을 RStudio 실행 시 불러옴
- **Save workspace to .RData on exit:** R workspace 자동 저장 (.RData) 여부
- **Always save history (even when not saving .RData)** : R 실행 명령 history 저장 여부(Always/Never/Ask)
- **Remove duplicate entries in history:** history 저장 시 중복 명령 제거 여부

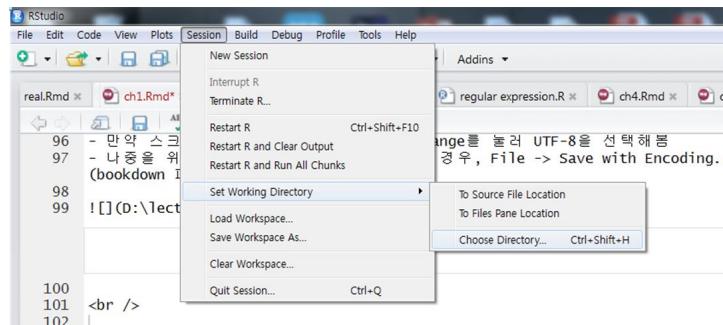
작업폴더(Working Directory)는 현재 R session에서 사용하는 기본 폴더로서 R 소스파일 및 데이터의 저장 및 로딩시 기본이 되는 폴더임.

- 소스파일이나 데이터를 불러들일 때 작업 폴더에 있는 파일은 경로명을 지정하지 않고 파일명만 사용해도 됨
- 작업폴더가 아닌 곳에 있는 파일을 불러들일 때는 경로명까지 써 주어야함.
- R 데이터를 저장할때도 파일명만 쓰면 기본적으로 작업폴더에 저장되며, 다른 폴더에 저장하기 위해서는 경로명까지 써 주어야 함.

처음 컴퓨터에 RStudio를 설치하면 Working directory는 Windows 사용자 폴더(예: user)의 Document 폴더가 기본값으로 설정되어 있음. 기본 작업폴더를 변경하려면 Figure 1.8에서 설정 가능.

현재 R session의 작업 디렉토리 설정 방법

- [Session] -> [Set Working Directoy] -> [Choose Directory]에서 설정



R 콘솔에서 다음과 같은 명령어로 작업폴더를 확인 및 변경 가능

```
getwd() # 작업폴더 확인
```

```
[1] "D:/Current-Workspace/Lecture/cnu-r-programming-lecture-note"
```

```
setwd("../") # 차상위 폴더로 이동
getwd()
```

```
[1] "D:/Current-Workspace/Lecture"
```

```
setwd("../..") # 차차상위 폴더로 이동
getwd()
```

```
[1] "D:/"
```

```
setwd("D:/Current-Workspace/Lecture/misc/") # 절대 폴더 명 입력
setwd("../")
# dir() # 폴더 내 파일 명 출력
getwd()
```

```
[1] "D:/Current-Workspace/Lecture"
```

```
setwd("misc") # Current-Workspace 하위폴더인 misc 으로 이동
getwd()
```

```
[1] "D:/Current-Workspace/Lecture/misc"
```

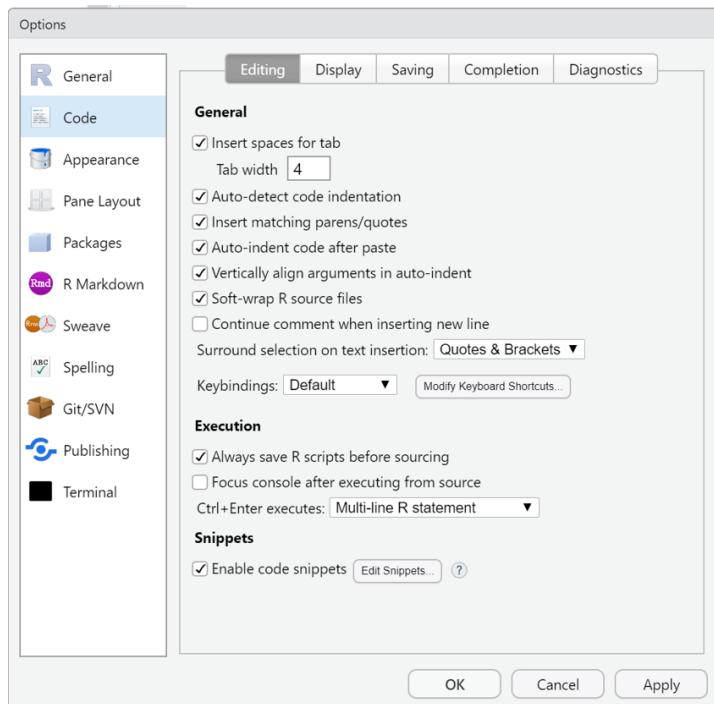
```
setwd("D:/Current-Workspace/Lecture/cnu-r-programming-lecture-note/")
getwd()
```

```
[1] "D:/Current-Workspace/Lecture/cnu-r-programming-lecture-note"
```



R에서 디렉토리 또는 폴더 구분자는 / 입. Windows에서 사용하는 구분자는 \인데, R에서 \는 특수문자로 간주하기 때문에 Windows의 폴더명을 그대로 사용 시 에러 메세지를 출력함. 이를 해결하기 위해 Windows 경로명을 그대로 복사한 경우 경로 구분자 \ 대신 \\로 변경
실습: C:\\r-project를 컴퓨터에 생성 후 해당 폴더를 default 작업폴더로 설정

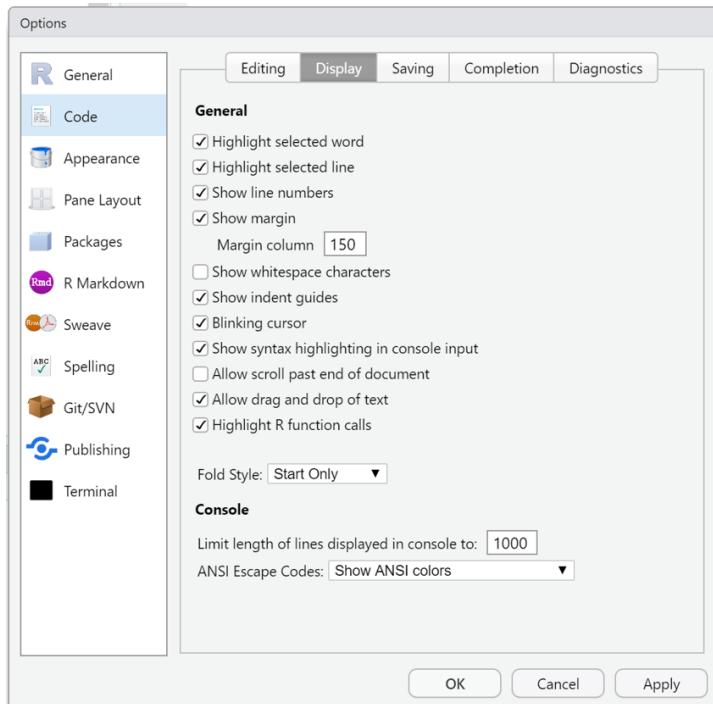
Code: Editing: 들여쓰기, 자동 줄바꿈 등 코드 편집에 대한 전반적 설정



- **Insert spaces for tab:** [Tab] 키를 눌렀을 때 공백(space) 개수 결정
(본 강의노트: Tab width = 4)

- **Auto-detect code indentation:** 코드 들여쓰기 자동 감지
- **Insert matching parens/quotes:** 따옴표, 괄호 입력 시 커서를 따옴 표/괄호 사이로 자동 이동
- **Auto-indent code after paste:** 코드 복사 시 들여쓰기 일괄 적용
- **Vertically align arguments in auto-indent:** 함수 작성 시 들여쓰기 레벨 유지 여부
- **Soft-wrap R source file:** 스크립트 편집기 너비를 초과하는 경우 R 코드 행을 자동 줄바꿈
- **Continue comment when inserting new line:** 주석 표시를 다음 행에도 자동 적용 여부
- **Surround selection on text insertino:** 스크립트 상 text 선택 후 자동 따옴표 및 괄호 적용 여부
- **Focus console after executing from source:** 스크립트 실행 후 커서 위치를 콘솔로 이동 여부

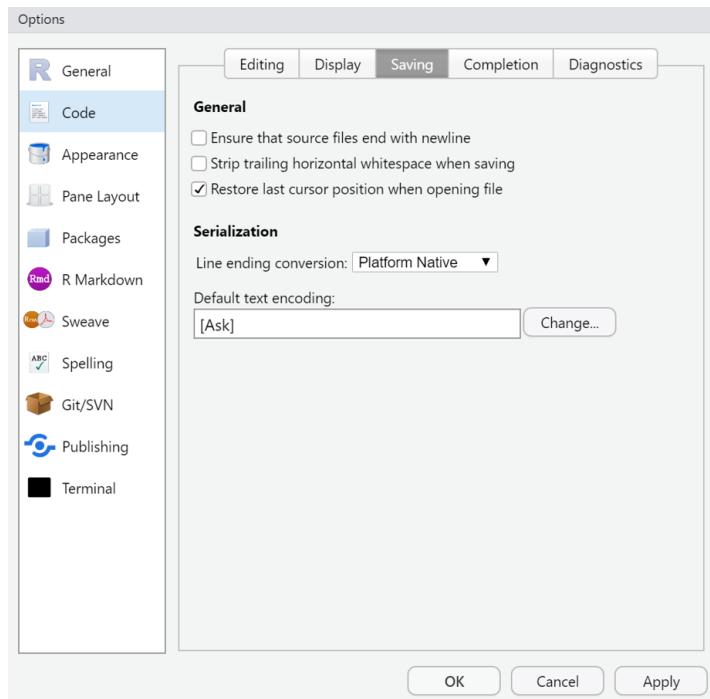
Code: Display: 스크립트(소스) 에디터 표시 화면 설정



- **Highlight selected word:** 스크립트 내 text 선택 시 동일한 text에 대해 배경강조 효과 여부
- **Highlight selected line:** 선택된 행에 대해 배경 강조효과 여부
- **Show line numbers:** 행 번호 보여주기 여부
- **Show margin:** 소스 에디터 오른 쪽에 지정한 margin column 보여주기 여부
- **Show whitespace characters:** 에디터에 공백 표시 여부
- **Show indent guides:** 현재 들여쓰기 열 표시 여부
- **Blinking cursor:** 커서 깜박임 여부
- **Show syntax highlighting in console output:** 콘솔 입력 라인에 R 구문 강조 표시 적용 여부
- **Allow scroll past end of document:** 문서 마지막 행 이후 스크롤 허용 여부

- **Allow drag and drop of text:** 선택한 복수의 행으로 구성된 text에 대해 마우스 drag 허용
- **Highlight R function calls:** R 내장 및 패키지 제공함수에 대해 강조 여부

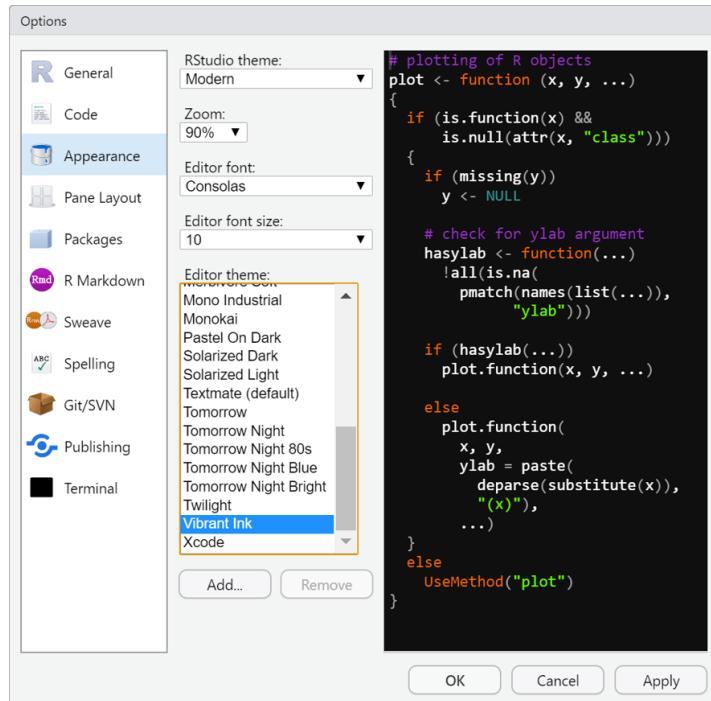
Code: Saving: 스크립트(소스) 에디터 저장 설정



- **Ensure that source file end with newline**
- **String trailing horizontal whitespace when saving**
- **Restore last cursor position when opening file**
- **Default text encoding:** 소스 에디터의 기본 설정 인코딩 설정 변경
 - RStudio의 Windows 버전 기본 text encoding은 CP949 입
 - Linux나 Mac OS의 경우 한글은 UTF-8로 인코딩이 설정되어 있음.

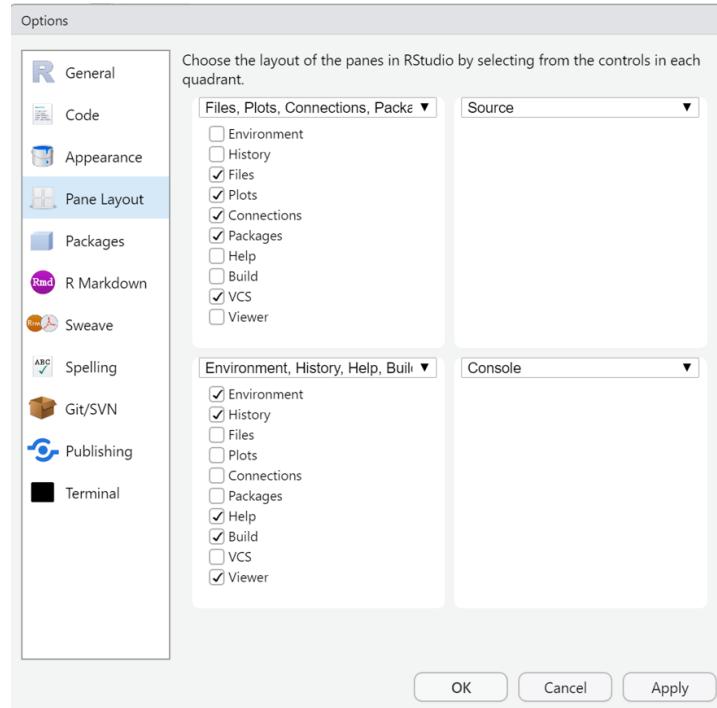
- R 언어는 Linux 환경에서 개발되었기 때문에 UTF-8 인코딩과 호환성이 더 좋음
- 스크립트 파일의 한글이 깨질 때는 [File] -> [Reopen with Encoding...]에서 encoding 방식 변경

Appearance: RStudio 전체 폰트, 폰트 크기, theme 설정



- 본인의 취향에 맞게 폰트 및 테마(theme) 설정
- 취향 → 가독성이 제일 좋고 편안한 theme

Pane Layout: RStudio 구성 패널들의 위치 및 항목 등을 수정/추가/삭제(4개 패널은 항상 유지)



실습: 개인 취향에 맞게 RStudio 에디터 및 theme을 변경해 보자!!

1.4.4 RStudio 프로젝트

1. 프로젝트

- 물리적 측면: 최종 산출물(문서)를 생성하기 위한 데이터, 사진, 그림 등을 모아 놓은 폴더
- 논리적 측면: R session 및 작업의 버전 관리

2. 프로젝트의 필요성

- 자료의 정합성 보장
- 다양한 확장자를 갖는 파일들이 한 폴더 내에 뒤섞일 때 고란해 질 수 있음

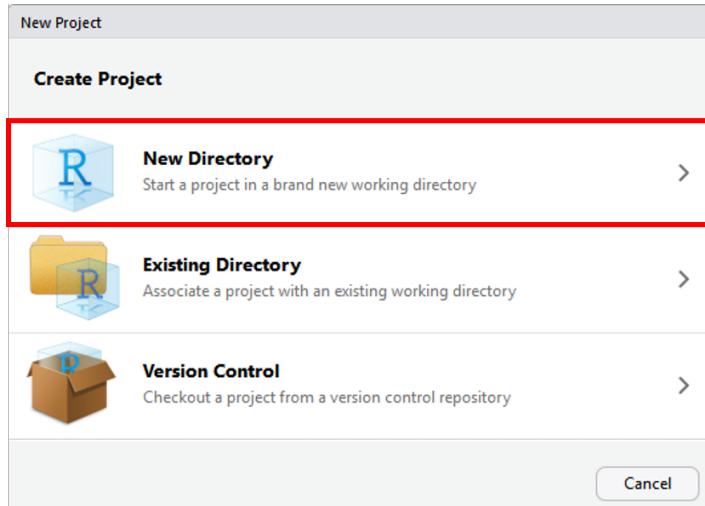
- 실제 분석 및 그래프 생성에 사용한 정확한 프로그램 또는 코드 연결이 어려움

3. 좋은 프로젝트 구성을 위한 방법

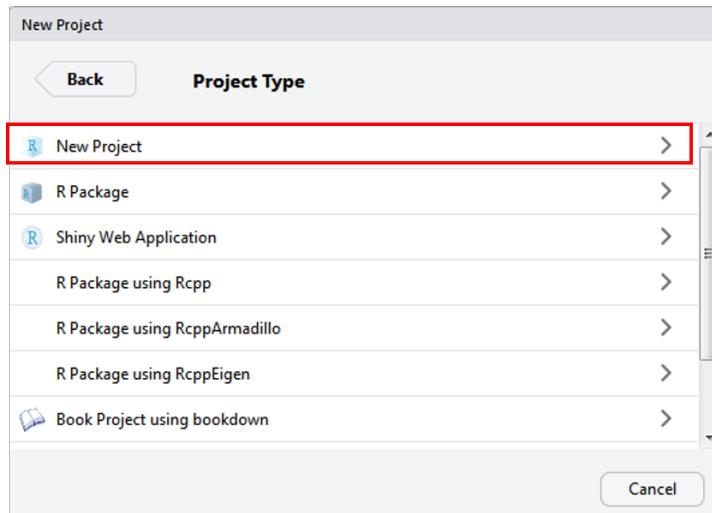
- 원자료(raw data)의 보호: 가급적 자료를 읽기 전용(read only) 형태로 다루기
- 데이터 정제(data wrangling 또는 data munging)를 위한 스크립트와 정제 자료를 보관하는 읽기 전용 데이터 디렉토리 생성
- 작성한 스크립트로 생성한 모든 산출물(테이블, 그래프 등)을 “일회용 품”처럼 처리 → 스크립트로 재현 가능
- 한 프로젝트 내 각기 다른 분석마다 다른 하위 디렉토리에 출력결과 저장하는 것이 유용

4. RStudio 새로운 프로젝트 생성

- RStudio의 강력하고 유용한 기능
- 새로운 프로젝트 생성: RStudio 메뉴에서 [File] → [New Project] 선택하면 아래와 같은 팝업 메뉴 생성

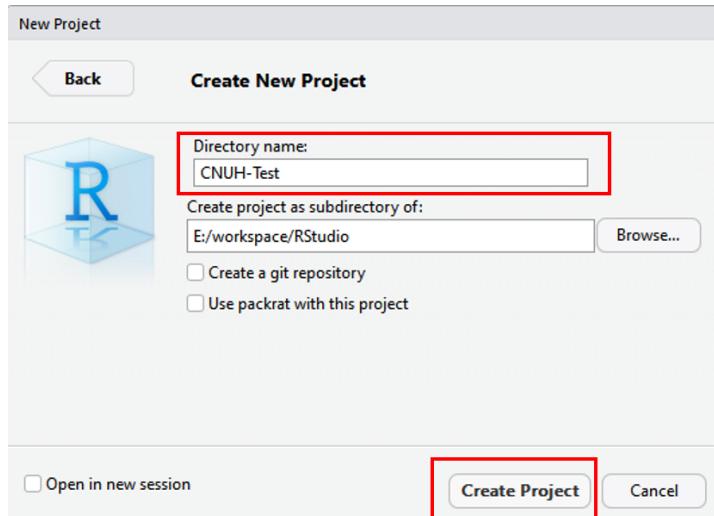


4. 위 그림에서 New Directory를 선택하면 아래와 같은 팝업 창이 나타나면 아래와 같은 프로젝트 유형이 나타남. 여기서는 New Project 선택



5. 다음 팝업창에서 새로운 프로젝트의 폴더명을 지정 후 Create Project 클릭

- 아래 [Create projects as subdirectories of]에서 생성하고자 하는 프로젝트의 상위 디렉토리 설정 → 보통 RStudio의 기본 작업폴더로 설정



6. 현재 R session 종료 후 새로운 프로젝트로 session 화면이 열리면 프로젝트 생성 완료



실습: 프로젝트 생성

- 위에서 설정한 작업폴더 내에 학번-r-programming 프로젝트 생성
- 생성한 프로젝트 폴더 내에 docs, figures, script 폴더 생성

1.5 R 패키지



R 패키지 (package): 특수 목적을 위한 로직으로 구성된 코드들의 집합으로 R에서 구동되는 분석툴을 통칭

- CRAN을 통해 배포: 3자가 이용하기 쉬움 → R 시스템 환경에서 패키지는 가장 중요한 역할
- CRAN available package by name¹⁶ 또는 available package by date¹⁷에서 현재 등재된 패키지 리스트 확인 가능
- R console에서 `available.packages()` 함수를 통해서도 확인 가능

- 현재 CRAN 기준(2020-03-17) 배포된 패키지의 개수는 16045 개임

목적: RStudio 환경에서 패키지를 설치하고 불러오기

1.5.1 R 패키지 경로 확인 및 변경

- 패키지 설치 시 일반적으로 R 환경에서 기본값으로 지정한 라이브러리 폴더에 저장
- 패키지 설치 전 R 패키지 설치 경로(path) 지정
- `.libPaths()` 함수를 통해 현재 설정된 패키지 저장 경로 확인

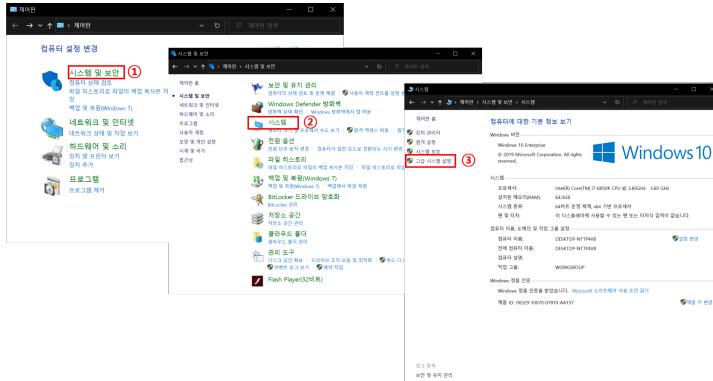
```
.libPaths()
```

```
[1] "C:/Users/user/Documents/R/win-library/3.6"  
[2] "C:/Program Files/R/R-3.6.3/library"
```

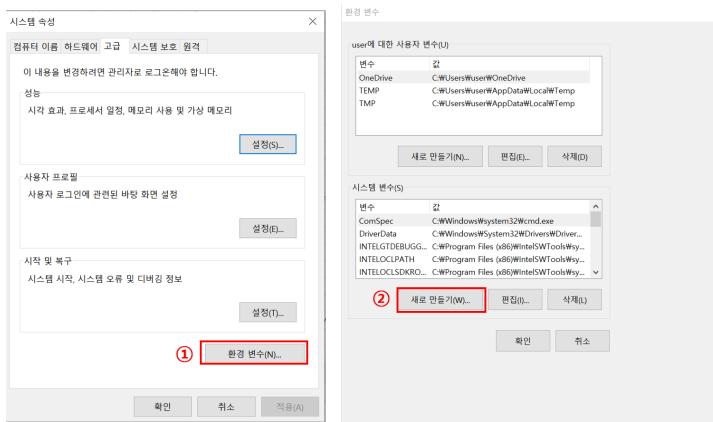
- 일반적으로 첫 번째 경로를 디폴트 라이브러리 폴더로 사용
- 사용자 지정 라이브러리 경로를 설정 하려면 아래와 같은 절차로 진행

실습: c:/r-library 폴더를 패키지 경로로 지정

- 1) C:\에서 [새로 만들기 (W)] -> [폴더 (F)] 선택 후 생성 폴더 이름을 r-library로 변경
- 2) 윈도우즈 [제어판] -> [시스템 및 보안] -> [시스템] -> [고급 시스템 설정] 클릭

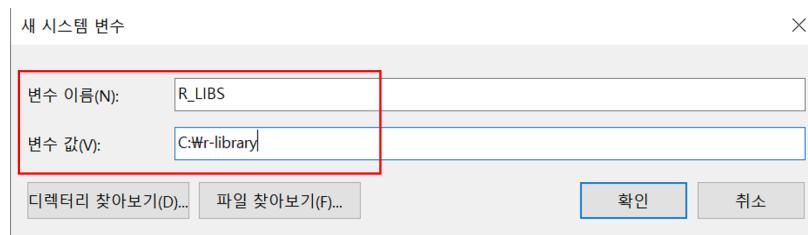


3) [환경변수(N)...] 선택 후 시스템 변수에서 [새로 만들기(W)...] 클릭



4) 아래 그림과 같이 변수 이름(N)에 R_LIBS, 변수 값(V)에 해당 디렉토리

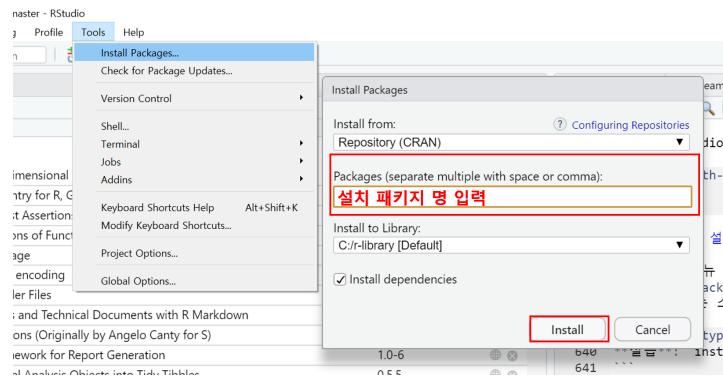
경로 C:\r-library 입력 후 확인 버튼 클릭



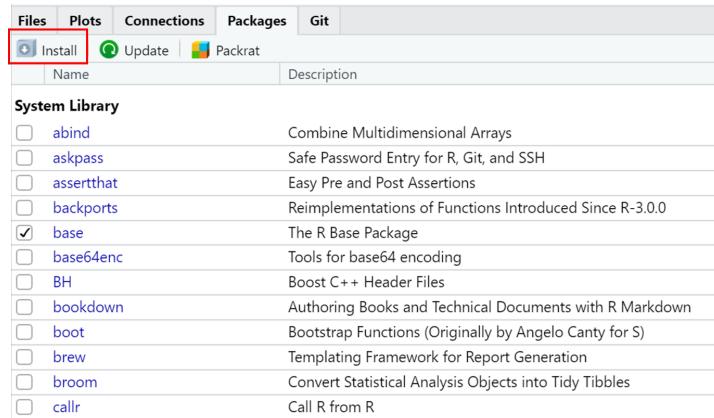
5) 현재 RStudio 종료 후 재실행한 다음 콘솔창에 .libPaths() 입력 후 라이브러리 경로 확인

1.5.2 R 패키지 설치하기

- RStudio 메뉴 [Tools] → [Install packages] 클릭 후 생성된 팝업창에서 설치하고자 하는 패키지 입력 후 설치



- RStudio Packages 창에서 [Install] 버튼 누르면 위와 동일한 팝업창이 나타남(위와 동일)



- R 콘솔 또는 스크립트 창에서 `install.packages(package_name)` 함수를 사용해서 패키지 설치



설습: `install.packages()` 함수를 이용해 `tidyverse` 패키지 설치

```
install.packages("tidyverse")
```

위 명령어를 실행하면 `tidyverse` 패키지 뿐 아니라 연관된 패키지들이 동시에 설치됨

1.5.3 R 패키지 불러오기

1. `library()` vs. `require()`

- `library()`: 불러오고자 하는 패키지가 시스템에 존재하지 않는 경우
에러 메세지 출력(에러 이후 명령어들이 실행되지 않음)
- `require()`: 패키지가 시스템에 존재하지 않는 경우 경고 메세지 출력
(경고 이후 명령어 정상적으로 실행)

2. 다중 패키지 동시에 불러오기

- RStudio Packages 창에서 설치하고자 하는 패키지 선택 버튼 클릭
하면 R workspace로 해당 패키지 로드 가능
- 스크립트 이용

 실습: `tidyverse` 패키지 불러오기

```
require(tidyverse)
```

필요한 패키지를 로딩중입니다: `tidyverse`

```
-- Attaching packages ---

v ggplot2     3.3.0     v purrr      0.3.3
v tibble       2.1.3     v dplyr      0.8.5
v tidyverse    1.0.2     v stringr    1.4.0
v readr        1.3.1     vforcats   0.5.0
```

```
-- Conflicts -----
x dplyr::filter()      masks stats::filter()
x dplyr::group_rows()  masks kableExtra::group_rows()
x dplyr::lag()         masks stats::lag()
```



실무에서 R의 활용능력은 패키지 활용 여부에 달려 있음. 즉, 목적에 맞는 업무를 수행하기 위해 가장 적합한 패키지를 찾고 활용하느냐에 따라 R 활용능력의 차이를 보임. 앞서 언급한 바와 같이 CRAN에 등록된 패키지는 16000 개가 넘지만, 이 중 많이 활용되고 있는 패키지의 수는 약 200 ~ 300 개 내외이고, 실제 데이터 분석 시 10 ~ 20개 정도의 패키지가 사용됨. 앞 예제에서 설치하고 불러온 **tidyverse** 패키지는 Hadley Wickham ([Wickham et al., 2019](#))이 개발한 데이터 전처리 및 시각화 패키지 번들이고, 현재 R 프로그램 환경에 지대한 영향을 미침. 본 강의 “데이터프레임 가공 및 시각화”에서 해당 패키지 활용 방법을 배울 예정

1.6 R 기초 문법



본 절에서 다루는 R 문법은 R 입문 시 객체(object)의 명명 규칙과 R 콘솔 창에서 가장 빈번하게 사용되는 기초적인 명령어만 다룰 예정임. 심화 내용은 2-3주 차에 다룰 예정임.

- R은 객체지향언어(object-oriented language)
 - 객체(object): 숫자, 데이터셋, 단어, 테이블, 분석결과 등 모든 것을 칭함
 - “객체지향”의 의미는 R의 모든 명령어는 객체를 대상으로 이루어진다는 것을 의미



알아두면 유용한(콘솔창에서 매우 많이 사용되는) 명령어 및 단축기

- **ls()**: 현재 R 작업공간에 저장된 모든 객체 리스트 출력
- **rm(object_name)**: **object_name**에 해당하는 객체 삭제

- `rm(list = ls())`: R 작업공간에 저장된 모든 객체들을 일괄 삭제
- 단축키 [Ctrl] + [L]: R 콘솔 창 일괄 청소
- 단축키 [Ctrl] + [Shift] + [F10]: R session 초기화

예시

```
x <- 7
y <- 1:30 # 1에서 30까지 정수 입력
ls() # 현재 작업공간 내 객체명 출력
```

```
[1] "a"           "b"           "cars"
[4] "def.chunk.hook" "fig_cap"      "hook_output"
[7] "tab"          "x"           "y"
[10] "도움말 보기 명령어" "사용법"      "설명"
```

```
rm(x) # 객체 x 삭제
ls()
```

```
[1] "a"           "b"           "cars"
[4] "def.chunk.hook" "fig_cap"      "hook_output"
[7] "tab"          "y"           "도움말 보기 명령어"
[10] "사용법"       "설명"
```

```
rm(a,b) # 객체 a, b 동시 삭제
ls()
```

```
[1] "cars"         "def.chunk.hook" "fig_cap"
[4] "hook_output"  "tab"          "y"
[7] "도움말 보기 명령어" "사용법"      "설명"
```

```
# rm(list = ls()) # 모든 객체 삭제
```

R 객체 입력 방법 및 변수 설정 규칙

객체를 할당하는 두 가지 방법 :=, <-

- 두 할당 지시자의 차이점
 - :=: 명령의 최상 수준에서만 사용 가능
 - <-: 어디서든 사용 가능
 - 함수 호출과 동시에 변수에 값을 할당할 목적으로는 <-만 사용 가능

```
# mean(): 입력 벡터의 평균 계산  
mean(y <- 1:5)
```

```
[1] 3
```

```
y
```

```
[1] 1 2 3 4 5
```

```
mean(x = 1:5)
```

```
[1] 3
```

```
x
```

Error in eval(expr, envir, enclos): 객체 'x'를 찾을 수 없습니다

객체 또는 변수의 명명 규칙

- 알파벳, 한글, 숫자, _, .의 조합으로 구성 가능 (-은 사용 불가)
- 변수명의 알파벳, 한글, .로 시작 가능
- .로 시작한 경우 뒤에 숫자 올 수 없음(숫자로 인지)
- 대소문자 구분

```
# 1:10은 1부터 10까지 정수 생성
# 'c()'는 벡터 생성 함수
x <- c(1:10)

# 1:10으로 구성된 행렬 생성
X <- matrix(c(1:10), nrow = 2, ncol = 5, byrow = T)
x
```

```
[1] 1 2 3 4 5 6 7 8 9 10
```

```
x
```

```
[,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
```

```
# 논리형 객체
.x <- TRUE

.x
```

```
[1] TRUE
```

```
# 알파벳 + 숫자
# seq(): 수열을 만드는 함수
# 1에서부터 (from) 10 까지 (to) 공차가 2(by)인 수열
a1 <- seq(from = 1, to = 10, by = 2)

# 한글 변수명
가수 <- c("Damian Rice", "Beatles", "최백호", "Queen", "Carlos Gardel", "BTS", "조용필")
가수
```

```
[1] "Damian Rice" "Beatles" "최백호"
"Queen"
[5] "Carlos Gardel" "BTS" "조용필"
```

3. 잘못된 객체 또는 변수 명명 예시

```
3x <- 7
```

```
Error: <text>:1:2: 예상하지 못한 기호(symbol)입니다.  
1: 3x  
^
```

```
_x <- c("M", "M", "F")
```

```
Error: <text>:1:1: 예상하지 못한 입력입니다.  
1: _  
^
```

```
.3 <- 10
```

```
Error in 0.3 <- 10: 대입에 유효하지 않은 (do_set) 좌변입니다
```

1.7 R Markdown (맛보기)



R **기초 문법** 결과 마찬가지로 R Markdown을 이용해 최소한의 문서(html 문서)를 작성하고 생성하는 방법에 대해 기술함. R Markdown에 대한 보다 상세한 내용은 본 수업의 마지막 주차에 다룰 예정임.

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 연구¹⁸ 가능

¹⁸과학적 연구의 결과물을 오픈소스로 내놓고 누구라도 검증 가능

- 신뢰성 있는 문서 작성
 - Copy & paste를 하지 않고 효율적 작업 가능
3. R Markdown 문서를 통해 최종 결과물 (pdf, html, docx)이 도출되는 process
- 현재 공식적인 프로세스는 knitr + rmarkdown + pandoc + RStudio + github

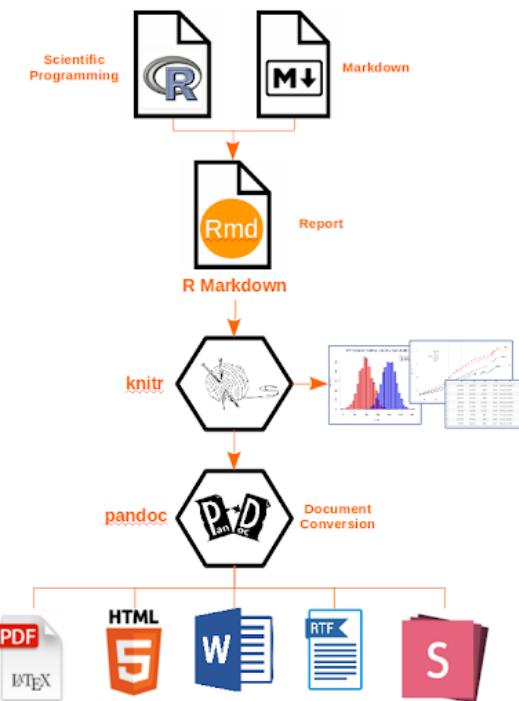


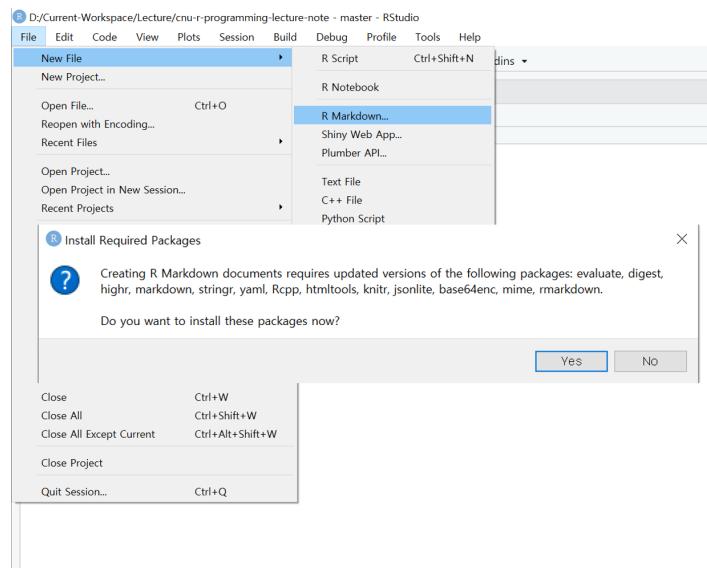
FIGURE 1.9: R Markdown의 최종 결과물 산출과정 (<http://appliedr.com/project-reporting-template/>)

R Markdown 문서 시작하기

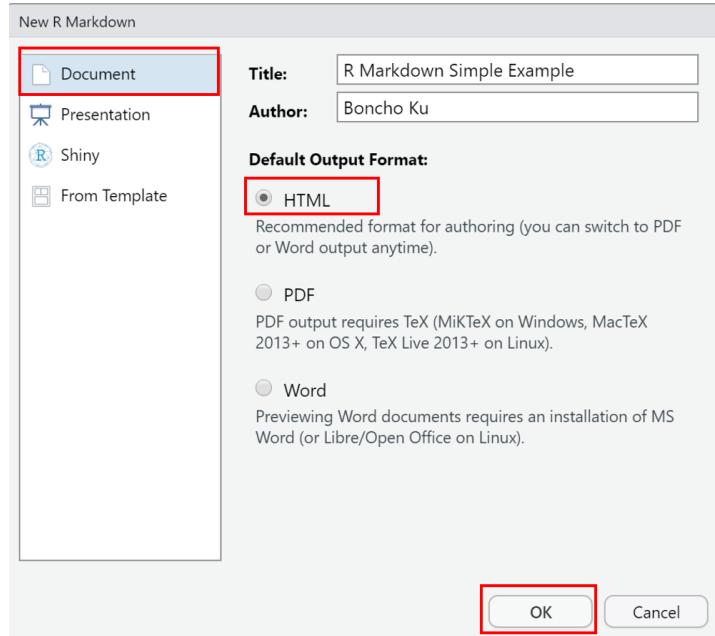
- R Markdown 문서 생성 : [File] -> [New File] -> [R Markdown..]을 선택



RStudio를 처음 설치하고 위와 같이 진행할 경우 아래와 같은 팝업 창이 나타남. 패키지 설치 여부를 묻는 팝업 창에 [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 저장¹⁹

```

1 <!-- 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
2 # Markdown see <a href="http://rmarkdown.rstudio.com">http://rmarkdown.rstudio.com.
3
4 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
5 within the document. You can embed an R code chunk like this:
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
16 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
17 within the document. You can embed an R code chunk like this:
18
19 +```{r cars}
20 summary(cars)
21
22 +## Including Plots
23
24 You can also embed plots, for example:
25
26 +```{r pressure, echo=FALSE}
27 plot(pressure)
28
29 Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.
30
31

```

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

¹⁹RStudio 프로젝트에서 생성한 폴더 내에 파일 저장

The screenshot shows the RStudio interface with several tabs at the top: Untitled1, README.md, preamble-krantz.tex, and _bookdown.yml. A context menu is open over the following R Markdown code:

```

1 -> Knit to HTML
2 au Knit to PDF
3 da Knit to Word
4 ou Knit with Parameters...
5 -> Knit Directory
6 << E>
7 kr Clear Knitr Cache...
8 << E>
9 << E>
10 << E>
11 << E>
12 ## R Markdown
13
14 This is an R Markdown document. Markdown is a simple form
      of plain text. For more details see <http://rmarkdown.rstudio.com>.

```

The 'Knit' button in the menu is highlighted. The menu items include 'Knit to HTML', 'Knit to PDF', 'Knit to Word', 'Knit with Parameters...', 'Knit Directory', 'Clear Knitr Cache...', and a separator line.

- 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 1.10: test.html 문서 화면(저장 풀더 내 ‘test.html’을 크롬 브라우저로 실행)

1.7.0.1 R Markdown 문서 구성

R Markdown 문서는 아래 그림과 같아 YAML, Markdown 텍스트, Code Chunk 세 부분으로 구성됨.

The screenshot shows the RStudio interface with several tabs open. The main editor tab contains R Markdown code. A red box highlights the YAML section at the top, which includes metadata like title, author, date, and output type. Another red box highlights the R Markdown text section, which contains a brief introduction to R Markdown. A third red box highlights a code chunk where the 'echo=FALSE' parameter is used to prevent printing of the R code.

```

1: title: "R Markdown Simple Example"
2: author: "Boncho Ku"
3: date: "2020-03-17"
4: output: html_document
5:
6: ```{r setup, include=FALSE}
7: knitr::opts_chunk$set(echo = TRUE)
8: ...
9:
10: ## R Markdown
11:
12: 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
13: Markdown see <http://rmarkdown.rstudio.com>.
14:
15: 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
16: within the document. You can embed an R code chunk like this:
17:
18: ```{r cars}
19: summary(cars)
20: ...
21:
22: ## Including Plots
23:
24: You can also embed plots, for example:
25:
26: ```{r pressure, echo=FALSE}
27: plot(pressure)
28:
29: Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.
30:
31:

```

1. YAML (YAML Ain't Markup Language)

- R Markdown 문서의 metadata로 문서의 맨 처음에 항상 포함되어야 함.
- R Markdown 문서의 최종 출력 형태, 제목, 저자, 날짜 등의 정보 등을 포함
- YAML 언어에 대한 사용 예시는 Xie (2016) 의 Appendix B.2²⁰ 참고
- 최소 형태의 YAML 예시

```

---
title: "Hello R Markdown"
author: "Zorba"
date: "2020-03-17"
output: html_document
---

```

2. Markdown 텍스트

- Markdown 문법은 15주 차 강의에서 배울 예정임
- R Markdown 레퍼런스 가이드²¹ 참조
- 그림 삽입: ![] (path/filename)

²⁰<https://bookdown.org/yihui/bookdown/r-markdown.html>

²¹<https://rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf>

그립 삽입 구문

```
! [] (figures/son.jpg)
```



3. Code Chunk

- 실제 R code가 실행되는 부분임
- Code chunk 실행 시 다양한 옵션들이 있으나 이 부분 역시 15주 차 강의에서 간략히 다룰 예정임
- 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
- Chunk option에 대한 상세 내용은 <https://yihui.org/knitr/options/> 또는 R Markdown 레퍼런스 가이드²² 참조

Code chunk 예시

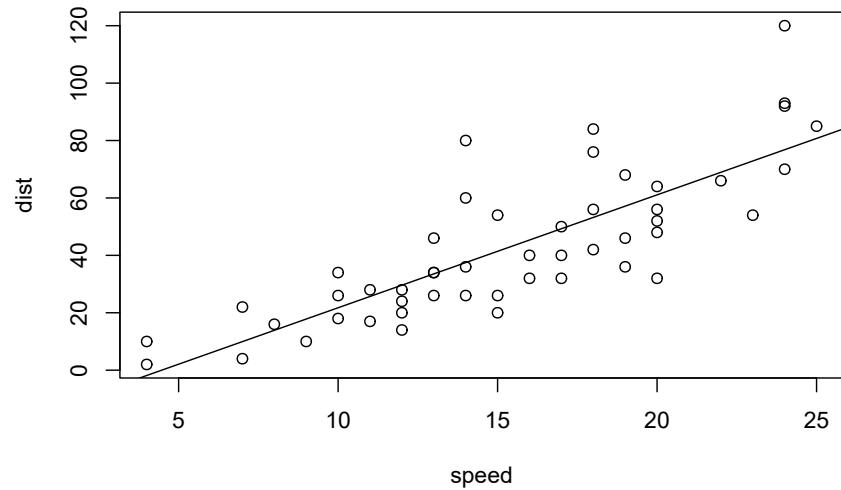
Xie의 R Markdown: The Definitive Guide에서 발췌

```
```{r}
fit = lm(dist ~ speed, data = cars)
b = coef(fit)
plot(cars)
abline(fit)
```

```

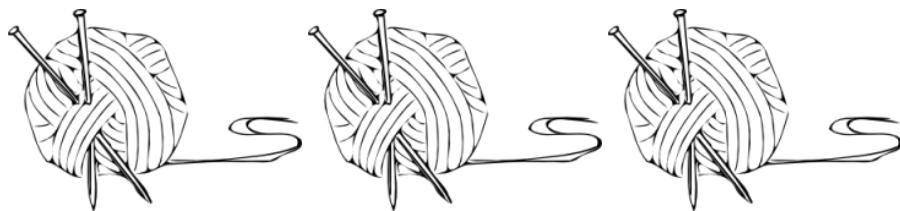
```
fit = lm(dist ~ speed, data = cars)
b   = coef(fit)
plot(cars)
abline(fit)
```

²²<https://rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf>



- Code chunk에서 외부 그림 파일 불러오기 (Xie et al. (2018) 에서 예시
발췌))

```
knitr::include_graphics(rep('figures/knit-logo.png', 3))
```



Homework 1: R Markdown 문서에 아래 내용을 포함한 문서를 html 파일 형식으로 출력 후 제출

- 간략한 자기소개 및 “통계 프로그래밍 언어” 수업에 대한 본인만의 목표 기술
- 본인이 setting 한 RStudio 구성 캡쳐 화면을 그림 파일로 저장하고 R Markdown 문서에 삽입(화면 캡쳐 시 생성 프로젝트 내 폴더 내용 반드시 포함)
- 패키지 `ggplot2`를 불러오고 `cars` 데이터셋의 2 차원 산점도 (`hint: help(geom_point)` 또는 googling 활용)를 문서에 포함



2

R 객체 (R object)



학습목표(2 주차): R에서 사용 가능한 데이터 타입에 대해 알아보고, 고유 데이터 타입으로 구성한 객체(스칼라, 벡터, 리스트)와 이와 연관된 함수들을 익힌다.

학습 필요성

- R 언어는 타 프로그래밍 언어와 유사한 데이터 타입(정수형, 실수형, 문자형 등)을 제공
- R 언어가 다른 언어와 차이점 → 데이터 분석에 특화된 벡터(vector), 행렬(matrix), 데이터프레임(data frame), 리스트(list)와 같은 객체¹ 제공
- R 패키지에서 제공되는 함수 사용 방법은 R의 객체에 따라 달라질 수 있음
- R 언어를 원활히 다룰 수 있으려면 R에서 데이터 객체의 형태, 자료 할당 및 그 연산 방법에 대한 이해가 필수적으로 선행되어야 함

R의 데이터 타입

- 수치형(numeric): 숫자(정수, 소수)
- 문자열(string): "충남대학교", "R강의"

¹R에서 사용자가 데이터 입력을 위해 생성 또는 읽어온 객체(object)는 종종 변수(variable)라는 말과 혼용. 본 문서에서는 최상위 데이터 저장장소를 객체라고 명명하며 데이터프레임과 같이 여러 종류의 데이터타입으로 이루어진 객체의 1차원 속성을 변수라고 칭함

- 논리형 (logical): TRUE/FALSE
- 결측값 (NA): 자료에서 발생한 결측 표현
- 공백 (NULL): 지정하지 않은 값
- 요인 (factor): 범주형 자료 표현 (수치 + 문자 결합 형태로 이해하면 편함)
- 기타: 숫자아님 (NaN), 무한대 (Inf) 등

R 객체의 종류

- 스칼라 (상수형, scalar 또는 atomic)
- 벡터 (vector): R의 기본연산 단위
- 리스트 (list)
- 행렬 (matrix)
- 배열 (array)
- 데이터프레임 (data frame)

아래 그림은 2~4 주차에 배운 R 주요 객체에 대한 개요도임

2.1 스칼라 (scalar)

- 단일 차원의 값 (하나의 값): 1×1 벡터로 표현 \rightarrow R 데이터 객체의 기본은 벡터!!
- 데이터 객체의 유형은 크게 숫자형, 문자열, 논리형이 있음

 스칼라를 입력시 R의 벡터 지정 함수인 `c()` (벡터 부분에서 상세 내용 학습)를 꼭 사용해서 입력할 필요가 없다. 단, 연속되지 않은 두 개 이상 스칼라면 벡터이므로 꼭 `c()`를 써야 한다.

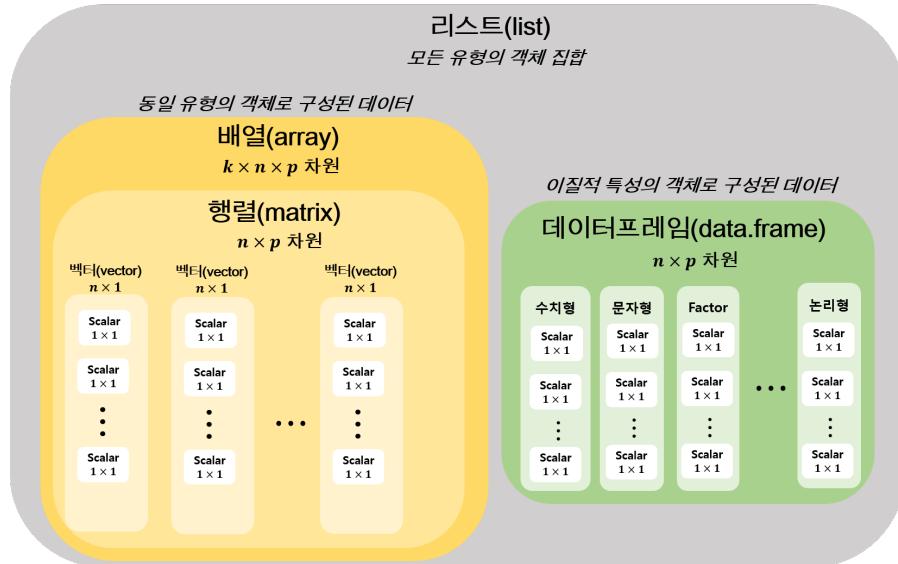


FIGURE 2.1: R 데이터 타입 구조 다이어그램: [R, Python 분석과 프로그래밍 (by R Friend)](<http://rfriend.tistory.com/>)에서 발췌 후 수정

2.1.1 선언

- 일반적으로 컴파일이 필요한 언어(예: C 언어)의 경우 변수 또는 객체를 사용 전에 선언이 필요

```
int x;
x = 1;
```

- 위 코드에서 `int x;` 없이 `x = 1;`을 입력 후 컴파일 하면 에러가 나타나지만 R 언어에서는 변수를 선언할 필요가 전혀 없음
- `z` 가 어떤 데이터 타입인지 언급할 필요가 전혀 없음 → Python, Perl, Matlab 등과 같은 스크립트 언어의 특징. 아래 코드 참조

```
z <- 3
z
```

```
[1] 3
```

2.1.2 숫자형

- 정수형 (integer)과 실수형 (double)로 구분됨
- 정수형 구분시 숫자 뒤 L을 표시

```
# 정수형 구분자 사용 예시
# typeof(): R 객체의 데이터 타입 반환하는 함수
typeof(10L)
```

```
[1] "integer"
```

```
typeof(10)
```

```
[1] "double"
```

- 수치연산 (+, -, *, ^, **, /, %%, %%) 가능: R은 함수형 언어이기 때문에 앞에 기술한 연산자도 하나의 함수로 인식함.
- 수치 연산자 (operator) 및 기본 수학 함수

TABLE 2.1: R언어의 기본 수치 연산자

| 수치형 연산자 | 설명 |
|-----------------|----------------|
| +, -, *, / | 사칙연산 |
| n %% m | n을 m 으로 나눈 나머지 |
| n %%/ m | n을 m 으로 나눈 몫 |
| n ^ m 또는 n ** m | n 의 m 승 |

숫자형 스칼라 연산 적용 예시

```
# 숫자형 스칼라  
a <- 3  
b <- 10  
a; b
```

```
[1] 3
```

```
[1] 10
```

```
# 덧셈  
c <- a + b  
c
```

```
[1] 13
```

```
# 덧셈을 함수로 입력  
# "+"(a, b)로 입력한 결과  
c <- "+"(a, b)
```

```
# 뺄셈  
d <- b - a  
d
```

```
[1] 7
```

```
# 곱셈  
m <- a * b  
m
```

```
[1] 30
```

```
# 나누기  
dd <- b/a  
dd
```

```
[1] 3.333333
```

```
# 멱승  
b^a
```

```
[1] 1000
```

```
# 나누기의 나머지 (remainder) 반환  
r <- b %% a  
r
```

```
[1] 1
```

```
# 나누기의 몫 (quotient) 반환  
q <- b %/% a  
q
```

```
[1] 3
```

```
# 연산 우선 순위  
nn <- (3 + 5)*3 - 4**2/4  
nn
```

```
[1] 20
```

2.1.3 문자형

- 수치형이 아닌 문자 형식의 단일 원소
- C와 같은 언어에서 볼수 있는 한개 문자에 대한 데이터 타입 존재하지 않음
- 수치연산 불가능
- 따옴표 (" 또는 ')로 문자를 묶어서 문자열 표시
- 문자열을 다루는 자세한 설명은 5주차에서 자세히 설명할 예정임

```
h1 <- c("Hello CNU!!")  
h2 <- c("R is not too difficult.")  
typeof(h1); typeof(h2)
```

```
[1] "character"
```

```
[1] "character"
```

```
h1
```

```
[1] "Hello CNU!!"
```

```
h2
```

```
[1] "R is not too difficult."
```

```
# 문자열의 문자 수 반환  
nchar(h1); nchar(h2)
```

```
[1] 11
```

```
[1] 23
```

```
# 문자열 연산 error 예시  
h1 - h2
```

```
Error in h1 - h2: 이항연산자에 수치가 아닌 인수입니다
```

2.1.4 논리형 스칼라

- 참(TRUE, T) 또는 거짓(FALSE, F)를 나타내는 값
- TRUE/FALSE: 예약어(reserved word)
- T/F: TRUE와 FALSE로 초기화된 전역 변수
 - T에 FALSE 또는 어떤 값도 할당 가능 → 가급적 TRUE/FALSE를 명시하는 것이 편함

- 논리형 연산자(logical operator)

TABLE 2.2: R 언어의 논리형 연산자

| 논리형 연산자 | 설명 |
|---------|------------------|
| & | AND (vectorized) |
| && | AND (atomic) |
| | OR (vectorized) |
| | OR (atomic) |
| ! | NOT |

- 비교 연산자를 적용할 경우 논리값을 반환

TABLE 2.3: R 언어의 비교 연산자

| 비교 연산자 | 설명 |
|--------|----------------------------|
| > | 크다(greater-than) |
| < | 작다(less-than) |
| == | 같다(equal) |
| >= | 크거나 같다(greater than equal) |
| <= | 작거나 같다(less than equal) |
| != | 같지 않다(not equal) |

Note:

기술한 비교 연산자는 수치형 및 논리형 데이터 타입 모두에 적용 가능하지만, 문자형은 비교 연산은 ==, != 만 가능함

참고

- 논리형 스칼라도 숫자형 연산 가능 → 컴퓨터는 TRUE/FALSE를 1과 0 숫자로 인식

- 수치 연산자는 스칼라 뿐 아니라 아래에서 다룰 벡터, 행렬, 리스트, 데이터프레임
객체의 연산에 사용 가능
- &/|와 &&/||는 동일하게 AND/OR를 의미하지만 연산 결과가 다름.
- &의 연산 대상이 벡터인 경우 벡터 구성 값 각각에 대해 & 연산을 실행 하지만 &&는
하나의 값(스칼라)에만 논리 연산이 적용(아래 예시 참고)

- 논리형 스칼라의 논리 및 비교 연산 예시

```
typeof(TRUE) # TRUE의 데이터 타입
```

```
[1] "logical"
```

```
TRUE & TRUE # TRUE 반환
```

```
[1] TRUE
```

```
TRUE & FALSE # FALSE 반환
```

```
[1] FALSE
```

```
# 아래 연산은 모두 TRUE 반환
```

```
TRUE | TRUE
```

```
[1] TRUE
```

```
TRUE | FALSE
```

```
[1] TRUE
```

```
# TRUE와 FALSE의 반대
```

```
! TRUE
```

```
[1] FALSE
```

```
! FALSE
```

```
[1] TRUE
```

```
# 전역변수 T에 FALSE 값 할당
T <- FALSE
T
```

```
[1] FALSE
```

```
T <- TRUE # 원상복귀
# TRUE/FALSE에 값을 할당할 수 없음
TRUE <- 1
```

Error in TRUE <- 1: 대입에 유효하지 않은 (do_set) 좌변입니다

```
TRUE <- FALSE
```

Error in TRUE <- FALSE: 대입에 유효하지 않은 (do_set) 좌변입니다

```
# &(/)와 &&(//)의 차이
l.01 <- c(TRUE, TRUE, FALSE, TRUE) # 논리형 값으로 구성된 벡터
l.02 <- c(FALSE, TRUE, TRUE, TRUE)
l.01 & l.02 # l.01과 l.02 각 원소 별 & 연산
```

```
[1] FALSE TRUE FALSE TRUE
```

```
l.01 && l.02 # l.01과 l.02의 첫 번째 원소에 대해 && 연산
```

```
[1] FALSE
```

```
# 비교 연산자
x <- 9
y <- 4
```

```
# x > y 의 반환값 데이터 타입  
typeof(x > y)
```

```
[1] "logical"
```

```
# 논리형 값 반환  
x > y
```

```
[1] TRUE
```

```
x < y
```

```
[1] FALSE
```

```
x == y
```

```
[1] FALSE
```

```
x != y
```

```
[1] TRUE
```

2.1.5 결측값 (missing value)

- 결측치 지정 상수: NA → R과 다른 언어의 가장 큰 차이점 중 하나
- 예를 들어 4명의 통계학과 학생 중 3명의 통계학 개론 중간고사 점수가 각각 80, 90, 75점이고 4번 째 학생의 점수가 없는 경우 NA로 결측값 표현
- is.na() 함수를 이용해 해당 값이 결측을 포함하고 있는지 확인

```
one <- 80; two <- 90; three <- 75; four <- NA  
four
```

```
[1] NA
```

```
# 'is.na()' 결측 NA가 포함되어 있으면 TRUE
is.na(four)
```

[1] TRUE

 `is.na(object_name)`: 객체를 구성하고 있는 원소 중 NA를 포함하고 있는지 확인 → NA를 포함하면 TRUE, 아니면 FALSE 반환

참고: 자료에 NA가 포함된 경우 연산 결과는 모두 NA가 반환

```
NA + 1
```

[1] NA

```
NA & TRUE
```

[1] NA

```
NA <= 3
```

[1] NA

2.1.6 NULL 값

- NULL: 초기화 되지 않은 변수 또는 객체를 지칭함
- `is.null()` 함수를 통해 객체가 NULL인지 판단

```
x <- NULL # NULL 지정
is.null(x) # NULL 객체인지 판단
```

[1] TRUE

```
x <- 1
is.null(x)
```

```
[1] FALSE
```



NA와 NULL의 차이점: 자료의 공백을 의미한다는 점에서 유사한 측면이 있으나 아래 내용처럼 큰 차이가 있음

- **NULL:** 값을 지정하지 않은 객체를 표현하는데 사용. 즉 아직 변수 또는 객체의 상태가 아직 미정인 상태를 나타냄
- **NA:** 데이터 값이 결측임을 지정해주는 논리형 상수

```
# NA와 NULL은 다른  
x <- NA  
is.null(NA)
```

```
[1] FALSE
```

```
is.na(NULL)
```

```
logical(0)
```

2.1.7 무한대/무한소/숫자아님

- **Inf:** 무한대 ($+\infty$, $1/0$)
- **-Inf:** 무한소 ($-\infty$, $-1/0$)
- **NaN:** 숫자아님 (Not a Number, $0/0$)
- **is.finite(), is.infinite(), is.nan()** 함수를 통해 객체가 Inf 또는 NaN을 포함하는지 확인

```
x <- Inf  
is.finite(x)
```

```
[1] FALSE
```

```
is.infinite(x)
```

```
[1] TRUE
```

```
x <- 0/0
is.nan(x)
```

```
[1] TRUE
```

```
is.infinite(x)
```

```
[1] FALSE
```



지금까지 요인형(factor)을 제외하고 R 언어에서 객체가 가질 수 있는 데이터 유형에 대해 알아봄. 요인형은 4주 차에 예정된 “R 자료형: 팩터, 테이블, 데이터 프레임”에서 상세하게 배울 예정임.

2.2 벡터 (vector)

2.2.1 벡터의 특징

- 타 프로그래밍 언어의 배열(array)의 개념으로 **동일한 유형**의 데이터 원소가 하나 이상($n \times 1, n \geq 1$)으로 구성된 자료 형태
- R 언어의 가장 기본적인 데이터 형태로 R에서 행해지는 모든 연산의 기본 (vectorization) → 벡터 연산 시 반복구문(예: `for loop`)이 필요 없음.
- 2.1** 절에서 기술한 **스칼라(scalar)**는 사실 1×1 벡터임
- 수학적으로 벡터는 아래와 같이 나타낼 수 있음

$$\mathbf{x} = [x_1, x_2, x_3, \dots, x_n]^T$$

- 벡터는 앞의 예시에서 본 바와 같이 `c()` 함수를 사용해 생성

```
# 숫자형 벡터  
x <- c(2, 0, 2, 0, 0, 3, 2, 4)  
x
```

```
[1] 2 0 2 0 0 3 2 4
```

```
# 문자형 벡터  
y <- c("Boncho Ku", "R programming", "Male", "sophomore", "2020-03-24")  
y
```

```
[1] "Boncho Ku"      "R programming" "Male"           "sophomore"  
[5] "2020-03-24"
```

- 두 개 이상의 벡터는 `c()` 함수를 통해 결합 가능
 - 함수 내 , 구분자를 통해 결합

```
# 두 벡터의 결합 (1)  
x <- 1:5  
y <- 10:6  
z <- c(x, y)  
x
```

```
[1] 1 2 3 4 5
```

```
y
```

```
[1] 10 9 8 7 6
```

```
z
```

```
[1] 1 2 3 4 5 10 9 8 7 6
```

```
x <- 5:10
x1 <- x[1:3] # x 벡터에서 1에서 4번째 원소 추출
x2 <- c(x1, 15, x[4])
x2
```

```
[1] 5 6 7 15 8
```

- 서로 다른 자료형으로 벡터를 구성한 경우 표현력이 높은 자료형으로 변환한
값 반환
 - 예: 문자열 + 숫자로 구성된 벡터 → 문자형 벡터
 - 변환 규칙: NULL < raw < logical < integer < double <
complex < character < list < expression

```
# 숫자형 벡터와 문자열 벡터 혼용
k <- c(1, 2, "3", "4")
k
```

```
[1] "1" "2" "3" "4"
```

```
is.numeric(k) # 벡터가 숫자형인지 판단하는 함수
```

```
[1] FALSE
```

```
is.character(k) # 벡터가 문자열인지 판단하는 함수
```

```
[1] TRUE
```

```
# 숫자형 벡터와 문자열 벡터 결합
x <- 1:3
y <- c("a", "b", "c")
z <- c(x, y)
z
```

```
[1] "1" "2" "3" "a" "b" "c"
```

```
is.numeric(z)
```

```
[1] FALSE
```

```
is.character(z)
```

```
[1] TRUE
```

```
# 숫자형 벡터와 논리형 벡터 결합  
x <- 9:4  
y <- c(TRUE, TRUE, FALSE)  
z <- c(x, y)  
  
z # TRUE/FALSE 가 1과 0으로 변환
```

```
[1] 9 8 7 6 5 4 1 1 0
```

```
is.numeric(z)
```

```
[1] TRUE
```

```
is.logical(z)
```

```
[1] FALSE
```

- 두 벡터는 중첩이 불가능 → 동일한 벡터 2개를 결합 시 단일 차원 벡터 생성

```
x <- y <- 1:3 # x와 y 동시에 [1, 2, 3] 할당  
x
```

```
[1] 1 2 3
```

```
y
```

```
[1] 1 2 3
```

```
z <- c(x, y)
z
```

```
[1] 1 2 3 1 2 3
```

- 벡터 각 원소에 이름 부여 가능
 - `names()` 함수를 이용해 원소 이름 지정
 - 사용 프로토타입: `names(x) <- 문자열 벡터`, 단 x와 이름에 입력할 문자열 벡터의 길이는 같아야 함.
 - `c()` 함수에서 직접 이름 지정 → `c(atom_name1 = value,`
`atom_name2 = value, ...)`

```
x <- c("Boncho Ku", "R programming", "Male", "sophomore", "2020-03-24")
```

```
# 벡터 원소 이름 지정
names(x) <- c("name", "course", "gender", "grade", "date")
x
```

| name | course | gender | grade | date |
|-------------|-----------------|--------|-------------|--------------|
| "Boncho Ku" | "R programming" | "Male" | "sophomore" | "2020-03-24" |

```
y <- c(a = 10, b = 6, c = 9)
names(y)
```

```
[1] "a" "b" "c"
```

- 벡터의 길이(차원) 확인
 - `length()` 또는 `NROW()` 사용

```
x <- 1:50
# 객체의 길이 반환
```

```
# length(): 벡터, 행렬인 경우 원소의 개수, 데이터프레임인 경우 열의 개수 반환
length(x)
```

```
[1] 50
```

```
# NROW(): 벡터인 경우 원소의 개수, 행렬, 데이터 프레임인 경우 행의 개수 반환
NROW(x)
```

```
[1] 50
```

2.2.2 벡터의 연산

- 원소 단위 사칙연산 및 비교연산 수행 → 벡터화 연산(vectorized operation)
 - 예를 들어 $x = [1, 2, 3]^T$ 이고, $y = [2, 3, 4]^T$ 라고 할 때 $x + y$ 의 연산은 아래와 같음

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \\ 7 \end{bmatrix}$$

- * 연산 시 행렬 대수학에서 벡터의 곱(product)과 다름을 주의

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} * \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 2 \\ 6 \\ 12 \end{bmatrix}$$

```
x <- 1:3; y <- 2:4
length(x); length(y)
```

```
[1] 3
```

```
[1] 3
```

```
x; y
```

```
[1] 1 2 3
```

```
[1] 2 3 4
```

```
# 사칙연산(+, -, *, /)
# 벡터 vs. 벡터
x + y
```

```
[1] 3 5 7
```

```
x - y
```

```
[1] -1 -1 -1
```

```
x * y
```

```
[1] 2 6 12
```

```
x / y
```

```
[1] 0.5000000 0.6666667 0.7500000
```

```
# 그외 연산
# 나머지 (remainder)
y %% x
```

```
[1] 0 1 1
```

```
# 몫 (quotient)
y %/% x
```

```
[1] 2 1 1
```

```
# 멱승 (exponent)
y ^ x
```

[1] 2 9 64

- 차원이 서로 맞지 않는 경우 작은 차원(짧은 쪽)의 벡터를 재사용함

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + [5] = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + \begin{bmatrix} 5 \\ 5 \\ 5 \end{bmatrix} = \begin{bmatrix} 6 \\ 7 \\ 8 \end{bmatrix}$$

```
# 벡터 ( $n$  by 1) vs. 스칼라 (1 by 1)
x * 5 # 5을 x의 길이 만큼 재사용(반복) 후 곱 연산 수행
```

[1] 5 10 15

```
x <- c(2, 1, 3, 5, 4); y <- c(2, 3, 4)
x
```

[1] 2 1 3 5 4

y

[1] 2 3 4

```
length(x); length(y)
```

[1] 5

[1] 3

```
# x의 길이가 5이고 y의 길이가 3이기 때문에 5를 맞추기 위해
# y의 원소 중 1-2 번째 원소를 재사용
x + y
```

```
Warning in x + y: 두 객체의 길이가 서로 배수관계에 있지 않습니다
```

```
[1] 4 4 7 7 7
```

```
x / y
```

```
Warning in x/y: 두 객체의 길이가 서로 배수관계에 있지 않습니다
```

```
[1] 1.0000000 0.3333333 0.7500000 2.5000000 1.3333333
```

- 연산 순서는 일반적인 사칙연산의 순서를 준용
 - 단 1단위 수열을 생성하는 : 연산자가 사칙연산을 우선함

```
# 연산 우선 순위  
1:5 * 3
```

```
[1] 3 6 9 12 15
```

```
1:(5 * 3)
```

```
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
```

- 논리형 값으로 구성된 벡터의 기본 연산 시 수치형으로 변환된 연산 결과를 반환

```
# 논리형 벡터  
b1 <- c(TRUE, TRUE, FALSE, TRUE, TRUE, TRUE, FALSE, FALSE)  
b2 <- c(FALSE, TRUE, TRUE, TRUE, TRUE, TRUE, FALSE, TRUE)  
  
is.numeric(b1); is.numeric(b2)
```

```
[1] FALSE
```

```
[1] FALSE
```

```
is.logical(b1); is.logical(b2)
```

```
[1] TRUE
```

```
[1] TRUE
```

```
# 논리형 벡터 연산  
b3 <- b1 + b2  
is.numeric(b3)
```

```
[1] TRUE
```

```
b3
```

```
[1] 1 2 1 2 2 2 0 1
```

```
b1 - b2
```

```
[1] 1 0 -1 0 0 0 0 -1
```

```
b1 * b2
```

```
[1] 0 1 0 1 1 1 0 0
```

```
b1/b2
```

```
[1] Inf 1 0 1 1 1 NaN 0
```

- 두 벡터 간 비교 연산은 사칙연산과 마찬가지로 각 원소단위 연산을 수행하고 논리형 벡터 반환
 - 재사용 규칙은 그대로 적용됨

```
# 두 벡터의 비교 연산  
x <- c(2, 4, 3, 10, 5, 9)  
y <- c(3, 4, 6, 2, 10, 7)
```

```
x == y
```

```
[1] FALSE TRUE FALSE FALSE FALSE FALSE
```

```
x != y
```

```
[1] TRUE FALSE TRUE TRUE TRUE TRUE
```

```
x > y
```

```
[1] FALSE FALSE FALSE TRUE FALSE TRUE
```

```
x < y
```

```
[1] TRUE FALSE TRUE FALSE TRUE FALSE
```

```
x >= y
```

```
[1] FALSE TRUE FALSE TRUE FALSE TRUE
```

```
x <= y
```

```
[1] TRUE TRUE TRUE FALSE TRUE FALSE
```

```
# 비교 연산 시 두 벡터의 길이가 다른 경우
```

```
x <- 1:5; y <- 2:4
```

```
x == y
```

```
Warning in x == y: 두 객체의 길이가 서로 배수관계에 있지 않습니다
```

```
[1] FALSE FALSE FALSE FALSE FALSE
```

```
x != y
```

Warning in x != y: 두 객체의 길이가 서로 배수관계에 있지 않습니다

```
[1] TRUE TRUE TRUE TRUE TRUE
```

```
x > y
```

Warning in x > y: 두 객체의 길이가 서로 배수관계에 있지 않습니다

```
[1] FALSE FALSE FALSE TRUE TRUE
```

```
x < y
```

Warning in x < y: 두 객체의 길이가 서로 배수관계에 있지 않습니다

```
[1] TRUE TRUE TRUE FALSE FALSE
```

```
x >= y
```

Warning in x >= y: 두 객체의 길이가 서로 배수관계에 있지 않습니다

```
[1] FALSE FALSE FALSE TRUE TRUE
```

```
x <= y
```

Warning in x <= y: 두 객체의 길이가 서로 배수관계에 있지 않습니다

```
[1] TRUE TRUE TRUE FALSE FALSE
```

- 문자열 벡터의 연산은 == 또는 != 만 가능(사칙연산 불가능)

```
# 문자열 벡터 연산 (==, !=)
c1 <- letters[1:5]
# a~z로 구성된 벡터에서 1~2, 6~8 번째 원소 추출
c2 <- letters[c(1:2, 6:8)]
c1
```

```
[1] "a" "b" "c" "d" "e"
```

```
c2
```

```
[1] "a" "b" "f" "g" "h"
```

```
c1 == c2
```

```
[1] TRUE TRUE FALSE FALSE FALSE
```

```
c1 != c2
```

```
[1] FALSE FALSE TRUE TRUE TRUE
```

- NA를 포함한 두 벡터 연산 시 동일 위치에 NA가 존재하면 어떤 연산이든 NA 값을 반환

```
# 결측을 포함한 벡터
x <- c(1:10, NA, NA)
y <- c(NA, NA, 1:10)
x
```

```
[1] 1 2 3 4 5 6 7 8 9 10 NA NA
```

```
y
```

```
[1] NA NA 1 2 3 4 5 6 7 8 9 10
```

```
is.na(x); is.na(y)
```

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE TRUE
```

```
[1] TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
# 결측을 포함한 벡터의 연산  
x + y
```

```
[1] NA NA 4 6 8 10 12 14 16 18 NA NA
```

```
x / y
```

```
[1] NA NA 3.000000 2.000000 1.666667 1.500000 1.400000 1.333333  
[9] 1.285714 1.250000 NA NA
```

```
x < y
```

```
[1] NA NA FALSE FALSE FALSE FALSE FALSE FALSE FALSE NA NA
```

```
x > y
```

```
[1] NA NA TRUE TRUE TRUE TRUE TRUE TRUE TRUE NA NA
```

- NULL이 벡터에 포함되더라도 벡터의 길이에는 변동이 없음

```
# NULL을 포함한 벡터  
x <- c(1, 2, 3, NULL, NULL, NULL) # 길이가 6?  
length(x)
```

```
[1] 3
```

```
x
```

```
[1] 1 2 3
```

2.2.3 벡터의 색인(indexing)

- 벡터의 특정 위치에 있는 원소를 추출

- 색인 (indexing)을 통해 벡터의 원소에 접근 가능
- 타 언어는 대체로 첫 번째 색인이 0에서 시작하지만, R은 1부터 시작
- $x[i]$: 벡터 x 의 i 번 째 요소
- $x[start:end]$: x 의 $start$ 부터 end 까지 값 반환

```
x <- c(1.2, 3.1, 4.2, 2.8, 3.3)
x[3] # x 원소 중 3 번째 원소 추출
```

```
[1] 4.2
```

```
# x 원소 중 2-3번째 원소 추출
x[2:3]
```

```
[1] 3.1 4.2
```

- $x[-i]$: 벡터 x 에서 i 번 째 요소를 제외한 나머지 값 반환

```
# x의 3 번째 원소 제거
x[-3]
```

```
[1] 1.2 3.1 2.8 3.3
```

```
# 맨 마지막 원소 (5 번째) 제거
# 아래 script는 동일한 결과 출력
x[1:(length(x) - 1)]
```

```
[1] 1.2 3.1 4.2 2.8
```

```
x[-length(x)]
```

```
[1] 1.2 3.1 4.2 2.8
```

- $x[idx_vec]$: idx_vec 가 인덱싱 벡터라고 할 때 idx_vec 에 지정된 요소를

얻어옴. 일반적으로 `idx_vec`는 벡터의 행 순서 번호 또는 각 벡터 원소의 이름에 대응하는 문자열 벡터를 인덱싱 벡터로 사용할 수 있음.

```
# 벡터를 이용한 인덱싱
# x 원소 중 1, 5번째 원소 추출
x[c(1, 5)] # c(1,5)는 벡터
```

[1] 1.2 3.3

```
v <- c(1, 4)
x[v]
```

[1] 1.2 2.8

```
# 인덱스 번호 중복 가능
x[c(1, 2, 2, 4)]
```

[1] 1.2 3.1 3.1 2.8

```
# 원소 이름으로 인덱싱
# 원소 이름 지정
names(x) <- paste0("x", 1:length(x)) # 문자열 "x"와 숫자 1:5(벡터 길이)를 결합한 문자열 반복
x["x3"]
```

x3

4.2

```
x[c("x2", "x4")]
```

x2 x4

3.1 2.8

- 필터링 (filtering): 특정한 조건을 만족하는 원소 추출
 - 비교 연산자를 이용한 조건 생성 → 논리값을 이용한 원소 추출

```
z <- c(5, 2, -3, 8)
# z의 원소 중 z의 제곱이 8보다 큰 원소 추출
w <- z[z^2 > 8]
w
```

[1] 5 -3 8

- 작동 원리

- $z^2 > 8$ 은 벡터 z의 모든 원소 제곱값이 8 보다 큰 케이스를 논리형 값으로 반환

z^2

[1] 25 4 9 64

```
idx <- z^2 > 8
idx
```

[1] TRUE FALSE TRUE TRUE

$z[idx]$

[1] 5 -3 8

- 특정 조건을 만족하는 벡터의 위치에 임의의 값을 치환할 수 있음

```
# 위 벡터 z 의 원소 중  $z^2 > 8$  인 원소의 값을 0으로 치환
z[idx] <- 0
```

2.2.4 벡터 관련 함수

- c() 함수 외에 R은 벡터 생성을 위해 몇 가지 유용한 함수를 제공함

seq 계열 함수

보다 자세한 사용 설명은 **help(seq)** 참고

seq(): 등차 수열 생성하는 함수로 **from**에서 **end** 까지 숫자 내에서 공차(간격)가 **by** 인 수열 생성

```
# seq(): 수열 생성 함수
seq(
  from, # 시작값
  to,   # 끝값
  by    # 공차(증가치)
)

# 기타 인수
# length.out = n
#   - 생성하고자 하는 벡터의 길이가 n인 수열 생성
# along.with = 1:n
#   - index가 1에서 n 까지 길이를 갖는 수열 생성
```

- 사용 예시

```
x <- seq(from = 2, to = 30, by = 2)
```

```
x
```

```
[1]  2  4  6  8 10 12 14 16 18 20 22 24 26 28 30
```

```
# 간격이 꼭 정수가 아니어도 사용 가능
```

```
x <- seq(from = 0, to = 3, by = 0.2)
```

```
# by 대신 length.out 으로 생성된 수열의 길이 조정
```

```
x <- seq(from = -3, to = 3, length.out = 10)
x

[1] -3.0000000 -2.3333333 -1.6666667 -1.0000000 -0.3333333  0.3333333
[7]  1.0000000  1.6666667  2.3333333  3.0000000
```

```
# from, to 인수 없이 length.out=10 인 경우
seq(length.out = 10)
```

```
[1]  1  2  3  4  5  6  7  8  9 10
```

```
# by 대신 along.width
seq(along.with=1:10)
```

```
[1]  1  2  3  4  5  6  7  8  9 10
```

```
seq(1, 5, along.with=1:10)
```

```
[1] 1.000000 1.444444 1.888889 2.333333 2.777778 3.222222 3.666667 4.111111
[9] 4.555556 5.000000
```

```
# 벡터 x에 seq() 함수 적용 시 1:length(x) 값 반환
seq(x)
```

```
[1]  1  2  3  4  5  6  7  8  9 10
```

seq_along(): 주어진 객체의 길이 만큼 1부터 1 간격의 수열 생성

- seq() 함수와 매우 유사하나, 무조건 1부터 시작해서 인수로 seq()의 along.with 값을 이용한 함수
- seq() 함수보다 조금 빠름
- 사용 예시

```
# 1부터 x 벡터의 길이 까지 1 단위 수열 값 반환  
seq_along(x)
```

```
[1] 1 2 3 4 5 6 7 8 9 10
```

seq_len(): 인수로 받은 값 만큼 1부터 해당 값 까지 1 간격의 수열 생성

- `seq()` 함수의 인수 중 `length.out` 값을 이용한 함수
- 사용 예시

```
# 1부터 n 까지 1 단위 수열 값 반환  
seq_len(10)
```

```
[1] 1 2 3 4 5 6 7 8 9 10
```

rep 계열 함수

`help(rep)`을 통해 상세 내용 참고

rep(): 주어진 벡터의 원소를 반복

```
# rep(): 벡터 또는 벡터의 개별 원소를 반복한 값 반환  
rep(  
  x, # 반복할 값이 저장된 벡터  
  times, # 전체 벡터의 반복 횟수  
  each # 개별 원소의 반복 횟수  
)
```

- 사용 예시

```
x <- rep(4, 5) # 4를 5번 반복  
x
```

```
[1] 4 4 4 4 4
```

```
# x <- c(1:3) 전체를 3번 반복한 벡터 반환
x <- c(1:3)
xr1 <- rep(x, times = 3)
xr1
```

```
[1] 1 2 3 1 2 3 1 2 3
```

```
# 벡터 x 의 각 원소를 4번씩 반복한 벡터 반환
xr2 <- rep(x, each = 4)
xr2
```

```
[1] 1 1 1 1 2 2 2 2 3 3 3 3
```

```
# 벡터 x 의 각 원소를 3번 반복하고 해당 벡터를 2회 반복
xr3 <- rep(x, each = 3, times = 2)
xr3
```

```
[1] 1 1 1 2 2 2 3 3 3 1 1 1 2 2 2 3 3 3
```

```
# 문자형 벡터의 반복
# 아래 sex 벡터의 각 원소를 2 번 반복하고 해당 벡터를 4회 반복
sex <- c("Male", "Female")
sexr <- rep(sex, each = 2, times = 4)
sexr
```

```
[1] "Male"   "Male"   "Female" "Female" "Male"   "Male"   "Female" "Female"
[9] "Male"   "Male"   "Female" "Female" "Male"   "Male"   "Female" "Female"
```

rep.int() & rep_len(): rep() 함수의 simple 버전으로 속도 (performance)가 요구되는 프로그래밍 시 사용

- 사용 예시

```
# 1:5 벡터를 3 번 반복  
rep.int(1:5, 3)
```

```
[1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
```

```
# 불완전한 사이클로 벡터 반복  
rep_len(1:5, length.out = 7)
```

```
[1] 1 2 3 4 5 1 2
```

Filtering 관련 함수

```
help(subset) 참고
```

subset(): 기존 필터링 방식과 비교할 때 NA를 처리하는 방식에서 차이를 보임

- 벡터 뿐 아니라 앞으로 배울 행렬 및 데이터프레임 객체에도 적용 가능

```
x <- c(6, 1:3, NA, NA, 12)  
x
```

```
[1] 6 1 2 3 NA NA 12
```

```
# 일반적 필터링 적용  
x[x > 5]
```

```
[1] 6 NA NA 12
```

```
# subset() 함수 적용  
subset(x, x > 5)
```

```
[1] 6 12
```

which(): 한 벡터에서 특정 조건에 맞는 위치(인덱스)를 반환

```
# which(): 논리형 벡터를 인수로 받고 해당 논리형 벡터가 참인 index 반환
which(
  logical_vec # 논리형 벡터
)
```

- 사용 예시

```
x <- c(3, 8, 3, 1, 7)
```

```
# x의 원소값이 3인 index 반환
which(x == 3)
```

```
[1] 1 3
```

```
# x의 원소가 4보다 큰 원소의 index 반환
```

```
which(x > 4)
```

```
[1] 2 5
```

```
# 9월 (Sep)과 12월 (Dec)와 같은 원소 index
```

```
# month.abb: R 내장 벡터로 월 약어 (Jan ~ Dec)를 저장한 문자열 벡터
which(month.abb == c("Sep", "Dec"))
```

```
[1] 9 12
```

```
# 조건을 만족하는 원소가 존재하지 않는다면?
```

```
x <- which(x > 9)
x
```

```
integer(0)
```

```
length(x) # 길이가 0인 벡터 반환 is.null(x) == TRUE ??
```

```
[1] 0
```

```
is.null(x)
```

```
[1] FALSE
```

```
# 특정 조건 만족 여부를 확인  
# any(condition) -> 하나라도 condition을 만족하는 원소가 존재하는지 판단  
# TRUE 또는 FALSE 값 반환  
any(x > 9)
```

```
[1] FALSE
```

집합 관련 함수

- 벡터는 숫자, 문자열의 묶음, 즉 원소들의 집합(set)으로 볼 수 있기 때문에 집합 연산이 가능
- 두 집합을 X 와 Y 로 정의 했을 때 아래와 같은 집합 연산 가능
- **setequal(X, Y)**: X 와 Y 가 동일한지 판단 ($X = Y$) \rightarrow 논리값 TRUE 또는 FALSE 반환

```
x <- y <- c(1, 9, 7, 3, 6)  
setequal(x, y)
```

```
[1] TRUE
```

- **union(X, Y)**: X 와 Y 의 합집합 ($X \cup Y$)

```
y <- c(1, 9, 8, 2, 0, 3)  
union(x, y)
```

```
[1] 1 9 7 3 6 8 2 0
```

- **intersect(X, Y)**: X 와 Y 의 교집합 ($X \cap Y$)

```
intersect(x, y)
```

```
[1] 1 9 3
```

- **setdiff(X, Y)**: X와 Y의 차집합 ($X - Y$)

```
setdiff(x, y)
```

```
[1] 7 6
```

```
setdiff(y, x)
```

```
[1] 8 2 0
```

- **X %in% Y**: X(기준)가 집합 Y의 원소인지 논리값 반환

```
x <- c("apple", "banana", "strawberry", "mango", "peach", "orange")
y <- c("strawberry", "orange", "mango")
```

```
x %in% y
```

```
[1] FALSE FALSE TRUE TRUE FALSE TRUE
```

```
y %in% x
```

```
[1] TRUE TRUE TRUE
```

두 벡터의 동일성 테스트

- 두 벡터가 동일한지 테스트 하기 위해 **x == y** 연산의 반환 값은 위의 예제에서 확인한 것처럼 각 원소에 대한 논리값을 반환(아래 예제 확인)

```
x <- 1:3
```

```
y <- c(1, 3, 4)  
x == y
```

```
[1] TRUE FALSE FALSE
```

- 단지 두 벡터가 동일한지 아닌지를 확인하기 위해서는 하나의 논리값만 필요한 경우 `all()` 사용

```
all(x == y)
```

```
[1] FALSE
```

- 보다 나은 방법으로 `identical()` 함수 적용

```
# 두 객체의 동일성 여부 테스트  
identical(x, y)
```

```
[1] FALSE
```

- `identical()` 함수는 벡터가 갖는 데이터 타입의 동일성 까지 체크함

```
x <- 1:5; y <- c(1, 2, 3, 4, 5)  
x
```

```
[1] 1 2 3 4 5
```

```
y
```

```
[1] 1 2 3 4 5
```

```
# all() 함수로 동일성 확인  
all(x == y)
```

```
[1] TRUE
```

```
# identical 함수로 동일성 확인
identical(x, y)
```

[1] FALSE

```
# x, y 데이터 타입 확인
typeof(x)
```

[1] "integer"

```
typeof(y)
```

[1] "double"

2.3 리스트(list)

- **리스트(list)**: (key, value) 형태로 데이터를 저장한 배열(벡터)
- 서로 다른 데이터 타입을 가진 객체를 원소로 가질 수 있는 벡터
 - 예: 한 리스트 안에는 상이한 데이터 타입(숫자형, 문자형, 논리형 등)을 갖는 원소(객체)들을 포함할 수 있음



리스트 예시: 통계프로그래밍언어 중간고사 성적 테이블

- 중간고사 성적 테이블은 이름, 학번, 출석률, 점수, 등급으로 이루어졌다고 가정하면 “김상자”의 성적 리스트는 다음과 같이 나타낼 수 있음
 - LIST(이름 = “김상자”, 학번 = “202015115”, 점수 = 95, 등급 = “A-”)
 - 위 record에서 보듯이 문자형과 숫자형이 LIST 안에 같이 표현되고 있음

- 위 record를 벡터 생성함수 `c()`로 생성한 경우

```
# 벡터로 위 record를 입력한 경우
vec <- c(`이름` = "김상자", `학번` = "202015115",
           `점수` = 95, `등급` = "A-")
vec
```

| | | | |
|-------|-------------|------|------|
| 이름 | 학번 | 점수 | 등급 |
| "김상자" | "202015115" | "95" | "A-" |

```
typeof(vec)
```

```
[1] "character"
```



객체 명칭 규칙을 벗어나는 이름을 객체명으로 사용하고 싶다면 다음과 같이 훌따옴표 'object_name' 표시를 통해 사용 가능함

```
> #공백이 있는 이름을 객체 명칭으로 사용
> `golf score` <- c(75, 82, 92)
> `golf score`
```

```
[1] 75 82 92
```

```
> `3x` <- c(3, 6, 9, 12)
> `3x`
```

```
[1] 3 6 9 12
```

2.3.1 리스트 생성

- `list()` 함수를 사용해 list 객체 생성

```
# list 함수 사용 prototype
list(name_1 = object_1, ..., name_m = object_m)
```

```
# name_1, ..., name_m: 리스트 원소 이름
# object_1, ..., object_m: 리스트 원소에 대응한 객체
```

- 중간고사 성적 테이블 예시

```
# lst 객체 생성
lst <- list(`이름` = "김상자",
            `학번` = "202015115",
            `점수` = 95,
            `등급` = "A-")
lst
```

\$이름

[1] "김상자"

\$학번

[1] "202015115"

\$점수

[1] 95

\$등급

[1] "A-"

```
# lst 내 객체의 데이터 타입 확인
# lapply(): lst 객체에 동일한 함수 적용 (추후 학습)
lapply(lst, typeof)
```

\$이름

[1] "character"

\$학번

[1] "character"

```
$점수
```

```
[1] "double"
```

```
$등급
```

```
[1] "character"
```

- 리스트 원소에 이름이 부여된 경우 names()를 통해 확인 가능

```
names(lst)
```

```
[1] "이름" "학번" "점수" "등급"
```

- 이름(name_1, ..., name_n) 없이도 리스트 생성 가능하나, 가급적 이름을 부여 하는 것이 더 명확

```
list("김상자", "202015115", 95, "A-")
```

```
[[1]]
```

```
[1] "김상자"
```

```
[[2]]
```

```
[1] "202015115"
```

```
[[3]]
```

```
[1] 95
```

```
[[4]]
```

```
[1] "A-"
```

- 리스트는 벡터이므로 vector() 함수를 통해 생성 가능

```
# 길이가 10이고 객체가 NULL인 리스트 생성
```

```
z <- vector(mode = "list", length=1)
z
```

[[1]]

NULL

- 리스트의 값이 어떤 객체든 관계 없음

```
x <- list(name = c("A", "B", "C"),
           salary = c(500, 450, 600), union = T)
x
```

\$name

[1] "A" "B" "C"

\$salary

[1] 500 450 600

\$union

[1] TRUE

2.3.2 리스트 색인

- 리스트에 포함된 객체에 접근은 기본적으로 벡터의 색인 방법과 동일하게 색인 번호 또는 키(이름)을 통해 접근 가능
- 리스트에 포함된 모든 객체의 원소값을 쉽게 확인하는 함수는 `unlist()`임

```
lval <- unlist(x)
typeof(lval)
```

[1] "character"

TABLE 2.4: 리스트 데이터 접근 방법

| 색인방법 | 동작 |
|--|---|
| <code>x\$name</code> | 리스트 <code>x</code> 에서 객체명 (<code>name</code>)에 해당하는 객체에 접근 |
| <code>x[[i]]</code> 또는 <code>x[[name]]</code> | 리스트 <code>x</code> 에서 <code>i</code> 번째 또는 <code>name</code> 에 해당하는 객체 반환 |
| <code>x[i]</code> 또는 <code>x[name]</code> | 리스트 <code>x</code> 에서 <code>i</code> 번째 또는 <code>name</code> 에 해당하는 부분 리스트 반환 |

- `x$name`을 통해 리스트 내 객체 접근

```
lst$`학번`
```

```
[1] "202015115"
```

- `x[[i]]` 또는 `x[[name]]` 을 통해 리스트 내 객체 접근

```
lst[[2]]
```

```
[1] "202015115"
```

```
z <- lst[["학번"]]
z
```

```
[1] "202015115"
```

```
typeof(z)
```

```
[1] "character"
```

- `x[i]` 또는 `x[name]` 을 통해 리스트 내 부분 리스트 추출

```
lst[2]
```

\$학번

```
[1] "202015115"
```

```
j <- lst["학번"]
```

```
j
```

\$학번

```
[1] "202015115"
```

```
typeof(j)
```

```
[1] "list"
```

- 리스트 또한 벡터로 볼 수 있기 때문에 여러 개의 부분 리스트 추출 가능

```
# 리스트 lst 에서 1 ~ 3 번째 까지 부분 리스트 추출
```

```
lst[1:3]
```

\$이름

```
[1] "김상자"
```

\$학번

```
[1] "202015115"
```

\$점수

```
[1] 95
```

- 리스트를 구성하는 객체 내 색인

```
x
```

\$name

```
[1] "A" "B" "C"
```

```
$salary  
[1] 500 450 600
```

```
$union  
[1] TRUE
```

```
# salary에서 2-3번째 원소 추출  
x$salary[2:3]
```

```
[1] 450 600
```

```
x[[2]][2:3]
```

```
[1] 450 600
```

```
x[["salary"]][2:3]
```

```
[1] 450 600
```

```
# 부분 리스트도 길이가 1인 리스트이므로,  
# 부분 리스트 내 객체 접근 시 리스트 접근이 선행  
# x의 2번째 부분 리스트에서 첫 번째 객체의 2-3번째 원소 추출  
x[2][[1]][2:3]
```

```
[1] 450 600
```

- 리스트의 길이 반환: 벡터와 마찬가지로 `length()` 함수 적용 가능

```
length(lst); length(x)
```

```
[1] 4
```

```
[1] 3
```

2.3.3 리스트에 원소 추가/제거

- 주어진 리스트 `x`에 새로운 원소를 `x$new_obj <- value` 명령어 형태로 추가
- 이미 존재하고 있는 리스트 원소 제거는 `x$exist_obj <- NULL` 형태로 제거

```
# 리스트 lst에 5회 차 퀴즈 점수 추가  
lst$quiz <- c(10, 8, 9, 9, 8)  
  
# 리스트 lst의 원소 quiz 제거  
lst$quiz <- NULL  
lst
```

\$이름

[1] "김상자"

\$학번

[1] "202015115"

\$점수

[1] 95

\$등급

[1] "A-"

```
# 벡터 색인을 이용해 원소 추가 가능
```

```
lst[[5]] <- c(10, 8, 9, 9, 8)  
lst
```

\$이름

[1] "김상자"

\$학번

```
[1] "202015115"
```

\$점수

```
[1] 95
```

\$등급

```
[1] "A-"
```

```
[[5]]
```

```
[1] 10 8 9 9 8
```

```
# 부분 리스트 괄호에서도 색인 통해 추가/삭제 가능
lst[5] <- NULL
lst
```

\$이름

```
[1] "김상자"
```

\$학번

```
[1] "202015115"
```

\$점수

```
[1] 95
```

\$등급

```
[1] "A-"
```

```
# 여러 개의 리스트 동시 추가/삭제 가능
lst[5:9] <- c(10, 8, 9, 9, 8)
lst
```

\$이름

```
[1] "김상자"
```

\$학번

```
[1] "202015115"
```

\$점수

```
[1] 95
```

\$등급

```
[1] "A-"
```

[[5]]

```
[1] 10
```

[[6]]

```
[1] 8
```

[[7]]

```
[1] 9
```

[[8]]

```
[1] 9
```

[[9]]

```
[1] 8
```

```
lst[5:9] <- NULL
```

```
lst
```

\$이름

```
[1] "김상자"
```

\$학번

```
[1] "202015115"
```

```
$점수
```

```
[1] 95
```

```
$등급
```

```
[1] "A-"
```

2.3.4 리스트의 결합

- 두 개 이상의 리스트를 결합 시 `c()` 사용

```
# 리스트 lst와 x 결합  
c(lst, x)
```

```
$이름
```

```
[1] "김상자"
```

```
$학번
```

```
[1] "202015115"
```

```
$점수
```

```
[1] 95
```

```
$등급
```

```
[1] "A-"
```

```
$name
```

```
[1] "A" "B" "C"
```

```
$salary
```

```
[1] 500 450 600
```

```
$union
```

```
[1] TRUE
```



리스트 내에 리스트를 가질 수 있다. 이를 재귀 리스트(recursive list)라고 한다. 예를 들어 위 예제에서 각 학생의 성적 데이터가 리스트로 구성되어 있다면, 전체 성적 데이터베이스는 리스트로 구성된 리스트임. 아래 예제 처럼 간단한 재귀 리스트 구현이 가능

```
kim <- list(id = "20153345", sex = "Male", score = 85, grade = "B+")
lee <- list(id = "20153348", sex = "Female", score = 75, grade = "B0")
```

```
gr <- list(kim=kim, lee=lee)
```

```
gr
```

```
$kim
```

```
$kim$id
```

```
[1] "20153345"
```

```
$kim$sex
```

```
[1] "Male"
```

```
$kim$score
```

```
[1] 85
```

```
$kim$grade
```

```
[1] "B+"
```

```
$lee
```

```
$lee$id
```

```
[1] "20153348"
```

```
$lee$sex
```

```
[1] "Female"
```

```
$lee$score
```

```
[1] 75
```

```
$lee$grade
```

```
[1] "BO"
```

2.4 행렬 (matrix)



학습목표(3 주차): 행렬, 배열, 요인형과 테이블에 대해 살펴보고, 이를 객체에 대한 연산과 연관된 함수에 대해 익힌다.

행렬의 정의

- 동일한 데이터 타입의 원소로 구성된 2차원 데이터 구조
- $n \times 1$ 차원 벡터 p 개로 둑여진 데이터 뉴어리 $\rightarrow n \times p$ 행렬로 명칭함
- 행렬의 형태

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \cdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}$$

- R에서 행렬은 동일한 유형의 데이터 타입으로 구성 가능 \rightarrow 첫 번째 행은 숫자형, 두 번째 행은 문자열로 입력해도 행렬을 만들 수 있지만, 표현력이 더 높은 문자형 행렬 반환
- 행렬의 내부 저장공간은 “열 우선 배열”
- 행렬 생성을 위한 R 함수는 `matrix()` 함수이고 사용 형태는 아래와 같음

```
# matrix(): 행렬 생성 함수
# 상세 내용은 help(matrix)를 통해 확인

matrix(data, # 행렬을 생성할 데이터 벡터
       nrow, # 행의 개수 (정수)
       ncol, # 열의 개수 (정수)
       byrow, # TRUE: 행 우선, FALSE: 열 우선
       # default = FALSE
       dimnames # 행렬을 각 차원에 부여할 이름 (리스트)
       )
```

- 행렬 생성 예시

```
# byrow = FALSE
```

```
x <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 3, ncol = 3)
x
```

```
[,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

```
# byrow = TRUE
```

```
x <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 3, ncol = 3, byrow = T)
x
```

```
[,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

- 행의 개수(nrow)나 열의 개수(ncol)로 나머지를 추정 가능하다면 둘 중 어떤 인수도 생략 가능

```
x <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), ncol = 3)
x
```

```
[,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

```
x <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9), nrow = 3)
x
```

```
[,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

- `nrow × ncol` 이 입력한 데이터(벡터)의 길이보다 작거나 큰 경우

```
# length(x) < nrow * ncol 인 경우
# nrow * ncol에 해당하는 길이 만큼
# x의 원소를 사용해 행렬 생성
x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)
y <- matrix(x, nrow = 3, ncol = 4)
```

Warning in `matrix(x, nrow = 3, ncol = 4)`: 데이터의 길이[9]가 열의 개수[4]의 배수
가 되지 않습니다

```
y
```

```
[,1] [,2] [,3] [,4]
[1,]    1    4    7    1
[2,]    2    5    8    2
[3,]    3    6    9    3
```

```
# length(x) > nrow * ncol 인 경우
# x의 첫 번째 원소부터 초과하는 만큼
# x 원소의 값을 재사용
z <- matrix(x, nrow = 2, ncol = 3)
```

`Warning in matrix(x, nrow = 2, ncol = 3): 데이터의 길이[9]가 행의 개수[2]의 배수
가 되지 않습니다`

```
z
```

```
[,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

- 행렬 구성 시 길이에 대한 약수가 아닌 값을 `nrow` 또는 `ncol`의 인수로 받은 경우

```
# x (length=9)로 행렬 생성 시 nrow=4 를
# 인수로 입력한 경우
h <- matrix(x, nrow = 4)
```

`Warning in matrix(x, nrow = 4): 데이터의 길이[9]가 행의 개수[4]의 배수가 되지 않
습니다`

```
h
```

```
[,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6    1
[3,]    3    7    2
[4,]    4    8    3
```

```
# x (length=9)로 행렬 생성 시 ncol=2 만
```

```
# 인수로 입력한 경우
h <- matrix(x, nrow = 2)
```

Warning in matrix(x, nrow = 2) : 데이터의 길이[9]가 행의 개수[2]의 배수가 되지 않습니다

```
h
```

```
[,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8    1
```

2.4.1 행렬의 연산

- 선형대수(linear algebra)에서 배우는 행렬-스칼라, 행렬-행렬 간 연산 가능

행렬-스칼라 연산

합 연산: 스칼라가 자동적으로 행렬의 차원에 맞춰서 재사용

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} + 4 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} + \begin{bmatrix} 4 & 4 & 4 \\ 4 & 4 & 4 \\ 4 & 4 & 4 \end{bmatrix} = \begin{bmatrix} 5 & 6 & 7 \\ 8 & 9 & 10 \\ 11 & 12 & 13 \end{bmatrix}$$

```
x <- matrix(1:9, 3, 3, byrow = T)
x + 4
```

```
[,1] [,2] [,3]
[1,]    5    6    7
[2,]    8    9   10
[3,]   11   12   13
```

곱 연산

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \times 4 = \begin{bmatrix} 4 & 8 & 12 \\ 16 & 20 & 24 \\ 28 & 32 & 36 \end{bmatrix}$$

```
x*4
```

```
[,1] [,2] [,3]
[1,]    4     8    12
[2,]   16    20    24
[3,]   28    32    36
```

행렬-행렬 연산

- 행렬 간 연산에서 스칼라 연산(일반 연산)과 다른 점은 차원이 개입

행렬 간 합(차)

- 두 행렬의 동일 차원 간 합 연산 수행 (+ 또는 - 연산자 사용)

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} + \begin{bmatrix} 1 & -1 & 2 \\ 3 & 2 & 4 \\ -6 & 3 & -7 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 5 \\ 7 & 7 & 10 \\ 1 & 11 & 2 \end{bmatrix}$$

```
x <- matrix(1:9, 3, 3, byrow = T)
y <- matrix(c(1, 3, -6, -1, 2, 3, 2, 4, -7), ncol = 3)
x + y
```

```
[,1] [,2] [,3]
[1,]    2     1     5
[2,]    7     7    10
[3,]    1    11     2
```

행렬 곱/나누기 (elementwise product/division)

- 연산자 * 또는 / 사용

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} * \begin{bmatrix} 1 & -1 & 2 \\ 3 & 2 & 4 \\ -6 & 3 & -7 \end{bmatrix} = \begin{bmatrix} 1 & -2 & 6 \\ 12 & 10 & 24 \\ -42 & 24 & -63 \end{bmatrix}$$

```
x * y
```

```
[,1] [,2] [,3]
[1,]    1   -2    6
[2,]   12   10   24
[3,]  -42   24  -63
```

- 행렬-행렬 합(차) 또는 곱(나누기) 연산 시 행렬의 열단위 원소가 재사용되지 않음

동일 차원 간 연산만 가능!!

```
z <- y[, 1:2] # y 행렬에서 1-2 번째 열 추출
z # 3 by 2 행렬
```

```
[,1] [,2]
[1,]    1   -1
[2,]    3    2
[3,]   -6    3
```

```
x + z
```

Error in x + z: 배열의 크기가 올바르지 않습니다

```
x * z
```

Error in x * z: 배열의 크기가 올바르지 않습니다

```
x / z
```

Error in x/z: 배열의 크기가 올바르지 않습니다

행렬 간 곱(matrix product)

- 두 행렬 $\mathbf{X}_{n \times m}$, $\mathbf{Y}_{m \times k}$ 이 주어졌을 때 두 행렬의 곱(matrix product) $\mathbf{Z} = \mathbf{X} \cdot \mathbf{Y}$ 는 $n \times k$ 행렬이고 \mathbf{Z} 원소 z_{ij} ($i = 1, \dots, n$, $j = 1, \dots, k$) 아래와 같이 정의됨

$$z_{ij} = \sum_{r=1}^m x_{ir}y_{rj}, \quad \forall \{i, j\}$$

- R에서 위와 같은 연산은 %*%를 사용

- 예시: 행렬 $\mathbf{X}_{2 \times 4}$, $\mathbf{Y}_{4 \times 3}$ 이 아래와 같이 주어졌을 때 두 행렬의 곱 $\mathbf{Z}_{2 \times 3} = \mathbf{X}_{2 \times 4} \mathbf{Y}_{4 \times 3}$ 은 아래와 같음

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} 1 & -2 & -1 \\ 1 & 1 & 2 \\ 1 & 3 & 1 \\ 1 & 2 & 2 \end{bmatrix}$$

$$\mathbf{Z} = \mathbf{XY} = \begin{bmatrix} 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & -2 & -1 \\ 1 & 1 & 2 \\ 1 & 3 & 1 \\ 1 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 2 & -2 & 2 \\ 2 & 2 & 0 \end{bmatrix}$$

```
X <- matrix(c(1,1,1,-1,-1,1,1,1), nrow = 2, ncol = 4)
Y <- matrix(c(1,1,1,1, -2, 1, 3, 2, -1, 2, 1, 2), nrow = 4, ncol = 3)
```

```
Z <- X %*% Y
Z
```

```
[,1] [,2] [,3]
[1,]    2   -2    2
[2,]    2    2    0
```

행렬-벡터 연산

- 행렬 \mathbf{X} 의 행 길이와 벡터 \mathbf{y} 의 길이가 같은 경우 $\rightarrow \mathbf{y}$ 를 열 단위로 재사용

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 4 \\ 1 & 3 & 2 \\ 1 & 2 & 1 \end{bmatrix}, \quad \mathbf{y} = [20, 18, 23]^T$$

$$\mathbf{X} + \mathbf{y} = \begin{bmatrix} 1 & 2 & 4 \\ 1 & 3 & 2 \\ 1 & 2 & 1 \end{bmatrix} + \begin{bmatrix} 20 & 20 & 20 \\ 18 & 18 & 18 \\ 23 & 23 & 23 \end{bmatrix} = \begin{bmatrix} 21 & 22 & 24 \\ 19 & 21 & 20 \\ 24 & 25 & 24 \end{bmatrix}$$

```
#행렬-벡터 합 연산
# X = 3 by 3 행렬; y = 3 by 1 벡터
x <- c(1, 1, 1, 2, 3, 2, 4, 2, 1)
X <- matrix(x, nrow = 3)
y <- c(20, 18, 23) # 재사용

X + y
```

```
[,1] [,2] [,3]
[1,]    21   22   24
[2,]    19   21   20
[3,]    24   25   24
```

- 행렬 \mathbf{X} 의 길이와 벡터 \mathbf{y} 의 길이가 같은 경우 \rightarrow 벡터 \mathbf{y} 를 자동으로 원소를 행렬(열단위)로 변환

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \quad \mathbf{y} = [1, 2, \dots, 9]^T$$

$$\mathbf{X} + \mathbf{y} = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix} + \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix} = \begin{bmatrix} 2 & 8 & 14 \\ 4 & 10 & 16 \\ 6 & 12 & 18 \end{bmatrix}$$

```
#행렬-벡터 합 연산
# 행렬 X의 길이와 벡터 y의 길이가 같은 경우
x <- c(1:9); X <- matrix(x, nrow = 3)
length(X); y <- x
```

```
[1] 9
```

```
X + y
```

```
[,1] [,2] [,3]
[1,]    2     8    14
[2,]    4    10    16
[3,]    6    12    18
```

```
# 길이가 다른 경우
# 1) 행렬 길이보다 큰 경우
y <- c(1:10)
X + y
```

Warning in X + y: 두 객체의 길이가 서로 배수관계에 있지 않습니다

Error in eval(expr, envir, enclos): dims [product 9]가 객체 [10]의 길이와 일치하지 않습니다

```
# 1) 행렬 길이의 약수가 아닌 경우
# y 재사용
y <- c(1:4)
X + y
```

Warning in X + y: 두 객체의 길이가 서로 배수관계에 있지 않습니다

```
[,1] [,2] [,3]
[1,]    2     8    10
[2,]    4     6    12
[3,]    6     8    10
```

- 행렬-벡터 `%*%` 적용 시 벡터는 $n \times 1$ 행렬로 간주하고 행렬 곱 연산 수행(단 \mathbf{X} 와 벡터 \mathbf{y} 의 길이는 같아야 함).

$$\mathbf{X}_{4 \times 3} = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 1 & 1 \\ 1 & 3 & 3 \\ 1 & 4 & 4 \end{bmatrix}, \quad \mathbf{y}_{3 \times 1} = [7, 6, 8]^T$$

$$\mathbf{X}\mathbf{y} = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 1 & 1 \\ 1 & 3 & 3 \\ 1 & 4 & 4 \end{bmatrix} \cdot \begin{bmatrix} 7 \\ 6 \\ 8 \end{bmatrix} = \begin{bmatrix} 27 \\ 21 \\ 49 \\ 63 \end{bmatrix}$$

```
x <- c(1, 1, 1, 1, 2, 1, 3, 4, 1, 1, 3, 4)
y <- c(7, 6, 8)
X <- matrix(x, nrow = 4, ncol = 3)
X %*% y
```

```
[,1]
[1,] 27
[2,] 21
[3,] 49
[4,] 63
```

행렬의 전치 (transpose)

- 전치 행렬(transpose matrix)는 임의의 행렬의 행과 열을 서로 맞바꾼 행렬임
- 행렬 \mathbf{X} 의 전치 행렬은 \mathbf{X}^T 또는 \mathbf{X}' 으로 나타냄
- 행렬 \mathbf{X} 가 다음과 같이 주어졌을 때 전치 행렬 결과

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad \mathbf{X}^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

- R에서 행렬을 전치시키는 함수는 `t()` 입

```
# t(object_name): 전치행렬 반환
x <- 1:6
X <- matrix(x, nrow = 2, ncol = 3, byrow = T)
t(X)
```

```
[,1] [,2]
[1,] 1 4
[2,] 2 5
[3,] 3 6
```

```
# 전치행렬과 행렬 간 곱
x <- c(1, 1, 1, 1, 22.3, 23.2, 21.5, 25.3, 28.0)
```

```
X <- matrix(x, nrow = 5)
t(X) %*% X
```

```
[,1]      [,2]
[1,]    5.0   120.30
[2,]  120.3  2921.87
```

- 벡터-벡터 곱 연산 (%*% 사용)

$$\mathbf{x} = [1, 2, 3, 4]^T$$

$$\mathbf{x}\mathbf{x}^T = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 6 & 8 \\ 3 & 6 & 9 & 12 \\ 4 & 8 & 12 & 16 \end{bmatrix}$$

$$\mathbf{x}^T \mathbf{x} = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} = 1 + 4 + 9 + 16 = 30$$

```
x <- 1:4
x %*% t(x) # 행렬 반환
```

```
[,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    2    4    6    8
[3,]    3    6    9   12
[4,]    4    8   12   16
```

```
t(x) %*% x # 스칼라 반환 x %*% x와 동일 결과 출력
```

[,1]

[1,] 30

 참고: 전치행렬의 성질(통계수학 II 강의내용 참고)

- $(\mathbf{X}^T)^T = \mathbf{X}$
- $(\mathbf{X} + \mathbf{Y})^T = \mathbf{X}^T + \mathbf{Y}^T$
- $(\mathbf{XY})^T = \mathbf{Y}^T \mathbf{X}^T$
- $(c\mathbf{X})^T = c\mathbf{X}^T$, c 는 임의의 상수

역행렬 (inverse matrix)

- 행렬의 나눗셈 형태
- 행렬 \mathbf{X} 가 $n \times n$ 정방행렬 (square matrix) 일 때, 아래를 만족하는 행렬 $\mathbf{Y}_{n \times n}$ 가 존재하면 \mathbf{Y} 를 \mathbf{X} 의 역행렬 (inverse matrix) 라고 하고 \mathbf{X}^{-1} 로 나타냄.

$$\mathbf{XX}^{-1} = \mathbf{X}^{-1}\mathbf{X} = \mathbf{I}_{n \times n}$$

- 여기서 $\mathbf{I}_{n \times n}$ 은 대각 원소가 1이고 나머지 원소는 0인 항등 행렬임
- 2×2 행렬의 역행렬은 아래와 같이 구함 (3×3 이상 역행렬 구하는 방법은 통계수학 II 강의 참고)

$$\mathbf{X} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \quad \mathbf{X}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

- R에서 정방 행렬의 역행렬은 `solve()` 함수를 사용해 구함

```
# 2 by 2 행렬의 역행렬
x <- c(1, 2, 3, 4)
X <- matrix(x, 2)
solve(X)
```

```
[,1] [,2]
[1,] -2 1.5
[2,] 1 -0.5
```

```
# 항등 행렬이 나오는지 확인
X %*% solve(X)
```

```
[,1] [,2]
[1,] 1 0
[2,] 0 1
```

 참고: 역행렬의 성질(통계수학 II 강의내용 참고)

- $(\mathbf{X}^{-1})^{-1} = \mathbf{X}$
- $(\mathbf{X}^T)^{-1} = (\mathbf{X}^{-1})^T$
- $(\mathbf{XY})^{-1} = \mathbf{Y}^{-1}\mathbf{X}^{-1}$

행렬식 (determinant)

- 행렬의 성질을 대표할 수 있는 하나의 값으로 $n \times n$ 정방행렬 (square matrix)에서 정의
- 역행렬을 구할 때 임의의 행렬이 0, 즉 위 2×2 행렬에서 $ad - bc$ 의 값이 0이라면 역행렬이 존재할 수 없는데 여기서 $ad - bc$ 가 2×2 행렬의 정방행렬임
- 임의의 정방행렬 \mathbf{X} 의 행렬식은 $|\mathbf{X}|$ 또는 $\det(\mathbf{X})$ 로 표시함
- 2×2 행렬의 행렬식은 넓이, 3×3 이상인 정방 행렬에서는 부피의 개념으로 이해할 수 있음

- 정방행렬 $\mathbf{X}_{n \times n} = \{x_{ij}\}$ 가 주어졌을 때, i 번째 행과 j 번째 열을 제외한 나머지 $(n-1) \times (n-1)$ 정방행렬의 행렬식을 $|\mathbf{X}_{ij}|$ 라고 하면 이를 x_{ij} 의 소행렬식(minor)이라 부르고 x_{ij} 의 여인수(co-factor) C_{ij} 는 아래와 같이 정의됨

$$c_{ij} = (-1)^{i+j} |\mathbf{X}_{ij}|$$

- 이때 $\mathbf{X}_{n \times n}$ 행렬식은 임의의 i 또는 j 에 대해 아래의 식을 통해 구할 수 있음

$$\det(\mathbf{X}) = \sum_{i=1}^n x_{ij} c_{ij} = \sum_{j=1}^n x_{ij} c_{ij}$$

- 행렬식 계산 예시

$$\mathbf{X} = \begin{bmatrix} 1 & 5 & 0 \\ 2 & 4 & -1 \\ 0 & -2 & 0 \end{bmatrix}$$

$$\det(\mathbf{X}) = x_{11} \det(\mathbf{X}_{11}) - x_{12} \det(\mathbf{X}_{12}) + x_{13} \det(\mathbf{X}_{13})$$

$$= 1 \begin{vmatrix} 4 & -1 \\ -2 & 0 \end{vmatrix} - 5 \begin{vmatrix} 2 & -1 \\ 0 & 0 \end{vmatrix} + 0 \begin{vmatrix} 2 & 4 \\ 0 & -2 \end{vmatrix} = -2$$

- R에서 임의 행렬의 행렬식은 `det()` 함수를 이용해 구함

```
X <- matrix(c(1, 2, 0, 5, 4, -2, 0, -1, 0), ncol = 3)
det(X)
```

[1] -2

 참고: 행렬식의 성질(통계수학 II 강의내용 참고)

- 행렬 \mathbf{X}, \mathbf{Y} 가 정방행렬이면 $\det(\mathbf{XY}) = \det(\mathbf{X})\det(\mathbf{Y})$
- $\det(\mathbf{X}) = \det(\mathbf{X}^T)$
- $\det(c\mathbf{X}) = c^n \det(\mathbf{X})$ 여기서 c 는 임의의 상수
- $\det(\mathbf{X}^{-1}) = \det(\mathbf{X})^{-1}$

그외 정칙(non-singular), 비정칙(non-singular), 양정치(positive definite) 행렬 모두 행렬식으로 정의할 수 있고 자세한 내용은 통계수학 II를 통해 학습. 추가적으로 여인수 c_{ij} 를 이용한 역행렬 공식은 아래와 같음

$$\mathbf{X}^{-1} = \frac{1}{\det(\mathbf{X})} \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix}$$



예습: 3×3 정방행렬 \mathbf{X} 가 아래와 같이 주어졌을 때, \mathbf{X} 의 행렬식과 역행렬 \mathbf{X}^{-1} 을 직접 계산해 보고, R에서 각각을 구하는 함수를 사용하여 계산 결과가 맞는지 확인

$$\mathbf{X} = \begin{bmatrix} 6 & 1 & 4 \\ 2 & 5 & 3 \\ 1 & 1 & 2 \end{bmatrix}$$

2.4.2 행렬의 색인

- R의 행렬 객체 내 데이터 접근은 벡터와 유사하게 행과 열에 대응하는 색인 또는 이름으로 접근 가능
- 행렬의 행과 열은 괹쇠 '[']' 안에서 ,(콤마)로 구분
- $X[\text{idx_row}, \text{idx_col}]$: 행렬 X의 idx_row 행, idx_col 열에 저장된 값 반환(색인번호는 1부터 시작)
- idx_row, idx_col을 지정하지 않으면 전체 행 또는 열을 선택

```
x <- 1:12
X <- matrix(x, ncol = 4)
X
```

```
[,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

```
# 1행만 선택
X[1, ]
```

```
[1] 1 4 7 10
```

```
# 3열만 선택
X[, 3]
```

```
[1] 7 8 9
```

```
# 1:3행만 선택
X[1:3, ]
```

```
[,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

```
# 1-2행, 3-4열 선택
X[1:2, 3:4]
```

```
[,1] [,2]
[1,]    7   10
[2,]    8   11
```

- 행렬의 각 행과 열에 이름 부여 가능 → `matrix()` 함수 인수 중 `dimnames`에 속성 부여와 동일
- `dimnames()` 함수를 통해 각 행과 열의 이름 확인 및 부여 가능
- `dimnames(object)[[i]]`, $i = 1, 2$ 를 통해 행($i = 1$)과 열($i = 2$) 이름 변경 및 부여 가능
- 위와 유사한 기능을 하는 함수
 - `rownames()`: 행 이름 반환 및 부여
 - `colnames()`: 열 이름 반환 및 부여

```
# matrix 함수 내에서 행렬 이름 동시 부여
X <- matrix(1:9, ncol = 3,
             dimnames = list(c("1", "2", "3"), # 행 이름
                             c("A", "B", "C"))))# 열 이름
X
```

```
A B C
1 1 4 7
2 2 5 8
3 3 6 9
```

```
# dimnames()를 이용한 이름 확인
dimnames(X) # 행렬에 대한 리스트 반환
```

```
[[1]]
[1] "1" "2" "3"
```

```
[[2]]
[1] "A" "B" "C"
```

```
# dimnames() 함수로 행 이름 변경
dimnames(X)[[1]] <- c("r1", "r2", "r3")
```

```
# dimnames() 함수로 열 이름 변경
dimnames(X)[[2]] <- c("c1", "c2", "c3")
dimnames(X)
```

```
[[1]]
[1] "r1" "r2" "r3"
```

```
[[2]]
[1] "c1" "c2" "c3"
```

```
X
```

| | c1 | c2 | c3 |
|----|----|----|----|
| r1 | 1 | 4 | 7 |
| r2 | 2 | 5 | 8 |
| r3 | 3 | 6 | 9 |

```
# rownames()를 통해 행 이름 확인
rownames(X)
```

```
[1] "r1" "r2" "r3"
```

```
# colnames()를 통해 열 이름 확인
colnames(X)
```

```
[1] "c1" "c2" "c3"
```

```
# rownames()를 이용해 행 이름 변경
rownames(X) <- c("apple", "strawberry", "orange")
rownames(X)
```

```
[1] "apple"      "strawberry" "orange"
```

```
# colnames()를 이용해 행 이름 변경
colnames(X) <- c("costco", "emart", "homeplus")
colnames(X)
```

```
[1] "costco"   "emart"    "homeplus"
```

```
X
```

| | costco | emart | homeplus |
|------------|--------|-------|----------|
| apple | 1 | 4 | 7 |
| strawberry | 2 | 5 | 8 |
| orange | 3 | 6 | 9 |

- 행과 열에 대한 이름이 존재한다면 벡터와 마찬가지로 이름으로 색인 가능

```
X[c("apple", "orange"), c("emart")]
```

```
apple orange
4       6
```

```
# 2번째 열에 해당(emart)를 제외한 나머지 열 반환
X[, colnames(X)[-2]]
```

| | costco | homeplus |
|------------|--------|----------|
| apple | 1 | 7 |
| strawberry | 2 | 8 |
| orange | 3 | 9 |

- 색인한 행렬 원소에 다른 값 할당

```
y <- c(1:12); Y <- matrix(y, ncol = 3)
Y
```

```
[,1] [,2] [,3]
[1,]    1    5    9
```

```
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
```

```
# 2, 4 행과 2-3열에 다른 값 할당
Y[c(2, 4), 2:3] <- matrix(c(1, 2, 1, 4), ncol = 2)

# 행렬 값 할당 다른 예시
X <- matrix(nrow = 4, ncol = 3) # NA 값으로 구성된 4 by 3 행렬
X
```

```
[,1] [,2] [,3]
[1,]    NA    NA    NA
[2,]    NA    NA    NA
[3,]    NA    NA    NA
[4,]    NA    NA    NA
```

```
y <- c(1, 0, 0, 1); Y <- matrix(y, ncol = 2)
X[3:4, 2:3] <- Y
X
```

```
[,1] [,2] [,3]
[1,]    NA    NA    NA
[2,]    NA    NA    NA
[3,]    NA     1     0
[4,]    NA     0     1
```

- 행렬 필터링 → 색인 대신 조건 사용(벡터와 동일)

```
X = matrix(c(1,2,4,3,2,3,5,6), nrow = 4, ncol = 2)

# X의 1열이 3보다 작거나 같은 행 필터링
X[X[,1] <= 3, ]
```

```
[,1] [,2]
[1,]    1    2
[2,]    2    3
[3,]    3    6

# 논리값을 활용한 필터링
idx <- X[, 1] <= 3; idx

[1] TRUE TRUE FALSE TRUE
```

```
X[idx, ]

[,1] [,2]
[1,]    1    2
[2,]    2    3
[3,]    3    6
```

2.4.3 행과 열 추가 및 제거

- 행렬 재할당 (re-assignment)를 통해 열이나 행을 직접 추가하거나 삭제 가능
- `cbind()` (열 붙이기), `column bind`, `rbind()` (행 붙이기), `row bind` 함수 사용

```
j <- rep(1, 4)
Z <- matrix(c(1:4, 1, 1, 0, 0, 1, 0, 1, 0), nrow = 4, ncol = 3)
Z
```

```
[,1] [,2] [,3]
[1,]    1    1    1
[2,]    2    1    0
[3,]    3    0    1
[4,]    4    0    0
```

```
cbind(j, Z) # 열 기준으로 붙이기
```

```
j
[1,] 1 1 1 1
[2,] 1 2 1 0
[3,] 1 3 0 1
[4,] 1 4 0 0
```

```
# 길이가 다른 경우 재사용
cbind(1, Z)
```

```
[,1] [,2] [,3] [,4]
[1,] 1 1 1 1
[2,] 1 2 1 0
[3,] 1 3 0 1
[4,] 1 4 0 0
```

```
# Z 행렬 앞에 j 열 붙혀서 새로운 Z 생성
Z <- cbind(j, Z)
```

```
# 행 기준으로 붙이기
Z <- rbind(Z, 2)
```

- 행 또는 열의 제거는 벡터에서와 마찬가지로 색인 앞에 - 사용

```
# 첫 번째 행 제거
Z[-1, ]
```

```
j
[1,] 1 2 1 0
[2,] 1 3 0 1
[3,] 1 4 0 0
[4,] 2 2 2 2
```

```
# 1, 5행, 3열 제거
Z[-c(1, 5), -3]
```

```
j
[1,] 1 2 0
[2,] 1 3 1
[3,] 1 4 0
```



`cbind()` 또는 `rbind()` 함수는 다음 주에 배울 데이터 프레임에도 적용 가능하다.

2.4.4 행렬 관련 함수

- `diag()`: 대각행렬 생성 또는 대각원소(diagonal elements) 추출
- 대각행렬: 주 대각선을 제외한 모든 원소가 0인 $n \times n$ 정방행렬로 다음과 같이 정의

$$\mathbf{D} = \{d_{ij}\}, \quad i, j \in \{1, 2, \dots, n\}, \quad \forall i \neq j \rightarrow d_{ij} = 0$$

```
D <- diag(c(1:5), 5)
D
```

```
[,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    0
[2,]    0    2    0    0    0
[3,]    0    0    3    0    0
[4,]    0    0    0    4    0
[5,]    0    0    0    0    5
```

```
# 3차원 항등 행렬(모든 대각원소가 1인 행렬)
I3 <- diag(1, 3)
```

```
# 대각원소 추출
```

```
diag(D)
```

```
[1] 1 2 3 4 5
```

```
# 대각원소 재할당
```

```
diag(D) <- rep(1, 5)
```



객체는 속성(attribute)을 갖고 그 속성에 따라 데이터의 구조가 정해짐. 즉 속성은 데이터에 대한 메타 데이터임. 객체의 속성은 대표적으로 이름(names), 차원(dimension), 클래스(class)로 정의되고 객체에 대한 자세한 정보를 파악하기 위해 제공되는 몇 가지 함수들에 대해 알아봄.

R은 앞서 언급한 바와 같이 객체지향언어(object oriented program, OOP)이고 세 가지 유형의 객체지향 시스템(S3, S4, S5)이 존재함. R의 핵심적인 함수 및 패키지는 S3 객체 시스템을 사용하고 있기 때문에 알아둘 필요가 있으나 본 강의의 범위를 벗어나기 때문에 이번 학기에는 다루지 않을 것임.

- `dim(object_name)`: 행렬 또는 데이터 프레임의 행과 열의 개수(차원)를 반환

```
# dim(): 객체의 차원 (dimension)을 반환
```

```
Z
```

```
j  
[1,] 1 1 1 1  
[2,] 1 2 1 0  
[3,] 1 3 0 1  
[4,] 1 4 0 0  
[5,] 2 2 2 2
```

```
dim(Z)
```

```
[1] 5 4
```

- `nrow()` 또는 `NROW()`: 행렬의 행 길이 반환
- `ncol()` 또는 `NCOL()`: 행렬의 행 길이 반환

```
nrow(Z); ncol(Z)
```

```
[1] 5
```

```
[1] 4
```

`nrow()`/`ncol()`과 `NROW()`/`NCOL()`의 차이점

- `nrow()`/`ncol()`은 행렬 또는 데이터 프레임에 적용되며 벡터가 인수로 사용될 때 `NULL` 값을 반환하는데 비해 `NROW()`/`NCOL()`은 벡터의 길이도 반환 가능

- `attributes()`: 객체가 갖는 속성을 반환함

```
x <- 1:9; X <- matrix(x, ncol = 3)  
# 객체의 속성 확인  
attributes(x)
```

```
NULL
```

```
attributes(X)
```

```
$dim
```

```
[1] 3 3
```

- `class()`: 객체의 클래스 명칭 반환 및 클래스 부여

```
# 객체의 class 확인  
class(x); class(X)
```

```
[1] "integer"
```

```
[1] "matrix"
```

```
# 객체의 class 부여
class(x) <- "this is a vector"
```

- `str()`: 객체가 갖고 있는 데이터의 구조 확인

```
# 객체의 구조 파악
str(x); str(X)
```

```
'this is a vector' int [1:9] 1 2 3 4 5 6 7 8 9
int [1:3, 1:3] 1 2 3 4 5 6 7 8 9
```

```
# x와 X에 이름 (name) 속성을 추가한 경우
names(x) <- paste0("x", 1:9)
dimnames(X) <- list(paste0("r", 1:3),
                      paste0("c", 1:3))
attributes(x); attributes(X)
```

```
$class
[1] "this is a vector"
```

```
$names
[1] "x1" "x2" "x3" "x4" "x5" "x6" "x7" "x8" "x9"
```

```
$dim
[1] 3 3
```

```
$dimnames
$dimnames[[1]]
[1] "r1" "r2" "r3"
```

```
$dimnames[[2]]
[1] "c1" "c2" "c3"
```

```
class(x); class(X)
```

```
[1] "this is a vector"
```

```
[1] "matrix"
```

```
str(x); str(X)
```

```
'this is a vector' Named int [1:9] 1 2 3 4 5 6 7 8 9  
- attr(*, "names")= chr [1:9] "x1" "x2" "x3" "x4" ...  
  
int [1:3, 1:3] 1 2 3 4 5 6 7 8 9  
- attr(*, "dimnames")=List of 2  
..$ : chr [1:3] "r1" "r2" "r3"  
..$ : chr [1:3] "c1" "c2" "c3"
```

- `attr(object, "attribute_name")`: 객체가 갖고 있는 속성을 지정해서 확인

```
# 객체 속성 요소 확인  
attr(x, "names")
```

```
[1] "x1" "x2" "x3" "x4" "x5" "x6" "x7" "x8" "x9"
```

```
attr(X, "dimnames")
```

```
[[1]]
```

```
[1] "r1" "r2" "r3"
```

```
[[2]]
```

```
[1] "c1" "c2" "c3"
```

2.4.5 벡터와 행렬의 차이점

- 행렬은 개념적으로 $n \times 1$ 벡터가 2 개 이상 묶여져서 행과 열의 속성을 갖지만 기본적으로는 벡터

```
z <- 1:8  
U <- matrix(z, 4, 2)  
length(z) # 입력 벡터 원소의 길이가 8
```

```
[1] 8
```

- R에서 U가 행렬임을 나타내기 위해 추가적인 속성(attribute)를 부여

```
class(z) # 벡터
```

```
[1] "integer"
```

```
attributes(z)
```

```
NULL
```

```
class(U) # 행렬
```

```
[1] "matrix"
```

```
attributes(U)
```

```
$dim
```

```
[1] 4 2
```

2.4.6 의도치 않은 차원축소 피하기

- 다음 행렬에서 한 행을 추출

```
Z <- matrix(c(1:8), 4, 2)
z <- Z[2, ]

attributes(Z) # 행과 열의 차원 수를 표시
```

```
$dim
```

```
[1] 4 2
```

```
# 객체 z의 속성 및 형태는?
```

```
attributes(z) # 차원이 존재하지 않음
```

```
NULL
```

- 차원축소를 방지하는 방법 → r을 벡터가 아닌 1×2 행렬로 인식

```
z <- Z[2, , drop = FALSE]
attributes(z)
```

```
$dim
```

```
[1] 1 2
```

- `as.matrix()`를 이용한 직접 변환

```
z <- as.matrix(Z[2, ])
class(z)
```

```
[1] "matrix"
```

```
z # 행렬이 변환됨을 유의
```

```
[,1]
[1,]    2
[2,]    6
```

2.5 배열 (array)

- 통계학의 관점에서 R의 행렬의 행은 조사 대상이 되는 사람, 동물 등 관측 대상에 해당하고, 열은 대상의 특성을 표현하는 변수(예: 몸무게, 키, 혈압 등)에 해당 → 2차원 구조
- 위와 같은 데이터를 네 단위로 수집한다면? → 한 대상자에 해당하는 변수들은 시간에 따라 변함 → 시간 차원이 하나 더 존재!
- R에서 이러한 형태의 데이터 구조를 배열 (array)이라고 지칭함

2.5.1 배열의 생성 및 색인

- 동일한 유형의 데이터가 2차원 이상으로 구성된 데이터 구조
- 동일한 차원 ($n \times p$)의 배열(행렬)이 k 개 방에 저장된 데이터 구조
- 배열 생성 함수

```
# array() 함수 인수 구조
array(data, # 저장할 데이터 벡터 또는 행렬
      dim, # 배열의 차원 지정
      dimnames # 배열 차원 명칭
      )
```

- 통계학과 3명의 학생에 대한 중간고사 기준 한 번의 퀴즈와 중간고사 점수, 그리고 기말고사 기준 한 번의 퀴즈와 기말고사 점수 데이터 가정

```
x <- c(75, 84, 93, 65, 78, 92)
y <- c(82, 78, 85, 88, 75, 88)

first_term <- matrix(x, nrow = 3, ncol = 2)
second_term <- matrix(y, nrow = 3, ncol = 2)
```

```
first_term
```

```
[,1] [,2]  
[1,] 75 65  
[2,] 84 78  
[3,] 93 92
```

```
second_term
```

```
[,1] [,2]  
[1,] 82 88  
[2,] 78 75  
[3,] 85 88
```

```
# 위 두 데이터를 2층 짜리 배열로 구성
```

```
Z <- array(data = c(first_term, second_term),  
            dim = c(3, 2, 2))  
Z
```

```
, , 1
```

```
[,1] [,2]  
[1,] 75 65  
[2,] 84 78  
[3,] 93 92
```

```
, , 2
```

```
[,1] [,2]  
[1,] 82 88  
[2,] 78 75  
[3,] 85 88
```

```
# Z의 속성
attributes(Z)
```

```
$dim
[1] 3 2 2
```

```
# Z의 클래스
class(Z)
```

```
[1] "array"
```

```
# Z의 구조
str(Z)
```

```
num [1:3, 1:2, 1:2] 75 84 93 65 78 92 82 78 85 88 ...
```

- 배열 내 데이터 접근은 색인을 통해 가능(벡터 행렬과 동일)

```
# 첫 번째 층만 추출
Z[, , 1]
```

```
[,1] [,2]
[1,]    75    65
[2,]    84    78
[3,]    93    92
```

```
# 두 번째 층에서 2-3행 만 추출
Z[2:3, , 2]
```

```
[,1] [,2]
[1,]    78    75
[2,]    85    88
```

2.5.2 배열의 확장 예제

데이터 사이언스 스쿨^{L2} 참고

- 배열 구조를 갖는 가장 대표적인 데이터 중 하나가 이미지(사진)
- 이미지 데이터는 픽셀(pixel)이라는 세분화된 작은 이미지를 직사각형 형태로 모은 형태
- 전체 이미지는 세로픽셀수 × 가로픽셀수로 표현됨 → 행렬
- 픽셀의 색을 숫자로 표현하는 방식을 색공간(color space)라고 명칭
- 대표적 색공간은 흑백스케일(grey scale), RGB (Red-Green-Blue), HSV(Hue-Saturation-Value) 방식
- RGB 색공간을 사용한 경우 각 색공간별로 동일한 크기의 행렬이 3개 층으로 저장된 상태 → 배열
- RGB는 0 ~ 255 까지 값을 갖고 빨강색 (255, 0, 0), 녹색 (0, 255, 0), 파란색은 (0, 0, 255)임

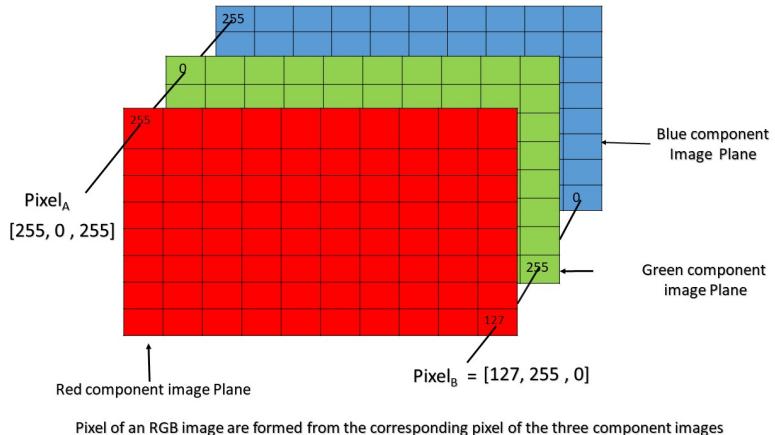


FIGURE 2.2: <https://www.geeksforgeeks.org/matlab-rgb-image-representation/>에서 발췌

 목표

- R에서 웹 url로 이미지를 불러오기
- 불러온 이미지를 R에서 plotting 해보기
- 이미지 데이터를 직접 수정 해보기

1. 이미지 입출력 패키지 installation

```
install.packages("jpeg") # jpeg 파일 입출력 관련 package  
install.packages("cowplot") # ggplot add-on package
```

2. 관련 패키지 불러오기

```
require(tidyverse)  
require(jpeg)  
require(cowplot)
```

3. 이미지 불러오기

```
myurl <- "https://img.livescore.co.kr/data/editor/1906/ba517de8162d92f4ea0e9de0ec98ba02.jpg"  
z <- tempfile()  
download.file(myurl,z,mode="wb")  
pic <- readJPEG(z)
```

4. 이미지 그래프 출력창에서 확인



5. 이미지 임의 부분 편집하기

```
pic[300:460, 440:520, 1] <- 0.5  
pic[300:460, 440:520, 2] <- 0.5  
pic[300:460, 440:520, 3] <- 0.5  
  
ggdraw() +  
  draw_image(pic)
```



6. RGB값을 무작위로 샘플링 후 매개변수로 노이즈 가중치 조절해 보기

```

pic <- readJPEG(z)
yr <- pic[300:460, 440:520, 1]
yg <- pic[300:460, 440:520, 2]
yb <- pic[300:460, 440:520, 3]
n <- nrow(yr); p <- ncol(yr)

t <- 0.2
wr <- t * yr + (1 - t)*matrix(runif(length(yr)), nrow = n, ncol = p)
wg <- t * yg + (1 - t)*matrix(runif(length(yg)), nrow = n, ncol = p)
wb <- t * yb + (1 - t)*matrix(runif(length(yb)), nrow = n, ncol = p)

pic[300:460, 440:520, 1] <- wr
pic[300:460, 440:520, 2] <- wg
pic[300:460, 440:520, 3] <- wb
  
```

```
ggdraw() +  
  draw_image(pic)
```



2.6 요인 (factor)과 테이블 (table)

- 요인 (factor) 데이터 타입은 통계학에서 범주형 변수 (categorical variable)을 표현하기 위한 R의 데이터 타입으로 범주형 자료는 크게 명목형 (nominal)과 순서형 (ordinal) 으로 구분
- 테이블 (table) 객체는 factor 객체에 대한 빈도를 나타내기 위해 사용

범주형 자료

- 데이터가 사전에 정해진 특정 유형으로만 분류되는 경우: 성별, 인종, 혈액형 등

- 범주형 자료는 명목형과 순서형으로 구분 가능
- 순서형 자료 예: 성적, 교육수준, 선호도, 중증도 등

2.6.1 요인 (factor)

- 범주형 자료를 표현하기 위한 R의 객체 클래스
- Factor는 정수형 벡터를 기반으로 levels (수준)이라는 속성이 추가된 객체임
- 숫자 또는 문자로 표현되었다 하더라도 범주형으로 이해
- Factor는 level에 해당하는 값만 가질 수 있는 벡터로 간주
- Factor 생성 함수

```
# factor 정의 함수
factor(data, # factor로 표현하고자 하는 값. 주로 문자형
       levels, # 요인의 수준, 미리 정한 값
       labels, # 수준에 대한 레이블링
       ordered # 순서형 자료 표시 여부
       # TRUE/FALSE, default = FALSE
       )
```

- 수치형을 factor로 만들어도 처음 입력 값은 문자형으로 변하고 level 값으로 치환
- 대신 (1, 2, 3)이 중심값이 됨 → 정수형 벡터임

```
score <- rep(c(4:6), each = 4)
fscore <- factor(score)

typeof(fscore) # factor의 기본 데이터 타입
```

[1] "integer"

```
attributes(fscore) # factor의 속성
```

```
$levels  
[1] "4" "5" "6"
```

```
$class  
[1] "factor"
```

```
# factor의 구조  
str(fscore)
```

```
Factor w/ 3 levels "4","5","6": 1 1 1 1 2 2 2 2 3 3 ...
```

```
# levels(): factor의 수준 (levels) 반환 함수  
levels(fscore)
```

```
[1] "4" "5" "6"
```

```
# nlevels(): level의 개수 반환  
nlevels(fscore)
```

```
[1] 3
```

- Factor를 벡터 결합 함수 `c()`로 결합

```
c(fscore, factor(4)) # 강제로 정수형 벡터로 변환
```

```
[1] 1 1 1 1 2 2 2 2 3 3 3 3 1
```

- Factor의 범주 수준 (level) 및 범주명 (label) 지정

```
x <- rep(c(1:2), each = 4)  
  
# factor의 범주 수준 지정
```

```
sex <- factor(x, levels = 1:2)
sex
```

```
[1] 1 1 1 1 2 2 2 2
Levels: 1 2
```

```
# factor의 범주 수준 및 범주 명칭 지정
sex <- factor(x, levels = 1:2, labels = c("male", "female"))
sex # level의 값이 명칭으로 변경
```

```
[1] male male male male female female female female
Levels: male female
```

```
str(sex)
```

```
Factor w/ 2 levels "male","female": 1 1 1 1 2 2 2 2
```

```
# 값은 존재하지 않으나 수준을 미리 정해 놓은 경우
severity <- factor(1:2, levels = c(1, 2, 3), labels = c("Mild", "Moderate", "Severe"))
severity[2] <- "Severe"

# 존재하지 않는 수준 할당
severity[1] <- "Good"
```

Warning in `<- .factor`(`*tmp*`, 1, value = "Good"): 요인의 수준
 (factor level)이 올바르지 않아 NA가 생성되었습니다.

```
severity
```

```
[1] <NA> Severe
Levels: Mild Moderate Severe
```

- 순서형 factor 생성

```

severity <- factor(rep(1:3, times = 3), levels = 1:3,
                     labels = c("Mild", "Moderate", "Severe"),
                     ordered = T)
severity

[1] Mild      Moderate Severe    Mild      Moderate Severe    Mild      Moderate
[9] Severe

Levels: Mild < Moderate < Severe

```

```
is.ordered(severity) # 순서형 범주 체크
```

```
[1] TRUE
```

요인형 객체에 적용되는 일반적인 함수

tapply() 함수

- 특정 요인 수준의 고유한 조합으로 각 그룹에 속한 값에 특정 함수를 적용한 결과를 반환
- 일반적인 함수 사용 형태는 아래와 같음

```

# tapply() 함수 사용 인수
tapply(
  x, # 벡터,
  INDEX, # 벡터를 그룹화할 색인 (factor)
  FUN, # 각 그룹마다 적용할 함수
)

```

- 예시: 2020년 4월 15일 총선의 연령별 지지율

```

# 문자열을 INDEX의 인수로 받은 경우

x <- c(48, 43, 27, 52, 38,
```

```

67, 23, 58, 72, 85) # 유권자 연령
f <- rep(c("더불어민주당", "미래통합당"), each = 5)
t <- tapply(x, f, mean) # f의 요인 수준 별 x (연령) 평균 계산
t

```

더불어민주당 미래통합당

41.6 61.0

```

# x, f 순서를 랜덤하게 섞은 다음 결과
set.seed(12345) # 난수 생성 결과 고정
idx <- order(runif(10))
x <- x[idx]
f <- f[idx]

tapply(x, f, mean)

```

더불어민주당 미래통합당

41.6 61.0

- Factor가 2개 이상인 경우 두 factor 객체의 수준의 조합(AND 조건)에 따른 그룹을 만든 후 그룹별 함수 적용

```

s <- rep(c("M", "F"), each = 6)
income <- c(35, 42, 68, 29, 85, 55,
          30, 40, 63, 27, 83, 52) * 100 # 단위: 만원
age <- c(32, 36, 44, 25, 55, 41,
        28, 33, 46, 23, 54, 44)

set.seed(12345) # 난수 생성 결과 고정
idx <- order(runif(12))
s <- s[idx]; income <- income[idx]; age <- age[idx]

# age <= 40 -> 1, 40 < age <= 50 -> 2,

```

```
# age >= 50 -> 3 할당: ifelse() 함수 사용
age <- ifelse(age <= 40, 1,
              ifelse(age <= 50, 2, 3))

tapply(income, list(sex = s, age = age), mean)
```

| | age | | |
|-----|----------|------|------|
| sex | 1 | 2 | 3 |
| F | 3233.333 | 5750 | 8300 |
| M | 3533.333 | 6150 | 8500 |



R에서 가장 많이 활용되는 함수 계열 중 하나로 *apply()를 들 수 있다. 벡터, 행렬 등과 같은 R 객체에 for loop 대신 반복적으로 동일한 함수를 적용할 때 활용된다. *apply() 계열 함수에 대해서는 데이터 프레임에서 더 상세하게 배울 것임

2.6.1.0.1 *split()* 함수

- tapply()는 주어진 요인의 수준에 따라 특정 함수를 적용하지만, split()은 데이터를 요인의 수준(그룹) 별로 데이터를 나누어 리스트 형태로 반환

```
# split() 함수 사용 인수
split(
  x, # 분리할 데이터(벡터)
  f, # 데이터를 분리할 기준이 되는 factor 지정
)
```

- split() 함수 사용 예시

```
# 성별의 수준 남녀 별 소득 수준 분리
split(income, s)
```

```
$F
[1] 8300 5200 3000 4000 6300 2700
```

```
$M
```

```
[1] 5500 8500 3500 6800 4200 2900
```

```
# 두 개 요인 조합으로 income 벡터 분리
split(income, list(s, age))
```

```
$F.1
```

```
[1] 3000 4000 2700
```

```
$M.1
```

```
[1] 3500 4200 2900
```

```
$F.2
```

```
[1] 5200 6300
```

```
$M.2
```

```
[1] 5500 6800
```

```
$F.3
```

```
[1] 8300
```

```
$M.3
```

```
[1] 8500
```

```
# 요인의 각 수준에 대한 인덱스를 반환하고자 하는 경우
```

```
abalone <- read.csv("http://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data",
header = FALSE) # 전복 데이터셋
# V1: 전복의 종류
# F=암컷; M=수컷, I=새끼
g <- abalone[, 1] # 전복종류만 추출
set.seed(20200410)
```

```
idx <- sample(1:length(g), size = 10)
g <- g[idx]
split(1:length(g), g)
```

\$F
[1] 1 6 8

\$I
[1] 2 3 5 7

\$M
[1] 4 9 10

2.6.2 테이블(table)

- 범주형 변수의 빈도 또는 분할표(교차표)를 표현하기 위한 객체(클래스)
- 범주 별 통계량(평균, 표준편차, 중위수, …) 요약

tapply() 함수를 이용한 테이블 만들기

- 길이가 12인 임의의 벡터 u를 수준의 개수가 각각 3, 2인 factor의 조합으로 부분벡터로 분리 후 length() 적용 → tapply() 함수 사용

```
u <- runif(12)
f1 <- factor(c(4, 4, 3, 5, 5, 4,
              3, 3, 4, 5, 5, 3))
f2 <- factor(c("a", "a", "a", "a", "b", "a",
              "b", "b", "a", "a", "b", "b"))
tapply(u, list(f1, f2), length)
```

a b
3 1 3
4 4 NA

```
5 2 2
```

- u의 값과 상관 없이 두 factor 형 변수 f1과 f2의 조합에 따른 개수 반환 → 분할표(contingency table)
- 위 예시에서 f1이 “4”이고 f2가 “b”인 경우는 없기 때문에 0 값이 있어야 하나, tapply() 함수 적용 시 결측값 NA를 반환
- table(): 하나 이상의 factor의 수준 또는 수준의 조합으로 분할표 생성
- Factor가 3개 이상인 경우 배열로 다차원 분할표 표현

```
# table() 적용 예시
t1 <- table(f1, f2)
t1
```

```
f2
f1  a b
 3 1 3
 4 4 0
 5 2 2
```

```
typeof(t1); attributes(t1); str(t1)
```

```
[1] "integer"
```

```
$dim
```

```
[1] 3 2
```

```
$dimnames
$dimnames$f1
[1] "3" "4" "5"
```

```
$dimnames$f2
[1] "a" "b"
```

```
$class  
[1] "table"  
  
'table' int [1:3, 1:2] 1 4 2 3 0 2  
- attr(*, "dimnames")=List of 2  
..$ f1: chr [1:3] "3" "4" "5"  
..$ f2: chr [1:2] "a" "b"
```

```
# factor가 한개인 경우  
table(f1)
```

```
f1  
3 4 5  
4 4 4
```

```
# factor가 3개인 경우  
year = c("1", "1", "2", "3", "3", "4")  
gender = c("M", "M", "F", "M", "F", "F")  
grade = c("A", "C", "B", "B", "A", "C")  
  
table(gender, grade, year)
```

```
, , year = 1
```

```
grade  
gender A B C  
F 0 0 0  
M 1 0 1
```

```
, , year = 2
```

```
grade  
gender A B C
```

```
F 0 1 0
M 0 0 0

, , year = 3
```

```
grade
gender A B C
F 1 0 0
M 0 1 0
```

```
, , year = 4
```

```
grade
gender A B C
F 0 0 1
M 0 0 0
```

테이블 관련 함수

2.6.2.0.1 *tabulate()* 함수

- 정수로 이루어진 벡터에 각 정수 값이 발생한 횟수를 카운팅한 결과를 반환
→ *table()* 함수의 핵심 함수

```
# tabulate() 함수 사용 인수(argument)
tabulate(
  bin, # 정수형(수치형) 벡터 또는 factor
  nbins, # 사용할 수준(bin)의 개수
)
```

- tabulate()* 함수 예시

```
x <- c(2, 2, 2, 1, 3, 4, 5, 5, 10, 8, 8)
tabulate(x)
```

```
[1] 1 3 1 1 2 0 0 2 0 1
```

```
tabulate(x, nbins = 3)
```

```
[1] 1 3 1
```

2.6.2.0.2 addmargins() 함수

- 테이블 객체(분할표)를 인수로 받아 각 요인의 수준 및 수준 조합 별 합계 값을 테이블과 동시 반환

```
# addmargins() 함수 사용 인수  
addmargins(  
  T # 테이블 또는 배열 객체  
)
```

- addmargins() 예시

```
t1 <- table(f1, f2)  
addmargins(t1)
```

```
f2  
f1      a   b Sum  
 3      1   3   4  
 4      4   0   4  
 5      2   2   4  
Sum    7   5  12
```

```
# 3차원 이상 테이블  
t2 <- table(gender, grade, year)  
is.table(t2); is.array(t2)
```

```
[1] TRUE
```

```
[1] TRUE
```

```
addmargins(t2)
```

```
, , year = 1
```

```
grade  
gender A B C Sum  
F 0 0 0 0  
M 1 0 1 2  
Sum 1 0 1 2
```

```
, , year = 2
```

```
grade  
gender A B C Sum  
F 0 1 0 1  
M 0 0 0 0  
Sum 0 1 0 1
```

```
, , year = 3
```

```
grade  
gender A B C Sum  
F 1 0 0 1  
M 0 1 0 1  
Sum 1 1 0 2
```

```
, , year = 4
```

```
grade  
gender A B C Sum  
F 0 0 1 1  
M 0 0 0 0
```

```
Sum 0 0 1 1
```

```
, , year = Sum
```

```
grade
gender A B C Sum
F   1 1 1 3
M   1 1 1 3
Sum 2 2 2 6
```

2.6.2.0.3 ftable() 함수

- “평평한(flat)” 교차표 생성
- 다차원 교차표 작성 시 행변수와 열변수 교환을 통해 재사용 가능

```
ftable(
  x, # factor, table 또는 ftable 클래스를 갖는 객체
  row.vars, # 행 변수 지정 색인(정수, 문자)
  col.vars # 열 변수 지정 색인(정수, 문자)
)
```

- ftable() 함수 사용 예시

```
t3 <- ftable(t2)
t3; attributes(t3); str(t3)
```

```
year 1 2 3 4
gender grade
F      A      0 0 1 0
          B      0 1 0 0
          C      0 0 0 1
M      A      1 0 0 0
          B      0 0 1 0
          C      1 0 0 0
```

```
$dim
[1] 6 4

$class
[1] "ftable"

$row.vars
$row.vars$gender
[1] "F" "M"

$row.vars$grade
[1] "A" "B" "C"

$col.vars
$col.vars$year
[1] "1" "2" "3" "4"

'ftable' int [1:6, 1:4] 0 0 0 1 0 1 0 1 0 0 ...
- attr(*, "row.vars")=List of 2
..$ gender: chr [1:2] "F" "M"
..$ grade : chr [1:3] "A" "B" "C"
- attr(*, "col.vars")=List of 1
..$ year: chr [1:4] "1" "2" "3" "4"
```

```
# 테이블 내 행 변수 바꾸기
t4 <- ftable(t2, row.vars = c("year", "gender"))
t4
```

| | | grade | A | B | C |
|------|--------|-------|---|---|---|
| year | gender | | | | |
| 1 | F | | 0 | 0 | 0 |
| | M | | 1 | 0 | 1 |
| 2 | F | | 0 | 1 | 0 |

| | | |
|---|---|-------|
| | M | 0 0 0 |
| 3 | F | 1 0 0 |
| | M | 0 1 0 |
| 4 | F | 0 0 1 |
| | M | 0 0 0 |

```
# 테이블 내 열 변수 바꾸기
t5 <- ftable(t2, col.vars = 1)
t5
```

| | gender | F | M |
|-------|--------|---|---|
| grade | year | | |
| A | 1 | 0 | 1 |
| | 2 | 0 | 0 |
| | 3 | 1 | 0 |
| | 4 | 0 | 0 |
| B | 1 | 0 | 0 |
| | 2 | 1 | 0 |
| | 3 | 0 | 1 |
| | 4 | 0 | 0 |
| C | 1 | 0 | 1 |
| | 2 | 0 | 0 |
| | 3 | 0 | 0 |
| | 4 | 1 | 0 |

2.6.2.0.4 margin.table() 함수

- 배열 형식으로 지정된 교차표 (table() 반환 결과)에서 지정된 차원 색인에 대한 표 합계 계산 결과 반환

```
margin.table(
  x, # table 또는 ftable 클래스를 갖는 객체
```

```
margin # 차원 색인 번호  
)
```

- margin.table() 예시

```
t2
```

```
, , year = 1
```

```
grade  
gender A B C  
F 0 0 0  
M 1 0 1
```

```
, , year = 2
```

```
grade  
gender A B C  
F 0 1 0  
M 0 0 0
```

```
, , year = 3
```

```
grade  
gender A B C  
F 1 0 0  
M 0 1 0
```

```
, , year = 4
```

```
grade  
gender A B C  
F 0 0 1
```

```
M O O O
```

```
margin.table(t2, 1) # 1 차원(행): 성별
```

```
gender
```

```
F M
```

```
3 3
```

```
margin.table(t2, 2) # 2 차원(열): 성적
```

```
grade
```

```
A B C
```

```
2 2 2
```

```
margin.table(t2, 3) # 3 차원(배열 방 번호): 학년
```

```
year
```

```
1 2 3 4
```

```
2 1 2 1
```

2.6.2.0.5 prop.table() 함수

- table 객체 빈도에 대한 비율 계산
- 전체, 차원 단위 비율 계산 가능

```
prop.table(
  x, # table 또는 ftable 클래스를 갖는 객체
  margin # 차원 색인 번호
)
```

- prop.table() 예시

- margin = NULL: 각 셀을 전체 cell의 합으로 나눈 비율
- margin = 1: 각 행 별 셀에 대해 각 행에 해당하는 cell 합으로 나눈 비율 ($n_{ij}/n_{i\cdot}$), $n_{i\cdot} = \sum_{j=1}^J n_{ij}$

– `margin = 2`: 각 열 별 셀에 대해 각 열에 해당하는 cell 합으로 나눈

$$\text{비율 } (n_{ij}/n_{.j}), n_{.j} = \sum_{i=1}^I n_{ij}$$

```
# 2차원 교차표
prop.table(t1) # margin = NULL
```

```
f2
f1      a      b
3 0.08333333 0.25000000
4 0.33333333 0.00000000
5 0.16666667 0.16666667
```

```
prop.table(t1, 1) # margin = 1 (row)
```

```
f2
f1      a      b
3 0.25 0.75
4 1.00 0.00
5 0.50 0.50
```

```
prop.table(t1, 2) # margin = 2 (column)
```

```
f2
f1      a      b
3 0.1428571 0.6000000
4 0.5714286 0.0000000
5 0.2857143 0.4000000
```

TABLE 2.5: 스프레드시트 기본 형태 예시

| 이름 | 직장 | 나이 |
|-----|------|----|
| 김어준 | 딴지일보 | 51 |
| 주진우 | 시사인 | 46 |
| 김용민 | 프리랜서 | 45 |
| 정봉주 | 정당인 | 59 |

2.7 데이터 프레임 (*data frame*)



학습목표(4 주차): 데이터 프레임 클래스에 대해 알아보고, 데이터 프레임을 생성, 병합 (merge), 연산에 대한 함수들에 대해 알아본다.

- Excel 스프레드시트와 같은 형태
- 데이터 프레임은 데이터 유형에 상관없이 2차원 형태의 데이터 구조
- 행렬과 리스트를 혼합한 자료 형태 → 동일한 길이의 벡터로 이루어진 리스트를 구성요소로 갖는 리스트
- 행렬과 유사한 구조를 갖고 있지만 각기 다른 유형의 자료형태로 자료행렬을 구성할 수 있다는 점에서 행렬과 차이를 갖음
- 행렬과 마찬가지로 변수(열)의 길이(행의 개수)는 모두 동일해야 함
- R에서 가장 빈번하게 활용되고 있는 데이터 클래스임
- 데이터 프레임의 각 열(컬럼)은 벡터로 간주

2.7.1 데이터 프레임 생성

- 데이터 프레임 생성 함수: `data.frame()`

```
data.frame(
  # 값 또는 이름(tag) = 값
  ...,
  # 논리값.
  # 변수명(열 이름)이 구문 상 유효한 변수인지 또는 중복이 있는지 확인
  check.names,
  # 논리값. 문자형 벡터의 factor 형 강제 변환 여부
  stringsAsFactors,
)
```

- 데이터 프레임 생성 예시: 모 병원에서 얻은 환자의 인구학적 정보

```
id <- c(1:10)
sex <- rep(c("Female", "Male"), each = 5)
age <- c(34, 22, 54, 43, 44, 39, 38, 28, 31, 42)
sbp <- c(112, 118, 132, 128, 128, 124, 121, 119, 124, 109)
height <- c(165, 158, 161, 160, 168, 172, 175, 182, 168, 162)
weight <- c(52, 48, 59, 60, 48, 72, 73, 82, 64, 60)

df <- data.frame(id, sex, age, sbp, height, weight,
                  stringsAsFactors = FALSE)
df
```

| | id | sex | age | sbp | height | weight |
|---|----|--------|-----|-----|--------|--------|
| 1 | 1 | Female | 34 | 112 | 165 | 52 |
| 2 | 2 | Female | 22 | 118 | 158 | 48 |
| 3 | 3 | Female | 54 | 132 | 161 | 59 |
| 4 | 4 | Female | 43 | 128 | 160 | 60 |
| 5 | 5 | Female | 44 | 128 | 168 | 48 |
| 6 | 6 | Male | 39 | 124 | 172 | 72 |
| 7 | 7 | Male | 38 | 121 | 175 | 73 |
| 8 | 8 | Male | 28 | 119 | 182 | 82 |
| 9 | 9 | Male | 31 | 124 | 168 | 64 |

```
10 10  Male  42 109    162     60
```

```
attributes(df); str(df); summary(df)
```

```
$names  
[1] "id"      "sex"     "age"     "sbp"     "height"   "weight"
```

```
$class  
[1] "data.frame"
```

```
$row.names  
[1] 1 2 3 4 5 6 7 8 9 10
```

```
'data.frame': 10 obs. of 6 variables:  
 $ id    : int 1 2 3 4 5 6 7 8 9 10  
 $ sex   : chr "Female" "Female" "Female" "Female" ...  
 $ age   : num 34 22 54 43 44 39 38 28 31 42  
 $ sbp   : num 112 118 132 128 128 124 121 119 124 109  
 $ height: num 165 158 161 160 168 172 175 182 168 162  
 $ weight: num 52 48 59 60 48 72 73 82 64 60
```

| | id | sex | age | sbp |
|---------|--------|------------------|---------------|---------------|
| Min. | : 1.00 | Length:10 | Min. :22.00 | Min. :109.0 |
| 1st Qu. | : 3.25 | Class :character | 1st Qu.:31.75 | 1st Qu.:118.2 |
| Median | : 5.50 | Mode :character | Median :38.50 | Median :122.5 |
| Mean | : 5.50 | | Mean :37.50 | Mean :121.5 |
| 3rd Qu. | : 7.75 | | 3rd Qu.:42.75 | 3rd Qu.:127.0 |
| Max. | :10.00 | | Max. :54.00 | Max. :132.0 |

| | height | weight |
|---------|--------|---------------|
| Min. | :158.0 | Min. :48.00 |
| 1st Qu. | :161.2 | 1st Qu.:53.75 |
| Median | :166.5 | Median :60.00 |
| Mean | :167.1 | Mean :61.80 |
| 3rd Qu. | :171.0 | 3rd Qu.:70.00 |

```
Max. :182.0  Max. :82.00
```

```
# stringsAsFactors = TRUE 인 경우 sex의 summary() 결과
df <- data.frame(id, sex, age, sbp, height, weight,
                  stringsAsFactors = TRUE)
summary(df)
```

| | id | sex | age | spp | height |
|---------|-----------|------------|---------------|---------------|---------------|
| Min. | 1.00 | Female:5 | Min. :22.00 | Min. :109.0 | Min. :158.0 |
| 1st Qu. | 3.25 | Male :5 | 1st Qu.:31.75 | 1st Qu.:118.2 | 1st Qu.:161.2 |
| Median | 5.50 | | Median :38.50 | Median :122.5 | Median :166.5 |
| Mean | 5.50 | | Mean :37.50 | Mean :121.5 | Mean :167.1 |
| 3rd Qu. | 7.75 | | 3rd Qu.:42.75 | 3rd Qu.:127.0 | 3rd Qu.:171.0 |
| Max. | 10.00 | | Max. :54.00 | Max. :132.0 | Max. :182.0 |
| | | | | | |
| | | | weight | | |
| Min. | | | :48.00 | | |
| 1st Qu. | | | :53.75 | | |
| Median | | | :60.00 | | |
| Mean | | | :61.80 | | |
| 3rd Qu. | | | :70.00 | | |
| Max. | | | :82.00 | | |

 `summary()` 함수는 객체의 클래스에 따라 요약 통계량을 출력해주는 함수로 특히 데이터 프레임이 가지고 있는 변수들의 특징을 손쉽게 알아볼 수 있기 때문에 가장 많이 호출되는 함수 중 하나임. 숫자형 벡터에 대해서는 최솟값(minimum), 1/4 분위수(1st quantile), 중앙값(median), 평균(mean), 3/4 분위수(3rd quantile), 최댓값을 출력하고, factor 형 객체에 대해서는 factor의 각 수준 별 빈도를 출력함. 2차원 이상 `table()` 객체에 적용 시 χ^2 검정(독립성 검정) 결과값을 출력함.

- 이미 정의된 데이터 프레임에 데이터를 추가 가능
 - 예를 들어 `dbp`라는 벡터에 이완기 혈압(diastolic blood pressure)

- 데이터가 입력되어 있고 df에 dbp 변수를 새롭게 추가 시 df\$dbp <- x 형태로 추가
- 위 형태로 이미 존재하고 있는 변수(열)에 새로운 값 재할당 가능
 - 이러한 형태로 문자형 벡터 추가 시 문자형 벡터는 자동으로 factor로 형 변환 되지 않음

```
x <- 1:nrow(df)
dbp <- c(73, 70, 88, 82, 75, 77, 74, 81, 72, 64)

# df에 "dbp" 열을 생성하고 x 값 대입
df$dbp <- x
df
```

| | id | sex | age | sbp | height | weight | dbp |
|----|----|--------|-----|-----|--------|--------|-----|
| 1 | 1 | Female | 34 | 112 | 165 | 52 | 1 |
| 2 | 2 | Female | 22 | 118 | 158 | 48 | 2 |
| 3 | 3 | Female | 54 | 132 | 161 | 59 | 3 |
| 4 | 4 | Female | 43 | 128 | 160 | 60 | 4 |
| 5 | 5 | Female | 44 | 128 | 168 | 48 | 5 |
| 6 | 6 | Male | 39 | 124 | 172 | 72 | 6 |
| 7 | 7 | Male | 38 | 121 | 175 | 73 | 7 |
| 8 | 8 | Male | 28 | 119 | 182 | 82 | 8 |
| 9 | 9 | Male | 31 | 124 | 168 | 64 | 9 |
| 10 | 10 | Male | 42 | 109 | 162 | 60 | 10 |

```
# df의 dbp에 dbp 벡터의 값을 재할당
df$dbp <- dbp
df
```

| | id | sex | age | sbp | height | weight | dbp |
|---|----|--------|-----|-----|--------|--------|-----|
| 1 | 1 | Female | 34 | 112 | 165 | 52 | 73 |
| 2 | 2 | Female | 22 | 118 | 158 | 48 | 70 |
| 3 | 3 | Female | 54 | 132 | 161 | 59 | 88 |

```

4   4 Female  43 128    160     60  82
5   5 Female  44 128    168     48  75
6   6   Male   39 124    172     72  77
7   7   Male   38 121    175     73  74
8   8   Male   28 119    182     82  81
9   9   Male   31 124    168     64  72
10 10   Male   42 109    162     60  64

```

```

# df0에 운동여부 exercyn 라는 변수 추가
# exercyn 는 "Y" 또는 "N" 두 값을 가짐
df$exercyn <- c("Y", "Y", "N", "Y", "N",
                 "N", "N", "Y", "N", "Y")
str(df)

```

```

'data.frame': 10 obs. of 8 variables:
 $ id      : int 1 2 3 4 5 6 7 8 9 10
 $ sex     : Factor w/ 2 levels "Female","Male": 1 1 1 1 1 2 2 2 2 2
 $ age     : num 34 22 54 43 44 39 38 28 31 42
 $ sbp     : num 112 118 132 128 128 124 121 119 124 109
 $ height : num 165 158 161 160 168 172 175 182 168 162
 $ weight  : num 52 48 59 60 48 72 73 82 64 60
 $ dbp     : num 73 70 88 82 75 77 74 81 72 64
 $ exercyn: chr "Y" "Y" "N" "Y" ...

```

- 행렬 및 벡터에서 언급 되었던 rownames(), colnames(), names(), dim(), ncol()/NCOL(), nrow()/NROW() 함수 적용 가능

```
rownames(df); colnames(df); names(df)
```

```

[1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"
[1] "id"     "sex"    "age"    "sbp"    "height"  "weight"  "dbp"
[8] "exercyn"
[1] "id"     "sex"    "age"    "sbp"    "height"  "weight"  "dbp"

```

```
[8] "exercyn"
```

```
dim(df); ncol(df); nrow(df)
```

```
[1] 10 8
```

```
[1] 8
```

```
[1] 10
```

```
# rownames() 함수를 통해 행이름 변경  
rownames(df) <- letters[1:10]  
df
```

| | id | sex | age | sbp | height | weight | dbp | exercyn |
|---|----|--------|-----|-----|--------|--------|-----|---------|
| a | 1 | Female | 34 | 112 | 165 | 52 | 73 | Y |
| b | 2 | Female | 22 | 118 | 158 | 48 | 70 | Y |
| c | 3 | Female | 54 | 132 | 161 | 59 | 88 | N |
| d | 4 | Female | 43 | 128 | 160 | 60 | 82 | Y |
| e | 5 | Female | 44 | 128 | 168 | 48 | 75 | N |
| f | 6 | Male | 39 | 124 | 172 | 72 | 77 | N |
| g | 7 | Male | 38 | 121 | 175 | 73 | 74 | N |
| h | 8 | Male | 28 | 119 | 182 | 82 | 81 | Y |
| i | 9 | Male | 31 | 124 | 168 | 64 | 72 | N |
| j | 10 | Male | 42 | 109 | 162 | 60 | 64 | Y |

```
#colnames() 함수를 통해 열 이름 변경  
varname_orig <- colnames(df)  
colnames(df) <- paste0("V", 1:ncol(df))  
df
```

| | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 |
|---|----|--------|----|-----|-----|----|----|----|
| a | 1 | Female | 34 | 112 | 165 | 52 | 73 | Y |
| b | 2 | Female | 22 | 118 | 158 | 48 | 70 | Y |
| c | 3 | Female | 54 | 132 | 161 | 59 | 88 | N |

```
d 4 Female 43 128 160 60 82 Y
e 5 Female 44 128 168 48 75 N
f 6 Male 39 124 172 72 77 N
g 7 Male 38 121 175 73 74 N
h 8 Male 28 119 182 82 81 Y
i 9 Male 31 124 168 64 72 N
j 10 Male 42 109 162 60 64 Y
```

```
# names() 함수와 colnames()는 거의 동일한 기능 수행
# 두 함수의 차이점?
names(df)
```

```
[1] "V1" "V2" "V3" "V4" "V5" "V6" "V7" "V8"
```

```
names(df) <- varname_orig
df
```

| | id | sex | age | sbp | height | weight | dbp | exercyn |
|---|----|--------|-----|-----|--------|--------|-----|---------|
| a | 1 | Female | 34 | 112 | 165 | 52 | 73 | Y |
| b | 2 | Female | 22 | 118 | 158 | 48 | 70 | Y |
| c | 3 | Female | 54 | 132 | 161 | 59 | 88 | N |
| d | 4 | Female | 43 | 128 | 160 | 60 | 82 | Y |
| e | 5 | Female | 44 | 128 | 168 | 48 | 75 | N |
| f | 6 | Male | 39 | 124 | 172 | 72 | 77 | N |
| g | 7 | Male | 38 | 121 | 175 | 73 | 74 | N |
| h | 8 | Male | 28 | 119 | 182 | 82 | 81 | Y |
| i | 9 | Male | 31 | 124 | 168 | 64 | 72 | N |
| j | 10 | Male | 42 | 109 | 162 | 60 | 64 | Y |

 참고: R Markdown에서 데이터 프레임을 손쉽게 html 파일에 출력하는 방법

- R Markdwon의 YAML 부분에 다음과 같이 옵션을 추가하면 별다른 함수 처리 없이 데이터 프레임을 테이블 형태로 html 문서에 붙일 수 있음. 아래 예시에서 `output` 이후 `df_print: paged` 옵션을 추가

- 옵션 추가 시 들여쓰기(탭 구분)은 YAML 문서의 트리 구조를 표현한 것이기 때문에
꼭 들여쓰기를 정확히 일치시켜야 함

```
---
```

```
title: "문서 제목"
author: "이름"
date: "`r Sys.Date()`"
output:
  html_document:
    df_print: paged
---
```

2.7.2 데이터 프레임 접근 및 필터링

접근방법

- 리스트 데이터 접근 방식

```
# 추출(접근) 연산자(함수) `df$col_name` 형태로 접근
df$height
```

```
[1] 165 158 161 160 168 172 175 182 168 162
```

```
# df[[index]] 또는 df[["col_name"]] 형태로 접근
df[[4]]
```

```
[1] 112 118 132 128 128 124 121 119 124 109
```

```
df[["sex"]]
```

```
[1] Female Female Female Female Female Male   Male   Male   Male   Male
Levels: Female Male
```

```
w <- df[[4]]
attributes(w); str(w)
```

NULL

```
num [1:10] 112 118 132 128 128 124 121 119 124 109
```

```
# df[index] 또는 df["col_name"] 형태로 접근
h <- df["height"]
attributes(h); str(h)
```

```
$names
[1] "height"
```

```
$row.names
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
```

```
$class
[1] "data.frame"

'data.frame':   10 obs. of  1 variable:
 $ height: num  165 158 161 160 168 172 175 182 168 162
```

- 행렬 데이터 접근 방식

```
# df[idx_row, idx_col] 또는 df[row_name, col_name]
# 형태 데이터 접근

# 열 index 접근
df[, 3];
```

```
[1] 34 22 54 43 44 39 38 28 31 42
```

```
# 형 강제 변환 방지  
df[, 3, drop = FALSE]
```

```
age  
a 34  
b 22  
c 54  
d 43  
e 44  
f 39  
g 38  
h 28  
i 31  
j 42
```

```
# 행 index 접근  
df[8, ]
```

```
id sex age sbp height weight dbp exercyn  
h 8 Male 28 119 182 82 81 Y
```

```
# 행과 열 index 접근  
df[1:4, 5:6]
```

```
height weight  
a 165 52  
b 158 48  
c 161 59  
d 160 60
```

```
# 열 이름으로 접근  
df[, c("sex", "sbp")]
```

```
sex sbp
a Female 112
b Female 118
c Female 132
d Female 128
e Female 128
f Male 124
g Male 121
h Male 119
i Male 124
j Male 109
```

```
# 행 이름으로 접근
```

```
df[c("d", "e", "f"), ]
```

| | id | sex | age | sbp | height | weight | dbp | exercyn |
|---|----|--------|-----|-----|--------|--------|-----|---------|
| d | 4 | Female | 43 | 128 | 160 | 60 | 82 | Y |
| e | 5 | Female | 44 | 128 | 168 | 48 | 75 | N |
| f | 6 | Male | 39 | 124 | 172 | 72 | 77 | N |

```
# 행과 열 이름으로 접근
```

```
df[c("a", "f"), c("sex", "height", "dbp")]
```

| | sex | height | dbp |
|---|--------|--------|-----|
| a | Female | 165 | 73 |
| f | Male | 172 | 77 |

```
# 행 또는 열 제외
```

```
df[-c(2:6), ]
```

| | id | sex | age | sbp | height | weight | dbp | exercyn |
|---|----|--------|-----|-----|--------|--------|-----|---------|
| a | 1 | Female | 34 | 112 | 165 | 52 | 73 | Y |
| g | 7 | Male | 38 | 121 | 175 | 73 | 74 | N |
| h | 8 | Male | 28 | 119 | 182 | 82 | 81 | Y |

```
i 9 Male 31 124    168     64 72      N
j 10 Male 42 109   162     60 64      Y
```

```
df[-c(1, 5:7), -c(1, 8)]
```

| | sex | age | sbp | height | weight | dbp |
|---|--------|-----|-----|--------|--------|-----|
| b | Female | 22 | 118 | 158 | 48 | 70 |
| c | Female | 54 | 132 | 161 | 59 | 88 |
| d | Female | 43 | 128 | 160 | 60 | 82 |
| h | Male | 28 | 119 | 182 | 82 | 81 |
| i | Male | 31 | 124 | 168 | 64 | 72 |
| j | Male | 42 | 109 | 162 | 60 | 64 |

필터링

- 벡터, 행렬과 마찬가지로 비교 연산자를 이용해 조건에 맞는 부분 데이터 추출 가능

```
# %in% 연산자를 이용해 데이터 프레임의 부분 변수 추출
# id, age 열을 제외한 나머지 데이터 프레임 추출
varname_df <- names(df)
df[, !varname_df %in% c("id", "age")]
```

| | sex | sbp | height | weight | dbp | exercyn |
|---|--------|-----|--------|--------|-----|---------|
| a | Female | 112 | 165 | 52 | 73 | Y |
| b | Female | 118 | 158 | 48 | 70 | Y |
| c | Female | 132 | 161 | 59 | 88 | N |
| d | Female | 128 | 160 | 60 | 82 | Y |
| e | Female | 128 | 168 | 48 | 75 | N |
| f | Male | 124 | 172 | 72 | 77 | N |
| g | Male | 121 | 175 | 73 | 74 | N |
| h | Male | 119 | 182 | 82 | 81 | Y |
| i | Male | 124 | 168 | 64 | 72 | N |
| j | Male | 109 | 162 | 60 | 64 | Y |

```
# 조건 연산자 사용
# sex 가 Female이고 나이가 40 이상인 데이터 추출
df[df$sex == "Female" & df$age >= 40, ]
```

| | id | sex | age | sbp | height | weight | dbp | exercyn |
|---|----|--------|-----|-----|--------|--------|-----|---------|
| c | 3 | Female | 54 | 132 | 161 | 59 | 88 | N |
| d | 4 | Female | 43 | 128 | 160 | 60 | 82 | Y |
| e | 5 | Female | 44 | 128 | 168 | 48 | 75 | N |

```
# id가 3보다 작은 데이터 추출
df[df[, 1] < 3, ]
```

| | id | sex | age | sbp | height | weight | dbp | exercyn |
|---|----|--------|-----|-----|--------|--------|-----|---------|
| a | 1 | Female | 34 | 112 | 165 | 52 | 73 | Y |
| b | 2 | Female | 22 | 118 | 158 | 48 | 70 | Y |

```
# subset 함수 이용한 데이터 추출
# sbp 가 120 이상이고 dbp 가 80 이상인 데이터 추출
subset(df, sbp >= 120 & dbp >= 80)
```

| | id | sex | age | sbp | height | weight | dbp | exercyn |
|---|----|--------|-----|-----|--------|--------|-----|---------|
| c | 3 | Female | 54 | 132 | 161 | 59 | 88 | N |
| d | 4 | Female | 43 | 128 | 160 | 60 | 82 | Y |

```
# 성별, 수축기, 이완기 혈압 변수만 추출
subset(df, select = c(sex, sbp, dbp))
```

```
sex sbp dbp
a Female 112 73
b Female 118 70
c Female 132 88
d Female 128 82
e Female 128 75
```

```
f   Male 124  77
g   Male 121  74
h   Male 119  81
i   Male 124  72
j   Male 109  64
```

```
# id 변수 제거
subset(df, select = -c(id))
```

| | sex | age | sbp | height | weight | dbp | exercyn |
|---|--------|-----|-----|--------|--------|-----|---------|
| a | Female | 34 | 112 | 165 | 52 | 73 | Y |
| b | Female | 22 | 118 | 158 | 48 | 70 | Y |
| c | Female | 54 | 132 | 161 | 59 | 88 | N |
| d | Female | 43 | 128 | 160 | 60 | 82 | Y |
| e | Female | 44 | 128 | 168 | 48 | 75 | N |
| f | Male | 39 | 124 | 172 | 72 | 77 | N |
| g | Male | 38 | 121 | 175 | 73 | 74 | N |
| h | Male | 28 | 119 | 182 | 82 | 81 | Y |
| i | Male | 31 | 124 | 168 | 64 | 72 | N |
| j | Male | 42 | 109 | 162 | 60 | 64 | Y |



데이터 프레임 또는 리스트 접근 시 `df$col_name`를 사용한다면 매번 데이터 프레임 이름과 \$을 반복하기 때문에 코드가 불필요하게 복잡해짐. R에서는 데이터 프레임 내부의 열 이름을 직접 접근할 수 있도록 도와주는 몇 가지 함수(예: `with()`, `attach()` 등)가 있는데, `with()`와 `within()` 활용법에 대해 간략히 알아봄.

- 위 예제에서 `sex` 가 `Female`이고 나이가 40 이상인 데이터 추출한다고 했을 때 `with()` 함수 사용

```
# with() 함수: 데이터 환경(객체 내)에서 주어진 표현식의 결과를 반환
with(
  data, #리스트 또는 데이터 프레임
```

```
expr, # 실제 명령을 수행할 표현식,
)
```

```
with(df, df[sex == "Female" & age >= 40, ])
```

| | id | sex | age | sbp | height | weight | dbp | exercyn |
|---|----|--------|-----|-----|--------|--------|-----|---------|
| c | 3 | Female | 54 | 132 | 161 | 59 | 88 | N |
| d | 4 | Female | 43 | 128 | 160 | 60 | 82 | Y |
| e | 5 | Female | 44 | 128 | 168 | 48 | 75 | N |

- `within()` 함수는 `with()`와 유사하지만 코드블록(`{...}`)을 이용해 보다 자유롭게 데이터 수정 및 추가 가능

```
df2 <- within(df, {
  hospital <- c("A", "B", "B", "A", "C",
               "A", "A", "B", "C", "B")
  mean_age <- mean(age)
})
```

2.7.3 데이터 프레임 관련 함수

유ти리티 함수

- 보통 데이터 분석은 외부에서 데이터를 읽은 후 특정 객체에 읽어온 데이터를 할당하는데, 이 경우 데이터가 저장된 객체는 대부분은 데이터 프레임 형태임.
- 읽어온 데이터는 보통 많은 행(표본)으로 구성되어 있기 때문에 데이터를 손쉽게 살펴보는 방법이 필요

2.7.3.0.1 *head()*/*tail()* 함수

- 객체(벡터, 행렬, 테이블, 데이터 프레임 등)의 처음 또는 끝에서부터 몇 개의 데이터(`default = 6L`)를 순차적으로 보여줌

```
# 앞에서 불러온 전복 데이터셋 확인  
dim(abalone)
```

```
[1] 4177    9
```

```
#처음 1에서 6행 까지 데이터 출력  
head(abalone)
```

| | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 |
|---|----|-------|-------|-------|--------|--------|--------|-------|----|
| 1 | M | 0.455 | 0.365 | 0.095 | 0.5140 | 0.2245 | 0.1010 | 0.150 | 15 |
| 2 | M | 0.350 | 0.265 | 0.090 | 0.2255 | 0.0995 | 0.0485 | 0.070 | 7 |
| 3 | F | 0.530 | 0.420 | 0.135 | 0.6770 | 0.2565 | 0.1415 | 0.210 | 9 |
| 4 | M | 0.440 | 0.365 | 0.125 | 0.5160 | 0.2155 | 0.1140 | 0.155 | 10 |
| 5 | I | 0.330 | 0.255 | 0.080 | 0.2050 | 0.0895 | 0.0395 | 0.055 | 7 |
| 6 | I | 0.425 | 0.300 | 0.095 | 0.3515 | 0.1410 | 0.0775 | 0.120 | 8 |

```
# 제일 마지막 행부터 위로 6개 데이터 까지 출력  
tail(abalone)
```

| | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 |
|------|----|-------|-------|-------|--------|--------|--------|--------|----|
| 4172 | M | 0.560 | 0.430 | 0.155 | 0.8675 | 0.4000 | 0.1720 | 0.2290 | 8 |
| 4173 | F | 0.565 | 0.450 | 0.165 | 0.8870 | 0.3700 | 0.2390 | 0.2490 | 11 |
| 4174 | M | 0.590 | 0.440 | 0.135 | 0.9660 | 0.4390 | 0.2145 | 0.2605 | 10 |
| 4175 | M | 0.600 | 0.475 | 0.205 | 1.1760 | 0.5255 | 0.2875 | 0.3080 | 9 |
| 4176 | F | 0.625 | 0.485 | 0.150 | 1.0945 | 0.5310 | 0.2610 | 0.2960 | 10 |
| 4177 | M | 0.710 | 0.555 | 0.195 | 1.9485 | 0.9455 | 0.3765 | 0.4950 | 12 |

2.7.3.0.2 *View()* 함수

- 2차원 데이터의 readable 한 스프레드시트 제공

```
View(abalone)
```

데이터 프레임 결합 및 분리 함수

2.7.3.0.3 rbind() / cbind() 함수

- 행렬에서 사용한 rbind() / cbind()를 데이터 프레임에도 적용 가능

```
a = data.frame(x1 = rep(0,5), x2 = rep("x",5))
b = data.frame(x1 = rep(1,5), x2 = rep("d",5))
c = data.frame(x3 = rep(2,5), x4 = rep("z",5))
d <- list(1, "d")
e <- list(x5 = rep(4, 5), x6 = rep("y", 5))
# rbind()를 이용해 두 데이터 프레임 a-b 합치기
ab <- rbind(a, b)
ab
```

```
x1 x2
1 0 x
2 0 x
3 0 x
4 0 x
5 0 x
6 1 d
7 1 d
8 1 d
9 1 d
10 1 d
```

```
# 변수명이 다른 경우
rbind(a, c) #변수명이 다르기 때문에 행으로 묶을 수 없다.
```

```
Error in match.names(clabs, names(xi)): names do not match previous names
```

```
# rbind()를 이용해 데이터 프레임-리스트 합치기  
abd <- rbind(ab, d)  
abd
```

```
x1 x2  
1 0 x  
2 0 x  
3 0 x  
4 0 x  
5 0 x  
6 1 d  
7 1 d  
8 1 d  
9 1 d  
10 1 d  
11 1 d
```

```
# cbind()를 이용해 두 데이터 프레임 a-c 합치기  
ac <- cbind(a, c)  
ac
```

```
x1 x2 x3 x4  
1 0 x 2 z  
2 0 x 2 z  
3 0 x 2 z  
4 0 x 2 z  
5 0 x 2 z
```

```
# 행 길이가 다르면 작은 길이의 데이터를 재사용  
cbind(a, d)
```

```
x1 x2 1 "d"  
1 0 x 1 d
```

```

2 0 x 1 d
3 0 x 1 d
4 0 x 1 d
5 0 x 1 d

# cbind()를 이용해 두 데이터 프레임-리스트 합치기
ace <- cbind(ac, e)

```

2.7.3.0.4 merge() 함수

- 두 데이터 프레임을 공통된 값을 기준으로 병합
- Excel의 vlookup() 함수 또는 데이터베이스 SQL 쿼리 중 join과 동일한 역할을 함
- cbind()의 경우는 단순히 열을 합치는 것이지만 merge()는 공통되는 열을 기준으로 두 데이터셋을 병합
- 공통된 데이터가 있을 때만 데이터 병합 수행

```

# merge() 함수 인수
merge(
  x, # 병합할 데이터 프레임
  y, # 병합할 데이터 프레임
  by, # 병합 기준으로 사용할 컬럼 (문자열 벡터)
  by.x, # 병합에 사용할 x와 y의 열 이름이 다른 경우
  by.y, # by.x와 by.y에 각각 공통 데이터에 해당하는 열 이름 지정
  # 둘 다 문자형 스칼라 또는 벡터값 인수로 받음
  all, # 논리값 이순
  # TRUE인 경우 x, y 중 공통된 값을 갖는 행이 없을 때
  # 해당 쪽을 NA를 채워 병합
  # 결과적으로 x, y 전체 행이 결과에 포함
  all.x, # x, y 중 특정 쪽에 공통된 값이 없더라도 항상
  all.y, # 결과에 포함
)

```

- `merge()` 함수 예시

```
d1 = data.frame(Name = c("Park", "Hanzo", "Mercy", "Soldier76"),
                 country = c("Korea", "Japan", "Swiss", "USA"))

d2 = data.frame(Age = c(19,38,37,56,31),
                Name = c("Park", "Hanzo", "Mercy", "Soldier76", "Mei" ) )

d1; d2
```

```
Name country
1 Park Korea
2 Hanzo Japan
3 Mercy Swiss
4 Soldier76 USA

Age      Name
1 19      Park
2 38      Hanzo
3 37      Mercy
4 56 Soldier76
5 31      Mei
```

```
dim(d1); dim(d2)
```

```
[1] 4 2
```

```
[1] 5 2
```

```
# 두 데이터 병합 01
merge(d1, d2, by = "Name")
```

```
Name country Age
1 Hanzo Japan 38
2 Mercy Swiss 37
3 Park Korea 19
4 Soldier76 USA 56
```

```
# 두 데이터 병합 02
names(d2)[2] <- "Surname"
merge(d1, d2, by.x = "Name", by.y = "Surname")
```

| | Name | country | Age |
|---|-----------|---------|-----|
| 1 | Hanzo | Japan | 38 |
| 2 | Mercy | Swiss | 37 |
| 3 | Park | Korea | 19 |
| 4 | Soldier76 | USA | 56 |

```
# 두 데이터 병합 03
merge(d1, d2,
      by.x = "Name", by.y = "Surname",
      all = T)
```

| | Name | country | Age |
|---|-----------|---------|-----|
| 1 | Hanzo | Japan | 38 |
| 2 | Mercy | Swiss | 37 |
| 3 | Park | Korea | 19 |
| 4 | Soldier76 | USA | 56 |
| 5 | Mei | <NA> | 31 |

2.7.3.0.5 *split()* 함수

- Factor 형에서 언급한 *split()* 함수를 통해 그룹 별로 데이터 분할
- 분할된 데이터는 리스트에 저장

```
split(df, df$sex)
```

```
$Female
  id   sex age sbp height weight dbp exercyn
a  1 Female  34 112     165      52  73       Y
b  2 Female  22 118     158      48  70       Y
```

```

c 3 Female 54 132    161      59  88      N
d 4 Female 43 128    160      60  82      Y
e 5 Female 44 128    168      48  75      N

```

\$Male

```

id sex age sbp height weight dbp exercyn
f 6 Male 39 124    172      72  77      N
g 7 Male 38 121    175      73  74      N
h 8 Male 28 119    182      82  81      Y
i 9 Male 31 124    168      64  72      N
j 10 Male 42 109   162      60  64      Y

```

데이터 정렬 함수

2.7.3.0.6 sort() 함수

- 데이터(벡터)의 정렬(오름차순 또는 내림차순) 결과 반환

```

# sort() 함수 인수
sort(
  x, # 정렬할 벡터
  decreasing, # 논리값, 내림차순 여부
  # default = FALSE
  na.last # 논리값. 결측 존재 시 NA 값 위치 지정
)          # TRUE: 정렬 후 결측은 마지막에 위치
           # FALSE: 맨 처음 NA 위치

```

- 예시

```

# 오름차순 정렬
sort(df2$age)

```

```
[1] 22 28 31 34 38 39 42 43 44 54
```

```
# 내림차순 정렬
sort(df$height, decreasing = TRUE)
```

```
[1] 182 175 172 168 168 165 162 161 160 158
```

2.7.3.0.7 *order()* 함수

- 데이터 정렬을 위해 순서에 대한 색인 생성 결과 반환
- 데이터 프레임에서 특정 열 기준으로 데이터 정렬 시 주로 사용

```
# 나이 기준으로 오름차순으로 데이터 정렬
with(df, df[order(age), ])
```

| | id | sex | age | sbp | height | weight | dbp | exercyn |
|---|----|--------|-----|-----|--------|--------|-----|---------|
| b | 2 | Female | 22 | 118 | 158 | 48 | 70 | Y |
| h | 8 | Male | 28 | 119 | 182 | 82 | 81 | Y |
| i | 9 | Male | 31 | 124 | 168 | 64 | 72 | N |
| a | 1 | Female | 34 | 112 | 165 | 52 | 73 | Y |
| g | 7 | Male | 38 | 121 | 175 | 73 | 74 | N |
| f | 6 | Male | 39 | 124 | 172 | 72 | 77 | N |
| j | 10 | Male | 42 | 109 | 162 | 60 | 64 | Y |
| d | 4 | Female | 43 | 128 | 160 | 60 | 82 | Y |
| e | 5 | Female | 44 | 128 | 168 | 48 | 75 | N |
| c | 3 | Female | 54 | 132 | 161 | 59 | 88 | N |

```
# 키 순으로 내림차순 정렬
df[order(df$height, decreasing = T), ]
```

| | id | sex | age | sbp | height | weight | dbp | exercyn |
|---|----|--------|-----|-----|--------|--------|-----|---------|
| h | 8 | Male | 28 | 119 | 182 | 82 | 81 | Y |
| g | 7 | Male | 38 | 121 | 175 | 73 | 74 | N |
| f | 6 | Male | 39 | 124 | 172 | 72 | 77 | N |
| e | 5 | Female | 44 | 128 | 168 | 48 | 75 | N |

| | | | | | | | | |
|---|----|--------|----|-----|-----|----|----|---|
| i | 9 | Male | 31 | 124 | 168 | 64 | 72 | N |
| a | 1 | Female | 34 | 112 | 165 | 52 | 73 | Y |
| j | 10 | Male | 42 | 109 | 162 | 60 | 64 | Y |
| c | 3 | Female | 54 | 132 | 161 | 59 | 88 | N |
| d | 4 | Female | 43 | 128 | 160 | 60 | 82 | Y |
| b | 2 | Female | 22 | 118 | 158 | 48 | 70 | Y |

2.7.4 *apply() 계열 함수

- `apply()`, `lapply()`, `sapply()` 등 `apply` 계열 함수는 R에서 가장 일반적으로 사용되는 함수 중 하나
- 보통 `for-loop`를 대신하기 위해 활용되며, R 객체를 입력 받아 원소 별 혹은 그루 별 함수를 적용
- 데이터 전체에 함수를 한번에 적용하는 vectorizing 연산을 수행함

`apply()` 함수

- 배열 또는 행렬에 주어진 함수를 적용한 뒤 그 결과를 벡터 또는 리스트로 반환
- 행 또는 열 차원 기준 함수 적용
- 동일한 유형의 벡터로 구성된 데이터셋에 적용

```
apply(
  X, # 배열, 행렬, 또는 같은 형태로 정의된 데이터 프레임
  MARGIN, # MARGIN = 1: 행 기준
            # MARGIN = 2: 열 기준
            # MARGIN = c(1,2): 행과 열 방향 모두
  FUN # 적용할 함수
)
```

- 예시1: 행렬 및 배열 `apply()` 적용

```
X <- matrix(1:9, nrow = 3)  
X
```

```
[,1] [,2] [,3]  
[1,]    1    4    7  
[2,]    2    5    8  
[3,]    3    6    9
```

```
# 행 기준으로 합계 계산  
apply(X, 1, sum)
```

```
[1] 12 15 18
```

```
# 열 기준으로 합계 계산  
apply(X, 2, sum)
```

```
[1] 6 15 24
```

```
# 배열에 apply 적용  
# 각 학생의 퀴즈와 중간-기말 각각 평균 계산  
Z # 성적
```

```
, , 1
```

```
[,1] [,2]  
[1,] 75 65  
[2,] 84 78  
[3,] 93 92
```

```
, , 2
```

```
[,1] [,2]  
[1,] 82 88
```

```
[2,] 78 75
[3,] 85 88
```

```
apply(Z, c(1,2), mean)
```

```
[,1] [,2]
[1,] 78.5 76.5
[2,] 81.0 76.5
[3,] 89.0 90.0
```

```
# 각 시점 별 개별 학생의 퀴즈-중간, 퀴즈-기말 평균 계산
apply(Z, c(1, 3), mean)
```

```
[,1] [,2]
[1,] 70.0 85.0
[2,] 81.0 76.5
[3,] 92.5 86.5
```

- 예시2: 데이터 프레임

- 위에서 사용한 df와 2.6.1 요인 (factor) 절에서 잠깐 예시로 사용된 전복(abalone) 데이터셋 사용



Abalone dataset 변수 설명

| | Origin | Variable | Name | Unit | Description |
|---|--------|------------|----------------|-------|-------------------------|
| 1 | V1 | sex | Sex | | 성별(M: 수컷; F: 암컷; I: 새끼) |
| 2 | V2 | length | Length | mm | 길이(최장길이) |
| 3 | V3 | diameter | Diameter | mm | 직경 |
| 4 | V4 | height | Height | mm | 껍질 내 육질의 높이 |
| 5 | V5 | whole.wt | Whole weight | grams | 전체 중량 |
| 6 | V6 | shucked.wt | Shucked weight | grams | 육질 무게 |
| 7 | V7 | viscera.wt | Viscera weight | grams | 내장 무게 |
| 8 | V8 | shell.wt | Shell weight | grams | 껍질 무게 |

| | | | | |
|---|----|-------|-------|-------|
| 9 | V9 | rings | Rings | 전복 나이 |
|---|----|-------|-------|-------|

```
names(abalone) <- c("sex", "length", "diameter",
                     "height", "whole.wt",
                     "shucked.wt", "viscera.wt",
                     "shell.wt", "rings")
head(abalone)
```

| | sex | length | diameter | height | whole.wt | shucked.wt | viscera.wt | shell.wt | rings |
|---|-----|--------|----------|--------|----------|------------|------------|----------|-------|
| 1 | M | 0.455 | 0.365 | 0.095 | 0.5140 | 0.2245 | 0.1010 | 0.150 | 15 |
| 2 | M | 0.350 | 0.265 | 0.090 | 0.2255 | 0.0995 | 0.0485 | 0.070 | 7 |
| 3 | F | 0.530 | 0.420 | 0.135 | 0.6770 | 0.2565 | 0.1415 | 0.210 | 9 |
| 4 | M | 0.440 | 0.365 | 0.125 | 0.5160 | 0.2155 | 0.1140 | 0.155 | 10 |
| 5 | I | 0.330 | 0.255 | 0.080 | 0.2050 | 0.0895 | 0.0395 | 0.055 | 7 |
| 6 | I | 0.425 | 0.300 | 0.095 | 0.3515 | 0.1410 | 0.0775 | 0.120 | 8 |

```
# sex를 제외한 나머지 수치형 변수에 대한 기초통계량 계산
# 평균: mean() 함수 사용
apply(abalone[, -1], 2, mean)
```

```
length      diameter      height      whole.wt shucked.wt viscera.wt      shell.wt
0.5239921  0.4078813  0.1395164  0.8287422  0.3593675  0.1805936  0.2388309
rings
9.9336845
```

```
# 표준편차: sd() 함수 사용
apply(abalone[, -1], 2, sd)
```

```
length      diameter      height      whole.wt shucked.wt viscera.wt      shell.wt
0.12009291 0.09923987 0.04182706 0.49038902 0.22196295 0.10961425 0.13920267
rings
3.22416903
```

```
# 개별 전복에 대해 내장, 육질, 껍질 무게 합계 계산
apply(abalone[, c("shucked.wt",
                 "viscera.wt",
                 "shell.wt")], 1,
      sum) -> ab_wt_sum

head(ab_wt_sum, 10)
```

```
[1] 0.4755 0.2180 0.6080 0.4845 0.1840 0.3385 0.7085 0.7035 0.4940 0.7855
```

```
# 데이터에 결측이 포함된 경우
# diameter 변수에 10개의 결측을 임의 생성
set.seed(20200410)

idx <- sample(1:NROW(abalone), 10) # 비복원 추출
ab2 <- abalone
ab2[idx, 3] <- NA

# 성별 제외한 나머지 변수의 평균 계산
apply(ab2[, -1], 2, mean)
```

| | length | diameter | height | whole.wt | shucked.wt | viscera.wt | shell.wt |
|-------|-----------|----------|-----------|-----------|------------|------------|-----------|
| | 0.5239921 | NA | 0.1395164 | 0.8287422 | 0.3593675 | 0.1805936 | 0.2388309 |
| rings | | | | | | | |
| | 9.9336845 | | | | | | |

```
# NA 결과를 피하려면?
apply(ab2[, -1], 2, mean, na.rm = TRUE)
```

| | length | diameter | height | whole.wt | shucked.wt | viscera.wt | shell.wt |
|-------|-----------|-----------|-----------|-----------|------------|------------|-----------|
| | 0.5239921 | 0.4079578 | 0.1395164 | 0.8287422 | 0.3593675 | 0.1805936 | 0.2388309 |
| rings | | | | | | | |
| | 9.9336845 | | | | | | |

참고 1: 결측이 포함된 벡터 연산 시 결측에 대한 처리 지정 없이 함수를 적용하면 결측값을 반환함. 따라서 R에서 제공되는 연산 관련 일반 함수는 결측처리에 대한 옵션을 인수로 받음. 보통 인수 형태는 `na.rm = T/F` 형태이고 다음의 함수를 통해 데이터에 결측 처리에 대한 속성 및 클래스를 부여함

- `na.omit()`/`na.exclude()`: NA가 포함되어 있는 행 생략

위 두 함수는 기본적으로 동일하지만, 특정 함수(예: 회귀분석을 수행하는 `lm()`) 함수에서는 다른 결과를 출력

참고 2: 위 예제에서 보여준 행 또는 열의 합 또는 평균 계산은 매우 자주 사용되기 때문에 `rowSums()`, `colSums()`, `rowMeans()`, `colMeans()` 함수가 제공됨

```
colMeans(abalone[,-1]) # apply 결과와 비교
```

```
length      diameter      height      whole.wt shucked.wt viscera.wt      shell.wt
0.5239921   0.4078813   0.1395164   0.8287422   0.3593675   0.1805936   0.2388309
      rings
9.9336845
```

tapply() 함수

- 2.6.1 요인 (factor) 절에서 설명
- `tapply()`는 1개의 벡터를 대상으로만 함수를 호출
- 데이터 프레임에 적용하려면? → `aggregate()` 함수 사용

2.7.4.0.1 aggregate()

- 데이터를 특정 factor의 수준 별로 나눈 후, 각 그룹마다 함수 적용
- `aggregate()`는 다음 두 가지 형태로 함수 적용 가능

```
# aggregate() 기본 인수
aggregate(
```

```

x, # R 객체, 주로 데이터 프레임
by, # 그룹으로 묶을 값의 리스트
FUN, # 그룹별 적용할 함수
)

aggregate(
  formula, # y ~ x 형태로 y는 계산에 사용할 값
  # x는 group 변수
  # y 대신 .은 그룹 변수를 제외하고
  # 적용할 함수의 대상이 되는 모든 변수
  data, # formula를 적용할 데이터
  FUN # 적용할 함수
)

```

- 예시: 임상연구 자료(df)

```

# 성별에 따라 연속형 변수의 평균 계산
aggregate(df[,-c(1:2, 8)], by =df[["sex"]], mean)

```

```

  sex   age   sbp height weight   dbp
1 Female 39.4 123.6 162.4   53.4 77.6
2   Male 35.6 119.4 171.8   70.2 73.6

```

```

# 수식 표현형 사용
aggregate(. ~ sex,
  data = df[,-8],
  mean)

```

```

  sex id   age   sbp height weight   dbp
1 Female  3 39.4 123.6 162.4   53.4 77.6
2   Male  8 35.6 119.4 171.8   70.2 73.6

```

```
# 요인의 2개인 경우
aggregate(df[,-c(1:2, 8)],
           by = list(sex = df[["sex"]], exercise = df[["exercyn"]]),
           mean)
```

| | sex | exercise | age | sbp | height | weight | dbp |
|---|--------|----------|-----|----------|----------|----------|----------|
| 1 | Female | N | 49 | 130.0000 | 164.5000 | 53.50000 | 81.50000 |
| 2 | Male | N | 36 | 123.0000 | 171.6667 | 69.66667 | 74.33333 |
| 3 | Female | Y | 33 | 119.3333 | 161.0000 | 53.33333 | 75.00000 |
| 4 | Male | Y | 35 | 114.0000 | 172.0000 | 71.00000 | 72.50000 |

```
aggregate(. ~ sex + exercyn,
           data = df[,-1],
           mean)
```

| | sex | exercyn | age | sbp | height | weight | dbp |
|---|--------|---------|-----|----------|----------|----------|----------|
| 1 | Female | N | 49 | 130.0000 | 164.5000 | 53.50000 | 81.50000 |
| 2 | Male | N | 36 | 123.0000 | 171.6667 | 69.66667 | 74.33333 |
| 3 | Female | Y | 33 | 119.3333 | 161.0000 | 53.33333 | 75.00000 |
| 4 | Male | Y | 35 | 114.0000 | 172.0000 | 71.00000 | 72.50000 |

`lapply()` 함수

- 특정 함수를 벡터, 리스트, 데이터 프레임 등에 적용하고 그 결과를 리스트로 반환

```
lapply(
  X, # 벡터, 리스트, 표현식, 또는 데이터 프레임
  FUN, # 적용할 함수
)
```

- 예시: abalone 데이터를 사용해 일변량 회귀분석 실시

```
# abalone 데이터를 이용해 단순회귀분석 결과 출력  
# 종속변수: rings  
# 설명변수: 성별을 제외한 모든 연속형 변수  
  
# lm() 함수를 이용한 일변량 회귀분석 실시  
univ_reg <- lapply(abalone[,-c(1, 9)], function(x) lm(abalone$rings ~ x))  
# univ_reg  
  
# 위 객체로부터 회귀모형 요약 통계량 결과  
summ_reg <- lapply(univ_reg, summary)  
# summ_reg  
  
# 추정 회귀계수를 데이터로 저장  
## summary(lm_object)의 속성 파악  
str(summ_reg[[1]])
```

```
List of 11  
$ call : language lm(formula = abalone$rings ~ x)  
$ terms :Classes 'terms', 'formula' language abalone$rings ~ x  
... .- attr(*, "variables")= language list(abalone$rings, x)  
... .- attr(*, "factors")= int [1:2, 1] 0 1  
... . .- attr(*, "dimnames")=List of 2  
... . . .$. : chr [1:2] "abalone$rings" "x"  
... . . .$. : chr "x"  
... .- attr(*, "term.labels")= chr "x"  
... .- attr(*, "order")= int 1  
... .- attr(*, "intercept")= int 1  
... .- attr(*, "response")= int 1  
... .- attr(*, ".Environment")=<environment: 0x000000039c1a000>  
... .- attr(*, "predvars")= language list(abalone$rings, x)  
... .- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"  
... . .- attr(*, "names")= chr [1:2] "abalone$rings" "x"
```

```
$ residuals      : Named num [1:4177] 6.0975 -0.3331 -1.0235 1.3217 -0.0342 ...
..- attr(*, "names")= chr [1:4177] "1" "2" "3" "4" ...
$ coefficients : num [1:2, 1:4] 2.102 14.946 0.186 0.345 11.328 ...
..- attr(*, "dimnames")=List of 2
... .$. : chr [1:2] "(Intercept)" "x"
... .$. : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
$ aliased       : Named logi [1:2] FALSE FALSE
..- attr(*, "names")= chr [1:2] "(Intercept)" "x"
$ sigma         : num 2.68
$ df            : int [1:3] 2 4175 2
$ r.squared     : num 0.31
$ adj.r.squared: num 0.31
$ fstatistic    : Named num [1:3] 1875 1 4175
..- attr(*, "names")= chr [1:3] "value" "numdf" "dendf"
$ cov.unscaled : num [1:2, 1:2] 0.0048 -0.0087 -0.0087 0.0166
..- attr(*, "dimnames")=List of 2
... .$. : chr [1:2] "(Intercept)" "x"
... .$. : chr [1:2] "(Intercept)" "x"
- attr(*, "class")= chr "summary.lm"
```

```
summ_reg[[1]]$coefficients
```

| | Estimate | Std. Error | t value | Pr(> t) |
|-------------|-----------|------------|----------|--------------|
| (Intercept) | 2.101883 | 0.1855477 | 11.32800 | 2.544353e-29 |
| x | 14.946411 | 0.3451570 | 43.30323 | 0.000000e+00 |

```
# summ_reg에서 coefficients만 추출
res <- lapply(summ_reg, function(lst) lst$coefficients)
res
```

```
$length
Estimate Std. Error t value Pr(>|t|)
(Intercept) 2.101883 0.1855477 11.32800 2.544353e-29
x           14.946411 0.3451570 43.30323 0.000000e+00
```

```
$diameter
      Estimate Std. Error t value Pr(>|t|)
(Intercept) 2.318574  0.1727366 13.42260 3.01241e-40
x           18.669921  0.4114954 45.37091 0.000000e+00

$height
      Estimate Std. Error t value Pr(>|t|)
(Intercept) 3.938464  0.1442530 27.30248 3.678478e-151
x           42.971441  0.9904086 43.38759 0.000000e+00

$whole.wt
      Estimate Std. Error t value Pr(>|t|)
(Intercept) 6.989239  0.08244313 84.77648 0.000000e+00
x           3.552909  0.08561680 41.49780 1.888678e-315

$shucked.wt
      Estimate Std. Error t value Pr(>|t|)
(Intercept) 7.736643  0.0861330 89.82206 0.000000e+00
x           6.113633  0.2039254 29.97976 5.087464e-179

$viscera.wt
      Estimate Std. Error t value Pr(>|t|)
(Intercept) 7.257427  0.08306853 87.36674 0.000000e+00
x           14.819227 0.39322428 37.68645 8.574726e-268

$shell.wt
      Estimate Std. Error t value Pr(>|t|)
(Intercept) 6.462117  0.07714642 83.76431      0
x           14.535675 0.27908233 52.08382      0

# res 결과를 2차원 배열 형태로 변환
# do.call() 함수 사용
```

```
# 리스트로 주어진 인자에 함수를 적용하여 결과 반환
res <- do.call(rbind, res)
res
```

| | Estimate | Std. Error | t value | Pr(> t) |
|-------------|-----------|------------|----------|---------------|
| (Intercept) | 2.101883 | 0.18554766 | 11.32800 | 2.544353e-29 |
| x | 14.946411 | 0.34515697 | 43.30323 | 0.000000e+00 |
| (Intercept) | 2.318574 | 0.17273658 | 13.42260 | 3.012410e-40 |
| x | 18.669921 | 0.41149538 | 45.37091 | 0.000000e+00 |
| (Intercept) | 3.938464 | 0.14425297 | 27.30248 | 3.678478e-151 |
| x | 42.971441 | 0.99040860 | 43.38759 | 0.000000e+00 |
| (Intercept) | 6.989239 | 0.08244313 | 84.77648 | 0.000000e+00 |
| x | 3.552909 | 0.08561680 | 41.49780 | 1.888678e-315 |
| (Intercept) | 7.736643 | 0.08613300 | 89.82206 | 0.000000e+00 |
| x | 6.113633 | 0.20392536 | 29.97976 | 5.087464e-179 |
| (Intercept) | 7.257427 | 0.08306853 | 87.36674 | 0.000000e+00 |
| x | 14.819227 | 0.39322428 | 37.68645 | 8.574726e-268 |
| (Intercept) | 6.462117 | 0.07714642 | 83.76431 | 0.000000e+00 |
| x | 14.535675 | 0.27908233 | 52.08382 | 0.000000e+00 |

sapply() 힘수

- lapply() 함수와 유사하나(lapply()의 wrapper 함수), 결과를 벡터 또는 행렬로 반환하는 점에서 차이를 보임
- 예시: 회귀분석

```
# 각 변수별 회귀계수(절편항과 설명변수) 반환
univ_reg2 <- sapply(abalone[,-c(1, 9)], function(x) {
  coef(lm(abalone$rings ~ x)) # lm 클래스에서 회귀계수 반환
})
univ_reg2
```

length diameter height whole.wt shucked.wt viscera.wt

```
(Intercept) 2.101883 2.318574 3.938464 6.989239 7.736643 7.257427  
x          14.946411 18.669921 42.971441 3.552909 6.113633 14.819227  
           shell.wt  
(Intercept) 6.462117  
x          14.535675
```

```
attributes(univ_reg2) # 행렬 반환
```

```
$dim
```

```
[1] 2 7
```

```
$dimnames
```

```
$dimnames[[1]]
```

```
[1] "(Intercept)" "x"
```

```
$dimnames[[2]]
```

```
[1] "length"      "diameter"    "height"      "whole.wt"    "shucked.wt"  
[6] "viscera.wt" "shell.wt"
```

```
# as.data.frame(univ_reg2)
```

mapply() 함수

- mapply()와 유사하지만 다수의 인수를 함수에 전달해 적용
- 임의의 함수 FUN()이 있고, FUN()이 사용할 인수가 데이터로 저장되어 있을 때 이를 불러들여 함수를 적용

```
mapply(  
  FUN, # 적용할 함수  
  ... , # 적용할 인수  
)
```

- 난수 생성 예시: rnorm() 함수 사용

- `rnorm(n, mean, sd)`: 평균이 `mean`이고 표준편차가 `sd`인 정규분포에서 `n`개의 난수 생성

```
# 평균이 각각 0, 1, 2, 4이고
# 표준편차가 1, 1, 1, 1인 정규난수를
# 각각 20, 40, 60, 100 개 생성

rn_res <- mapply(rnorm,
                  c(20, 40, 60, 100),
                  c(0:2, 4),
                  rep(1, 4))

# rn_res

# 생성한 난수의 평균과 표준편차 확인
sapply(rn_res, mean); sapply(rn_res, sd)
```

[1] 0.1277941 0.8607565 1.9070456 3.9012846

[1] 0.8954115 0.7916029 0.9990486 0.9088889

2.8 객체의 유형 판별 및 변환

지금까지 R 객체를 알아보면서 `is.na()`, `is.null()` 등 스칼라의 데이터 타입을 확인하는 함수부터 `str()`, `attributes()`, `class()`와 같이 객체의 속성 및 구조에 대해 확인하는 함수들에 대해 간략히 소개함. 앞서 언급한 것처럼 R은 스크립트 언어이기 때문에 모든 명령 실행이 함수 기반으로 이루어짐. 특정 객체에만 적용할 수 있는 함수들이 있는 반면, 함수를 통해 새로운 속성을 갖는 객체가 생성되기도 함. 그렇기 때문에 함수 적용 또는 적용 후 반환 후 생성된

객체의 타입을 확인하거나 객체의 유형 변환 작업은 R을 통해 데이터를 분석하는 과정에서 빈번하게 발생함.

R에서는 객체 유형 판별을 위해 `is.type_name()`, 객체 타입 변환을 위해 `as.type_name()` 형태의 함수를 제공함. 지금까지 배운 R 객체에 대한 `is.`과 `as.` 계열 함수는 아래와 같음.

TABLE 2.6: R 객체 타입 판별 및 변환 함수

| is 계열 함수 | as 계열 함수 | 설명 |
|------------------------------|------------------------------|--|
| <code>is.factor()</code> | <code>as.factor()</code> | 주어진 객체가 <code>factor</code> 형인지 판단/변환 |
| <code>is.ordered()</code> | <code>as.ordered</code> | 주어진 객체가 순서형 <code>factor</code> 인지 판단/변환 |
| <code>is.numeric()</code> | <code>as.numeric()</code> | 주어진 객체가 수치형인지 판단/변환 |
| <code>is.character()</code> | <code>as.character()</code> | 주어진 객체가 문자형인지 판단/변환 |
| <code>is.matrix()</code> | <code>as.matrix()</code> | 주어진 객체가 행렬인지 판단/변환 |
| <code>is.array()</code> | <code>as.array()</code> | 주어진 객체가 배열인지 판단/변환 |
| <code>is.list()</code> | <code>as.list()</code> | 주어진 객체가 리스트인지 판단/변환 |
| <code>is.data.frame()</code> | <code>as.data.frame()</code> | 주어진 객체가 데이터 프레임인지 판단/변환 |

- `is/as계열` 함수 사용 예시

```
x <- c("M", "F"); f <- factor(x)
# x가 문자열인가?
is.character(x)
```

```
[1] TRUE
```

```
# f가 factor인가?
is.factor(f)
```

```
[1] TRUE
```

```
# f가 숫자형인가?  
is.numeric(f)
```

[1] FALSE

```
# f를 수치형으로 변환  
f <- as.numeric(f)  
is.numeric(f)
```

[1] TRUE

f

[1] 2 1

```
# 다시 f를 factor형으로 변환  
as.factor(f)
```

[1] 2 1

Levels: 1 2

- 2차원 데이터 객체 유형 판별 및 변환

```
X <- matrix(rnorm(9), 3)  
d <- data.frame(group = rep(LETTERS[1:3], each = 2),  
                 meas = c(mapply(rnorm,  
                               c(2, 2, 2),  
                               c(1, 2, 3),  
                               c(1, 1, 1))))  
  
# 객체 유형 확인  
is.matrix(X); is.data.frame(X)
```

[1] TRUE

```
[1] FALSE
```

```
is.matrix(d); is.data.frame(d)
```

```
[1] FALSE
```

```
[1] TRUE
```

```
# 객체 유형 변환  
as.data.frame(X); as.matrix(d)
```

| | V1 | V2 | V3 |
|---|------------|-------------|------------|
| 1 | 0.3288415 | -0.02180829 | -0.3734678 |
| 2 | 0.2479955 | 1.35400192 | -2.2851782 |
| 3 | -0.7521625 | -0.15786514 | -1.1439634 |

```
group meas  
[1,] "A"    "2.028870"  
[2,] "A"    "1.996936"  
[3,] "B"    "2.835169"  
[4,] "B"    "1.105901"  
[5,] "C"    "1.095014"  
[6,] "C"    "2.927883"
```

```
as.list(X); as.list(d)
```

```
[[1]]
```

```
[1] 0.3288415
```

```
[[2]]
```

```
[1] 0.2479955
```

```
[[3]]
```

```
[1] -0.7521625
```

```
[[4]]
```

```
[1] -0.02180829
```

```
[[5]]
```

```
[1] 1.354002
```

```
[[6]]
```

```
[1] -0.1578651
```

```
[[7]]
```

```
[1] -0.3734678
```

```
[[8]]
```

```
[1] -2.285178
```

```
[[9]]
```

```
[1] -1.143963
```

```
$group
```

```
[1] A A B B C C
```

```
Levels: A B C
```

```
$meas
```

```
[1] 2.028870 1.996936 2.835169 1.105901 1.095014 2.927883
```

2.9 Homework #2

1. `seq()` 함수를 사용하여 $\log(\exp(10))$ 부터 0 까지 길이가 100인 벡터를 생성 후 객체 `lambda`를 생성하시오.

2. 두 벡터 $p = c(1, 4, 2, 3, 4, 7, 9, 12)$, $q = c(4, 5, 3, 2)$ 의
사칙연산 결과를 출력하고, 왜 이런 형태로 계산이 이루어졌는지 기술하시오.
3. 집합 $A = \{1, 3, 5, 7, 8, 9, 12, 15\}$ 이고 집합 $B = \{3, 6, 9, 12, 15, 18\}$ 일
때, $A \cup B$, $A \cap B$, $A - B$ 의 결과를 출력하시오.
4. `year`라는 객체에 $\{2000, 2001, \dots, 2020\}$, `month` 객체에 $\{\text{Jan}, \text{Feb}, \dots, \text{Dec}\}$, `day` 객체에 $\{1, \dots, 31\}$ 을 저장하고 `Date`라는
`list`를 생성 후 생성 결과를 출력 하시오.
5. `x` 벡터에 $\{23, 22, 24.5, \text{NA}, \text{NA}, 28, 27.8, 31, \text{NA}, \text{NA}\}$ 를 입력
하고 결측의 개수를 구하시오.
6. `tidyverse` 패키지를 불러온 후 `mpg` 데이터 셋에서 `hwy` 변수을 `x`라는 객체
에 저장한 후, `x` 객체에서 24보다 작은 값들의 개수를 구하시오.
7. 1부터 150 까지 1 단위 수열을 생성 후 객체 `x`에 저장하고 `x`에서 홀수 값만
추출 하시오.
8. 두 벡터 $\{1, 2, 3, 0, -1, -2, -1, 0, 7\}$ 와 $\{6, -3, 0, 0, 4, -5, 0, 0, 2\}$ 를 각각 `x`와 `y` 객체에 저장하고, 해당 객체를 이용해 다음
행렬을 생성하시오

$$\mathbf{X} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & -1 & -2 \\ -1 & 0 & 7 \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} 6 & 0 & 0 \\ -3 & 4 & 0 \\ 0 & -5 & 2 \end{bmatrix}$$

9. 위 두 행렬의 연산 결과를 출력 하시오

- \mathbf{XX}^T
- \mathbf{XY}
- \mathbf{YX}

- $\det(\mathbf{X})$
- \mathbf{Y}^{-1}

10. `runif()` 함수를 이용해 난수 200개를 생성하여 `x`라는 객체에 저장 하시오.

- 생성한 `x` 를 이용해 `x`가 0.5 보다 작으면 0, 0.5 보다 크거나 같으면 1 값을
재할당 하시오.
- 수준이 0, 1이고 수준이름이 각각 “Male”, “Female” 인 요인형 객체 `sex`를
생성하시오.

과제 제출 방식

- R Markdown 문서 (`Rmd`) 파일과 해당 문서를 컴파일 후 생성된 `html` 파일 모두
제출할 것
- 모든 문제에 대해 작성한 R 코드 및 결과가 `html` 문서에 포함되어야 함.
- 해당 과제에 대한 R Markdown 문서 템플릿은 https://github.com/zorba78/cnu-r-programming-lecture-note/blob/master/assignment/homework2_template.Rmd에서 다운로드 또는 스크레이핑 가능
- 최종 파일명은 학번-성명.Rmd, 학번-성명.html로 저장

2.10 Homework #3

1. 2.5절 배열에서 다른 확장 예제 “RGB값을 무작위로 샘플링 후 매개변수로
노이즈 가중치 조절해 보기” 명령 스크립트 중 다음 아래에 해당하는 구문의
반복 명령을 최소화한 스크립트 작성 후 해당 스크립트가 정상적으로 작동
하는지 그림 출력을 통해 확인하시오(라인별 의미에 대한 주석 추가할 것).
단, 그림은 2.5절 예제와 동일한 그림을 사용(Hint: `*apply()` 계열 함수,
코드블록(`{}`), `return()`, `unlist()`, `array()` 함수 사용)

```
require(tidyverse)
require(jpeg)
require(cowplot)

myurl <- paste0("https://img.livescore.co.kr/data/editor/1906/",
                 "ba517de8162d92f4ea0e9de0ec98ba02.jpg")
z <- tempfile()
download.file(myurl,z,mode="wb")

pic <- readJPEG(z)
dim_pic <- dim(pic)
t <- 0.2; nl <- length(300:460); pl <- length(440:520)

# 다음 아래(문제 1에 해당)
yr <- pic[300:460, 440:520, 1]
yg <- pic[300:460, 440:520, 2]
yb <- pic[300:460, 440:520, 3]

t <- 0.2
wr <- t * yr + (1 - t)*matrix(runif(length(yr)), nrow = nl, ncol = pl)
wg <- t * yg + (1 - t)*matrix(runif(length(yg)), nrow = nl, ncol = pl)
wb <- t * yb + (1 - t)*matrix(runif(length(yb)), nrow = nl, ncol = pl)

pic[300:460, 440:520, 1] <- wr
pic[300:460, 440:520, 2] <- wg
pic[300:460, 440:520, 3] <- wb
```

2. R에 기본으로 내장된 mtcars 데이터셋은 다음과 같이 11개의 변수로 구성되어 있다.

| 변수명 | 변수 설명 |
|------|--|
| mpg | Miles/(US) gallon, 연비 |
| cyl | Number of cylinders, 엔진 기통수 |
| disp | Displacement (cu.in.), 배기량(cc 단위) |
| hp | Gross horsepower, 마력 |
| drat | Rear axle ratio, 뒤차축비 |
| wt | Weight (1000 lbs), 무게 |
| qsec | 1/4 mile time, 1/4 mile 도달시간 |
| vs | Engine (0 = V-shaped, 1 = straight) |
| am | Transmission (0 = automatic, 1 = manual), 변속기여 |
| gear | Number of forward gears, 전진기여 갯수 |
| carb | Number of carburetors, 기화기 갯수 |

- a) `mtcars`의 데이터 구조를 파악하고 자료의 전반적인 형태에 대해 기술 하시오.
- b) 위 코드북을 참고하여 엔진과 변속기여에 해당하는 변수를 factor로 변환 후 해당 데이터 프레임을 df 객체에 저장 하시오.
- c) df 데이터셋에서 변속기여 (am)에 따른 mpg, disp, hp, drat, wt, qsec에 대한 평균과 표준편차를 구하시오 (Hint: `mean()`, `sd()` 함수 사용). 단 각 결과는 테이블 형태로 반환되어야 함(한 객체에 모든 변수의 평균 또는 표준편차가 저장, 테이블 객체가 반환을 의미하는 것은 아님).
- d) df 데이터셋을 엔진형태(vs) 별로 나눈 후, 연비를 종속변수(y)로 놓고 무게(wt)를 독립변수로 사용한 일변량 회귀식을 반환 하시오.

3. 1912년 4월 14일 타이타닉호 침몰 사고의 생존자 정보를 담고 있는 `titanic` 데이터셋은 통계학적으로 범주형 데이터 분석의 예시로서 널리 사용되고 있으며, 기계학습 및 데이터 과학 커뮤니티인 Kaggle³에서도 기계학습 알고리즘 경연을 위한 학습자료로 활용되고 있다. 해당 데이터는 <http://biostat>.

³<https://kaggle.com>

mc.vanderbilt.edu/wiki/pub/Main/DataSets/titanic3.csv에
서 다운로드가 가능하다. 타이타닉 데이터의 주요 변수에 대한 설명은 아래와
같다.

| 변수명 | 변수설명(영문) | 변수설명(국문) |
|----------|---|--------------|
| pclass | Passenger Class (1=1st; 2=2nd; 3=3rd) | 선실 등급 |
| survived | Survival (0=No; 1=Yes) | 생존여부 |
| name | Passenger name | 탑승객 성명 |
| sex | Sex | 성별 |
| age | Age | 나이 |
| sibsp | # of siblings/spouses abroad | 동승한 형제/배우자 수 |
| parch | # of parents/children abroad | 동승한 부모/자녀 수 |
| ticket | Ticket number | 티켓번호 |
| fare | Passenger fare | 티켓요금 |
| cabin | Cabin | 선실번호 |
| embarked | Port of embarkation (C=Cherbourg; Q=Queenstown; S=Southhampton) | 탑승 장소 |

- a) 위 코드북의 내용을 `codebook_tit` 이란 데이터 프레임 객체에 저장 후
출력하시오.
- b) 위 URL 주소로부터 타이타닉 데이터 파일을 읽은 후 `titanic` 객체에 저장
한 뒤 위 코드북에서 제시한 변수만 추출한 다음 `df_titanic`이란 객체에
저장한 결과에 대해 개괄적 형태 및 데이터 특성에 대해 기술하시오.
- c) `age` 변수에 포함된 결측값을 `age`의 전체 평균값으로 대체 하시오.
- d) `df_titanic`에서 `age`를 다음과 같이 그룹화 후 `age_group` 변수(factor)
를 생성 하시오.

```
# 0 살 이상 15살 미만: Children
# 15살 이상: Adult
```

- e) 선실 등급에 따른 남녀 별 그리고 연령 집단 별 생존 빈도 및 비율에 대해 각각 테이블로 나타내시오.

4. 아래와 같은 데이터셋이 주어졌을 때

- Dataset #1

| surname | nationality |
|----------|-------------|
| Tukey | US |
| Venables | Australia |
| Tierney | US |
| Ripley | UK |
| McNeil | Australia |

- Dataset #2

| name | title |
|----------|-------------------------------|
| Tukey | Exploratory Data Analysis |
| Venables | Modern Applied Statistics ... |
| Tierney | LISP-STAT |
| Ripley | Spatial Statistics |
| Ripley | Stochastic Simulation |
| McNeil | Interactive Data Analysis |
| R Core | An Introduction to R |

- a) Dataset #1에 해당하는 데이터를 `authors`, dataset #2에 해당하는 데이터를 `books`에 저장한 객체를 생성 하시오(단, 데이터 프레임을 구성하는 모든 변수는 문자열 변수로 저장).
- b) 두 데이터 셋을 `authors` 기준으로 병합한 데이터셋을 생성하시오.
- c) 두 데이터 셋의 모든 값들을 포함한 결함 데이터 셋을 생성 하시오.

과제 제출 방식

- R Markdown 문서 (**Rmd**) 파일과 해당 문서를 컴파일 후 생성된 **html** 파일 모두 제출할 것
- 모든 문제에 대해 작성한 R 코드 및 결과가 **html** 문서에 포함되어야 함.
- 모든 코드(스크립트)에는 라인 별 의미에 대한 주석을 달 것.
- 해당 과제에 대한 R Markdown 문서 템플릿은 https://github.com/zorba78/cnu-r-programming-lecture-note/blob/master/assignment/homework3_template.Rmd에서 다운로드 또는 스크레이핑 가능
- 최종 파일명은 학번-성명.Rmd, 학번-성명.html로 저장
- 압축파일은 *.zip 형태로 생성할 것



3

문자열 처리와 정규표현식



학습 목표

- 텍스트 문자 처리에 있어 가장 기본인 정규 표현식 (regular expression)에 대해 알아본다.
- R에서 기본으로 제공하는 문자열 차리 함수에 대해 알아본다

학습 필요성

- 실제 데이터는 다양한 형태의 텍스트(문자열)을 포함
- R에서 문자열을 이용한 반복 계산 가능
- 대규모 텍스트 데이터(웹문서, 블로그, SNS, 뉴스, 논문, 상품평, ...)로부터 새로운 정보 및 지식을 도출하기 위한 텍스트 마이닝이 대두 되면서 텍스트 처리에 대한 기본적 이해 필요
- 여러 문자열로 이루어진 방대한 텍스트 벡터에서 특정 패턴을 갖고 있는 구문을 선별해야 할 경우, 패턴을 도식화 할 수 있는 함축적 표현이 필요 →
정규 표현식

정규 표현식의 기본함수

- `grep()`, `grep1()`: 문자형 벡터에서 정규 표현식 또는 문자 패턴의 일치를 검색.

- `grep()`: 일치하는 특정 문자열을 포함하는 문자형 벡터 또는 인덱스를 반환
- `grep1()`: 문자열 포함 여부에 대한 논리값 반환
- `regexpr()`, `gregexpr()`: 문자형 벡터에서 정규 표현식 또는 문자 패턴과 일치하는 원소를 검색하고, 일치가 시작되는 문자열의 인덱스와 일치 길이를 반환
- `sub()`, `gsub()`: 문자열 벡터에서 정규 표현식 또는 문자 패턴과 일치하는 원소를 검색하고 해당 문자열을 다른 문자열로 변경
- `regexec()`: `regexpr()`과 동일하게 일치가 시작되는 문자열의 인덱스를 반환하지만 괄호로 묶인 하위 표현식의 위치를 추가로 반환

Note: 정규 표현식 및 문자열 처리를 위한 함수의 종류는 매우 다양하지만, 본 강의에서는 정규 표현식의 이해를 위해 일부만 소개할 것임

문자열 기초

- 탈출 지시자(escape indicator): \
- 키보드로 입력할 수 없는 문자를 입력하기 위해 사용
- 문자열에 백슬래쉬 \를 입력하려면 \\로 표시

```
# 문자열에 따옴표(single or double quote, ', ") 입력
double_quote <- "\\""
double_quote
```

```
[1] "\\"
```

```
single_quote <- '\'''
single_quote
```

```
[1] " "
```

```
x <- c("\\", "\\\\", '\\')  
writeLines(x)
```

```
"
```

```
\
```

```
'
```

```
# 백슬래쉬가 포함된 문자열  
x <- "abc\n\tabc"
```

```
# \n: Enter
```

```
# \t: tab 문자를 표현
```

```
writeLines(x)
```

```
abc
```

```
abc
```

```
# 특수문자 표현
```

```
x <- "\u00b5" # 그리스 문자 mu 표현 (유니코드)
```

```
x
```

```
[1] " "
```

참고자료

- Youtube 동영상¹: 영어 강의가 옥외 티...
- regexr.com²: 정규 표현식의 패턴 확인 가능
- Wikibooks R programming: Text processing³

3.1 유용한 문자열 관련 함수

3.1.1 nchar()

- 인간이 눈으로 읽을 수 있는 문자의 개수(길이)를 반환
- 공백, 줄바꿈 표시자(예: \n)도 하나의 문자 개수로 인식
- 한글의 한 글자는 2 바이트(byte)지만 한 글자로 인식 → byte 단위 반환 가능

```
# 문자열을 구성하는 문자 개수 반환
nchar(
  x, # 문자형 벡터
  type # "bytes": 바이트 단위 길이 반환
    # "char": 인간이 읽을 수 있는 글자 길이 반환
    # "width": 문자열이 표현된 폭의 길이 반환
)
```

- 예시

```
x <- "Carlos Gardel's song: Por Una Cabeza"
nchar(x)
```

```
[1] 36
```

```
y <- "abcde\nfghij"
nchar(y)
```

```
[1] 11
```

```
z <- "양준일: 가나다라마바사"
nchar(z)
```

```
[1] 12
```

```
# 문자열 벡터  
str <- sentences[1:10]  
nchar(str)
```

```
[1] 42 43 38 40 36 37 43 43 35 40
```

```
s <- c("abc", "가나다", "1234[]", "R programming\n", "\\\"R\\\"")  
  
nchar(s, type = "char")
```

```
[1] 3 3 6 14 3
```

```
nchar(s, type = "byte")
```

```
[1] 3 6 6 14 3
```

```
nchar(s, type = "width")
```

```
[1] 3 6 6 14 3
```



벡터의 원소 개수를 반환하는 `length()` 함수와는 다름.

3.1.2 `paste()`, `paste0()`

- 하나 이상의 문자열을 연결하여 하나의 문자열로 만들어주는 함수
- Excel의 문자열 연결자인 &와 거의 동일한 기능을 수행

```
paste(  
  ..., # 한 개 이상의 R 객체. 강제로 문자형 변환  
  sep  # 연결 구분자: 디플트 값은 공백(" ")  
  collapse # 묶을 객체가 하나의 문자열 벡터인 경우
```

```
# 모든 원소를 collapse 구분자로 묶은 길이가 1인 벡터 반환
)
```

- paste0()은 paste()의 wrapper 함수이고 paste()의 구분자 인수 sep = "" 일 때와 동일한 결과 반환
- 예시

```
i <- 1:length(letters)
```

```
paste(letters, i) # sep = " "
```

```
[1] "a 1"  "b 2"  "c 3"  "d 4"  "e 5"  "f 6"  "g 7"  "h 8"  "i 9"  "j 10"
[11] "k 11" "l 12" "m 13" "n 14" "o 15" "p 16" "q 17" "r 18" "s 19" "t 20"
[21] "u 21" "v 22" "w 23" "x 24" "y 25" "z 26"
```

```
paste(letters, i, sep = "_") # sep = "-"
```

```
[1] "a_1"  "b_2"  "c_3"  "d_4"  "e_5"  "f_6"  "g_7"  "h_8"  "i_9"  "j_10"
[11] "k_11" "l_12" "m_13" "n_14" "o_15" "p_16" "q_17" "r_18" "s_19" "t_20"
[21] "u_21" "v_22" "w_23" "x_24" "y_25" "z_26"
```

```
paste0(letters, i) # paste(letters, i, sep = "") 동일
```

```
[1] "a1"  "b2"  "c3"  "d4"  "e5"  "f6"  "g7"  "h8"  "i9"  "j10" "k11" "l12"
[13] "m13" "n14" "o15" "p16" "q17" "r18" "s19" "t20" "u21" "v22" "w23" "x24"
[25] "y25" "z26"
```

```
# collapse 인수 활용
paste(letters, collapse = "")
```

```
[1] "abcdefghijklmnopqrstuvwxyz"
```

```
writeLines(paste(str, collapse = "\n"))
```

The birch canoe slid on the smooth planks.
 Glue the sheet to the dark blue background.
 It's easy to tell the depth of a well.
 These days a chicken leg is a rare dish.
 Rice is often served in round bowls.
 The juice of lemons makes fine punch.
 The box was thrown beside the parked truck.
 The hogs were fed chopped corn and garbage.
 Four hours of steady work faced us.
 Large size in stockings is hard to sell.

```
# 37# 이상 객체 묶기  
paste("Col", 1:2, c(TRUE, FALSE, TRUE), sep = " ", collapse = "<->")
```

```
[1] "Col 1 TRUE<->Col 2 FALSE<->Col 1 TRUE"
```

```
# paste 함수 응용  
# 스트링 명령어 실행  
exprs <- paste("lm(mpg ~", names(mtcars)[3:5], ", data = mtcars)")  
exprs
```

```
[1] "lm(mpg ~ disp , data = mtcars)" "lm(mpg ~ hp , data = mtcars)"  
[3] "lm(mpg ~ drat , data = mtcars)"
```

```
sapply(1:length(exprs), function(i) coef(eval(parse(text = exprs[i]))))
```

| [,1] | [,2] | [,3] |
|-------------|-------------|-------------|
| (Intercept) | 29.59985476 | 30.09886054 |
| disp | -0.04121512 | -0.06822828 |
| | 7.678233 | |

3.1.3 sprintf()

- C 언어의 `sprintf()` 함수와 동일하며 특정 변수들의 값을 이용해 문자열을 반환함
- 수치형 값의 소수점 자리수를 맞추거나 할 때 유용하게 사용
- 포맷팅 문자열을 통해 수치형의 자릿수를 지정 뿐 아니라 전체 문자열의 길이 및 정렬 가능
- 대표적인 포맷팅 문자열은 아래 표와 같음.

| Format | 설명 |
|--------|----------|
| %s | 문자열 |
| %d | 정수형 |
| %f | 부동 소수점 수 |
| %e, %E | 지수형 |

- 예시

```
options()$digits #
```

```
[1] 7
```

```
pi # 파이 값
```

```
[1] 3.141593
```

```
sprintf("%f", pi)
```

```
[1] "3.141593"
```

```
# 소수점 자리수 3자리 까지 출력
```

```
sprintf("%.3f", pi)
```

```
[1] "3.142"
```

```
# 소수점 출력 하지 않음  
sprintf("%1.0f", pi)
```

```
[1] "3"
```

```
# 출력 문자열의 길이를 5로 고정 후  
# 소수점 한 자리까지 출력  
sprintf("%5.1f", pi)
```

```
[1] " 3.1"
```

```
nchar(sprintf("%5.1f", pi))
```

```
[1] 5
```

```
# 빈 공백에 0값 대입  
sprintf("%05.1f", pi)
```

```
[1] "003.1"
```

```
# 양수/음수 표현  
sprintf("%+f", pi)
```

```
[1] "+3.141593"
```

```
sprintf("%+f", -pi)
```

```
[1] "-3.141593"
```

```
# 출력 문자열의 첫 번째 값을 공백으로  
sprintf("% f", pi)
```

```
[1] " 3.141593"
```

```
# 왼쪽 정렬  
sprintf("%-10.3f", pi)
```

```
[1] "3.142      "
```

```
# 수치형에 정수 포맷을 입력하면?  
sprintf("%d", pi)
```

Error in sprintf("%d", pi): '%d'는 유효하지 않은 포맷입니다; 수치형 객체들에는 포맷 %f, %e, %g 또는 %a를 사용해 주세요

```
 sprintf("%d", 100); sprintf("%d", 20L)
```

```
[1] "100"
```

```
[1] "20"
```

```
# 지수형  
sprintf("%e", pi)
```

```
[1] "3.141593e+00"
```

```
 sprintf("%E", pi)
```

```
[1] "3.141593E+00"
```

```
 sprintf("%.2E", pi)
```

```
[1] "3.14E+00"
```

```
# 문자열  
sprintf("%s = %.2f", "Mean", pi)
```

```
[1] "Mean = 3.14"
```

```
# 응용
mn <- apply(cars, 2, mean)
std <- apply(cars, 2, sd)

# Mean ± SD 형태로 결과 출력 (소수점 2자리 고정)
res <- sprintf("%.2f \U00B1 %.2f", mn, std)
resp <- paste(paste0(names(cars), ":", res), collapse = "\n")
writeLines(resp)
```

speed: 15.40 ± 5.29

dist: 42.98 ± 25.77

3.1.4 substr()

- 문자열에서 특정 부분을 추출하는 함수
- 보통 한 문자열이 주어졌을 때 start에서 end 까지 추출

```
substr(
  x, # 문자형 벡터
  start, # 문자열 추출 시작 위치
  stop # 문자열 추출 종료 위치
)
```

- 예시

```
cnu <- "충남대학교 자연과학대학 정보통계학과"
substr(cnu, start = 14, stop = nchar(str))
```

[1] "정보통계학과"

```
# 문자열 벡터에서 각 원소 별 적용
substr(str, 5, 15)
```

```
[1] "birch canoe" " the sheet " " easy to te" "e days a ch" " is often s"
[6] "juice of le" "box was thr" "hogs were f" " hours of s" "e size in s"
```

3.1.5 `tolower()`, `toupper()`

- 대문자를 소문자(`tolower()`) 혹은 소문자를 대문자(`toupper()`)로 변환

```
LETTERS; tolower(LETTERS)
```

```
[1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S"
[20] "T" "U" "V" "W" "X" "Y" "Z"

[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
[20] "t" "u" "v" "w" "x" "y" "z"
```

```
letters; toupper(letters)
```

```
[1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
[20] "t" "u" "v" "w" "x" "y" "z"

[1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S"
[20] "T" "U" "V" "W" "X" "Y" "Z"
```

3.2 정규표현식 기본 함수

3.2.1 `grep()`, `grep1()`

정규표현식을 이용한 특정 문자 패턴 검색 시 가장 빈번히 사용되는 함수들 중 하나임.

`grep()`

특정 문자 벡터에서 찾고자 하는 패턴과 일치하는 원소의 인덱스, 원소값 반환

```
# 일치하는 특정 문자열을 포함하는 원소값(문자형) 또는 인덱스(정수)를 반환
```

```
grep(  
  pattern, # 정규 표현식 또는 문자 패턴  
  string, # 패턴을 검색할 문자열 벡터  
  value    # 논리값  
  # TRUE: pattern에 해당하는 원소값 반환  
  # FALSE: pattern이 있는 원소의 색인 반환  
)
```

```
x <- c("Equator", "North Pole", "South Pole")  
  
# x에서 Pole 이 있는 원소의 문자열 반환  
grep("Pole", x, value = T)
```

```
[1] "North Pole" "South Pole"
```

```
# x에서 Pole 이 있는 원소의 색인 반환  
grep("Pole", x, value = F)
```

```
[1] 2 3
```

```
# x에서 Eq를 포함한 원소 색인 반환  
grep("Eq", x)
```

```
[1] 1
```

```
grep1()
```

grep()과 유사한 기능을 갖지만, 함수의 반환값이 논리형 벡터임

```
# 일치하는 특정 문자열을 포함하는 원소 색인에 대한 논리값 반환
```

```
grepl(
  pattern, # 정규 표현식 또는 문자 패턴
  string   # 패턴을 검색할 문자열 벡터
)
```

- 사용 예시

```
# grep() 예시
# Titanic data 불러오기
url1 <- "https://raw.githubusercontent.com/"
url2 <- "agconti/kaggle-titanic/master/data/train.csv"
titanic <- read.csv(paste0(url1, url2),
                     stringsAsFactors = FALSE)
```

```
# 승객이름 추출
pname <- titanic$name

# 승객 이름이 James 인 사람만 추출
g <- grep("James", pname)
pname[g]
```

```
[1] "Moran, Mr. James"
[2] "Crease, Mr. Ernest James"
[3] "Sobey, Mr. Samuel James Hayden"
[4] "Bateman, Rev. Robert James"
[5] "Watt, Mrs. James (Elizabeth \"Bessie\" Inglis Milne)"
[6] "Smith, Mr. James Clinch"
[7] "Brown, Mrs. James Joseph (Margaret Tobin)"
[8] "Bracken, Mr. James H"
[9] "Reed, Mr. James George"
[10] "Baxter, Mrs. James (Helene DeLaudeniere Chaput)"
[11] "Drew, Mrs. James Vivian (Lulu Thorne Christian)"
```

```
[12] "Flynn, Mr. James"  
[13] "Scanlan, Mr. James"  
[14] "Webber, Mr. James"  
[15] "McGough, Mr. James Robert"  
[16] "Farrell, Mr. James"  
[17] "Sharp, Mr. Percival James R"  
[18] "Downton, Mr. William James"  
[19] "Elsbury, Mr. William James"  
[20] "Kelly, Mr. James"  
[21] "Hawksford, Mr. Walter James"  
[22] "Lester, Mr. James"  
[23] "Slemen, Mr. Richard James"  
[24] "Banfield, Mr. Frederick James"
```

3.2.2 `regexpr()`, `gregexpr()`

`grep()`과 `grepl()`의 한계점 보완: 특정 문자 패턴의 일치여부에 대한 정보를 제공하지만 위치 및 정규식의 일치 여부를 알려주지는 않음

`regexpr()`

- 문자열에서 패턴이 일치하는 문자(표현)가 첫 번째 등장하는 위치와 몇 개의 문자로 구성(길이) 되어 있는지를 반환
- 예시

```
x <- c("Darth Vader: If you only knew the power of the Dark Side.  
       Obi-Wan never told you what happend to your father",  
       "Luke: He told me enough! It was you who killed him!",  
       "Darth Vader: No. I'm your father")  
  
# grep 계열 함수  
grep("you", x); grepl("you", x)
```

```
[1] 1 2 3
[1] TRUE TRUE TRUE
```

```
# regexpr()
regexpr("you", x) # 각 x의 문자열에서 you가 처음 나타난 위치 및 길이 반환
```

```
[1] 17 33 22
attr(,"match.length")
[1] 3 3 3
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE
```

```
regexpr("father", x) # 패턴을 포함하지 않은 경우 -1 반환
```

```
[1] 111 -1 27
attr(,"match.length")
[1] 6 -1 6
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE
```

- `substr()` 함수와 `regexpr()` 함수를 이용해 텍스트 내 원하는 문자 추출 가능

```
idx <- regexpr("father", x)
substr(x, idx, idx + attr(idx, "match.length") - 1)
```

```
[1] "father" "" "father"
```

gregexpr()

- 영역에 걸쳐 패턴과 일치하는 문자의 위치 및 길이 반환(`regexpr()`의 global 버전)

```
gregexpr("you", x) # 각 x의 문자열에서 you가 나타난 모든 위치 및 길이 반환
```

```
[[1]]
```

```
[1] 17 86 106  
attr("match.length")  
[1] 3 3 3  
attr("index.type")  
[1] "chars"  
attr("useBytes")  
[1] TRUE
```

```
[[2]]
```

```
[1] 33  
attr("match.length")  
[1] 3  
attr("index.type")  
[1] "chars"  
attr("useBytes")  
[1] TRUE
```

```
[[3]]
```

```
[1] 22  
attr("match.length")  
[1] 3  
attr("index.type")  
[1] "chars"  
attr("useBytes")  
[1] TRUE
```

```
gregexpr("father", x) # 패턴을 포함하지 않은 경우 -1 반환
```

```
[[1]]  
[1] 111  
attr("match.length")  
[1] 6  
attr("index.type")  
[1] "chars"  
attr("useBytes")  
[1] TRUE
```

```
[[2]]  
[1] -1  
attr("match.length")  
[1] -1  
attr("index.type")  
[1] "chars"  
attr("useBytes")  
[1] TRUE
```

```
[[3]]  
[1] 27  
attr("match.length")  
[1] 6  
attr("index.type")  
[1] "chars"  
attr("useBytes")  
[1] TRUE
```

3.2.3 sub(), gsub()

- 검색하고자 하는 패턴을 원하는 문자로 변경

- 문자열 벡터의 패턴을 일치시키거나 문자열 정리가 필요할 때 사용

sub()

- 문자열에서 첫 번째 일치하는 패턴만 변경

```
sub(pattern, # 검색하고자 하는 문자, 패턴, 표현  
      replacement, # 검색할 패턴 대신 변경하고자 하는 문자 및 표현  
      x # 문자형 벡터  
      )
```

- 예시

```
jude <- c("Hey Jude, don't make it bad",  
        "Take a sad song and make it better",  
        "Remember to let her into your heart",  
        "Then you can start to make it better")  
  
sub("a", "X", jude)
```

```
[1] "Hey Jude, don't mXke it bad"  
[2] "TXke a sad song and make it better"  
[3] "Remember to let her into your heXrt"  
[4] "Then you cXn start to make it better"
```

gsub()

- 문자열에서 일치하는 모든 패턴 변경
- 예시

```
sub(" ", "_", jude)
```

```
[1] "Hey_Jude, don't make it bad"  
[2] "Take_a sad song and make it better"
```

```
[3] "Remember_to let her into your heart"
[4] "Then_you can start to make it better"
```

```
gsub(" ", "_", jude)
```

```
[1] "Hey_Jude,_don't_make_it_bad"
[2] "Take_a_sad_song_and_make_it_better"
[3] "Remember_to_let_her_into_your_heart"
[4] "Then_you_can_start_to_make_it_better"
```

```
gsub("a", "X", jude)
```

```
[1] "Hey Jude, don't mXke it bXd"
[2] "TXke X sXd song Xnd mXke it better"
[3] "Remember to let her into your heXrt"
[4] "Then you cXn stXrt to mXke it better"
```

3.2.4 `regexec()`

`regexpr()`과 유사하게 작동하지만 팔호(())로 묶인 하위 표현식에 대한 인덱스를 제공

- (): 정규 표현식의 메타 문자 중 하나로 그룹을 나타냄 → 정규표현식 내 논리적 테스트 수행 가능

```
bla <- c("I like statistics",
       "I like R programming",
       "I like bananas",
       "Estates and statues are too expensive")

grepl("like", bla)
```

```
[1] TRUE TRUE TRUE FALSE
```

```
grep1("are", bla)
```

```
[1] FALSE FALSE FALSE TRUE
```

```
grep1("(like|are)", bla)
```

```
[1] TRUE TRUE TRUE TRUE
```

- 찾고자 하는 패턴을 두 그룹으로 나눌 때 유용
- 예시

```
gregexpr("stat", bla)
```

```
[[1]]
```

```
[1] 8
```

```
attr(),"match.length")
```

```
[1] 4
```

```
attr(),"index.type")
```

```
[1] "chars"
```

```
attr(),"useBytes")
```

```
[1] TRUE
```

```
[[2]]
```

```
[1] -1
```

```
attr(),"match.length")
```

```
[1] -1
```

```
attr(),"index.type")
```

```
[1] "chars"
```

```
attr(),"useBytes")
```

```
[1] TRUE
```

```
[[3]]
```

```
[1] -1
```

```
attr("match.length")
[1] -1
attr("index.type")
[1] "chars"
attr("useBytes")
[1] TRUE

[[4]]
[1] 2 13
attr("match.length")
[1] 4 4
attr("index.type")
[1] "chars"
attr("useBytes")
[1] TRUE
```

```
gregexpr("(st)(at)", bla)
```

```
[[1]]
[1] 8
attr("match.length")
[1] 4
attr("index.type")
[1] "chars"
attr("useBytes")
[1] TRUE

[[2]]
[1] -1
attr("match.length")
[1] -1
attr("index.type")
[1] "chars"
```

```
attr("useBytes")
[1] TRUE

[[3]]
[1] -1
attr("match.length")
[1] -1
attr("index.type")
[1] "chars"
attr("useBytes")
[1] TRUE

[[4]]
[1] 2 13
attr("match.length")
[1] 4 4
attr("index.type")
[1] "chars"
attr("useBytes")
[1] TRUE

# "at"에 대한 패턴을 찾지 못하고
# "stat" 패턴과 동일한 결과 반환

regexec("(st)(at)", bla)

[[1]]
[1] 8 8 10
attr("match.length")
[1] 4 2 2
attr("index.type")
[1] "chars"
attr("useBytes")
```

```
[1] TRUE
```

```
[[2]]
```

```
[1] -1
```

```
attr(,"match.length")
```

```
[1] -1
```

```
attr(,"index.type")
```

```
[1] "chars"
```

```
attr(,"useBytes")
```

```
[1] TRUE
```

```
[[3]]
```

```
[1] -1
```

```
attr(,"match.length")
```

```
[1] -1
```

```
attr(,"index.type")
```

```
[1] "chars"
```

```
attr(,"useBytes")
```

```
[1] TRUE
```

```
[[4]]
```

```
[1] 2 2 4
```

```
attr(,"match.length")
```

```
[1] 4 2 2
```

```
attr(,"index.type")
```

```
[1] "chars"
```

```
attr(,"useBytes")
```

```
[1] TRUE
```

```
# "stat" 패턴도 동시에 반환됨을 유의
```

```
# 첫 번째 일치 패턴만 반환
```

3.2.5 strsplit()

- 문자열에서 매칭되는 특정 패턴(문자)을 기준으로 문자열을 분할함

```
strsplit(  
  x,      # 문자형 벡터  
  split # 분할 구분 문자(정규표현식 포함)  
)
```

- 예시

```
jude_w1 <- strsplit(jude, " ")  
jude_w1
```

```
[[1]]
```

```
[1] "Hey"    "Jude," "don't" "make"   "it"     "bad"
```

```
[[2]]
```

```
[1] "Take"   "a"       "sad"     "song"    "and"    "make"   "it"     "better"
```

```
[[3]]
```

```
[1] "Remember" "to"      "let"     "her"     "into"   "your"   "heart"
```

```
[[4]]
```

```
[1] "Then"    "you"    "can"    "start"   "to"     "make"   "it"     "better"
```

```
# 공백, 쉼표가 있는 경우 구분
```

```
jude_w2 <- strsplit(jude, "(\s|,)")  
jude_w2
```

```
[[1]]
```

```
[1] "Hey"    "Jude"   ""        "don't"  "make"   "it"     "bad"
```

```
[[2]]
```

```
[1] "Take"     "a"      "sad"      "song"      "and"      "make"      "it"       "better"  
[[3]]  
[1] "Remember" "to"      "let"      "her"      "into"      "your"      "heart"  
[[4]]  
[1] "Then"     "you"     "can"      "start"    "to"       "make"      "it"       "better"
```

3.3 정규 표현식 (regular expression)

- 주어진 문자열에 특정한 패턴이 있는 경우, 해당 패턴을 일반화(수식화)한 문자열
- 특정 패턴을 표현한 문자열을 메타 문자(meta character)라고 지칭
- 일반적으로 특정 규칙 또는 패턴이 문자열을 찾고(to find), 해당 규칙에 해당하는 문자열을 대체(replace, substitute)하기 위해 사용
- R 언어 뿐 아니라 타 프로그래밍 언어(C, Perl, Python 등) 워드 프로세서, 텍스트 편집기, 검색 엔진, 운영체제(Windows, Linux 등)에서도 범용적으로 사용
- 정규식이라고도 불리우며 영어로는 regex 또는 regexp로 명칭됨

3.3.1 기본 메타 문자

TABLE 3.1: 정규표현식 메타 문자: 기본

| Expression | Name | 설명 |
|------------|---------------|---|
| \. | Period (마침표) | 무엇이든 한 글자를 의미 |
| \+ | Plus | \+ 앞에 오는 표현이 하나 이상 포함 |
| * | Asterisk | * 앞에 오는 표현이 0 또는 하나 이상 포함 |
| ? | Question mark | ? 앞에 오는 표현이 0 또는 하나 포함 |
| ^ | Caret | ^ 뒤에 오는 표현으로 시작 |
| \$ | Dollar | \$ 앞에 오는 표연으로 끝나는 경우 |
| {} | Curly bracket | { } 앞에 정확히 {}에 있는 숫자만큼 반복되는 패턴 (예시 참고) |
| () | Parenthesis | () 정규 표현식 내 하위(그룹) 표현식 (예시 참고) |
| | Vertical bar | 의 왼쪽 또는 오른쪽 표현이 존재하는지 |

- 메타 문자를 메타 문자가 아닌 문자 자체로 인식하기 위해서는 해당 문자 앞에 \\를 붙임

```
# 마침표가 있는 위치 반환
str2 <- str[1:2]
regexp(".", str2)
```

```
[1] 1 1
attr("match.length")
[1] 1 1
attr("index.type")
[1] "chars"
attr("useBytes")
[1] TRUE
```

```
# 에러 출력
grepexpr("\.", str2)
```

Error: ""\."로 시작하는 문자열 중에서 '\.'는 인식할 수 없는 이스케이프입니다

```
# 정확한 표현
grepexpr("\\.", str2)
```

```
[1] 42 43
attr(,"match.length")
[1] 1 1
attr(,"index.type")
[1] "chars"
attr(,"useBytes")
[1] TRUE
```

. 마침표(period)

- 어떤 임의의 한 문자를 의미

```
# 문자열 자체가 존재하니까 참값 반환
grepl(".", jude)
```

```
[1] TRUE TRUE TRUE TRUE
```

```
grepl(".", "#@%$@FDSAGF$%")
```

```
[1] TRUE
```

```
# 문자없음 ""
grepl(".", "")
```

```
[1] FALSE
```

```
# a로 시작하고 중간에 어떤 글자가 하나 존재하고 b로 끝나는 패턴
bla2 <- c("aac", "aab", "accb", "acadb")
g <- grep1("a.b", bla2)
bla2[g]
```

```
[1] "aab"    "acadb"
```

```
# a와 b 사이 어떤 두 문자 존재하는 패턴
g <- grep1("a..b", bla2)
bla2[g]
```

```
[1] "accb"
```

+ (plus)

- + 에 선행된 패턴이 한 번 이상 매칭 → + 앞에 문자를 1개 이상 포함

```
# "a"를 적어도 하나 이상 포함한 원소 반환
grep1("a+", c("ab", "aa", "aab", "aaab", "b"))
```

```
[1] TRUE TRUE TRUE TRUE FALSE
```

```
# "l"과 "n" 사이에 "o"가 하나 이상인 원소 반환
grep1("lo+n", c("bloon", "blno", "leno", "lnooon", "lololon"))
```

```
[1] TRUE FALSE FALSE FALSE TRUE
```

* (asterisk)

- * 앞에 선행된 문자 또는 패턴이 0번 이상 매치 → * 앞에 문자를 0개 또는 1개 이상 포함

```
# xx가 "a"를 0 또는 1개 이상 포함하고 있는가?
xx <- c("bbb", "acb", "def", "cde", "zde", "era", "xsery")
```

```
# "a" 존재와 상관 없이 모든 문자열이 조건에 부합
g <- grep1("a*", xx)
xx[g]
```

```
[1] "bbb"   "acb"   "def"   "cde"   "zde"   "era"   "xsery"
```

```
# "aab"와 "c" 사이에 "d"가 없거나 하나 이상인 경우
# "caabec"인 경우 "aab"와 "c" 사이에 "e"가 존재하기 때문에 FALSE
grep1("aabdd*c", c("aabddc", "caabec", "aabc"))
```

```
[1] TRUE FALSE TRUE
```

? (question)

- ? 앞에 항목은 선택 사항이며 많아야 한 번 매치 → ? 앞의 문자를 0개 또는 1개 포함

```
xx <- c("ac", "abbc", "abc", "abcd", "abbdc")
g <- grep1("ab?c", xx) ## "a"와 "c" 사이에 "b"가 0개 또는 1개 포함
xx[g]
```

```
[1] "ac"   "abc"   "abcd"
```

```
yy <- c("aabc", "aabbc", "daabec", "aabbbc", "aabbbbc")
g <- grep1("aabb?c", yy) ## "aab"와 "c" 사이에 "b"가 0개 또는 1개 있는 경우 일치
yy[g]
```

```
[1] "aabc"   "aabbc"
```

^ (caret)

- ^ 뒤에 나오는 문자(열)로 시작하는 문자열 반환

```
# str에서 "The"로 시작하는 문자열 반환
g <- grep1("The", str)
str[g]
```

```
[1] "The birch canoe slid on the smooth planks."
[2] "These days a chicken leg is a rare dish."
[3] "The juice of lemons makes fine punch."
[4] "The box was thrown beside the parked truck."
[5] "The hogs were fed chopped corn and garbage."
```

- [^]: 대괄호(straight bracket) 안에 첫 번째 문자가 ^인 경우 ^뒤에 있는 문자들을 제외한 모든 문자와 매치

```
xx <- c("abc", "def", "xyz", "werx", "wbcsp", "cba")
# "a", "b", "c"를 순서 상관 없이 동시에 포함하지 않은 문자열 반환
g <- grep1("[^abc]", xx)
xx[g]
```

```
[1] "def"    "xyz"    "werx"   "wbcsp"
```

- ^[]: [] 안에 들어간 문자 중 어느 한 단어로 시작하는 문자열 반환

```
xx <- c("def", "wasp", "sepcial", "statisitc", "abbey load", "cross", "batman")
g <- grep1("^abc", xx)
xx[g]
```

```
[1] "abbey load" "cross"      "batman"
```

\$ (dollar)

- \$ 앞에 나오는 문자 및 패턴과 문자열의 맨 마지막 문자 패턴과 매치

```
g <- grep1("father$", x)
writeLines(x[g])
```

Darth Vader: If you only knew the power of the Dark Side.

Obi-Wan never told you what happened to your father

Darth Vader: No. I'm your father

{ } (curly bracket)

- {} 앞의 문자 패턴이 {} 안에 숫자만큼 반복되는 패턴을 매치
 - {n}: 정확히 n 번 매치
 - {n,m}: n 번에서 m 번 매치
 - {n, }: 적어도 n 번 이상 매치

```
xx <- c("tango", "jazz", "swing jazz", "hip hop",
       "groove", "rock'n roll", "heavy metal")
```

```
# "z"가 정확히 2번 반복되는 원소 반환
g <- grep1("z{2}", xx)
xx[g]
```

```
[1] "jazz"      "swing jazz"
```

```
# "e"가 2번 이상 반복되는 원소 반환
yy <- c("deer", "abacd", "abcd", "daaeb", "eel", "greeeg")
g <- grep1("e{2,}", yy)
xx[g]
```

```
[1] "tango"      "groove"      "rock'n roll" "heavy metal"
```

```
# "b"가 2번 이상 4번 이하 반복되고 앞에 "a"가 있는 원소 반환
zz <- c("ababababab", "abbb", "cbbe", "xabbabbc")
g <- grep1("ab{2,4}", zz)
zz[g]
```

```
[1] "abbb"      "xabbabbc"
```



참고: 위에서 소개한 메타 문자 중 *는 {0,}, +는 {1,}, ?는 {0,1}과 동일한 의미를 가짐

() (parenthesis)

- 특정 문자열을 ()로 grouping
- 한 개 이상의 그룹 지정 가능

```
# ab가 1~4회 이상 반복되는 문자열 반환
g <- grep1("(ab){1,4}", zz)
zz[g]
```

```
[1] "ababababab" "abbb" "xabbbbbcd"
```

```
# "The"로 시작하고 "punch"가 포함된 문자열 반환
g <- grep1("^(The)+.*punch)", str)
str[g]
```

```
[1] "The juice of lemons makes fine punch."
```

| (vertical bar)

- |를 기준으로 좌우 문자 패턴 중 하나를 의미하며 OR 조건과 동일한 의미를 가짐
- [] 의 경우 메타문자나 문자 한글자에 대해서만 적용되는 반면 |는 문자를 묶어 문자열로 지정 가능

```
g <- grep1("(is|was)", str)
str[g]
```

```
[1] "These days a chicken leg is a rare dish."
[2] "Rice is often served in round bowls."
[3] "The box was thrown beside the parked truck."
[4] "Large size in stockings is hard to sell."
```

```
g <- grep1("(are|were)", str)
str[g]
```

```
[1] "These days a chicken leg is a rare dish."
[2] "The hogs were fed chopped corn and garbage."
```

3.3.2 문자 집합

TABLE 3.2: 정규표현식 메타 문자: 문자집합

| Expression | 설명 |
|------------|--|
| \w | 문자(letter), 숫자(digit), 또는 _(underscore) 포함 |
| \d | 숫자 0에서 9 |
| \s | 공백문자(line break, tab, spaces) |
| \W | \w에 포함하지 않는 표현 |
| \D | 숫자가 아닌 표현 |
| \S | 공백이 아닌 표현 |

```
# \w 를 이용해 email 추출

email <- c("demo@naver.com",
          "sample@gmail.com",
          "coffee@daum.net",
          "redbull@nate.com",
          "android@gmail.com",
          "secondmoon@gmail.com",
          "zorba1997@korea.re.kr")

# 이메일 주소가 naver 또는 gmail만 추출
```

```
g <- grep1("\\w+@(naver|gmail)\\.\\w+", email)
email[g]
```

```
[1] "demo@naver.com"      "sample@gmail.com"      "android@gmail.com"
[4] "secondmoon@gmail.com"
```

```
# 숫자를 포함하는 문자열 추출: \d
ex <- c("ticket", "51203", "+-.,!@#", "ABCD", "_", "010-123-4567")
g <- grep1("\\d", ex)
ex[g]
```

```
[1] "51203"      "010-123-4567"
```

```
# 뒤쪽 공백문자 제거
xx <- c("some text on the line 1; \n and then some text on line two      ")
sub("\\s+$", "", xx)
```

```
[1] "some text on the line 1; \n and then some text on line two"
```

```
# 영문자(소문자 및 대문자 포함), 숫자, 언더바(_)를 제외한 문자 포함
g <- grep1("\\W", ex)
ex[g]
```

```
[1] "+-.,!@#"      "010-123-4567"
```

```
# 숫자를 제외한 모든 문자 반화
g <- grep1("\\D", ex)
ex[g]
```

```
[1] "ticket"      "+-.,!@#"      "ABCD"      "_"      "010-123-4567"
```

```
# 영문자, 숫자, 언더바를 제외한 모든 문자 포함하고
# 숫자와 특수문자를 포함하는 문자열도 제외
```

```

g <- grep1("\\W\\D", ex)
ex[g]

[1] "+-.,!@#"

## 공백, 탭을 제외한 모든 문자 포함

blank <- c(" ", "_", "abcd", "\t", "%^$##*")
g <- grep1("\\S", blank)
blank[g]

[1] "_"         "abcd"      "%^$##*"

```

3.3.3 문자 클래스

- 문자 집합을 더 세분화하여 특정 목적에 맞는 정규 표현형
- 대괄호([]) 안에 특정 패턴에 해당하는 문자로 규칙 표현하고 하이픈(-)을 사용해 특정 문자의 범위 지정 가능
- 응용 가능한 문자 클래스

TABLE 3.3: 정규표현식 주요 문자 클래스

| Expression | 설명 |
|-------------|------------------------|
| [a-z] | 알파벳 소문자 중 하나 |
| [A-Z] | 알파벳 대문자 중 하나 |
| [0-9] | 0에서 9까지 숫자 중 하나 |
| [a-zA-Z] | 모든 알파벳 중 하나 |
| [a-zA-Z0-9] | 알파벳 소문자나 숫자 중 한 문자 |
| [가-힝] | 모든 한글 중 하나 |
| [(abc)d] | 문자열 'abc'와 문자 'd' 중 하나 |

- POSIX (Portable Operating System Interface): 서로 다른 UNIX OS의 API를 정리하여 이식성이 높은 유닉스 응용 프로그램을 개발하기 위한 목적으로 IEEE가 책정한 애플리케이션 인터페이스 규격 ([POS, 4 16](#))

TABLE 3.4: 정규표현식: POSIX 문자 클래스

| Expression | 설명 |
|-------------------------|---|
| <code>[:punct:]</code> | 구둣점 문자 [] [!#\$%&()<>*+,./:;=>?@\\^_`{ }~-] |
| <code>[:alpha:]</code> | 알파벳 [A-Za-z]와 동일한 표현 |
| <code>[:lower:]</code> | 소문자 알파벳 [a-z]와 동일 |
| <code>[:upper:]</code> | 대문자 알파벳 [A-Z]와 동일 |
| <code>[:digit:]</code> | 숫자 0 ~ 9 [0-9]와 동일 |
| <code>[:alnum:]</code> | 알파벳과 숫자 [0-9A-Za-z]와 동일 |
| <code>[:cntrl:]</code> | 제어문자 b |
| <code>[:print:]</code> | 모든 인쇄 가능한 문자 |
| <code>[:space:]</code> | 공백문자 \\t\\r\\n\\v\\f |
| <code>[:blank:]</code> | 공백문자 중 \\t \\n |
| <code>[:xdigit:]</code> | 16 진수 |

```
movie <- c("terminator 3: rise of the machiens",
          "les miserables",
          "avengers: infinity war",
          "iron man",
          "indiana jones: the last crusade",
          "irish man",
          "mission impossible",
          "the devil wears prada",
          "parasite (gisaengchung)",
          "once upon a time in hollywood")
```

```
# 각 영화제목의 첫글자를 대문자로 변경
# \b는 단어의 양쪽 가장 자리의 빈 문자를 의미
# \\1은 () 첫 번째 그룹, \\U는 대문자(perl)
gsub("\b(\w)", "\U\1", movie, perl = T)
```

```
[1] "Terminator 3: Rise Of The Machiens" "Les Miserables"
[3] "Avengers: Infinity War"           "Iron Man"
[5] "Indiana Jones: The Last Crusade"   "Irish Man"
[7] "Mission Impossible"                 "The Devil Wears Prada"
[9] "Parasite (Gisaengchung)"          "Once Upon A Time In Hollywood"
```

```
# 엑셀에서 ()로 표시된 음수 데이터를 읽어온 경우
# 이를 음수로 표시
num <- c("0.123", "0.456", "(0.45)", "1.25")
gsub("\\(([0-9.]+)\\)", "-\\1", num)
```

```
[1] "0.123" "0.456" "-0.45" "1.25"
```

3.3.4 정규 표현식 예시

- 텍스트 데이터를 처리할 때 일반적으로 많이 활용되는 정규 표현식
- 정제되지 않은 데이터 가공 시 유용하게 활용

공백 제거

- 선행 예제에서 문자열 뒤에 존재하는 공백 제거 예시 확인
- 다음 예시들은 선행 및 모든 공백 제거에 대한 정규 표현식에 대해 살펴봄

필요 표현식

- 공백을 다른 문자로 교체 해주는 함수 → gsub()
- 공백 character class: \\s
- 처음과 끝 지정 meta character: ^, \$

4) 조건을 찾기 위한 meta character: +, |

- 모든 공백을 제거하려면 → `\s+`
- 앞쪽 공백만 제거하려면? → `^\\s+`
- 뒤쪽 공백만 제거하려면? → `\\s+\$`
- 양쪽 공백 모두를 제거하려면? 문장의 맨 앞에 공백이 하나 이상 존재하거나
(OR, |), 문장 맨 끝에 공백이 하나 이상 존재 → `(^\\s+|\\s+\$)`

```
txt <- c("신종 코로나바이러스 감염증(코로나19) 환자 가운데 회복해서 항체가
생긴 사람 중 절반가량은 체내에 바이러스가 남아 있는 것으로 나타났다. ")
```

```
txt
```

```
[1] "신종 코로나바이러스 감염증(코로나19) 환자 가운데 회복해서 항체가
\n생긴 사람 중 절반가량은 체내에 바이러스가 남아 있는 것으로 나타났다. "
```

```
# 모근 공백 제거
gsub("\\s+", "", txt)
```

```
[1] "신종코로나바이러스감염증(코로나19)환자가운데회복해서항체가생긴사람중절반가량은체내에
바이러스가남아있는것으로나타났다."
```

```
# 앞쪽 공백만 제거
gsub("^\\s+", "", txt)
```

```
[1] "신종 코로나바이러스 감염증(코로나19) 환자 가운데 회복해서 항체가\n생긴 사
람 중 절반가량은 체내에 바이러스가 남아 있는 것으로 나타났다. "
```

```
# 뒤쪽 공백만 제거
gsub("\\s+\$", "", txt)
```

```
[1] "신종 코로나바이러스 감염증(코로나19) 환자 가운데 회복해서 항체가
\n생긴 사람 중 절반가량은 체내에 바이러스가 남아 있는 것으로 나타났다."
```

```
# 양쪽에 존재하는 공백들 제거
gsub("(^\s+|\s+$)", "", txt)
```

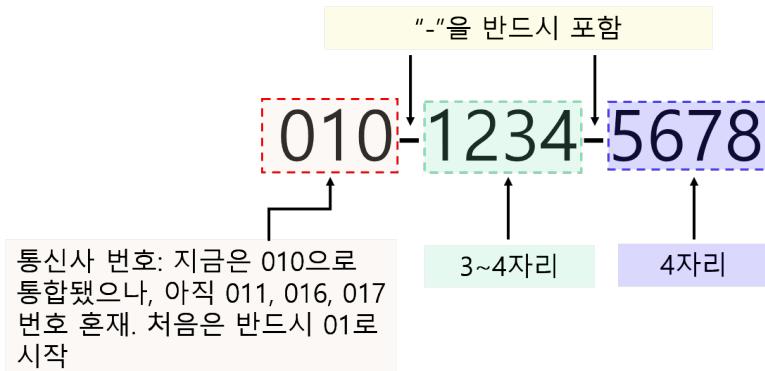
[1] "신종 코로나바이러스 감염증(코로나19) 환자 가운데 회복해서 항체가\n생긴 사람 중 절반가량은 체내에 바이러스가 남아 있는 것으로 나타났다."

```
# 가운데 \n 뒤에 존재하는 공백들을 없애려면??
gsub("(^\s+| {2,}|\s+$)", "", txt)
```

[1] "신종 코로나바이러스 감염증(코로나19) 환자 가운데 회복해서 항체가\n생긴 사람 중 절반가량은 체내에 바이러스가 남아 있는 것으로 나타났다."

핸드폰 번호 추출

- 대한민국 핸드폰 번호의 형태



필요한 표현식

- 처음 세 자리: ^{01}\d{1}
- 가운데 3~4자리: -\d{3,4}
- 마지막 4자리: -\d{4}

```
phone <- c("042-868-9999", "02-3345-1234",
          "010-5661-7578", "016-123-4567",
```

```

"063-123-5678", "070-5498- 1904",
"011-423-2912", "010-6745-2973")

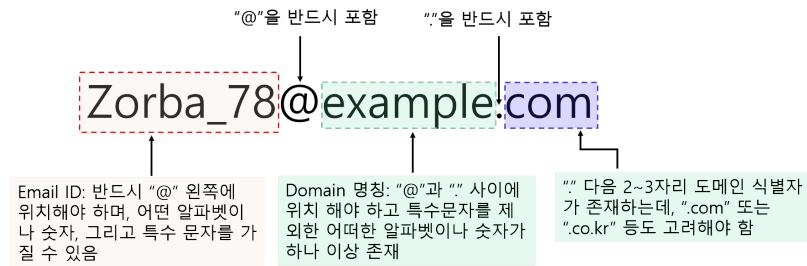
g <- grep1("^(01)\\d{1}-\\d{3,4}-\\d{4}", phone)
phone[g]

[1] "010-5661-7578" "016-123-4567"  "011-423-2912"  "010-6745-2973"

```

이메일 추출

- 정규 표현식을 이용해 이메일(e-mail) 주소만 텍스트 문서에서 추출
- 이메일 주소 구성



필요한 표현식

- E-mail ID(@ 왼쪽): 어떤 알파벳, 숫자, 특수문자가 1개 이상 → `[A-Za-z0-9[:punct:]]+`
- E-mail ID(@ 오른쪽-1): 어떤 알파벳이나 숫자가 하나 이상 존재하고 특수 문자 포함 (.xx.xx 추출에 필요) → `@[A-Za-z0-9[:punct:]]+`
- E-mail ID(@ 오른쪽-2): .xx 또는 .xxx 검색 → `\.\.[A-Za-z]+`

예시

- 네이버 뉴스 크롤링⁴ (nav, 2018)

⁴<https://dr-hkim.github.io/Naver-News-Web-Scraping-using-Keywords-in-R/>

- 검색 포털: 네이버
- 검색범위: 2020년 4월 21일
- 검색 keyword: 21대 국회위원 선거
- 검색 뉴스 개수: 39개
- 검색결과 저장 파일: dataset/news-scraping.csv
- 개별 기사에 해당하는 URL로부터 ID 생성
- 각 뉴스로부터 기자들의 e-mail 추출
- 추출 후 기사 ID, 기사제목, e-mail 주소로 구성된 데이터 프레임 생성

```
# 크롤링한 데이터 불러오기
news_naver <- read.csv("dataset/news-scraping.csv", header = T,
                      stringsAsFactors = FALSE)

# regmatches 함수: regexpr(), gregexpr(), regexec()로 검색한 패턴을
# 텍스트(문자열)에서 추출

# ID 추출
id <- regmatches(news_naver$url, regexpr("\d{10}", news_naver$url))
contents <- news_naver$news_content
news_naver2 <- data.frame(id, title = news_naver$news_title,
                           stringsAsFactors = FALSE)
tmp <- regmatches(contents,
                  gregexpr("\b[A-Za-z0-9[:punct:]]+@[A-Za-z0-9[:punct:]]+\.\w+", contents))
names(tmp) <- id
x <- t(sapply(tmp, function(x) x[1:2], simplify = "array"))
colnames(x) <- paste0("email", 1:2)
email <- data.frame(id = row.names(x), x, stringsAsFactors = F)
res <- merge(news_naver2, email, by = "id")
head(res)
```

id

```

1 0000026769
2 0000091243
3 0000425146
4 0000425152
5 0000489588
6 0000535054

```

| title | |
|-------|--|
| 1 | 그림으로 보는 제21대 국회의원 선거 결과 |
| 2 | '스트레이트' 꼼수로 얼룩진 21대 총선… 후퇴한 정치개혁 |
| 3 | [뉴스1번지] 20대 국회 막판 달구는 긴급재난지원금 논쟁 |
| 4 | [뉴스1번지] 주호영 당선인에게 듣는 슬기로운 국회생활 |
| 5 | [엄창섭의 몸과 삶] 나쁜 유전자, 나쁜 국회의원 |
| 6 | 고용진·전재수·유동수 의원 다시 금배지…'보험업계 숙원법안' 21대 국회 통과 기대 |
| | email1 email2 |
| 1 | cyr@sisain.co.kr <NA> |
| 2 | <NA> <NA> |
| 3 | <NA> <NA> |
| 4 | <NA> <NA> |
| 5 | <NA> <NA> |
| 6 | huropa@inews24.com <NA> |

```
# stringr 패키지 사용
```

```

# email <- str_extract_all(contents,
#                           "\b[A-Za-z0-9[:punct:]]+@[A-Za-z0-9[:punct:]]+\.\[A-Za-z\]+",
#                           simplify = TRUE)

# email <- data.frame(email, stringsAsFactors = FALSE)
# names(email) <- paste0("email", 1:2)

# res <- data.frame(id, title = news_naver$news_title, email,
#                     stringsAsFactors = FALSE)

# head(res)

```



4

데이터 핸들링 (Data handling)



학습 목표

- Hadley Wickham이 개발한 데이터의 전처리 및 시각화를 위해 각광받는 tidyverse 패키지에 대해 알아본다
- 데이터를 읽고, 저장하고, 목적에 맞게 가공하고, tidyverse 하에서 반복 계산 방법에 대해 알아본다.

데이터 분석과정

- 1) 데이터를 R 작업환경 (workspace)에 불러오고 (**import**),
- 2) 불러온 데이터를 가공하고 (**data management, data preprocessing**),
- 3) 가공한 데이터를 분석 (**analysis, modeling**) 및 시각화 (**visualization**) 후,
- 4) 분석 결과를 저장 (**save**) 및 외부 파일로 내보낸 (**export**) 후,
- 5) 이를 통해 전문가와 소통 (**communicate**)

R의 데이터 가공(관리) 방법

1. 기본 R을 활용: 지금까지 배워온 방법으로 분석을 위한 데이터 가공 (색인, 필터, 병합 등)

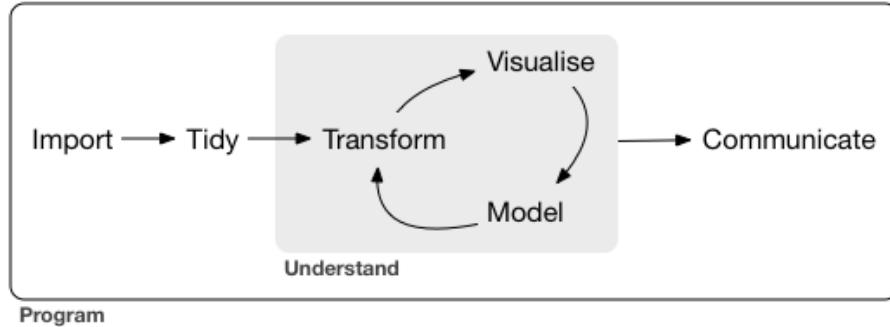


FIGURE 4.1: Data 분석의 과정. @wickham-2016r에서 발췌

2. tidyverse 패키지 활용

- 직관적 코드 작성 가능
- 빠른 실행속도

3. data.table 패키지 활용(본 강의에서는 다루지 않음)

- 빠른 실행속도

다양한 통계 함수와 최신 분석에 대한 여러 패키지 및 함수를 R 언어를 통해 활용 가능함에도 불구하고, 타 통계 소프트웨어(SAS, SPSS, Stata, Minitab 등)에 비해 데이터 가공 및 처리가 직관적이지 않고 불편했던 점은 R이 갖고 있던 큰 단점 중 하나임. RStudio의 수석 데이터 과학자인 Hadley Wickham의 tidyverse는 이러한 단점을 최대한 보완했고, 현재는 R을 통한 데이터 분석에서 핵심적인 도구로 자리매김 하고 있음. Tidyverse의 철학은 R 언어의 생태계에 혁신적인 변화를 가져왔을 뿐 아니라 지속적으로 진화하고 있기 때문에 해당 패키지들이 제공하는 언어 형태를 이해할 필요가 있음.

4.1 Prerequisites

4.1.1 외부데이터 불러오기 및 저장



R 기본 함수를 이용해서 데이터 저장 파일의 가장 기본적인 형태인 텍스트 파일을 읽고 저장하는 방법에 대해 먼저 살펴봄. Base R에서 외부 데이터를 읽고 저장을 위한 함수는 매우 다양하지만 가장 많이 사용되고 있는 함수들에 대해 살펴볼 것임

- 기본 R(base R)에서 제공하는 함수를 이용해 외부 데이터를 읽고, 내보내고, 저장하는 방법에 대해 살펴봄.
- 가장 일반적인 형태의 데이터는 보통 텍스트 파일 형태로 저장되어 있음, 일반적으로
 - 첫 번째 줄: 변수명
 - 두 번째 줄부터: 데이터 입력

```
id sex age edulev height
1 Male 65 12 168
2 Female 74 9 145
3 Male 61 12 171
4 Male 85 6 158
5 Female 88 0 134
```

- 데이터의 자료값과 자료값을 구분하는 문자를 구분자(separator)라고 하며 주로 공백(), 콤마(,), tab 문자(\t) 등이 사용됨
- 주로 확장자 명이 *.txt 이며, 콤마 구분자인 경우 보통은 *.csv (comma separated values)로 저장

```
#titanic3.csv 파일 일부
```

```
"pclass","survived","name","sex","age",
1,1,"Allen, Miss. Elisabeth Walton","female"
1,1,"Allison, Master. Hudson Trevor","male"
1,0,"Allison, Miss. Helen Loraine", "female"
1,0,"Allison, Mr. Hudson Joshua Creighton","male"
1,0,"Allison, Mrs. Hudson J C (Bessie Waldo Daniels)","female"
```

텍스트 파일 입출력



외부 데이터를 불러온다는 것은 외부에 저장되어 있는 파일을 R 작업환경으로 읽어온다는 의미이기 때문에, 현재 작업공간의 작업 디렉토리 (working directory) 확인이 필요.

- `read.table()/write.table()`:
 - 가장 범용적으로 외부 텍스트 데이터를 R 작업공간으로 데이터 프레임으로 읽고 저장하는 함수
 - 텍스트 파일의 형태에 따라 구분자 지정 가능

```
# read.table(): 텍스트 파일 읽어오기
read.table(
  file, # 파일명. 일반적으로 폴더명 구분자
  # 보통 folder/파일이름.txt 형태로 입력
  header = FALSE, # 첫 번째 행을 헤더(변수명)으로 처리할지 여부
  sep = "", # 구분자 ",", "\t" 등의 형태로 입력
  comment.char = "#", # 주석문자 지정
  stringsAsFactors = TRUE, # 문자형 변수를 factor으로 변환할지 여부
  encoding = "unknown" # 텍스트의 encoding 보통 CP949 또는 UTF-8
  # 한글이 입력된 데이터가 있을 때 사용
)
```

- `read.table()` 예시



예시에 사용된 데이터들은 Clinical trial data analysis using R (Chen and Peace, 2010)에서 제공되는 데이터임.

```
# tab 구분자 데이터 불러오기  
# dataset 폴더에 저장되어 있는 DBP.txt 파일 읽어오기  
dbp <- read.table("dataset/DBP.txt", sep = "\t", header = TRUE)  
str(dbp)
```

```
'data.frame': 40 obs. of 9 variables:  
 $ Subject: int 1 2 3 4 5 6 7 8 9 10 ...  
 $ TRT     : Factor w/ 2 levels "A","B": 1 1 1 1 1 1 1 1 1 1 ...  
 $ DBP1    : int 114 116 119 115 116 117 118 120 114 115 ...  
 $ DBP2    : int 115 113 115 113 112 112 111 115 112 113 ...  
 $ DBP3    : int 113 112 113 112 107 113 100 113 113 108 ...  
 $ DBP4    : int 109 103 104 109 104 104 109 102 109 106 ...  
 $ DBP5    : int 105 101 98 101 105 102 99 102 103 97 ...  
 $ Age     : int 43 51 48 42 49 47 50 61 43 51 ...  
 $ Sex     : Factor w/ 2 levels "F","M": 1 2 1 1 2 2 1 2 2 2 ...
```

```
# 문자형 값들을 factor로 변환하지 않는 경우  
dbp2 <- read.table("dataset/DBP.txt", sep = "\t",  
                     header = TRUE,  
                     stringsAsFactors = FALSE)  
str(dbp2)
```

```
'data.frame': 40 obs. of 9 variables:  
 $ Subject: int 1 2 3 4 5 6 7 8 9 10 ...  
 $ TRT     : chr "A" "A" "A" "A" ...  
 $ DBP1    : int 114 116 119 115 116 117 118 120 114 115 ...  
 $ DBP2    : int 115 113 115 113 112 112 111 115 112 113 ...  
 $ DBP3    : int 113 112 113 112 107 113 100 113 113 108 ...  
 $ DBP4    : int 109 103 104 109 104 104 109 102 109 106 ...
```

```
$ DBP5    : int 105 101 98 101 105 102 99 102 103 97 ...
$ Age     : int 43 51 48 42 49 47 50 61 43 51 ...
$ Sex     : chr "F" "M" "F" "F" ...
```

```
# 데이터 형태 파악
head(dbp)
```

| | Subject | TRT | DBP1 | DBP2 | DBP3 | DBP4 | DBP5 | Age | Sex | |
|---|---------|-----|------|------|------|------|------|-----|-----|---|
| 1 | | 1 | A | 114 | 115 | 113 | 109 | 105 | 43 | F |
| 2 | | 2 | A | 116 | 113 | 112 | 103 | 101 | 51 | M |
| 3 | | 3 | A | 119 | 115 | 113 | 104 | 98 | 48 | F |
| 4 | | 4 | A | 115 | 113 | 112 | 109 | 101 | 42 | F |
| 5 | | 5 | A | 116 | 112 | 107 | 104 | 105 | 49 | M |
| 6 | | 6 | A | 117 | 112 | 113 | 104 | 102 | 47 | M |

```
# 콤마 구분자 데이터 불러오기
# dataset 폴더에 저장되어 있는 diabetes_csv.txt 파일 읽어오기
diab <- read.table("dataset/diabetes_csv.txt", sep = ",", header = TRUE)
str(diab)
```

```
'data.frame': 403 obs. of 19 variables:
 $ id      : int 1000 1001 1002 1003 1005 1008 1011 1015 1016 1022 ...
 $ chol    : int 203 165 228 78 249 248 195 227 177 263 ...
 $ stab.glu: int 82 97 92 93 90 94 92 75 87 89 ...
 $ hdl     : int 56 24 37 12 28 69 41 44 49 40 ...
 $ ratio   : num 3.6 6.9 6.2 6.5 8.9 ...
 $ glyhb  : num 4.31 4.44 4.64 4.63 7.72 ...
 $ location: Factor w/ 2 levels "Buckingham","Louisa": 1 1 1 1 1 1 1 1 1 ...
 $ age     : int 46 29 58 67 64 34 30 37 45 55 ...
 $ gender  : Factor w/ 2 levels "female","male": 1 1 1 2 2 2 2 2 2 1 ...
 $ height  : int 62 64 61 67 68 71 69 59 69 63 ...
 $ weight  : int 121 218 256 119 183 190 191 170 166 202 ...
 $ frame   : Factor w/ 4 levels "", "large", "medium", ...: 3 2 2 2 3 2 3 3 2 4 ...
 $ bp.1s   : int 118 112 190 110 138 132 161 NA 160 108 ...
```

```
$ bp.1d    : int  59 68 92 50 80 86 112 NA 80 72 ...
$ bp.2s    : int  NA NA 185 NA NA NA 161 NA 128 NA ...
$ bp.2d    : int  NA NA 92 NA NA NA 112 NA 86 NA ...
$ waist   : int  29 46 49 33 44 36 46 34 34 45 ...
$ hip     : int  38 48 57 38 41 42 49 39 40 50 ...
$ time.ppn: int  720 360 180 480 300 195 720 1020 300 240 ...
```

head(diab)

| | id | chol | stab.glu | hdl | ratio | glyhb | location | age | gender | height | weight | frame | |
|---|-------|-------|----------|-------|-------|-------|----------|------------|--------|--------|--------|-------|--------|
| 1 | 1000 | 203 | | 82 | 56 | 3.6 | 4.31 | Buckingham | 46 | female | 62 | 121 | medium |
| 2 | 1001 | 165 | | 97 | 24 | 6.9 | 4.44 | Buckingham | 29 | female | 64 | 218 | large |
| 3 | 1002 | 228 | | 92 | 37 | 6.2 | 4.64 | Buckingham | 58 | female | 61 | 256 | large |
| 4 | 1003 | 78 | | 93 | 12 | 6.5 | 4.63 | Buckingham | 67 | male | 67 | 119 | large |
| 5 | 1005 | 249 | | 90 | 28 | 8.9 | 7.72 | Buckingham | 64 | male | 68 | 183 | medium |
| 6 | 1008 | 248 | | 94 | 69 | 3.6 | 4.81 | Buckingham | 34 | male | 71 | 190 | large |
| | bp.1s | bp.1d | bp.2s | bp.2d | waist | hip | time.ppn | | | | | | |
| 1 | 118 | 59 | NA | NA | 29 | 38 | 720 | | | | | | |
| 2 | 112 | 68 | NA | NA | 46 | 48 | 360 | | | | | | |
| 3 | 190 | 92 | 185 | 92 | 49 | 57 | 180 | | | | | | |
| 4 | 110 | 50 | NA | NA | 33 | 38 | 480 | | | | | | |
| 5 | 138 | 80 | NA | NA | 44 | 41 | 300 | | | | | | |
| 6 | 132 | 86 | NA | NA | 36 | 42 | 195 | | | | | | |

Encoding이 다른 경우(한글데이터 포함):

한글자 사전 데이터 (CP949 encoding으로 저장)

```
# tab 구분자 데이터 사용
```

UTF-8로 읽어오기

| | herb | herb_normal |
|---|----------------------------------|----------------------------------|
| 1 | <U+02B8><ed><U+00AD> | <U+02B8><ed><U+00AD> |
| 2 | <U+02B8><ed><U+00AD><eb><U+00BF> | <U+02B8><ed><U+00AD><eb><U+00BF> |
| 3 | <U+02B8><ed><U+00AD><f9><U+00AB> | <U+02B8><ed><U+00AD><f9><U+00AB> |
| 4 | <d3><d7><cf><fd> | <d3><d7><cf><fd> |
| 5 | <d3><d7><cf><fd><U+06AD> | <d3><d7><cf><fd> |
| 6 | <d3><d7><cf><fd><e3><f3> | <d3><d7><cf><fd> |
| | korean | |
| 1 | <U+00B0><U+00A1><c0><da> | |
| 2 | <U+00B0><U+00A1><c0><da><U+0030> | |
| 3 | <U+00B0><U+00A1><c0><da><c7><c7> | |
| 4 | <U+00B4><e7><U+00B1><cd> | |
| 5 | <U+00B4><e7><U+00B1><cd> | |
| 6 | <U+00B4><e7><U+00B1><cd> | |

```
# CP949로 읽어오기
herb <- read.table("dataset/herb_dic_sample.txt", sep = "\t",
                     header = TRUE,
                     encoding = "CP949",
                     stringsAsFactors = FALSE)
head(herb)
```

| | herb | herb_normal | korean |
|---|------|-------------|--------|
| 1 | 詞子 | 詞子 | 가자 |
| 2 | 詞子肉 | 詞子肉 | 가자육 |
| 3 | 詞子皮 | 詞子皮 | 가자피 |
| 4 | 當歸 | 當歸 | 당귀 |
| 5 | 當歸尾 | 當歸尾 | 당귀 |
| 6 | 當歸身 | 當歸身 | 당귀 |

- `read.table()` + `textConnection()`: 웹사이트나 URL에 있는 데이터를 Copy + Paste 해서 읽어올 경우 유용하게 사용

- `textConnection()`: 텍스트에서 한 줄씩 읽어 문자형 벡터처럼 인식 할 수 있도록 해주는 함수

```
# Plasma 데이터: http://lib.stat.cmu.edu/datasets/Plasma_Retinol



input2 <- ("AGE: Age (years)



SEX: Sex (1=Male, 2=Female).



SMOKSTAT: Smoking status (1=Never, 2=Former, 3=Current Smoker)



QUETELET: Quetelet (weight/(height2))



VITUSE: Vitamin Use (1=Yes, fairly often, 2=Yes, not often, 3>No)



CALORIES: Number of calories consumed per day.



FAT: Grams of fat consumed per day.



FIBER: Grams of fiber consumed per day.



ALCOHOL: Number of alcoholic drinks consumed per week.



CHOLESTEROL: Cholesterol consumed (mg per day).



BETADIET: Dietary beta-carotene consumed (mcg per day).



RETDIET: Dietary retinol consumed (mcg per day)



BETAPLASMA: Plasma beta-carotene (ng/ml)



RETPLASMA: Plasma Retinol (ng/ml)")



```
plasma <- read.table(textConnection(input1), sep = "\t", header = F)
codebook <- read.table(textConnection(input2), sep = ":", header = F)
```


```

```
varname <- gsub("^\\s+", "", codebook$V1) # 변수명 지정

names(plasma) <- varname
head(plasma)
```

| | AGE | SEX | SMOKSTAT | QUETELET | VITUSE | CALORIES | FAT | FIBER | ALCOHOL | CHOLESTEROL |
|---|----------|---------|------------|-----------|--------|----------|------|-------|---------|-------------|
| 1 | 64 | 2 | 2 | 21.48380 | 1 | 1298.8 | 57.0 | 6.3 | 0.0 | 170.3 |
| 2 | 76 | 2 | 1 | 23.87631 | 1 | 1032.5 | 50.1 | 15.8 | 0.0 | 75.8 |
| 3 | 38 | 2 | 2 | 20.01080 | 2 | 2372.3 | 83.6 | 19.1 | 14.1 | 257.9 |
| 4 | 40 | 2 | 2 | 25.14062 | 3 | 2449.5 | 97.5 | 26.5 | 0.5 | 332.6 |
| 5 | 72 | 2 | 1 | 20.98504 | 1 | 1952.1 | 82.6 | 16.2 | 0.0 | 170.8 |
| 6 | 40 | 2 | 2 | 27.52136 | 3 | 1366.9 | 56.0 | 9.6 | 1.3 | 154.6 |
| | BETADIET | RETDIET | BETAPLASMA | RETPLASMA | | | | | | |
| 1 | 1945 | 890 | 200 | 915 | | | | | | |
| 2 | 2653 | 451 | 124 | 727 | | | | | | |
| 3 | 6321 | 660 | 328 | 721 | | | | | | |
| 4 | 1061 | 864 | 153 | 615 | | | | | | |
| 5 | 2863 | 1209 | 92 | 799 | | | | | | |
| 6 | 1729 | 1439 | 148 | 654 | | | | | | |

- `write.table()`: R의 객체(벡터, 행렬, 데이터 프레임)를 저장 후 외부 텍스트 파일로 내보내기 위한 함수

```
# write.table() R 객체를 텍스트 파일로 저장하기
write.table(
  data_obj, # 저장할 객체 이름
  file, # 저장할 위치 및 파일명 또는
  # 또는 "파일쓰기"를 위한 연결 명칭
  sep, # 저장 시 사용할 구분자
  row.names = TRUE # 행 이름 저장 여부
)
```

- 예시

```
# 위에서 읽어온 plasma 객체를 dataset/plasma.txt로 내보내기  
# 행 이름은 생략, tab으로 데이터 구분  
  
write.table(plasma, "dataset/plasma.txt",  
            sep = "\t", row.names = F)
```

- 파일명 대신 Windows clipboard로 내보내기 가능

```
# clipboard로 복사 후 excel 시트에 해당 데이터 붙여넣기  
# Ctrl + V  
  
write.table(plasma, "clipboard",  
            sep = "\t", row.names = F)
```

- `read.csv()`/`write.csv()`: `read.table()` 함수의 wrapper 함수로 구분자 인수 `sep`이 콤마(,)로 고정(예시 생략)

R 바이너리(binary) 파일 입출력

R 작업공간에 존재하는 한 개 이상의 객체들을 저장하고 읽기 위한 함수

- R 데이터 관련 바이너리 파일은 한 개 이상의 객체가 저장된 바이너리 파일인 경우 `*.Rdata` 형태를 갖고, 단일 객체를 저장할 경우 보통 `*.rds` 파일 확장자로 저장
- `*.Rdata` 입출력 함수
 - `load()`: `*.Rdata` 파일 읽어오기
 - `save()`: 한 개 이상 R 작업공간에 존재하는 객체를 `.Rdata` 파일로 저장
 - `save.image()`: 현재 R 작업공간에 존재하는 모든 객체를 `.Rdata` 파일로 저장

```

# 현재 작업공간에 존재하는 모든 객체를 "output" 폴더에 저장
# output 폴더가 존재하지 않는 경우 아래 명령 실행
# dir.create("output")
ls()

[1] "codebook"      "dbp"          "dbp2"         "def.chunk.hook"
[5] "diab"          "herb"         "hook_output"   "input1"
[9] "input2"         "plasma"       "varname"

save.image(file = "output/all_obj.Rdata")

rm(list = ls())
ls()

character(0)

# 저장된 binary 파일(all_obj.Rdata) 불러오기
load("output/all_obj.Rdata")
ls()

[1] "codebook"      "dbp"          "dbp2"         "def.chunk.hook"
[5] "diab"          "herb"         "hook_output"   "input1"
[9] "input2"         "plasma"       "varname"

# dnp, plasma 데이터만 output 폴더에 sub_obj.Rdata로 저장
save(dbp, plasma, file = "output/sub_obj.Rdata")
rm(list = c("dbp", "plasma"))

ls()

[1] "codebook"      "dbp2"         "def.chunk.hook" "diab"
[5] "herb"          "hook_output"   "input1"        "input2"
[9] "varname"

```

```
# sub_obj.Rdata 파일 불러오기
load("output/sub_obj.Rdata")
ls()
```

```
[1] "codebook"      "dbp"          "dbp2"         "def.chunk.hook"
[5] "diab"          "herb"         "hook_output"  "input1"
[9] "input2"         "plasma"       "varname"
```

- *.rds 입출력 함수

- readRDS() / saveRDS(): 단일 객체가 저장된 *.rds 파일을 읽거나 저장
- 대용량 데이터를 다룰 때 유용함
- read.table() 보다 데이터를 읽는 속도가 빠르며, 다른 확장자 명의 텍스트 파일보다 높은 압축율을 보임

```
# 대용량 파일 dataset/pulse.csv 불러오기
# system.time(): 명령 실행 시가 계산 함수
system.time(pulse <- read.csv("dataset/pulse.csv", header = T))
```

사용자 시스템 elapsed

20.64 0.04 20.70

```
# saveRDS() 함수를 이용해 output/pulse.rds 파일로 저장
saveRDS(pulse, "output/pulse.rds")
rm(pulse); ls()
```

```
[1] "codebook"      "dbp"          "dbp2"         "def.chunk.hook"
[5] "diab"          "herb"         "hook_output"  "input1"
[9] "input2"         "plasma"       "varname"
```

```
system.time(pulse <- readRDS("output/pulse.rds"))
```

사용자 시스템 elapsed

0.08 0.00 0.08

4.2 Tidyverse

- “Tidy” + “Universe”의 조어로 “tidy data”의 기본 설계 철학, 문법 및 데이터 구조를 공유하는 RStudio 수석 과학자인 Hadley Wickham이 개발한 패키지 묶음(변들) 또는 메타 패키지로, 데이터 과학(data science)을 위한 R package를 표방 (Wickham, 2019b)
- 데이터 분석 과정 중 가장 긴 시간을 할애하는 데이터 전처리 (data pre-processing, data management, data wrangling, data munging 등으로 표현)를 위한 다양한 함수들을 제공하며, 특히 파이프(pipe) 연산자로 지칭되는 `%>%`를 통한 코드의 간결성 및 가독성을 최대화 하는 것이 tidyverse 패키지들의 특징
- Hadley Wickham이 주창한 Tidy Tools Manifesto¹에 따르면, tidyverse 가 추구하는 프로그래밍 인터페이스에 대한 4 가지 원칙을 제시

- 1) 기존 데이터의 구조를 재사용
- 2) 파이프 연산자를 이용한 최대한 간결한 함수 작성
- 3) R의 특징 중 하나인 functional programming 수용
- 4) 사람이 읽기 쉬운 프로그램으로 설계

- Tidyverse를 구성하는 주요 패키지(알파벳 순)

¹<https://mran.microsoft.com/web/packages/tidyverse/vignettes/manife sto.html>

- 1) **dplyr**: 가장 일반적인 데이터 가공 및 처리 해결을 위한 “동사”(함수)로 구성된 문법 제공
- 2) **forcats**: 범주형 변수 처리를 위해 Rdmrl factor와 관련된 일반적인 문제 해결을 위한 함수 제공
- 3) **ggplot2**: 그래픽 문법을 기반으로 2차원 그래픽을 생성하기 위해 고안된 시스템
- 4) **purrr**: 함수 및 벡터의 반복 작업을 수행할 수 있는 도구를 제공
- 5) **readr**: base R에서 제공하는 파일 입출력 함수보다 효율적인 성능을 갖는 입출력 함수로 구성
- 6) **stringr**: 가능한 한 쉬운 방법으로 문자열을 다룰 수 있는 함수 제공
- 7) **tibble**: Tidyverse에서 재해석한 데이터 프레임 형태로 tidyverse에서 다루는 데이터의 기본 형태
- 8) **tidyverse**: 데이터를 정리하고 “tidy data”를 도출하기 위한 일련의 함수 제공



- 그 밖에 유용한 tidyverse에 소속되어 있는 패키지
- **haven**: 타 통계 프로그램(SAS, SPSS, Stata)의 데이터 포맷 입출력 함수 제공

- **readxl**: Excel 파일 입력 함수 제공
- **lubridate**: 시간(년/월/일/시/분) 데이터 가공 및 연산 함수 제공
- **magrittr**: Tidyverse의 문법(함수)를 연결 시켜주는 파이프 연산자 제공.
예전에는 독립적인 패키지였으나 지금은 모든 tidyverse 패키지에 내장되어 있음

4.3 readr 패키지

- 기본적으로 4.1.1 절에서 학습했던 `read.table()`, `read.csv()`와 거의 동일하게 작동하지만, 읽고 저장하는 속도가 base R에서 제공하는 기본 입출력 함수보다 월등히 뛰어남. 최근 `readr` 패키지에서 제공하는 입출력 함수보다 더 빠르게 데이터 입출력이 가능한 `feather` 패키지 (Wickham, 2019a) 제공
- 데이터를 읽는 동안 사소한 문제가 있는 경우 해당 부분에 경고 표시 및 행, 관측 정보를 표시해줌 → 데이터 디버깅에 유용
- 주요 함수²
 - `read_table()`, `write_table()`
 - `read_csv()`, `write_csv()`
- `readr vignette`³을 통해 더 자세한 예시를 살펴볼 수 있음

```
read_csv(
  file, # 파일 명
  col_names = TRUE, # 첫 번째 행을 변수명으로 처리할 것인지 여부
```

²주요 함수들의 사용방법은 거의 유사하기 때문에 `read_csv()` 함수에 대해서만 살펴봄

³<https://cran.r-project.org/web/packages/readr/vignettes/readr.html>

```
# read.table(), read.csv()의 header 인수와 동일
col_types = NULL, # 열(변수)의 데이터 형 지정
            # 기본적으로 데이터 유형을 자동으로 감지하지만,
            # 입력 텍스트의 형태에 따라 데이터 유형을
            # 잘못 추측할 수 있기 때문에 간혹 해당 인수 입력 필요
            # col_* 함수 또는 compact string으로 지정 가능
            # c=character, i=integer, n=number, d=double,
            # l=logical, f=factor, D=date, T=date time, t=time
            # ?:guess, _/- skip column
progress, # 데이터 읽기/쓰기 진행 progress 표시 여부
)
```

- 예시

```
# dataset/titanic3.csv 불러오기
titanic <- read_csv("dataset/titanic3.csv")
```

```
Parsed with column specification:
cols(
  pclass = col_double(),
  survived = col_double(),
  name = col_character(),
  sex = col_character(),
  age = col_double(),
  sibsp = col_double(),
  parch = col_double(),
  ticket = col_character(),
  fare = col_double(),
  cabin = col_character(),
  embarked = col_character(),
  boat = col_character(),
  body = col_double(),
  home.dest = col_character()
```

)

titanic

```
# A tibble: 1,309 x 14
  pclass survived name   sex   age sibsp parch ticket   fare cabin embarked
  <dbl>     <dbl> <chr> <chr> <dbl> <dbl> <dbl> <chr> <dbl> <chr> <chr>
1     1       1 Allen, Miss. Elisabeth Walton female 29.00
2     1       1 Allison, Master. Hudson Trevor male 0.92
3     1       0 Allison, Miss. Helen Loraine female 2.00
4     1       0 Allison, Mr. Hudson Joshua Creighton male 30.00
5     1       0 Allison, Mrs. Hudson J C (Bessie Waldo Daniels) female 25.00
6     1       1 Anderson, Mr. Harry male 48.00
7     1       1 Andrews, Miss. Kornelia Theodosia female 63.00
8     1       0 Andrews, Mr. Thomas Jr male 39.00
9     1       1 Appleton, Mrs. Edward Dale (Charlotte Lamson) female 53.00
10    1       0 Artagaveytia, Mr. Ramon male 71.00
# ... with 1,299 more rows, and 3 more variables: boat <chr>, body <dbl>,
#   home.dest <chr>
```

```
# read.csv와 비교
head(read.csv("dataset/titanic3.csv", header = T), 10)
```

| | pclass | survived | | name | sex | age |
|----|--------|----------|--|---|--------|-------|
| 1 | 1 | 1 | | Allen, Miss. Elisabeth Walton | female | 29.00 |
| 2 | 1 | 1 | | Allison, Master. Hudson Trevor | male | 0.92 |
| 3 | 1 | 0 | | Allison, Miss. Helen Loraine | female | 2.00 |
| 4 | 1 | 0 | | Allison, Mr. Hudson Joshua Creighton | male | 30.00 |
| 5 | 1 | 0 | | Allison, Mrs. Hudson J C (Bessie Waldo Daniels) | female | 25.00 |
| 6 | 1 | 1 | | Anderson, Mr. Harry | male | 48.00 |
| 7 | 1 | 1 | | Andrews, Miss. Kornelia Theodosia | female | 63.00 |
| 8 | 1 | 0 | | Andrews, Mr. Thomas Jr | male | 39.00 |
| 9 | 1 | 1 | | Appleton, Mrs. Edward Dale (Charlotte Lamson) | female | 53.00 |
| 10 | 1 | 0 | | Artagaveytia, Mr. Ramon | male | 71.00 |

| | sibsp | parch | ticket | fare | cabin | embarked | boat | body |
|----|-------|-------|----------|----------|---------------------------------|----------|------|------|
| 1 | 0 | 0 | 24160 | 211.3375 | B5 | S | 2 | NA |
| 2 | 1 | 2 | 113781 | 151.5500 | C22 C26 | S | 11 | NA |
| 3 | 1 | 2 | 113781 | 151.5500 | C22 C26 | S | | NA |
| 4 | 1 | 2 | 113781 | 151.5500 | C22 C26 | S | | 135 |
| 5 | 1 | 2 | 113781 | 151.5500 | C22 C26 | S | | NA |
| 6 | 0 | 0 | 19952 | 26.5500 | E12 | S | 3 | NA |
| 7 | 1 | 0 | 13502 | 77.9583 | D7 | S | 10 | NA |
| 8 | 0 | 0 | 112050 | 0.0000 | A36 | S | | NA |
| 9 | 2 | 0 | 11769 | 51.4792 | C101 | S | D | NA |
| 10 | 0 | 0 | PC 17609 | 49.5042 | | C | | 22 |
| | | | | | home.dest | | | |
| 1 | | | | | St Louis, MO | | | |
| 2 | | | | | Montreal, PQ / Chesterville, ON | | | |
| 3 | | | | | Montreal, PQ / Chesterville, ON | | | |
| 4 | | | | | Montreal, PQ / Chesterville, ON | | | |
| 5 | | | | | Montreal, PQ / Chesterville, ON | | | |
| 6 | | | | | New York, NY | | | |
| 7 | | | | | Hudson, NY | | | |
| 8 | | | | | Belfast, NI | | | |
| 9 | | | | | Bayside, Queens, NY | | | |
| 10 | | | | | Montevideo, Uruguay | | | |

```
# column type을 변경
titanic2 <- read_csv("dataset/titanic3.csv",
                      col_types = "iicfdiicdcfcic")
titanic2

# A tibble: 1,309 x 14
   pclass survived name    sex     age sibsp parch ticket   fare cabin embarked
   <int>     <int> <chr> <fct> <dbl> <int> <int> <chr> <dbl> <chr> <fct>
1      1        1 Alle~ fema~  29       0     0 24160  211.   B5     S
2      1        1 Alli~ male   0.92     1     2 113781 152.   C22 ~ S
```

```

3     1      0 Alli~ fema~  2      1      2 113781 152.  C22 ~ S
4     1      0 Alli~ male   30     1      2 113781 152.  C22 ~ S
5     1      0 Alli~ fema~  25     1      2 113781 152.  C22 ~ S
6     1      1 Ande~ male   48     0      0 19952   26.6 E12   S
7     1      1 Andr~ fema~  63     1      0 13502    78.0 D7   S
8     1      0 Andr~ male   39     0      0 112050   0 A36   S
9     1      1 Appl~ fema~  53     2      0 11769    51.5 C101   S
10    1      0 Arta~ male   71     0      0 PC 17~  49.5 <NA>  C
# ... with 1,299 more rows, and 3 more variables: boat <chr>, body <int>,
#   home.dest <chr>

```

```

# 특정 변수만 불러오기
titanic3 <- read_csv("dataset/titanic3.csv",
                      col_types = cols_only(
                        pclass = col_integer(),
                        survived = col_integer(),
                        sex = col_factor(),
                        age = col_double()
                      ))
titanic3

```

```

# A tibble: 1,309 x 4
  pclass survived sex       age
  <int>     <int> <fct>    <dbl>
1     1         1 female  29
2     1         1 male   0.92
3     1         0 female  2
4     1         0 male   30
5     1         0 female  25
6     1         1 male   48
7     1         1 female  63
8     1         0 male   39
9     1         1 female  53

```

```
10      1      0 male    71  
# ... with 1,299 more rows
```

```
# 대용량 데이터셋 읽어올 때 시간 비교  
# install.packages("feather") # feather package  
require(feather)
```

필요한 패키지를 로딩중입니다: **feather**

```
system.time(pulse <- read.csv("dataset/pulse.csv", header = T))
```

사용자 시스템 elapsed

```
20.31    0.03   20.34
```

```
write_feather(pulse, "dataset/pulse.feather")  
system.time(pulse <- readRDS("output/pulse.rds"))
```

사용자 시스템 elapsed

```
0.08    0.00    0.08
```

```
system.time(pulse <- read_csv("dataset/pulse.csv"))
```

Parsed with column specification:

```
cols(  
  .default = col_double()  
)
```

See `spec(...)` for full column specifications.

사용자 시스템 elapsed

```
14.80    0.00   14.81
```

```
system.time(pulse <- read_feather("dataset/pulse.feather"))
```

사용자 시스템 elapsed

```
0.31    0.00    0.31
```

4.3.1 Excel 파일 입출력

- R에서 기본적으로 제공하는 파일 입출력 함수는 대부분 텍스트 파일 (*.txt, *.csv, *.tsv⁴)을 대상으로 하고 있음
- readr** 패키지에서도 이러한 원칙은 유지됨
- Excel 파일을 R로 읽어오기(과거 방법)
 - *.xls 또는 *.xlsx 파일을 엑셀로 읽은 후 해당 데이터를 위 텍스트 파일 형태로 내보낸 후 해당 파일을 R로 읽어옴
 - xlsx 패키지 등을 이용해 엑셀 파일을 직접 읽어올 수 있으나, Java 기반으로 개발된 패키지이기 때문에 Java Runtime Environment를 운영체제에 설치해야만 작동
- 최근 tidyverse 중 하나인 **readxl** 패키지를 이용해 간편하게 R 작업환경에 엑셀 파일을 읽어오는 것이 가능 (Hadley Wickham이 개발…)
 - tidyverse의 한 부분임에도 불구하고 tidyverse 패키지 번들에는 포함되어 있지 않기 때문에 별도 설치 필요

readxl 패키지 구성 주요 함수

- read_xls()**, **read_xlsx()**, **read_excel**: 엑셀 파일을 읽어오는 함수로 각각 Excel 97 ~ 2003, Excel 2007 이상, 또는 버전 상관 없이 저장된 엑셀 파일에 접근함
- excel_sheets()**: 엑셀 파일 내 시트 이름 추출 → 한 엑셀 파일의 복수 시트에 데이터가 저장되어 있는 경우 활용
- 예시: 2020년 4월 23일 COVID-19 유병률 데이터 (Our World in Data⁵)

```
read_xlsx(
  path, # Excel 폴더 및 파일 이름
```

⁴tab separated values

⁵<https://github.com/owid/covid-19-data/tree/master/public/data>

```
sheet = NULL, # 불러올 엑셀 시트 이름  
# default = 첫 번째 시트  
col_names = TRUE, # read_csv()의 인수와 동일한 형태 입력  
col_types = NULL # read_csv()의 인수와 동일한 형태 입력  
)
```

```
# 2020년 4월 21일자 COVID-19 국가별 유병률 및 사망률 집계 자료  
# dataset/owid-covid-data.xlsx 파일 불러오기  
# install.packages("readxl")  
require(readxl)
```

필요한 패키지를 로딩중입니다: *readxl*

```
covid19 <- read_xlsx("dataset/covid-19-dataset/owid-covid-data.xlsx")  
covid19
```

```
# A tibble: 14,315 x 16  
# ... with 14,305 more rows, and 9 more variables:  
#   total_cases_per_million <dbl>, new_cases_per_million <dbl>,  
#   total_deaths_per_million <dbl>, new_deaths_per_million <dbl>,  
#   total_tests <dbl>, new_tests <dbl>, total_tests_per_thousand <dbl>,  
#   total_gdp <dbl>, new_gdp <dbl>, total_gdp_per_capita <dbl>,  
#   iso_code <chr>, location <chr>, date <chr>, total_cases <dbl>,  
#   new_cases <dbl>, total_deaths <dbl>, new_deaths <dbl>  
# ... with 14,305 more rows, and 9 more variables:  
#   total_cases_per_million <dbl>, new_cases_per_million <dbl>,  
#   total_deaths_per_million <dbl>, new_deaths_per_million <dbl>,  
#   total_tests <dbl>, new_tests <dbl>, total_tests_per_thousand <dbl>,  
#   total_gdp <dbl>, new_gdp <dbl>, total_gdp_per_capita <dbl>,  
#   iso_code <chr>, location <chr>, date <chr>, total_cases <dbl>,  
#   new_cases <dbl>, total_deaths <dbl>, new_deaths <dbl>
```

```
#   new_tests_per_thousand <dbl>, tests_units <chr>

# 여러 시트를 동시에 불러올 경우
# dataset/datR4CTDA.xlsx 의 모든 시트 불러오기
path <- "dataset/datR4CTDA.xlsx"
sheet_name <- excel_sheets(path)
dL <- lapply(sheet_name, function(x) read_xlsx(path, sheet = x))
names(dL) <- sheet_name

# Tidyverse 에서는? (맛보기)
path %>%
  excel_sheets %>%
  set_names %>%
  map(~read_xlsx(path = path, sheet = .x)) -> dL2
```

4.3.2 tibble 패키지

- `readr` 또는 `readxl` 패키지에서 제공하는 함수를 이용해 외부 데이터를 읽어온 후, 확인할 때 기존 데이터 프레임과 미묘한 차이점이 있다는 것을 확인
- 프린트된 데이터의 맨 윗 부분을 보면 A `tibble`: 데이터 차원 이 표시된 부분을 볼 수 있음
- `tibble`은 tidyverse 생태계에서 사용되는 데이터 프레임 → 데이터 프레임 을 조금 더 빠르고 사용하기 쉽게 수정한 버전의 데이터 프레임

`tibble` 생성하기

- 기본 R 함수에서 제공하는 `as.*` 계열 함수처럼 `as_tibble()` 함수를 통해 기존 일반적인 형태의 데이터 프레임을 `tibble`로 변환 가능

```
head(iris)
```

| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|--------------|-------------|--------------|-------------|---------|
| 1 | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 2 | 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| 3 | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4 | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5 | 5.0 | 3.6 | 1.4 | 0.2 | setosa |
| 6 | 5.4 | 3.9 | 1.7 | 0.4 | setosa |

```
as_tibble(iris)
```

```
# A tibble: 150 x 5
```

| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|----|--------------|-------------|--------------|-------------|---------|
| | <dbl> | <dbl> | <dbl> | <dbl> | <fct> |
| 1 | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 2 | 4.9 | 3 | 1.4 | 0.2 | setosa |
| 3 | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4 | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5 | 5 | 3.6 | 1.4 | 0.2 | setosa |
| 6 | 5.4 | 3.9 | 1.7 | 0.4 | setosa |
| 7 | 4.6 | 3.4 | 1.4 | 0.3 | setosa |
| 8 | 5 | 3.4 | 1.5 | 0.2 | setosa |
| 9 | 4.4 | 2.9 | 1.4 | 0.2 | setosa |
| 10 | 4.9 | 3.1 | 1.5 | 0.1 | setosa |

```
# ... with 140 more rows
```

- 개별 벡터로부터 tibble 생성 가능
- 방금 생성한 변수 참조 가능
- 문자형 변수가 입력된 경우 데이터 프레임과 다르게 별다른 옵션이 없어도
 강제로 factor로 형 변환을 하지 않음

```
# 벡터로부터 tibble 객체 생성
tibble(x = letters, y = rnorm(26), z = y^2)
```

```
# A tibble: 26 x 3
  x         y     z
  <chr>   <dbl> <dbl>
1 a      -0.0764 0.00584
2 b      -0.233  0.0545 
3 c       0.742  0.551 
4 d       1.25   1.56  
5 e       1.34   1.78  
6 f       0.828  0.685 
7 g      -0.540  0.292 
8 h      -1.88   3.52  
9 i       0.462  0.213 
10 j      0.642  0.412
# ... with 16 more rows
```

```
# 데이터 프레임으로 위와 동일하게 적용하면?
data.frame(x = letters, y = rnorm(26), z = y^2)
```

Error in data.frame(x = letters, y = rnorm(26), z = y^2): 객체 'y'를 찾을 수 없습니다

```
# 벡터의 길이가 다른 경우
# 길이가 1인 벡터는 재사용 가능
tibble(x = 1, y = rep(0:1, each = 4), z = 2)
```

```
# A tibble: 8 x 3
  x     y     z
  <dbl> <int> <dbl>
1 1     0     2
2 1     0     2
```

```

3     1     0     2
4     1     0     2
5     1     1     2
6     1     1     2
7     1     1     2
8     1     1     2

```

```

# 데이터 프레임과 마찬가지로 비정상적 문자를 변수명으로 사용 가능
# 역따옴표(``)
tibble(`2000` = "year",
      `:`)` = "smile",
      `:(` = "sad")

```

```

# A tibble: 1 x 3
`2000` `:`)` `:(`
<chr> <chr> <chr>
1 year   smile sad

```

- **tribble()** 함수 사용: transposed (전치된) tibble의 약어로 데이터를 직접 입력 시 유용

```

tribble(
  ~x, ~y, ~z,
  "M", 172, 69,
  "F", 156, 45,
  "M", 165, 73,
)

```

```

# A tibble: 3 x 3
  x       y     z
<chr> <dbl> <dbl>
1 M      172    69
2 F      156    45
3 M      165    73

```

tibble()과 **data.frame()**의 차이점

- 가장 큰 차이점은 데이터 처리의 속도 및 데이터의 프린팅
- tibble이 데이터 프레임 보다 간결하고 많은 정보 확인 가능
- `str()`에서 확인할 수 있는 데이터 유형 확인 가능

```
head(iris)
```

| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|--------------|-------------|--------------|-------------|---------|
| 1 | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 2 | 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| 3 | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4 | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5 | 5.0 | 3.6 | 1.4 | 0.2 | setosa |
| 6 | 5.4 | 3.9 | 1.7 | 0.4 | setosa |

```
dd <- as_tibble(iris)
dd
```

A tibble: 150 x 5

| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|----|--------------|-------------|--------------|-------------|---------|
| | <dbl> | <dbl> | <dbl> | <dbl> | <fct> |
| 1 | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 2 | 4.9 | 3 | 1.4 | 0.2 | setosa |
| 3 | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4 | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5 | 5 | 3.6 | 1.4 | 0.2 | setosa |
| 6 | 5.4 | 3.9 | 1.7 | 0.4 | setosa |
| 7 | 4.6 | 3.4 | 1.4 | 0.3 | setosa |
| 8 | 5 | 3.4 | 1.5 | 0.2 | setosa |
| 9 | 4.4 | 2.9 | 1.4 | 0.2 | setosa |
| 10 | 4.9 | 3.1 | 1.5 | 0.1 | setosa |

... with 140 more rows

4.4 *dplyr* 패키지



*dplyr*에서 제공하는 “동사”(함수)로 데이터(데이터 프레임) 전처리 방법에 대해 익힌다.

- *dplyr*은 tidyverse에서 데이터 전처리를 담당하는 패키지로 데이터 전처리 과정을 쉽고 빠르게 수행할 수 있는 함수로 구성된 패키지임.
- 데이터 핸들링을 위해 Hadley Wickham이 개발한 *plyr* 패키지를 최적화한 패키지로 C++로 코딩되어 성능이 *plyr*에 비해 월등히 우수함.
- base R에 존재하는 함수만으로도 전처리는 충분히 가능하나, *dplyr*은 아래와 같은 장점을 가짐
 - a. 파이프 연산자 (%>%)로 코드의 가독성 극대화
 - b. 코드 작성이 쉬움
 - 전통적인 R의 데이터 처리에서 사용되는 [, [[, \$와 같은 색인 연산자 최소화
 - *dplyr*은 몇 가지 “동사”를 조합하여 사용
 - c. RStudio를 사용할 경우 코드 작성이 빨라짐
 - d. 접근 방법이 SQL 문과 유사함
- *dplyr*은 초기 데이터 프레임만을 다루지만, *purrr* 패키지를 통해 행렬, 배열, 리스트 등에도 적용 가능
- *dplyr*에서 제공하는 가장 기본 “동사”는 다음과 같음
 - `filter()`: 각 행(row)을 조건에 따라 선택

- `arrange()`: 선택한 변수(column)에 해당하는 행(row)을 기준으로 정렬
- `select()`: 변수(column) 선택
- `mutate()`: 새로운 변수를 추가하거나 이미 존재하는 변수를 변환
- `summarize()` 또는 `summarise()`: 변수 집계(평균, 표준편차, 최댓값, 최솟값, …)
- `group_by()` : 위 열거한 모든 동사들을 그룹별로 적용
- base R 제공 함수와 비교

TABLE 4.1: dplyr 패키지 함수와 R base 패키지 함수 비교

| 동사(함수) | 내용 | R base 패키지 함수 |
|--------------------------|----------------|--|
| <code>filter()</code> | 행 추출 | <code>subset()</code> |
| <code>arrange()</code> | 내림차순/오름차순 정렬 | <code>order()</code> , <code>sort()</code> |
| <code>select()</code> | 열 선택 | <code>data[,</code> <code>c('var_name01',</code> <code>'var_name03')]</code> |
| <code>mutate()</code> | 열 추가 및 변환 | <code>transform()</code> |
| <code>summarise()</code> | 집계 | <code>aggregate()</code> |
| <code>group_by()</code> | 그룹별 집계 및 함수 적용 | |

- dplyr 기본 동사와 연동해서 사용되는 주요 함수
 - `slice()`: 행 색인을 이용한 추출 → `data[n:m,]`과 동일
 - `distinct()`: 행 레코드 중 중복 항복 제거 → base R 패키지의 `unique()` 함수와 유사
 - `sample_n()`, `sample_frac()`: 데이터 레코드를 랜덤하게 샘플링

- `rename()`: 변수명 변경
- `inner_join`, `right_join()`, `left_join()`, `full_join`: 두 데이터셋 병합 → `merge()` 함수와 유사
- `tally()`, `count()`, `n()`: 데이터셋의 행의 길이(개수)를 반환하는 함수로 (그룹별) 집계에 사용: `length()`, `nrow()`/`NROW()` 함수와 유사
- `*_all`, `*_at`, `*_if`: `dplyr`에서 제공하는 기본 동사(`group_by()` 제외) 사용 시 적용 범위를 설정해 기본 동사와 동일한 기능을 수행하는 함수



R에서 데이터 전처리 및 분석을 수행할 때, 간혹 동일한 이름의 함수명들이 중복된 채 R 작업공간에 읽어오는 경우가 있는데, 이 경우 가장 마지막에 읽어온 패키지의 함수를 작업공간에서 사용한다. 예를 들어 R base 패키지의 `filter()` 함수는 시계열 데이터의 노이즈를 제거하는 함수이지만, tidyverse 패키지를 읽어온 경우, dplyr 패키지의 `filter()` 함수와 이름이 중복되기 때문에 R 작업공간 상에서는 dplyr 패키지의 `filter()`가 작동을 함. 만약 stats 패키지의 `filter()` 함수를 사용하고자 하면 `stats::filter()`를 사용. 이를 더 일반화 하면 현재 컴퓨터 상에 설치되어 있는 R 패키지의 특정 함수는 `::` 연산자를 통해 접근할 수 있으며, `package_name::function_name()` 형태로 접근 가능함.

4.4.1 파이프 연산자: `%>%`

- Tidyverse 세계에서 `tidy`를 담당하는 핵심적인 함수
- 여러 함수를 연결(chain)하는 역할을 하며, 이를 통해 불필요한 임시변수를 정의할 필요가 없어짐
- `function_1(x) %>% function_2(y) = function_2(function_1(x), y)`: `function_1(x)`에서 반환한 값을 `function_2()`의 첫 번째 인자로 사용

- `x %>% f(y) %>% g(z) = ?`
- 기존 R 문법과 차이점
 - 기존 R: 동사(목적어, 주변수, 나머지 변수)
 - Pipe 연결 방식: 목적어 `%>%` 동사(주변수, 나머지 변수)
- 예시

```
# base R 문법 적용
print(head(iris, 4))
```

| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|--------------|-------------|--------------|-------------|---------|
| 1 | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 2 | 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| 3 | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4 | 4.6 | 3.1 | 1.5 | 0.2 | setosa |

```
# %>% 연산자 이용
iris %>% head(4) %>% print
```

| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|--------------|-------------|--------------|-------------|---------|
| 1 | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 2 | 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| 3 | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4 | 4.6 | 3.1 | 1.5 | 0.2 | setosa |

```
# setosa 종에 해당하는 변수들의 평균 계산
apply(iris[iris$Species == "setosa", -5], 2, mean)
```

| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width |
|--|--------------|-------------|--------------|-------------|
| | 5.006 | 3.428 | 1.462 | 0.246 |

```
# tidyverse의 pipe 연산자 이용
# require(tidyverse)

iris %>%
  filter(Species == "setosa") %>%
  select(-Species) %>%
  summarise_all(mean)

Sepal.Length Sepal.Width Petal.Length Petal.Width
1      5.006      3.428      1.462      0.246

# Homework #3 b-c 풀이를 위한 pipe 연산 적용
# df <- within(df, {
#   am <- factor(am, levels = 0:1,
#                 labels = c("automatic", "manual"))
# })
# ggregate(cbind(mpg, disp, hp, drat, wt, qsec) ~ am,
#           data = df,
#           mean)
# aggregate(cbind(mpg, disp, hp, drat, wt, qsec) ~ am,
#           data = df,
#           sd)

mtcars %>%
  mutate(am = factor(vs,
                     levels = 0:1,
                     labels = c("automatic", "manual"))) %>%
  group_by(am) %>%
  summarise_at(vars(mpg, disp:qsec),
               list(mean = mean,
                    sd = sd))

# A tibble: 2 x 13
```

```

am      mpg_mean disp_mean hp_mean drat_mean wt_mean qsec_mean mpg_sd disp_sd
<fct>    <dbl>     <dbl>    <dbl>     <dbl>    <dbl>     <dbl>    <dbl>    <dbl>
1 auto~     16.6     307.    190.      3.39     3.69     16.7     3.86    107.
2 manu~     24.6     132.    91.4      3.86     2.61     19.3     5.38    56.9
# ... with 4 more variables: hp_sd <dbl>, drat_sd <dbl>, wt_sd <dbl>,
#   qsec_sd <dbl>

```

4.4.2 filter()

행 (row, case, observation) 조작 동사

- 데이터 프레임 (또는 tibble)에서 특정 조건을 만족하는 레코드 (row) 추출

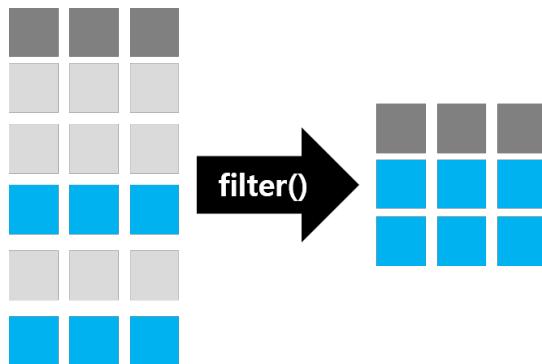


FIGURE 4.2: filter() 함수 다이어그램

- R base 패키지의 `subset()` 함수와 유사하게 작동하지만 성능이 더 좋음 (속도가 더 빠르다).
- 추출을 위한 조건은 2.1.4 절 논리형 스칼라에서 설명한 비교 연산자를 준용함. 단 `filter()` 함수 내에서 and (&) 조건은 ,(콤마, comma)로 표현 가능
- `filter()`에서 가능한 불린(boolean) 연산

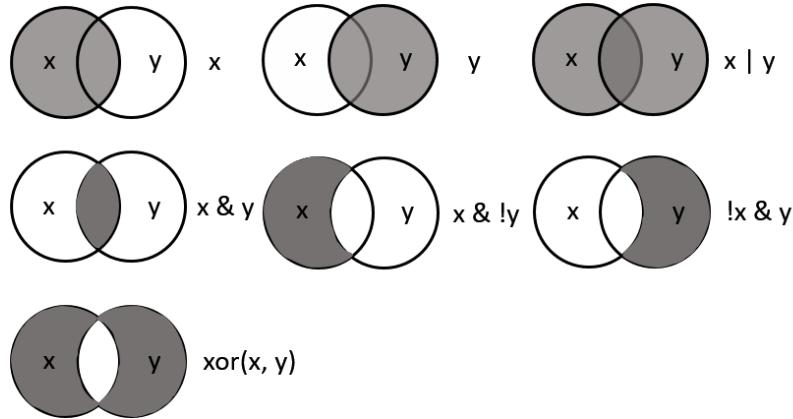


FIGURE 4.3: 가능한 모든 boolean 연산 종류: x는 좌변, y는 우변을 의미하고 음영은 연산 이후 선택된 부분을 나타냄.

```
# filter() 동사 prototype
dplyr::filter(x, # 데이터 프레임 또는 티블 객체
              condition_01, # 첫 번째 조건
              condition_02, # 두 번째 조건
              # 두 번째 인수 이후 조건들은
              # condition_1 & condition_2 & ... & condition_n 일
              ...)
```

- 예시 1: mpg 데이터 (ggplot2 패키지 내장 데이터)
 - mpg 데이터 코드북
 - 데이터 구조 확인을 위해 dplyr 패키지에서 제공하는 glimpse() 함수 (`str()` 유사) 사용

TABLE 4.2: mpg 데이터셋 설명(코드북)

| 변수명 | 변수설명(영문) | 변수설명(국문) |
|--------------|---|---|
| manufacturer | manufacturer name | 제조사 |
| model | model name | 모델명 |
| displ | engine displacement, in litres | 배기량(리터) |
| year | year of manufacture | 제조년도 |
| cyl | number of cylinders | 엔진 기통 수 |
| trans | type of transmission | 트렌스미션 |
| drv | the type of drive train, where f = front-wheel drive, r = rear wheel drive, 4 = 4wd | 구동 유형: f = 전륜구동, r = 후륜 구동, 4 = 4륜 구동 |
| cty | city miles per gallon | 시내 연비 |
| hwy | highway miles per gallon | 고속 연비 |
| fl | fuel type: e = E85, d = diesel, r = regular, p = premium, c = CNG | 연료: e = 에탄올 85%, r = 가솔린, p = 프리미엄, d = 디젤, c = CNG |
| class | 'type' of car | 자동차 타입 |

```
glimpse(mpg)
```

```
Observations: 234
Variables: 11
$ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi"...
$ model <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro"...
$ displ <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, ...
$ year <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, ...
$ cyl <int> 4, 4, 4, 4, 6, 6, 4, 4, 4, 6, 6, 6, 6, 6, 8, ...
```

```
$ trans      <chr> "auto(15)", "manual(m5)", "manual(m6)", "auto(av)", "a...
$ drv        <chr> "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", ...
$ cty        <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17...
$ hwy        <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25...
$ fl         <chr> "p", ...
$ class      <chr> "compact", "compact", "compact", "compact", "compact", ...
```

```
# 현대 차만 추출
## 기본 문법 사용
# mpg[mpg$manufacturer == "hyundai", ]
# subset(mpg, manufacturer == "hyundai")

## filter() 함수 사용
# filter(mpg, manufacturer == "hyundai")

## pipe 연산자 사용
mpg %>%
  filter(manufacturer == "hyundai")
```

```
# A tibble: 14 x 11
  manufacturer model  displ   year   cyl trans   drv     cty   hwy fl   class
  <chr>       <chr>   <dbl> <int> <int> <chr>   <chr> <int> <int> <chr> <chr>
1 hyundai     sonata    2.4  1999     4 auto(l~ f      18    26 r   midsize
2 hyundai     sonata    2.4  1999     4 manual~ f     18    27 r   midsize
3 hyundai     sonata    2.4  2008     4 auto(l~ f      21    30 r   midsize
4 hyundai     sonata    2.4  2008     4 manual~ f     21    31 r   midsize
5 hyundai     sonata    2.5  1999     6 auto(l~ f      18    26 r   midsize
6 hyundai     sonata    2.5  1999     6 manual~ f     18    26 r   midsize
7 hyundai     sonata    3.3  2008     6 auto(l~ f      19    28 r   midsize
8 hyundai     tibur~     2    1999     4 auto(l~ f      19    26 r   subcom~
9 hyundai     tibur~     2    1999     4 manual~ f     19    29 r   subcom~
10 hyundai    tibur~     2    2008     4 manual~ f     20    28 r   subcom~
11 hyundai    tibur~     2    2008     4 auto(l~ f      20    27 r   subcom~
```

```
12 hyundai      tibur~    2.7 2008      6 auto(l~ f          17     24 r      subcom~
13 hyundai      tibur~    2.7 2008      6 manual~ f        16     24 r      subcom~
14 hyundai      tibur~    2.7 2008      6 manual~ f        17     24 r      subcom~
```

```
# 시내 연비가 20 mile/gallon 이상이고 타입이 suv 차량 추출
```

```
## 기본 문법 사용
```

```
# mpg[mpg$cty >= 20 & mpg$class == "suv", ]
```

```
# subset(mpg, cty >= 20 & class == "suv")
```

```
## filter() 함수 사용
```

```
# filter(mpg, cty >= 20, class == "suv")
```

```
## pipe 연산자 사용
```

```
mpg %>%
```

```
filter(cty >= 20,
```

```
       class == "suv")
```

```
# A tibble: 2 x 11
```

| manufacturer | model | displ | year | cyl | trans | drv | cty | hwy | fl | class |
|--------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| <chr> | <chr> | <dbl> | <int> | <int> | <chr> | <chr> | <int> | <int> | <chr> | <chr> |

```
1 subaru      forester~    2.5 2008      4 manual~ 4          20     27 r      suv
2 subaru      forester~    2.5 2008      4 auto(l~ 4          20     26 r      suv
```

```
# 제조사가 audi 또는 volkswagen 이고 고속 연비가 30 miles/gallon 인 차량만 추출
```

```
mpg %>%
```

```
filter(manufacturer == "audi" | manufacturer == "volkswagen",
```

```
       hwy >= 30)
```

```
# A tibble: 5 x 11
```

| manufacturer | model | displ | year | cyl | trans | drv | cty | hwy | fl | class |
|--------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| <chr> | <chr> | <dbl> | <int> | <int> | <chr> | <chr> | <int> | <int> | <chr> | <chr> |

```
1 audi         a4        2   2008      4 manual~ f          20     31 p      compact
2 audi         a4        2   2008      4 auto(a~ f          21     30 p      compact
```

```
3 volkswagen    jetta      1.9  1999      4 manual~ f      33    44 d    compact
4 volkswagen    new be~    1.9  1999      4 manual~ f      35    44 d    subcom~
5 volkswagen    new be~    1.9  1999      4 auto(l~ f      29    41 d    subcom~
```

4.4.3 arrange()

행(row, case, observation) 조작 동사

- 지정한 열을 기준으로 데이터의 레코드(row)를 오름차순(작은 값부터 큰 값)으로 정렬
- 내림차순(큰 값부터 작은 값) 정렬 시 desc() 함수 이용

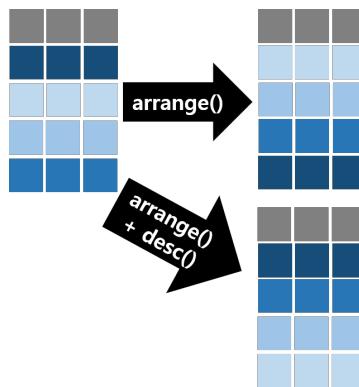


FIGURE 4.4: arrange() 함수 다이어그램

```
arrange(data, # 데이터 프레임 또는 티블 객체  
        var1, # 기준 변수 1  
        var2, # 기준 변수 2  
        ...)
```

- 예시 1: mpg 데이터셋

```
# 시내 연비를 기준으로 오름차순 정렬
## R 기본 문법 사용
# mpg[order(mpg$cty), ]

## arrange 함수 사용
# arrange(mpg, cty)

## pipe 사용
mpg_asc <- mpg %>% arrange(cty)
mpg_asc %>% print
```

```
# A tibble: 234 x 11
  manufacturer model      displ year   cyl trans drv   cty   hwy fl class
  <chr>        <chr>     <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
1 dodge        dakota p~    4.7  2008     8 auto(~ 4          9    12 e   pick~
2 dodge        durango ~   4.7  2008     8 auto(~ 4          9    12 e   suv
3 dodge        ram 1500~  4.7  2008     8 auto(~ 4          9    12 e   pick~
4 dodge        ram 1500~  4.7  2008     8 manua~ 4         9    12 e   pick~
5 jeep         grand ch~  4.7  2008     8 auto(~ 4          9    12 e   suv
6 chevrolet    c1500 su~  5.3  2008     8 auto(~ r          11   15 e   suv
7 chevrolet    k1500 ta~  5.3  2008     8 auto(~ 4         11   14 e   suv
8 chevrolet    k1500 ta~  5.7  1999     8 auto(~ 4         11   15 r   suv
9 dodge        caravan ~  3.3  2008     6 auto(~ f          11   17 e   mini~
10 dodge       dakota p~  5.2  1999     8 manua~ 4        11   17 r   pick~
# ... with 224 more rows
```

```
# 시내 연비는 오름차순, 차량 타입은 내림차순(알파벳 역순) 정렬
## R 기본 문법 사용
#### 문자형 벡터의 순위 계산을 위해 rank() 함수 사용
mpg_sortb <- mpg[order(mpg$cty, -rank(mpg$class)), ]

## arrange 함수 사용
```

```
mpg_sortt <- mpg %>% arrange(cty, desc(class))
mpg_sortt %>% print

# A tibble: 234 x 11
  manufacturer model       displ  year   cyl trans drv   cty   hwy fl class
  <chr>        <chr>     <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
1 dodge        durango ~    4.7  2008     8 auto(~ 4      9    12 e   suv 
2 jeep         grand ch~  4.7  2008     8 auto(~ 4      9    12 e   suv 
3 dodge        dakota p~  4.7  2008     8 auto(~ 4      9    12 e   pick~ 
4 dodge        ram 1500~ 4.7  2008     8 auto(~ 4      9    12 e   pick~ 
5 dodge        ram 1500~ 4.7  2008     8 manua~ 4      9    12 e   pick~ 
6 chevrolet    c1500 su~ 5.3  2008     8 auto(~ r      11   15 e   suv 
7 chevrolet    k1500 ta~ 5.3  2008     8 auto(~ 4      11   14 e   suv 
8 chevrolet    k1500 ta~ 5.7  1999     8 auto(~ 4      11   15 r   suv 
9 dodge        durango ~ 5.2  1999     8 auto(~ 4      11   16 r   suv 
10 dodge       durango ~ 5.9  1999     8 auto(~ 4      11   15 r   suv 
# ... with 224 more rows
```

```
# 두 데이터 셋 동일성 여부
identical(mpg_sortb, mpg_sortt)
```

[1] TRUE

4.4.4 `select()`

열(변수) 조작 동사

- 데이터셋을 구성하는 열(column, variable)을 선택하는 함수

```
select(
  data, # 데이터 프레임 또는 티블 객체
  var_name1, # 변수 이름 (따옴표 없이도 가능)
```

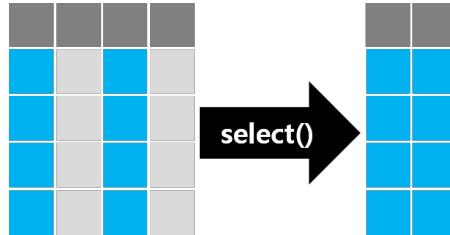


FIGURE 4.5: select() 함수 다이어그램

```
var_name2,
...
)
```

```
# 제조사(manufacturer), 모델명(model), 배기량(displ)
# 제조년도(year), 시내연비(cty)만 추출

## 기본 R 문법 이용한 변수 추출
glimpse(mpg[, c("manufacturer", "model", "displ", "year", "cty")])
```

Observations: 234

Variables: 5

```
$ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi"...
$ model      <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro"...
$ displ       <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, ...
$ year        <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, ...
$ cty         <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17...
```

```
# glimpse(mpg[, c(1:4, 8)])
# glimpse(mpg[, names(mpg) %in% c("manufacturer", "displ", "model",
#                               "year", "cty")])

## select() 함수 이용
### 아래 스크립트는 모두 동일한 결과를 반환
```

```
# mpg %>% select(1:4, 8)
#
# mpg %>%
#   select(c("manufacturer", "model", "displ", "year", "cty"))

mpg %>%
  select("manufacturer", "model", "displ", "year", "cty") %>%
  glimpse
```

Observations: 234

Variables: 5

```
$ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi"...
$ model       <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro"...
$ displ        <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, ...
$ year         <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, ...
$ cty          <int> 18, 21, 20, 21, 16, 18, 18, 16, 20, 19, 15, 17, 17...
```

- R 기본 문법과 차이점

- 선택하고자 하는 변수 입력 시 따옴표가 필요 없음
- : 연산자를 이용해 선택 변수의 범위 지정 가능
- - 연산자를 이용해 선택 변수 제거

```
# 제조사(manufacturer), 모델명(model), 배기량(displ)
# 제조년도(year), 시내연비(cty)만 추출
## select() 따옴표 없이 변수명 입력

mpg %>%
  select(manufacturer, model, displ, year, cty) %>%
  glimpse
```

Observations: 234

Variables: 5

```
$ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi"...
```

```
$ model      <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro"...
$ displ       <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, ...
$ year        <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, ...
$ cty         <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17...
```

: 연산자로 변수 범위 지정

```
mpg %>%
  select(manufacturer:year, cty) %>%
  glimpse
```

Observations: 234

Variables: 5

```
$ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi"...
$ model      <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro"...
$ displ       <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, ...
$ year        <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, ...
$ cty         <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17...
```

관심 없는 열을 -로 제외 가능

```
mpg %>%
  select(-cyl, -trans, -drv, -hwy, -fl, -class) %>%
  glimpse
```

Observations: 234

Variables: 5

```
$ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi"...
$ model      <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro"...
$ displ       <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, ...
$ year        <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, ...
$ cty         <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17...
```

조금 더 간결하게 (~: `와`-` 연산 조합)

```
mpg %>%
```

```
select(-cyl:-drv, -hwy:-class) %>%  
glimpse  
  
Observations: 234  
Variables: 5  
$ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi"...  
$ model          <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro"...  
$ displ           <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0,...  
$ year            <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, ...  
$ cty             <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17...
```

동일한 기능: -는 괄호로 묶을 수 있음

```
mpg %>%  
  select(-(cyl:drv), -(hwy:class)) %>%  
  glimpse
```

```
Observations: 234  
Variables: 5  
$ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi"...  
$ model          <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro"...  
$ displ           <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0,...  
$ year            <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, ...  
$ cty             <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17...
```

```
# select() 함수를 이용해 변수명 변경 가능  
mpg %>%  
  select(city_mpg = cty) %>%  
  glimpse
```

```
Observations: 234  
Variables: 1  
$ city_mpg <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17, 15...
```

- `select()`와 조합 시 유용한 선택 함수
 - `starts_with("abc")`: “abc”로 시작하는 변수 선택
 - `ends_with("xyz")`: “xyz”로 끝나는 변수 선택
 - `contains("def")`: “def”를 포함하는 변수 선택
 - `matches("^F[0-9]")`: 정규표현식과 일치하는 변수 선택. “F”와 한 자리 숫자로 시작하는 변수 선택
 - `everything()`: `select()` 함수 내에서 미리 선택한 변수를 제외한 모든 변수 선택

```
# "m"으로 시작하는 변수 제거
## 기존 select() 함수 사용

mpg %>%
  select(-manufacturer, -model) %>%
  glimpse
```

Observations: 234
 Variables: 9

```
$ displ <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, 2.8, 2...
$ year  <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, 2008, 2...
$ cyl   <int> 4, 4, 4, 4, 6, 6, 4, 4, 4, 6, 6, 6, 6, 6, 8, 8, 8...
$ trans <chr> "auto(15)", "manual(m5)", "manual(m6)", "auto(av)", "auto(15)...
$ drv   <chr> "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", "4", ...
$ cty   <int> 18, 21, 20, 21, 16, 18, 18, 16, 20, 19, 15, 17, 17, 15, 1...
$ hwy   <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25, 25, 2...
$ fl    <chr> "p", ...
$ class <chr> "compact", "compact", "compact", "compact", "compact", "compa...
```

```
## select() + starts_with() 함수 사용

mpg %>%
  select(-starts_with("m")) %>%
  glimpse
```

```
Observations: 234
Variables: 9
$ displ <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, 2.8, 2...
$ year  <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, 2008, 2...
$ cyl   <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6, 6, 6, 6, 8, 8, 8...
$ trans <chr> "auto(15)", "manual(m5)", "manual(m6)", "auto(av)", "auto(15)...
$ drv   <chr> "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", "4", "4", "...
$ cty   <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17, 15, 1...
$ hwy   <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25, 25, 2...
$ fl    <chr> "p", "...
$ class <chr> "compact", "compact", "compact", "compact", "compact", "compact", "compa...
```

```
# "l"로 끝나는 변수 선택: ends_with() 함수 사용
mpg %>%
  select(ends_with("l")) %>%
  glimpse
```

```
Observations: 234
Variables: 4
$ model <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro", "a4 q...
$ displ <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, 2.8, 2...
$ cyl   <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6, 6, 6, 6, 8, 8, 8...
$ fl    <chr> "p", "...
```

```
# dplyr 내장 데이터: starwars에서 이름, 성별, "color"를 포함하는 변수 선택
## contains() 함수 사용
starwars %>%
  select(name, gender, contains("color")) %>%
  head
```

```
# A tibble: 6 x 5
  name      gender hair_color skin_color eye_color
  <chr>     <chr>   <chr>     <chr>     <chr>
1 Luke Skywalker male    blond     fair     blue
```

```

2 C-3PO      <NA>    <NA>        gold      yellow
3 R2-D2      <NA>    <NA>    white, blue red
4 Darth Vader male   none       white      yellow
5 Leia Organa female brown    light      brown
6 Owen Lars   male   brown, grey light      blue

```

```

# 다시 mpg 데이터...
## 맨 마지막 문자가 "y"로 끝나는 변수 선택(제조사, 모델 포함)
## matches() 사용

mpg %>%
  select(starts_with("m"), matches("y$")) %>%
  glimpse

Observations: 234
Variables: 4
$ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi"...
$ model         <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro"...
$ cty           <int> 18, 21, 20, 21, 16, 18, 18, 16, 20, 19, 15, 17, 17...
$ hwy           <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25...

# cty, hwy 변수를 각각 1-2 번째 열에 위치
mpg %>%
  select(matches("y$"), everything()) %>%
  glimpse

Observations: 234
Variables: 11
$ cty           <int> 18, 21, 20, 21, 16, 18, 18, 16, 20, 19, 15, 17, 17...
$ hwy           <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25...
$ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi"...
$ model         <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro"...
$ displ          <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, ...
$ year           <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, ...

```

```
$ cyl      <int> 4, 4, 4, 4, 6, 6, 4, 4, 4, 6, 6, 6, 6, 6, 6, 8, ...
$ trans    <chr> "auto(15)", "manual(m5)", "manual(m6)", "auto(av)", "a...
$ drv      <chr> "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", ...
$ fl       <chr> "p", ...
$ class    <chr> "compact", "compact", "compact", "compact", "compact", ...
```

4.4.5 mutate()

열(변수) 조작 동사: 새로운 열을 추가하는 함수로 기본 R 문법의
`data$new_variable <- value` 와 유사한 기능을 함

- 주어진 데이터 셋(데이터 프레임)에 이미 존재하고 있는 변수를 이용해 새로운 값 변환 시 유용
- 새로 만든 열을 `mutate()` 함수 내에서 사용 가능 → R base 패키지에서 재공하는 `transform()` 함수는 `mutate()` 함수와 거의 동일한 기능을 하지만, `transform()` 함수 내에서 생성한 변수의 재사용이 불가능

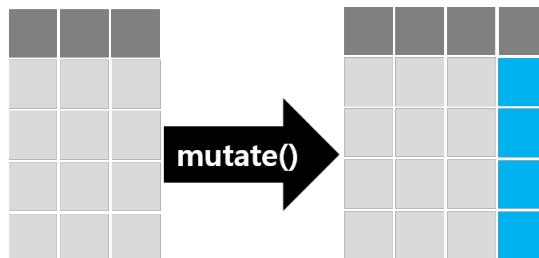


FIGURE 4.6: `mutate()` 함수 다이어그램

```
# mpg 데이터 셋의 연비 단위는 miles/gallon으로 입력 -> kmh/l로 변환
mile <- 1.61 #km
gallon <- 3.79 #litres
kpl <- mile/gallon
```

```
## 기본 R 문법
glimpse(transform(mpg,
  cty_kpl = cty * kpl,
  hwy_kpl = hwy * kpl))
```

```
Observations: 234
Variables: 13
$ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi"...
$ model      <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro"...
$ displ       <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, ...
$ year        <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, ...
$ cyl         <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 6, 6, 6, 6, 6, 8, ...
$ trans       <chr> "auto(15)", "manual(m5)", "manual(m6)", "auto(av)", "a...
$ drv          <chr> "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", ...
$ cty          <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17...
$ hwy          <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25...
$ fl           <chr> "p", ...
$ class        <chr> "compact", "compact", "compact", "compact", "compact", ...
$ cty_kpl     <dbl> 7.646438, 8.920844, 8.496042, 8.920844, 6.796834, 7.64...
$ hwy_kpl     <dbl> 12.319261, 12.319261, 13.168865, 12.744063, 11.044855, ...
```

```
## tidyverse 문법
mpg %>%
  mutate(cty_kpl = cty*kpl,
    hwy_kpl = hwy*kpl) %>%
  glimpse
```

```
Observations: 234
Variables: 13
$ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi"...
$ model      <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro"...
$ displ       <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, ...
$ year        <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, ...
```

```
$ cyl      <int> 4, 4, 4, 4, 6, 6, 4, 4, 4, 6, 6, 6, 6, 6, 6, 8, ...
$ trans    <chr> "auto(15)", "manual(m5)", "manual(m6)", "auto(av)", "a...
$ drv      <chr> "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", ...
$ cty      <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17...
$ hwy      <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25...
$ fl       <chr> "p", ...
$ class    <chr> "compact", "compact", "compact", "compact", "compact", ...
$ cty_kpl  <dbl> 7.646438, 8.920844, 8.496042, 8.920844, 6.796834, 7.64...
$ hwy_kpl  <dbl> 12.319261, 12.319261, 13.168865, 12.744063, 11.044855,...
```

```
# 새로 생성한 변수를 이용해 변환 변수를 원래 단위로 바꿔보기
## R 기본 문법: transform() 사용
glimpse(transform(mpg,
  cty_kpl = cty * kpl,
  hwy_kpl = hwy * kpl,
  cty_raw = cty_kpl/kpl,
  hwy_raw = hwy_kpl/kpl,
))
```

```
Error in eval(substitute(list(...)), `_data`, parent.frame()): 객
체 'cty_kpl'을 찾을 수 없습니다
```

```
## Tidyverse 문법
mpg %>%
  mutate(cty_kpl = cty*kpl,
        hwy_kpl = hwy*kpl,
        cty_raw = cty_kpl/kpl,
        hwy_raw = hwy_kpl/kpl) %>%
  glimpse
```

```
Observations: 234
Variables: 15
$ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi"...
$ model         <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro"...
```

```
$ displ      <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, ...
$ year       <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, ...
$ cyl        <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 6, 6, 6, 6, 6, 6, 8, ...
$ trans      <chr> "auto(15)", "manual(m5)", "manual(m6)", "auto(av)", "a...
$ drv         <chr> "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", ...
$ cty        <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17...
$ hwy        <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25...
$ fl          <chr> "p", ...
$ class       <chr> "compact", "compact", "compact", "compact", "compact", ...
$ cty_kpl    <dbl> 7.646438, 8.920844, 8.496042, 8.920844, 6.796834, 7.64...
$ hwy_kpl    <dbl> 12.319261, 12.319261, 13.168865, 12.744063, 11.044855, ...
$ cty_raw    <dbl> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17...
$ hwy_raw    <dbl> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25...
```

4.4.6 `transmute()`

열(변수) 조작 동사: `mutate()` 와 유사한 기능을 하지만 추가 또는 변환된 변수만 반환

```
`연비` <- mpg %>%
  transmute(cty_kpl = cty*kpl,
            hwy_kpl = hwy*kpl,
            cty_raw = cty_kpl/kpl,
            hwy_raw = hwy_kpl/kpl)
`연비` %>% print
```

```
# A tibble: 234 x 4
  cty_kpl hwy_kpl cty_raw hwy_raw
  <dbl>    <dbl>   <dbl>    <dbl>
1     7.65    12.3     18      29
2     8.92    12.3     21      29
```

```
3   8.50   13.2    20    31.  
4   8.92   12.7    21    30  
5   6.80   11.0    16    26  
6   7.65   11.0    18    26  
7   7.65   11.5    18    27  
8   7.65   11.0    18    26  
9   6.80   10.6    16    25  
10  8.50   11.9    20    28  
# ... with 224 more rows
```

4.4.7 summarise()

변수 요약 및 집계: 변수를 집계하는 함수로 R stat 패키지(R 처음 실행 시 기본으로 불러오는 패키지 중 하나)의 aggregate() 함수와 유사한 기능을 함

- 보통 mean(), sd(), var(), median(), min(), max() 등 요약 통계량을 반환하는 함수와 같이 사용
- 데이터 프레임(티블) 객체 반환
- 변수의 모든 레코드에 집계 함수를 적용하므로 summarise()만 단일로 사용하면 1개의 행만 반환

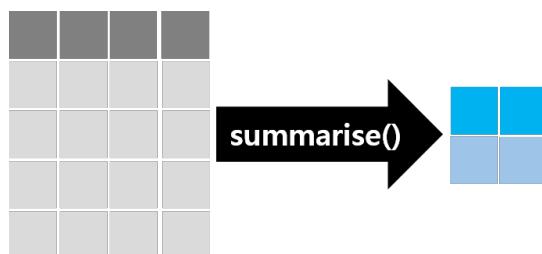


FIGURE 4.7: summarise() 함수 다이어그램

```
# cty, hwy의 평균과 표준편차 계산
mpg %>%
  summarise(mean_cty = mean(cty),
            sd_cty = sd(cty),
            mean_hwy = mean(hwy),
            sd_hwy = sd(hwy))
```

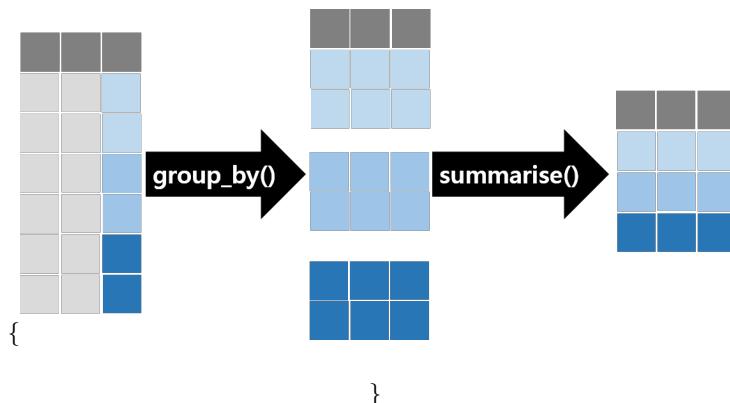
```
# A tibble: 1 x 4
  mean_cty sd_cty mean_hwy sd_hwy
  <dbl>   <dbl>     <dbl>   <dbl>
1     16.9    4.26     23.4    5.95
```

4.4.8 group_by()

행(row, case, observation) 그룹화

- 보통 `summarise()` 함수는 `aggregate()` 함수와 유사하게 그룹 별 요약 통계량을 계산할 때 주로 사용됨
- `group_by()`는 “주어진 그룹에 따라(by group)”, 즉 지정한 그룹(변수) 별 연산을 지정

\begin{figure}



```
\caption{group_by() 함수 다이어그램} \end{figure}
```

```
# 모델, 년도에 따른 cty, hwy 평균 계산
by_mpg <- group_by(mpg, model, year)
## 그룹 지정 check
by_mpg %>%
  head %>%
  print

# A tibble: 6 x 11
# Groups:   model, year [2]
  manufacturer model displ year cyl trans     drv     cty     hwy fl class
  <chr>        <chr>  <dbl> <int> <int> <chr>   <chr>   <int>   <int> <chr> <chr>
1 audi         a4      1.8  1999     4 auto(15) f       18     29 p   compa~
2 audi         a4      1.8  1999     4 manual(m5) f      21     29 p   compa~
3 audi         a4      2    2008     4 manual(m6) f      20     31 p   compa~
4 audi         a4      2    2008     4 auto(av)   f      21     30 p   compa~
5 audi         a4      2.8  1999     6 auto(15) f      16     26 p   compa~
6 audi         a4      2.8  1999     6 manual(m5) f      18     26 p   compa~

## 통계량 계산
by_mpg %>%
  summarise(mean_cty = mean(cty),
            mean_hwy = mean(hwy)) %>%
  print

# A tibble: 76 x 4
# Groups:   model [38]
  model      year mean_cty mean_hwy
  <chr>     <int>   <dbl>    <dbl>
1 4runner 4wd  1999     15.2     19
2 4runner 4wd  2008      15       18.5
3 a4        1999     18.2     27.5
```

```

4 a4          2008    19.7    29.3
5 a4 quattro 1999    16.5    25.2
6 a4 quattro 2008    17.8    26.2
7 a6 quattro 1999    15      24
8 a6 quattro 2008    16.5    24
9 altima      1999    20      28
10 altima     2008    21      29
# ... with 66 more rows

```

```

## by_group() chain 연결
mean_mpg <- mpg %>%
  group_by(model, year) %>%
  summarise(mean_cty = mean(cty),
            mean_hwy = mean(hwy))

```

- `group_by()` 이후 적용되는 동사는 모두 그룹 별 연산 수행

```

# 제조사 별 시내연비가 낮은 순으로 정렬
audi <- mpg %>%
  group_by(manufacturer) %>%
  arrange(cty) %>%
  filter(manufacturer == "audi")

audi %>% print

```

```

# A tibble: 18 x 11
# Groups:   manufacturer [1]
  manufacturer model   displ  year   cyl trans   drv     cty   hwy fl class
  <chr>        <chr>   <dbl> <int> <int> <chr>   <chr> <int> <int> <chr> <chr>
1 audi         a4     2.8   1999     6 auto(l~ 4       15    25 p   comp~
2 audi         a4     3.1   2008     6 manual~ 4      15    25 p   comp~
3 audi         a6     2.8   1999     6 auto(l~ 4       15    24 p   mids~
4 audi         a4     2.8   1999     6 auto(l~ f      16    26 p   comp~

```

| | | | | | | | |
|---------|----------|-----|------|-------------|----|------|-------|
| 5 audi | a4 quat~ | 1.8 | 1999 | 4 auto(l~ 4 | 16 | 25 p | comp~ |
| 6 audi | a6 quat~ | 4.2 | 2008 | 8 auto(s~ 4 | 16 | 23 p | mids~ |
| 7 audi | a4 quat~ | 2.8 | 1999 | 6 manual~ 4 | 17 | 25 p | comp~ |
| 8 audi | a4 quat~ | 3.1 | 2008 | 6 auto(s~ 4 | 17 | 25 p | comp~ |
| 9 audi | a6 quat~ | 3.1 | 2008 | 6 auto(s~ 4 | 17 | 25 p | mids~ |
| 10 audi | a4 | 1.8 | 1999 | 4 auto(l~ f | 18 | 29 p | comp~ |
| 11 audi | a4 | 2.8 | 1999 | 6 manual~ f | 18 | 26 p | comp~ |
| 12 audi | a4 | 3.1 | 2008 | 6 auto(a~ f | 18 | 27 p | comp~ |
| 13 audi | a4 quat~ | 1.8 | 1999 | 4 manual~ 4 | 18 | 26 p | comp~ |
| 14 audi | a4 quat~ | 2 | 2008 | 4 auto(s~ 4 | 19 | 27 p | comp~ |
| 15 audi | a4 | 2 | 2008 | 4 manual~ f | 20 | 31 p | comp~ |
| 16 audi | a4 quat~ | 2 | 2008 | 4 manual~ 4 | 20 | 28 p | comp~ |
| 17 audi | a4 | 1.8 | 1999 | 4 manual~ f | 21 | 29 p | comp~ |
| 18 audi | a4 | 2 | 2008 | 4 auto(a~ f | 21 | 30 p | comp~ |



그룹화된 데이터셋을 다시 그룹화 하지 않은 원래 데이터셋으로 변경할 때 `ungroup()` 함수를 사용

4.4.9 dplyr 관련 유용한 함수

- 데이터 핸들링 시 dplyr 기본 함수와 같이 사용되는 함수 모음

`slice()`

행(row, case, observation) 조작 동사: `filter()`의 특별한 케이스로
데이터의 색인을 직접 설정하여 원하는 row 추출

```
# 1 ~ 8행에 해당하는 데이터 추출
slice_mpg <- mpg %>% slice(1:8)
slice_mpg %>% print
```

```
# A tibble: 8 x 11
```

| | manufacturer | model | displ | year | cyl | trans | drv | cty | hwy | fl | class |
|---|--------------|----------|-------|-------|-------|-----------|-------|-------|-------|-------|--------|
| | <chr> | <chr> | <dbl> | <int> | <int> | <chr> | <chr> | <int> | <int> | <chr> | <chr> |
| 1 | audi | a4 | 1.8 | 1999 | 4 | auto(l~ f | | 18 | 29 | p | compa~ |
| 2 | audi | a4 | 1.8 | 1999 | 4 | manual~ f | | 21 | 29 | p | compa~ |
| 3 | audi | a4 | 2 | 2008 | 4 | manual~ f | | 20 | 31 | p | compa~ |
| 4 | audi | a4 | 2 | 2008 | 4 | auto(a~ f | | 21 | 30 | p | compa~ |
| 5 | audi | a4 | 2.8 | 1999 | 6 | auto(l~ f | | 16 | 26 | p | compa~ |
| 6 | audi | a4 | 2.8 | 1999 | 6 | manual~ f | | 18 | 26 | p | compa~ |
| 7 | audi | a4 | 3.1 | 2008 | 6 | auto(a~ f | | 18 | 27 | p | compa~ |
| 8 | audi | a4 quat~ | 1.8 | 1999 | 4 | manual~ 4 | | 18 | 26 | p | compa~ |

```
# 각 모델 별 첫 번째 데이터만 추출
slice_mpg_grp <- mpg %>%
  group_by(model) %>%
  slice(1) %>%
  arrange(model)

slice_mpg_grp %>% print
```

```
# A tibble: 38 x 11
# Groups:   model [38]
  manufacturer model   displ  year   cyl trans drv  cty   hwy fl class
  <chr>        <chr>   <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
  1 toyota      4runner~  2.7  1999     4 manua~ 4       15    20 r    suv 
  2 audi        a4        1.8  1999     4 auto(~ f    18    29 p    compa~
  3 audi        a4 quat~  1.8  1999     4 manua~ 4       18    26 p    compa~
```

4 audi a6 quat~ 2.8 1999 6 auto(~ 4 15 24 p midsi~

5 nissan altima 2.4 1999 4 manua~ f 21 29 r compa~

6 chevrolet c1500 s~ 5.3 2008 8 auto(~ r 14 20 r suv
 7 toyota camry 2.2 1999 4 manua~ f 21 29 r midsi~

8 toyota camry s~ 2.2 1999 4 auto(~ f 21 27 r compa~

9 dodge caravan~ 2.4 1999 4 auto(~ f 18 24 r miniv~

10 honda civic 1.6 1999 4 manua~ f 28 33 r subco~

```
# ... with 28 more rows
```

`top_n()`

**행(row, case, observation) 조작 동사: 선택한 변수를 기준으로 상위 n
개의 데이터(행)만 추출**

```
# mpg 데이터에서 시내 연비가 높은 데이터 5개 추출
```

```
mpg %>%
  top_n(5, cty) %>%
  arrange(desc(cty))
```

```
# A tibble: 5 x 11
```

| | manufacturer | model | displ | year | cyl | trans | drv | cty | hwy | f1 | class |
|---|--------------|---------|-------|-------|-------|---------|-------|-------|-------|-------|---------|
| | <chr> | <chr> | <dbl> | <int> | <int> | <chr> | <chr> | <int> | <int> | <chr> | <chr> |
| 1 | volkswagen | new be~ | 1.9 | 1999 | 4 | manual~ | f | 35 | 44 | d | subcom~ |
| 2 | volkswagen | jetta | 1.9 | 1999 | 4 | manual~ | f | 33 | 44 | d | compact |
| 3 | volkswagen | new be~ | 1.9 | 1999 | 4 | auto(l~ | f | 29 | 41 | d | subcom~ |
| 4 | honda | civic | 1.6 | 1999 | 4 | manual~ | f | 28 | 33 | r | subcom~ |
| 5 | toyota | corolla | 1.8 | 2008 | 4 | manual~ | f | 28 | 37 | r | compact |

`distinct()`

**행(row, case, observation) 조작 동사: 선택한 변수를 기준으로 중복 없는
유일한(unique)한 행 추출 시 사용**

- R base 패키지의 `unique()` 또는 `unique.data.frame()`과 유사하게 작동하지만 처리 속도 면에서 뛰어남

```
# mpg 데이터에서 중복데이터 제거 (모든 열 기준)
```

```
mpg_uniq <- mpg %>% distinct()
mpg_uniq %>% print
```

```
# A tibble: 225 x 11
  manufacturer model   displ year cyl trans drv cty hwy fl class
  <chr>        <chr>   <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
1 audi         a4      1.8  1999    4 auto(l~ f     18    29 p   comp~
2 audi         a4      1.8  1999    4 manual~ f    21    29 p   comp~
3 audi         a4      2    2008    4 manual~ f    20    31 p   comp~
4 audi         a4      2    2008    4 auto(a~ f    21    30 p   comp~
5 audi         a4      2.8  1999    6 auto(l~ f    16    26 p   comp~
6 audi         a4      2.8  1999    6 manual~ f   18    26 p   comp~
7 audi         a4      3.1  2008    6 auto(a~ f    18    27 p   comp~
8 audi         a4 quat~ 1.8  1999    4 manual~ 4   18    26 p   comp~
9 audi         a4 quat~ 1.8  1999    4 auto(l~ 4   16    25 p   comp~
10 audi        a4 quat~ 2    2008    4 manual~ 4   20    28 p   comp~
# ... with 215 more rows
```

```
# model을 기준으로 중복 데이터 제거
mpg_uniq2 <- mpg %>%
  distinct(model, .keep_all = TRUE) %>%
  arrange(model)

mpg_uniq2 %>% head(6) %>% print
```

```
# A tibble: 6 x 11
  manufacturer model   displ year cyl trans drv cty hwy fl class
  <chr>        <chr>   <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
1 toyota       4runner ~  2.7  1999    4 manual~ 4     15    20 r   suv
2 audi         a4      1.8  1999    4 auto(l~ f    18    29 p   comp~
3 audi         a4 quatt~ 1.8  1999    4 manual~ 4     18    26 p   comp~
4 audi         a6 quatt~ 2.8  1999    6 auto(l~ 4     15    24 p   mids~
5 nissan       altima   2.4  1999    4 manual~ f    21    29 r   comp~
```

```
6 chevrolet    c1500 suv   5.3 2008     8 auto(l~ r      14    20 r     suv
```

```
# 위 그룹별 slice(1) 예제와 비교
identical(slice_mpg_grp %>% ungroup, mpg_uniq2)
```

```
[1] TRUE
```

`sample_n() / sample_frac()`

행(row, case, observation) 조작 동사: 데이터의 행을 랜덤하게 추출하는
함수

- `sample_(n)`: 전체 N 행에서 n 행을 랜덤하게 추출
- `sample_frac(p)`: 전체 N 행에서 비율 p ($0 \leq p \leq 1$) 만큼 추출

```
# 전체 234개 행에서 3개 행을 랜덤하게 추출
mpg %>% sample_n(3)
```

```
# A tibble: 3 x 11
  manufacturer model      displ year cyl trans drv   cty   hwy fl class
  <chr>        <chr>     <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
1 dodge       ram 1500 ~    4.7  2008     8 auto(~ 4       13    17 r   pick~
2 subaru      forester ~   2.5  1999     4 manua~ 4       18    25 r   suv 
3 ford        f150 pick~   4.6  2008     8 auto(~ 4       13    17 r   pick~
```

```
# 전체 234개 행의 5%에 해당하는 행을 랜덤하게 추출
mpg %>% sample_frac(0.05)
```

```
# A tibble: 12 x 11
  manufacturer model      displ year cyl trans drv   cty   hwy fl class
  <chr>        <chr>     <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
1 toyota       camry s~   2.4  2008     4 auto(~ f       22    31 r   compa~
2 ford         f150 pi~   4.6  1999     8 auto(~ 4       13    16 r   pickup
```

| | | | | | | | | | |
|--------------|----------|-----|------|----------|---|----|----|---|--------|
| 3 nissan | altima | 3.5 | 2008 | 6 manua~ | f | 19 | 27 | p | midsi~ |
| 4 volkswagen | jetta | 2 | 2008 | 4 manua~ | f | 21 | 29 | p | compa~ |
| 5 chevrolet | c1500 s~ | 5.3 | 2008 | 8 auto(~ | r | 14 | 20 | r | suv |
| 6 nissan | altima | 2.5 | 2008 | 4 auto(~ | f | 23 | 31 | r | midsi~ |
| 7 toyota | camry s~ | 2.2 | 1999 | 4 manua~ | f | 21 | 29 | r | compa~ |
| 8 toyota | camry s~ | 3 | 1999 | 6 auto(~ | f | 18 | 26 | r | compa~ |
| 9 ford | expedit~ | 5.4 | 2008 | 8 auto(~ | r | 12 | 18 | r | suv |
| 10 hyundai | sonata | 3.3 | 2008 | 6 auto(~ | f | 19 | 28 | r | midsi~ |
| 11 subaru | impreza~ | 2.5 | 1999 | 4 manua~ | 4 | 19 | 26 | r | subco~ |
| 12 ford | explore~ | 4 | 1999 | 6 auto(~ | 4 | 14 | 17 | r | suv |

rename()

열(변수) 조작 동사: 변수의 이름을 변경하는 함수

- `rename(new_variable_name = old_variable_name)` 형태로 변경

```
# 변수명 변성
## R 기본 문법으로 변수명 변경
varn_mpg <- names(mpg) # 원 변수명 저장
names(mpg)[5] <- "cylinder" # cyl을 cylinder로 변경
names(mpg) <- varn_mpg # 

## Tidyverse 문법: rename() 사용
mpg %>%
  rename(cylinder = cyl) %>%
  head %>%
  print

# A tibble: 6 x 11
  manufacturer model displ year cylinder trans     drv     cty     hwy fl     class
  <chr>        <chr>   <dbl> <int>      <int> <chr>    <chr>   <int>   <int> <chr> <chr>
1 audi         a4       1.8  1999       4 auto(15) f        18     29 p     comp~
```

```

2 audi      a4      1.8 1999        4 manual(~ f      21    29 p    comp~
3 audi      a4      2    2008        4 manual(~ f      20    31 p    comp~
4 audi      a4      2    2008        4 auto(av) f     21    30 p    comp~
5 audi      a4      2.8 1999       6 auto(15) f     16    26 p    comp~
6 audi      a4      2.8 1999       6 manual(~ f     18    26 p    comp~

```

```

## cty를 city_mpg, hwy를 hw_mpg로 변경
mpg %>%
  rename(city_mpg = cty,
        hw_mpg = hwy) %>%
  print

```

```

# A tibble: 234 x 11
  manufacturer model  displ  year   cyl trans drv   city_mpg hw_mpg fl   class
  <chr>        <chr>  <dbl> <int> <int> <chr> <chr>    <int>   <int> <chr> <chr>
1 audi         a4      1.8  1999     4 auto~ f      18    29 p    comp~
2 audi         a4      1.8  1999     4 manu~ f      21    29 p    comp~
3 audi         a4      2    2008     4 manu~ f      20    31 p    comp~
4 audi         a4      2    2008     4 auto~ f      21    30 p    comp~
5 audi         a4      2.8  1999     6 auto~ f      16    26 p    comp~
6 audi         a4      2.8  1999     6 manu~ f      18    26 p    comp~
7 audi         a4      3.1  2008     6 auto~ f      18    27 p    comp~
8 audi         a4 qu~   1.8  1999     4 manu~ 4     18    26 p    comp~
9 audi         a4 qu~   1.8  1999     4 auto~ 4     16    25 p    comp~
10 audi        a4 qu~   2    2008     4 manu~ 4     20    28 p    comp~
# ... with 224 more rows

```

Count 관련 동사(함수)

데이터셋의 행 개수를 집계하는 함수들로 데이터 요약 시 유용하게 사용

tally()

- 총계, 행의 개수를 반환하는 함수⁶
- 데이터 프레임(티블) 객체 반환

```
# 전체 행 개수 (nrow(data))와 유사
```

```
mpg %>%
  tally %>%
  print
```

```
# A tibble: 1 x 1
```

```
  n
  <int>
1 234
```

```
# 제조사, 년도별 데이터 개수
```

```
mpg %>%
  group_by(manufacturer, year) %>%
  tally %>%
  ungroup %>%
  print
```

```
# A tibble: 30 x 3
```

| | manufacturer | year | n |
|---|--------------|-------|-------|
| | <chr> | <int> | <int> |
| 1 | audi | 1999 | 9 |
| 2 | audi | 2008 | 9 |
| 3 | chevrolet | 1999 | 7 |
| 4 | chevrolet | 2008 | 12 |
| 5 | dodge | 1999 | 16 |
| 6 | dodge | 2008 | 21 |
| 7 | ford | 1999 | 15 |

⁶tally의 사전적 의미: calculate the total number of

```
8 ford      2008    10
9 honda     1999     5
10 honda    2008    4
# ... with 20 more rows
```

count()

- tally() 함수와 유사하나 개수 집계 전 group_by()와 집계 후 ungroup()을 실행

```
# 제조사, 년도별 데이터 개수: tally() 예시와 비교
mpg %>%
  count(manufacturer, year) %>%
  print
```

```
# A tibble: 30 x 3
  manufacturer   year     n
  <chr>        <int> <int>
1 audi          1999     9
2 audi          2008     9
3 chevrolet    1999     7
4 chevrolet    2008    12
5 dodge         1999    16
6 dodge         2008    21
7 ford          1999    15
8 ford          2008    10
9 honda         1999     5
10 honda        2008    4
# ... with 20 more rows
```

n()

- 위에서 소개한 함수와 유사하게 행 개수를 반환하지만, 기본 동사 (summarise(), mutate(), filter()) 내에서만 사용

```
# 제조사, 년도에 따른 배기량, 시내연비의 평균 계산(그룹 별 n 포함)
mpg %>%
  group_by(manufacturer, year) %>%
  summarise(
    N = n(),
    mean_displ = mean(displ),
    mean_hwy = mean(cty)) %>%
  print
```

```
# A tibble: 30 x 5
# Groups:   manufacturer [15]
  manufacturer     year     N   mean_displ   mean_hwy
  <chr>        <int> <int>      <dbl>      <dbl>
1 audi            1999     9       2.36      17.1
2 audi            2008     9       2.73      18.1
3 chevrolet       1999     7       4.97      15.1
4 chevrolet       2008    12       5.12      14.9
5 dodge           1999    16       4.32      13.4
6 dodge           2008    21       4.42      13.0
7 ford            1999    15       4.45      13.9
8 ford            2008    10       4.66      14.1
9 honda           1999     5       1.6       24.8
10 honda          2008    4       1.85      24
# ... with 20 more rows
```

```
# mutate, filter에서 사용하는 경우
mpg %>%
  group_by(manufacturer, year) %>%
  mutate(N = n()) %>%
  filter(n() < 4) %>%
  print
```

```
# A tibble: 18 x 12
```

```
# Groups: manufacturer, year [9]
# ... with 1 more variable: N <int>
  manufacturer model displ year cyl trans drv cty hwy fl class
<chr>       <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
1 jeep        gran~   4    1999    6 auto~ 4      15    20 r    suv
2 jeep        gran~   4.7  1999    8 auto~ 4      14    17 r    suv
3 land rover  rang~   4    1999    8 auto~ 4      11    15 p    suv
4 land rover  rang~   4.2  2008    8 auto~ 4      12    18 r    suv
5 land rover  rang~   4.4  2008    8 auto~ 4      12    18 r    suv
6 land rover  rang~   4.6  1999    8 auto~ 4      11    15 p    suv
7 lincoln     navi~   5.4  1999    8 auto~ r     11    17 r    suv
8 lincoln     navi~   5.4  2008    8 auto~ r     11    16 p    suv
9 lincoln     navi~   5.4  2008    8 auto~ r     12    18 r    suv
10 mercury    moun~   4    1999    6 auto~ 4     14    17 r    suv
11 mercury    moun~   4    2008    6 auto~ 4     13    19 r    suv
12 mercury    moun~   4.6  2008    8 auto~ 4     13    19 r    suv
13 mercury    moun~   5    1999    8 auto~ 4     13    17 r    suv
14 pontiac    gran~   3.1  1999    6 auto~ f     18    26 r    mids~
15 pontiac    gran~   3.8  1999    6 auto~ f     16    26 p    mids~
16 pontiac    gran~   3.8  2008    6 auto~ f     17    27 r    mids~
17 pontiac    gran~   3.8  2008    6 auto~ f     18    28 r    mids~
18 pontiac    gran~   5.3  2008    8 auto~ f     16    25 p    mids~
```

4.4.10 부가 기능

- 위에서 소개한 dplyr 패키지의 기본 동사 함수를 조금 더 효율적으로 사용
(예: 특정 조건을 만족하는 두 개 이상의 변수에 함수 적용)하기 위한 범위 지정 함수로서 아래와 같은 부사(adverb), 접속사 또는 전치사가 본 동사 뒤에 붙음

- *_all: 모든 변수에 적용
- *_at: vars() 함수를 이용해 선택한 변수에 적용

c. `*_if`: 조건식 또는 조건 함수로 선택한 변수에 적용

- 여기서 `*` = {`filter`, `arrange`, `select`, `rename`, `mutate`, `transmute`, `summarise`, `group_by`}
- 적용할 변수들은 대명사(pronoun)로 지칭되며, `.`로 나타냄
- `vars()`는 `*_at` 계열 함수 내에서 변수를 선택해주는 함수로 `select()` 함수와 동일한 문법으로 사용 가능(단독으로는 사용하지 않음)

`filter_all()`, `filter_at()`, `filter_if()`

- `filter_all()`:`all_vars()` 또는 `any_vars()`라는 조건 함수와 같이 사용되며, 해당 함수 내에 변수는 대명사 `.`로 나타냄
- `filter_at()`: 변수 선택 지시자 `vars()`와 `vars()`에서 선택한 변수에 적용할 조건식 또는 조건 함수를 인수로 받음. 조건식 설정 시 `vars()`에 포함된 변수들은 대명사 `.`으로 표시
- `filter_if()`: 조건을 만족하는 변수들을 선택한 후, 선택한 변수들 중 모두 또는 하나라도 입력한 조건을 만족하는 행 추출

```
# mtcars 데이터셋 사용
## filter_all()
### 모든 변수들이 100 보다 큰 케이스 추출
mtcars %>%
  filter_all(all_vars(. > 100)) %>%
  print
```

```
[1] mpg cyl disp hp drat wt qsec vs am gear carb
<0 행> <또는 row.names의 길이가 0입니다>
```

```
### 모든 변수들 중 하나라도 300 보다 큰 케이스 추출
```

```
mtcars %>%
  filter_all(any_vars(. > 300)) %>%
  print
```

| | mpg | cyl | disp | hp | drat | wt | qsec | vs | am | gear | carb |
|----|------|-----|------|-----|------|-------|-------|----|----|------|------|
| 1 | 18.7 | 8 | 360 | 175 | 3.15 | 3.440 | 17.02 | 0 | 0 | 3 | 2 |
| 2 | 14.3 | 8 | 360 | 245 | 3.21 | 3.570 | 15.84 | 0 | 0 | 3 | 4 |
| 3 | 10.4 | 8 | 472 | 205 | 2.93 | 5.250 | 17.98 | 0 | 0 | 3 | 4 |
| 4 | 10.4 | 8 | 460 | 215 | 3.00 | 5.424 | 17.82 | 0 | 0 | 3 | 4 |
| 5 | 14.7 | 8 | 440 | 230 | 3.23 | 5.345 | 17.42 | 0 | 0 | 3 | 4 |
| 6 | 15.5 | 8 | 318 | 150 | 2.76 | 3.520 | 16.87 | 0 | 0 | 3 | 2 |
| 7 | 15.2 | 8 | 304 | 150 | 3.15 | 3.435 | 17.30 | 0 | 0 | 3 | 2 |
| 8 | 13.3 | 8 | 350 | 245 | 3.73 | 3.840 | 15.41 | 0 | 0 | 3 | 4 |
| 9 | 19.2 | 8 | 400 | 175 | 3.08 | 3.845 | 17.05 | 0 | 0 | 3 | 2 |
| 10 | 15.8 | 8 | 351 | 264 | 4.22 | 3.170 | 14.50 | 0 | 1 | 5 | 4 |
| 11 | 15.0 | 8 | 301 | 335 | 3.54 | 3.570 | 14.60 | 0 | 1 | 5 | 8 |

```
## filter_at()
```

```
### 기어 개수(gear)와 기화기 개수(carb)가 짝수인 케이스만 추출
```

```
mtcars %>%
  filter_at(vars(gear, carb),
            ~ . %% 2 == 0) %>% # 대명사 앞에 ~ 표시를 꼭 사용해야 함
  print
```

| | mpg | cyl | disp | hp | drat | wt | qsec | vs | am | gear | carb |
|---|------|-----|-------|-----|------|-------|-------|----|----|------|------|
| 1 | 21.0 | 6 | 160.0 | 110 | 3.90 | 2.620 | 16.46 | 0 | 1 | 4 | 4 |
| 2 | 21.0 | 6 | 160.0 | 110 | 3.90 | 2.875 | 17.02 | 0 | 1 | 4 | 4 |
| 3 | 24.4 | 4 | 146.7 | 62 | 3.69 | 3.190 | 20.00 | 1 | 0 | 4 | 2 |
| 4 | 22.8 | 4 | 140.8 | 95 | 3.92 | 3.150 | 22.90 | 1 | 0 | 4 | 2 |
| 5 | 19.2 | 6 | 167.6 | 123 | 3.92 | 3.440 | 18.30 | 1 | 0 | 4 | 4 |
| 6 | 17.8 | 6 | 167.6 | 123 | 3.92 | 3.440 | 18.90 | 1 | 0 | 4 | 4 |
| 7 | 30.4 | 4 | 75.7 | 52 | 4.93 | 1.615 | 18.52 | 1 | 1 | 4 | 2 |

```
8 21.4 4 121.0 109 4.11 2.780 18.60 1 1 4 2
```

```
## filter_if()
### 내림한 값이 원래 값과 같은 변수들 모두 0이 아닌 케이스 추출
mtcars %>%
  filter_if(~ all(floor(.) == .),
            all_vars(. != 0)) %>%
  # filter_if(~ all(floor(.) == .),
  #           ~ . != 0) %>%
  print
```

| | mpg | cyl | disp | hp | drat | wt | qsec | vs | am | gear | carb |
|---|------|-----|-------|-----|------|-------|-------|----|----|------|------|
| 1 | 22.8 | 4 | 108.0 | 93 | 3.85 | 2.320 | 18.61 | 1 | 1 | 4 | 1 |
| 2 | 32.4 | 4 | 78.7 | 66 | 4.08 | 2.200 | 19.47 | 1 | 1 | 4 | 1 |
| 3 | 30.4 | 4 | 75.7 | 52 | 4.93 | 1.615 | 18.52 | 1 | 1 | 4 | 2 |
| 4 | 33.9 | 4 | 71.1 | 65 | 4.22 | 1.835 | 19.90 | 1 | 1 | 4 | 1 |
| 5 | 27.3 | 4 | 79.0 | 66 | 4.08 | 1.935 | 18.90 | 1 | 1 | 4 | 1 |
| 6 | 30.4 | 4 | 95.1 | 113 | 3.77 | 1.513 | 16.90 | 1 | 1 | 5 | 2 |
| 7 | 21.4 | 4 | 121.0 | 109 | 4.11 | 2.780 | 18.60 | 1 | 1 | 4 | 2 |

`select_all(), select_at(), select_if()`

변수 선택과 변수명 변경을 동시에 수행

```
# select_all() 예시
## mpg 데이터셋의 모든 변수명을 대문자로 변경
mpg %>%
  select_all(~toupper(.)) %>%
  print
```

```
# A tibble: 234 x 11
  MANUFACTURER MODEL     DISPL   YEAR   CYL TRANS   DRV     CTY     HWY FL     CLASS
  <fct>    <fct>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
```

```

<chr>     <chr>     <dbl> <int> <int> <chr>     <chr> <int> <int> <chr> <chr>
1 audi      a4          1.8  1999     4 auto(l~ f       18    29 p     comp~
2 audi      a4          1.8  1999     4 manual~ f      21    29 p     comp~
3 audi      a4          2    2008     4 manual~ f      20    31 p     comp~
4 audi      a4          2    2008     4 auto(a~ f       21    30 p     comp~
5 audi      a4          2.8  1999     6 auto(l~ f       16    26 p     comp~
6 audi      a4          2.8  1999     6 manual~ f      18    26 p     comp~
7 audi      a4          3.1  2008     6 auto(a~ f       18    27 p     comp~
8 audi      a4 quat~    1.8  1999     4 manual~ 4      18    26 p     comp~
9 audi      a4 quat~    1.8  1999     4 auto(l~ 4       16    25 p     comp~
10 audi     a4 quat~   2    2008     4 manual~ 4      20    28 p     comp~

# ... with 224 more rows

```

```

# select_if() 예시
## 문자형 변수를 선택하고 선택한 변수의 이름을 대문자로 변경
mpg %>%
  select_if(~ is.character(.), ~ toupper(.)) %>%
  print

```

```

# A tibble: 234 x 6
  MANUFACTURER MODEL      TRANS     DRV     FL     CLASS
  <chr>     <chr>     <chr>     <chr> <chr> <chr>
1 audi      a4          auto(15)   f      p     compact
2 audi      a4          manual(m5) f      p     compact
3 audi      a4          manual(m6) f      p     compact
4 audi      a4          auto(av)    f      p     compact
5 audi      a4          auto(15)   f      p     compact
6 audi      a4          manual(m5) f      p     compact
7 audi      a4          auto(av)    f      p     compact
8 audi      a4 quattro  manual(m5) 4      p     compact
9 audi      a4 quattro  auto(15)   4      p     compact
10 audi     a4 quattro  manual(m6) 4      p     compact

# ... with 224 more rows

```

```
# select_at() 예시
## model에서 cty 까지 변수를 선택하고 선택한 변수명을 대문자로 변경
mpg %>%
  select_at(vars(model:cty), ~ toupper(.)) %>%
  print

# A tibble: 234 x 7
  MODEL     DISPL   YEAR   CYL TRANS    DRV     CTY
  <chr>     <dbl> <int> <int> <chr>    <chr> <int>
1 a4         1.8   1999     4 auto(l5) f       18
2 a4         1.8   1999     4 manual(m5) f      21
3 a4         2     2008     4 manual(m6) f      20
4 a4         2     2008     4 auto(av)  f      21
5 a4         2.8   1999     6 auto(l5) f      16
6 a4         2.8   1999     6 manual(m5) f      18
7 a4         3.1   2008     6 auto(av)  f      18
8 a4 quattro 1.8   1999     4 manual(m5) 4     18
9 a4 quattro 1.8   1999     4 auto(l5)  4     16
10 a4 quattro 2     2008     4 manual(m6) 4    20
# ... with 224 more rows

  mutate_all(), mutate_at(), mutate_if()
```

실제 데이터 전처리 시 가장 많이 사용

- `mutate_all()`: 모든 변수에 적용(모든 데이터가 동일한 데이터 타입인 경우 유용)
- `mutate_at()`: 선택한 변수에 적용. `vars()` 함수로 선택
- `mutate_if()`: 특정 조건을 만족하는 변수에 적용

```
# mutate_all() 예시
## 문자형 변수를 선택 후 모든 변수를 요인형으로 변환
mpg %>%
  select_if(~is.character(.)) %>%
  mutate_all(~factor(.)) %>%
  head %>%
  print
```

```
# A tibble: 6 x 6
  manufacturer model trans      drv   fl    class
  <fct>        <fct> <fct>      <fct> <fct> <fct>
  1 audi         a4    auto(15)    f     p    compact
  2 audi         a4    manual(m5)  f     p    compact
  3 audi         a4    manual(m6)  f     p    compact
  4 audi         a4    auto(av)    f     p    compact
  5 audi         a4    auto(15)    f     p    compact
  6 audi         a4    manual(m5)  f     p    compact
```

```
# mutate_at() 예시
## cty, hwy 단위를 km/l로 변경
mpg %>%
  mutate_at(vars(cty, hwy),
            ~ . * kpl) %>%  # 원래 변수를 변경
  head %>%
  print
```

```
# A tibble: 6 x 11
  manufacturer model displ year cyl trans      drv     cty   hwy fl    class
  <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <dbl> <dbl> <chr> <chr>
  1 audi         a4    1.8  1999     4 auto(15)    f      7.65 12.3 p    compact
  2 audi         a4    1.8  1999     4 manual(m5)  f      8.92 12.3 p    compact
  3 audi         a4    2    2008     4 manual(m6)  f      8.50 13.2 p    compact
  4 audi         a4    2    2008     4 auto(av)    f      8.92 12.7 p    compact
```

```
5 audi      a4      2.8 1999      6 auto(15)   f      6.80 11.0 p      compa-
6 audi      a4      2.8 1999      6 manual(m5) f      7.65 11.0 p      compa-
```

```
## "m"으로 시작하거나 "s"로 끝나는 변수 선택 후 요인형으로 변환
```

```
mpg %>%
  mutate_at(vars(starts_with("m")),
            ends_with("s")),
  ~ factor(.)) %>%
  summary
```

| | manufacturer | model | displ | year |
|------------|--------------|---------------------|---------------|--------------------------------|
| dodge | :37 | caravan 2wd | : 11 | Min. :1.600 Min. :1999 |
| toyota | :34 | ram 1500 pickup 4wd | : 10 | 1st Qu.:2.400 1st Qu.:1999 |
| volkswagen | :27 | civic | : 9 | Median :3.300 Median :2004 |
| ford | :25 | dakota pickup 4wd | : 9 | Mean :3.472 Mean :2004 |
| chevrolet | :19 | jetta | : 9 | 3rd Qu.:4.600 3rd Qu.:2008 |
| audi | :18 | mustang | : 9 | Max. :7.000 Max. :2008 |
| (Other) | :74 | (Other) | :177 | |
| | cyl | trans | drv | cty |
| Min. | :4.000 | auto(14) | :83 | Length:234 Min. : 9.00 |
| 1st Qu. | :4.000 | manual(m5) | :58 | Class :character 1st Qu.:14.00 |
| Median | :6.000 | auto(15) | :39 | Mode :character Median :17.00 |
| Mean | :5.889 | manual(m6) | :19 | Mean :16.86 |
| 3rd Qu. | :8.000 | auto(s6) | :16 | 3rd Qu.:19.00 |
| Max. | :8.000 | auto(16) | : 6 | Max. :35.00 |
| | | (Other) | :13 | |
| | hwy | fl | class | |
| Min. | :12.00 | Length:234 | 2seater : 5 | |
| 1st Qu. | :18.00 | Class :character | compact :47 | |
| Median | :24.00 | Mode :character | midsize :41 | |
| Mean | :23.44 | | minivan :11 | |
| 3rd Qu. | :27.00 | | pickup :33 | |
| Max. | :44.00 | | subcompact:35 | |

```
suv      :62
```

```
# mutate_if() 예시
## 문자형 변수를 요인형으로 변환
mpg %>%
  mutate_if(~ is.character(.), ~ factor(.)) %>%
  summary
```

| | manufacturer | model | displ | year | |
|------------|--------------|---------------------|-------|----------------------------|-----------------------------|
| dodge | :37 | caravan 2wd | : 11 | Min. :1.600 Min. :1999 | |
| toyota | :34 | ram 1500 pickup 4wd | : 10 | 1st Qu.:2.400 1st Qu.:1999 | |
| volkswagen | :27 | civic | : 9 | Median :3.300 Median :2004 | |
| ford | :25 | dakota pickup 4wd | : 9 | Mean :3.472 Mean :2004 | |
| chevrolet | :19 | jetta | : 9 | 3rd Qu.:4.600 3rd Qu.:2008 | |
| audi | :18 | mustang | : 9 | Max. :7.000 Max. :2008 | |
| (Other) | :74 | (Other) | :177 | | |
| | cyl | trans | drv | cty | hwy |
| Min. | :4.000 | auto(14) | :83 | 4:103 | Min. : 9.00 Min. :12.00 |
| 1st Qu. | :4.000 | manual(m5) | :58 | f:106 | 1st Qu.:14.00 1st Qu.:18.00 |
| Median | :6.000 | auto(15) | :39 | r: 25 | Median :17.00 Median :24.00 |
| Mean | :5.889 | manual(m6) | :19 | | Mean :16.86 Mean :23.44 |
| 3rd Qu. | :8.000 | auto(s6) | :16 | | 3rd Qu.:19.00 3rd Qu.:27.00 |
| Max. | :8.000 | auto(16) | : 6 | | Max. :35.00 Max. :44.00 |
| | | (Other) | :13 | | |
| | fl | class | | | |
| c: | 1 | 2seater | : | 5 | |
| d: | 5 | compact | : | 47 | |
| e: | 8 | midsize | : | 41 | |
| p: | 52 | minivan | : | 11 | |
| r: | 168 | pickup | : | 33 | |
| | | subcompact | : | 35 | |
| | | suv | : | 62 | |

summarise_all(), summarise_at(), summarise_if()

- 사용 방법은 `mutate_all`, `mutate_at`, `mutate_all` 과 동일
- 다중 변수 요약 시 코드를 효율적으로 작성할 수 있음.

```
# summary_all() 예시
## 모든 변수의 최솟값과 최댓값 요약
## 문자형 변수는 알파벳 순으로 최솟값과 최댓값 반환
## 복수의 함수를 적용 시 list()함수 사용
mpg %>%
  summarise_all(list(min = ~ min(.),
                     max = ~ max(.))) %>%
  glimpse
```

```
Observations: 1
Variables: 22
$ manufacturer_min <chr> "audi"
$ model_min          <chr> "4runner 4wd"
$ displ_min          <dbl> 1.6
$ year_min           <int> 1999
$ cyl_min            <int> 4
$ trans_min          <chr> "auto(av)"
$ drv_min             <chr> "4"
$ cty_min            <int> 9
$ hwy_min             <int> 12
$ fl_min              <chr> "c"
$ class_min          <chr> "2seater"
$ manufacturer_max <chr> "volkswagen"
$ model_max           <chr> "toyota tacoma 4wd"
$ displ_max           <dbl> 7
$ year_max            <int> 2008
$ cyl_max              <int> 8
```

```
$ trans_max      <chr> "manual(m6)"
$ drv_max        <chr> "r"
$ cty_max        <int> 35
$ hwy_max        <int> 44
$ fl_max         <chr> "r"
$ class_max      <chr> "suv"

# summary_at() 예시
## dipl, cyl, city, hwy0 대해 제조사 별 n수와 평균과 표준편차 계산
mpg %>%
  add_count(manufacturer) %>% # 행 집계 결과를 열 변수로 추가하는 함수
  group_by(manufacturer, n) %>%
  summarise_at(vars(displ, cyl, cty:hwy),
    list(mean = ~ mean(.),
         sd = ~ sd(.))) %>%
  print

# A tibble: 15 x 10
# Groups:   manufacturer [15]
  manufacturer     n displ_mean cyl_mean cty_mean hwy_mean displ_sd cyl_sd
  <chr>       <int>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
1 audi           18      2.54      5.22     17.6      26.4     0.673     1.22
2 chevrolet      19      5.06      7.26      15       21.9     1.37      1.37
3 dodge          37      4.38      7.08     13.1      17.9     0.868     1.12
4 ford           25      4.54      7.2       14       19.4     0.541     1.
5 honda          9       1.71      4         24.4      32.6     0.145      0
6 hyundai        14      2.43      4.86     18.6      26.9     0.365     1.03
7 jeep            8       4.58      7.25     13.5      17.6     1.02      1.04
8 land rover     4       4.3       8         11.5      16.5     0.258      0
9 lincoln         3       5.4       8         11.3      17       0         0
10 mercury        4       4.4       7         13.2      18       0.490     1.15
11 nissan         13      3.27      5.54     18.1      24.6     0.864     1.20
12 pontiac        5       3.96      6.4       17       26.4     0.808     0.894
```

```

13 subaru      14     2.46     4     19.3    25.6   0.109   0
14 toyota       34     2.95    5.12    18.5    24.9   0.931   1.32
15 volkswagen   27     2.26    4.59    20.9    29.2   0.443   0.844
# ... with 2 more variables: cty_sd <dbl>, hwy_sd <dbl>

# summary_if() 예시

## 1) 문자형 변수이거나 모든 값이 1999보다 크거나 같은 변수이거나
##     8보다 작거나 같고 정수인 변수를 factor 변환
## 2) 수치형 변수에 대한 제조사 별 n, 평균, 표준편차를 구한 후
## 3) 평균 cty (cty_mean) 기준 내림차순으로 정렬

mpg %>%
  mutate_if(~ is.character(.) | all(. >= 1999) |
    (all(. <= 8) & is.integer(.)),
    ~ factor(.)) %>%
  add_count(manufacturer) %>%
  group_by(manufacturer, n) %>%
  summarise_if(~ is.numeric.,
    list(mean = ~ mean(.),
        sd = ~ sd(.))) %>%
  arrange(desc(cty_mean)) %>%
  print

# A tibble: 15 x 8
# Groups:   manufacturer [15]
  manufacturer     n displ_mean cty_mean hwy_mean displ_sd cty_sd hwy_sd
  <fct>     <int>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
1 honda         9     1.71    24.4    32.6    0.145    1.94    2.55
2 volkswagen   27     2.26    20.9    29.2    0.443    4.56    5.32
3 subaru       14     2.46    19.3    25.6    0.109    0.914    1.16
4 hyundai      14     2.43    18.6    26.9    0.365    1.50    2.18
5 toyota        34     2.95    18.5    24.9    0.931    4.05    6.17
6 nissan        13     3.27    18.1    24.6    0.864    3.43    5.09
7 audi          18     2.54    17.6    26.4    0.673    1.97    2.18

```

| | | | | | | | |
|---------------|----|------|------|------|-------|-------|------|
| 8 pontiac | 5 | 3.96 | 17 | 26.4 | 0.808 | 1 | 1.14 |
| 9 chevrolet | 19 | 5.06 | 15 | 21.9 | 1.37 | 2.92 | 5.11 |
| 10 ford | 25 | 4.54 | 14 | 19.4 | 0.541 | 1.91 | 3.33 |
| 11 jeep | 8 | 4.58 | 13.5 | 17.6 | 1.02 | 2.51 | 3.25 |
| 12 mercury | 4 | 4.4 | 13.2 | 18 | 0.490 | 0.5 | 1.15 |
| 13 dodge | 37 | 4.38 | 13.1 | 17.9 | 0.868 | 2.49 | 3.57 |
| 14 land rover | 4 | 4.3 | 11.5 | 16.5 | 0.258 | 0.577 | 1.73 |
| 15 lincoln | 3 | 5.4 | 11.3 | 17 | 0 | 0.577 | 1 |

4.4.11 데이터 연결

- 분석용 데이터를 만들기 위해 **연관된** 복수의 데이터 테이블을 결합하는 작업이 필수임
- 서로 연결 또는 연관된 데이터를 관계형 데이터 (relational data)라고 칭함
- 관계는 항상 한 쌍의 데이터 테이블 간의 관계로 정의
- 관계형 데이터 작업을 위해 설계된 3 가지 “동사” 유형
 - Mutating join**: 두 데이터 프레임 결합 시 두 테이블 간 행이 일치하는 경우 첫 번째 테이블에 새로운 변수 추가
 - Filtering join**: 다른 테이블의 관측치와 일치 여부에 따라 데이터 테이블의 행을 필터링
 - Set operation**: 데이터 셋의 관측치를 집합 연산으로 조합



본 강의에서는 **mutating join**에 대해서만 다룸

- R base 패키지에서 제공하는 `merge()` 함수로 **mutating join**에 해당하는 두 데이터 간 병합이 가능하지만 앞으로 배울 `*_join()`로도 동일한 기능을 수행할 수 있고, 다음과 같은 장점을 가짐

- 행 순서를 보존
- `merge()`에 비해 코드가 직관적이고 빠름

예제

- 데이터: `nycflights13` (2013년 미국 New York에서 출발하는 항공기 이착륙 기록 데이터)
- `flights`, `airlines`, `airports`, `planes`, `weather` 총 5 개의 데이터 프레임으로 구성되어 있으며, 데이터 구조와 코드북은 다음과 같음

```
# install.packages("nycflights13")
require(nycflights13)
```

필요한 패키지를 로딩중입니다: `nycflights13`

- `flights`: 336,776 건의 비행에 대한 기록과 19개의 변수로 구성되어 있는 데이터셋

```
# A tibble: 336,776 x 19
  year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
  <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
1 2013     1     1      517          515       2     830          819
2 2013     1     1      533          529       4     850          830
3 2013     1     1      542          540       2     923          850
4 2013     1     1      544          545      -1    1004         1022
5 2013     1     1      554          600      -6     812          837
6 2013     1     1      554          558      -4     740          728
7 2013     1     1      555          600      -5     913          854
8 2013     1     1      557          600      -3     709          723
9 2013     1     1      557          600      -3     838          846
10 2013    1     1      558          600      -2     753          745
# ... with 336,766 more rows, and 11 more variables: arr_delay <dbl>,
```

```
# carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
# air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

TABLE 4.3: flights 데이터셋 코드북

| 변수 | 설명 |
|----------------------|-----------------------|
| year, month, day | 출발년도, 월, 일 |
| dep_time, arr_time | 실제 출발-도착 시간(현지시각) |
| sched_dep_time, | 예정 출발-도착 시간(현지시각) |
| sched_arr_time | |
| dep_delay, arr_delay | 출발 및 도착 지연 시간(분, min) |
| carrier | 항공코드 약어(두개 문자) |
| tailnum | 비행기 일련 번호 |
| flight | 항공편 번호 |
| origin, dest | 최초 출발지, 목적지 |
| air_time | 비행 시간(분, min) |
| distance | 비행 거리(마일, mile) |
| hour, minutes | 예정 출발 시각(시, 분)으로 분리 |
| time_hour | POSIXct 포맷으로 기록된 예정 |
| | 항공편 날짜 및 시간 |

- `airlines`: 항공사 이름 및 약어 정보로 구성

```
# A tibble: 16 x 2
  carrier name
  <chr>   <chr>
1 9E      Endeavor Air Inc.
2 AA      American Airlines Inc.
3 AS      Alaska Airlines Inc.
4 B6      JetBlue Airways
```

```

5 DL      Delta Air Lines Inc.
6 EV      ExpressJet Airlines Inc.
7 F9      Frontier Airlines Inc.
8 FL      AirTran Airways Corporation
9 HA      Hawaiian Airlines Inc.
10 MQ     Envoy Air
11 OO     SkyWest Airlines Inc.
12 UA     United Air Lines Inc.
13 US     US Airways Inc.
14 VX     Virgin America
15 WN     Southwest Airlines Co.
16 YV     Mesa Airlines Inc.

```

- airports: 각 공항에 대한 정보를 포함한 데이터셋이고 faa는 공항 코드

```

# A tibble: 1,458 x 8
  faa    name          lat   lon   alt   tz dst tzone
  <chr> <chr>        <dbl> <dbl> <dbl> <dbl> <chr> <chr>
1 04G  Lansdowne Airport  41.1 -80.6 1044  -5 A  America/New_Yo-
2 06A  Moton Field Municipal A- 32.5 -85.7  264  -6 A  America/Chicago
3 06C  Schaumburg Regional   42.0 -88.1  801  -6 A  America/Chicago
4 06N  Randall Airport      41.4 -74.4  523  -5 A  America/New_Yo-
5 09J  Jekyll Island Airport 31.1 -81.4   11  -5 A  America/New_Yo-
6 0A9  Elizabethton Municipal ~ 36.4 -82.2 1593  -5 A  America/New_Yo-
7 0G6  Williams County Airport 41.5 -84.5  730  -5 A  America/New_Yo-
8 0G7  Finger Lakes Regional A- 42.9 -76.8  492  -5 A  America/New_Yo-
9 0P2  Shoestring Aviation Air~ 39.8 -76.6 1000  -5 U  America/New_Yo-
10 0S9 Jefferson County Intl  48.1 -123.   108  -8 A  America/Los_An-
# ... with 1,448 more rows

```

TABLE 4.4: airports 데이터셋 코드북

| 변수 | 설명 |
|-------|---|
| faa | FAA 공항 코드 |
| name | 공항 명칭 |
| lat | 위도 |
| lon | 경도 |
| alt | 고도 |
| tz | 타임존 차이(GMT로부터) |
| dst | 일광 절약 시간제(섬머타임) : A= 미국 표준 DST, U=unknown, N=no DST |
| tzone | IANA 타임존 |

- planes: 항공기 정보(제조사, 일련번호, 유형 등)에 대한 정보를 포함한 데이터셋

```
# A tibble: 3,322 x 9
   tailnum year type      manufacturer model engines seats speed engine
   <chr>   <int> <chr>      <chr>       <chr>   <int> <int> <int> <chr>
1 N10156  2004 Fixed wing m~ EMBRAER    EMB-1~     2     55   NA Turbo--
2 N102UW   1998 Fixed wing m~ AIRBUS INDUST~ A320--     2    182   NA Turbo--
3 N103US   1999 Fixed wing m~ AIRBUS INDUST~ A320--     2    182   NA Turbo--
4 N104UW   1999 Fixed wing m~ AIRBUS INDUST~ A320--     2    182   NA Turbo--
5 N10575   2002 Fixed wing m~ EMBRAER    EMB-1~     2     55   NA Turbo--
6 N105UW   1999 Fixed wing m~ AIRBUS INDUST~ A320--     2    182   NA Turbo--
7 N107US   1999 Fixed wing m~ AIRBUS INDUST~ A320--     2    182   NA Turbo--
8 N108UW   1999 Fixed wing m~ AIRBUS INDUST~ A320--     2    182   NA Turbo--
9 N109UW   1999 Fixed wing m~ AIRBUS INDUST~ A320--     2    182   NA Turbo--
10 N110UW  1999 Fixed wing m~ AIRBUS INDUST~ A320--     2    182   NA Turbo--
```

... with 3,312 more rows

TABLE 4.5: planes 데이터셋 코드북

| 변수 | 설명 |
|--------------|----------|
| tailnum | 항공기 일련번호 |
| year | 제조년도 |
| type | 항공기 유형 |
| manufacturer | 제조사 |
| model | 모델명 |
| engines | 엔진 개수 |
| seats | 좌석 수 |
| speed | 속력 |
| engine | 엔진 유형 |

- weather: 뉴욕시 각 공항 별 날씨 정보

```
# A tibble: 26,115 x 15
```

| | origin | year | month | day | hour | temp | dewp | humid | wind_dir | wind_speed |
|----|--------|-------|-------|-------|-------|-------|-------|-------|----------|------------|
| | <chr> | <int> | <int> | <int> | <int> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> |
| 1 | EWR | 2013 | 1 | 1 | 1 | 39.0 | 26.1 | 59.4 | 270 | 10.4 |
| 2 | EWR | 2013 | 1 | 1 | 2 | 39.0 | 27.0 | 61.6 | 250 | 8.06 |
| 3 | EWR | 2013 | 1 | 1 | 3 | 39.0 | 28.0 | 64.4 | 240 | 11.5 |
| 4 | EWR | 2013 | 1 | 1 | 4 | 39.9 | 28.0 | 62.2 | 250 | 12.7 |
| 5 | EWR | 2013 | 1 | 1 | 5 | 39.0 | 28.0 | 64.4 | 260 | 12.7 |
| 6 | EWR | 2013 | 1 | 1 | 6 | 37.9 | 28.0 | 67.2 | 240 | 11.5 |
| 7 | EWR | 2013 | 1 | 1 | 7 | 39.0 | 28.0 | 64.4 | 240 | 15.0 |
| 8 | EWR | 2013 | 1 | 1 | 8 | 39.9 | 28.0 | 62.2 | 250 | 10.4 |
| 9 | EWR | 2013 | 1 | 1 | 9 | 39.9 | 28.0 | 62.2 | 260 | 15.0 |
| 10 | EWR | 2013 | 1 | 1 | 10 | 41 | 28.0 | 59.6 | 260 | 13.8 |

```
#   precip <dbl>, pressure <dbl>, visib <dbl>, time_hour <dttm>
```

TABLE 4.6: weather 데이터셋 코드북

| 변수 | 설명 |
|------------------------|-----------------------|
| origin | 기상관측소 |
| year, month, day, hour | 년도, 월, 일, 시간 |
| temp, dewp | 기온, 이슬점 (F) |
| humid | 습도 |
| wind_dir, wind_speed, | 바람방향(degree), 풍속 및 돌풍 |
| wind_gust | 속도(mph) |
| precip | 강수량(inch) |
| pressure | 기압(mbar) |
| visib | 가시거리(mile) |
| time_hour | POSIXct 포맷 일자 및 시간 |

열거한 각 테이블은 한 개 또는 복수의 변수로 연결 가능

- flights \leftrightarrow planes (by tailnum)
 - flights \leftrightarrow airlines (by carrier)
 - flights \leftrightarrow airports (by origin, dest)
 - flights \leftrightarrow weather (by origin, year, month, day, hour)
-
- 각 쌍의 데이터를 연결하는데 사용한 변수를 키(key)라고 지칭
 - a. 기준 테이블(여기서는 flights 데이터셋)의 키 \rightarrow 기본키(primary key)
 - b. 병합할 테이블의 키 \rightarrow 외래키(foreign key)
 - c. 다수의 변수를 이용한 기본키 및 외래키 생성 가능

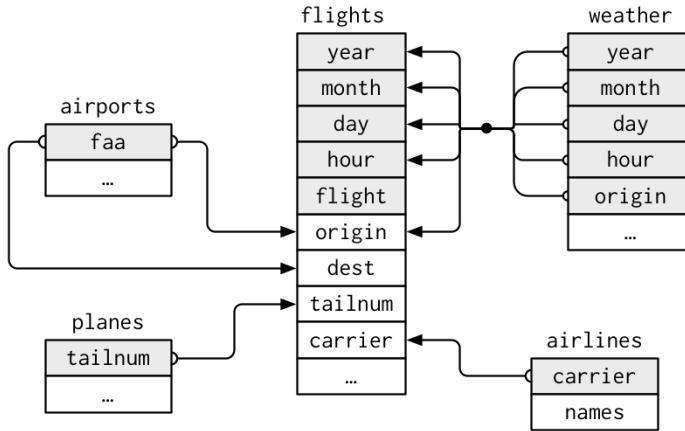


FIGURE 4.8: NYC flight 2013 데이터 관계도 (<https://r4ds.had.co.nz/>에서
발췌)

inner_join

두 데이터셋 모두에 존재하는 키 변수가 일치하는 행을 기준으로 병합

```
x <- tribble(
  ~key, ~val_x,
  1, "x1",
  2, "x2",
  3, "x3"
)
y <- tribble(
  ~key, ~val_y,
  1, "y1",
  2, "y2",
  4, "y3"
)

inner_join(x, y, by = "key") %>% print
```

```
# A tibble: 2 x 3
  key val_x val_y
  <dbl> <chr> <chr>
1     1 x1    y1
2     2 x2    y2

left_join()
```

두 데이터셋 관계 중 왼쪽(x) 데이터셋의 행은 모두 보존

```
left_join(x, y, by = "key") %>% print
```

```
# A tibble: 3 x 3
  key val_x val_y
  <dbl> <chr> <chr>
1     1 x1    y1
2     2 x2    y2
3     3 x3    <NA>
```

```
right_join()
```

두 데이터셋 관계 중 오른쪽(y) 데이터셋의 행은 모두 보존

```
right_join(x, y, by = "key") %>% print
```

```
# A tibble: 3 x 3
  key val_x val_y
  <dbl> <chr> <chr>
1     1 x1    y1
```

```
2      2 x2     y2
3      4 <NA>   y3
```

full_join

두 데이터셋의 관측치 모두를 보존

```
full_join(x, y, by = "key") %>% print
```

```
# A tibble: 4 x 3
  key val_x val_y
  <dbl> <chr> <chr>
1     1 x1     y1
2     2 x2     y2
3     3 x3     <NA>
4     4 <NA>   y3
```

4.4.11.1 NYC flights 2013 데이터 join 예시

```
# flights 데이터 간소화(일부 열만 추출)
flights2 <- flights %>%
  select(year:day, hour, origin, dest, tailnum, carrier)

# flights2 와 airlines 병합
flights2 %>%
  left_join(airlines, by = "carrier") %>%
  print

# A tibble: 336,776 x 9
  year month   day   hour origin dest   tailnum carrier name
  <int> <int> <int> <dbl> <chr> <chr> <chr>   <chr> <chr>
1  2013     1     1     5 EWR    IAH   N14228  UA     United Air Lines Inc.
```

```

2 2013     1     1     5 LGA   IAH   N24211  UA   United Air Lines Inc.
3 2013     1     1     5 JFK   MIA   N619AA  AA   American Airlines Inc.
4 2013     1     1     5 JFK   BQN   N804JB  B6   JetBlue Airways
5 2013     1     1     6 LGA   ATL   N668DN  DL   Delta Air Lines Inc.
6 2013     1     1     5 EWR   ORD   N39463  UA   United Air Lines Inc.
7 2013     1     1     6 EWR   FLL   N516JB  B6   JetBlue Airways
8 2013     1     1     6 LGA   IAD   N829AS  EV   ExpressJet Airlines Inc.
9 2013     1     1     6 JFK   MCO   N593JB  B6   JetBlue Airways
10 2013    1     1     6 LGA  ORD   N3ALAA  AA   American Airlines Inc.

# ... with 336,766 more rows

```

```

# flights2와 airline, airports 병합
## airports 데이터 간소화
airports2 <- airports %>%
  select(faa:name, airport_name = name) %>%
  print

```

```

# A tibble: 1,458 x 2
  faa    airport_name
  <chr> <chr>
1 04G    Lansdowne Airport
2 06A    Moton Field Municipal Airport
3 06C    Schaumburg Regional
4 06N    Randall Airport
5 09J    Jekyll Island Airport
6 0A9    Elizabethton Municipal Airport
7 0G6    Williams County Airport
8 0G7    Finger Lakes Regional Airport
9 0P2    Shoestring Aviation Airfield
10 0S9   Jefferson County Intl

# ... with 1,448 more rows

```

```

flights2 %>%
  left_join(airlines, by = "carrier") %>%
  left_join(airports2, by = c("origin" = "faa")) %>%
  print

# A tibble: 336,776 x 10
  year month   day hour origin dest tailnum carrier name      airport_name
  <int> <int> <int> <dbl> <chr>  <chr> <chr>  <chr>  <chr>    <chr>
1 2013     1     1     5 EWR    IAH    N14228  UA     United Ai~ Newark Liber~
2 2013     1     1     5 LGA    IAH    N24211  UA     United Ai~ La Guardia
3 2013     1     1     5 JFK    MIA    N619AA  AA     American ~ John F Kenne~
4 2013     1     1     5 JFK    BQN    N804JB  B6     JetBlue A~ John F Kenne~
5 2013     1     1     6 LGA    ATL    N668DN  DL     Delta Air~ La Guardia
6 2013     1     1     5 EWR    ORD    N39463  UA     United Ai~ Newark Liber~
7 2013     1     1     6 EWR    FLL    N516JB  B6     JetBlue A~ Newark Liber~
8 2013     1     1     6 LGA    IAD    N829AS  EV     ExpressJe~ La Guardia
9 2013     1     1     6 JFK    MCO    N593JB  B6     JetBlue A~ John F Kenne~
10 2013    1     1     6 LGA   ORD    N3ALAA  AA     American ~ La Guardia
# ... with 336,766 more rows

# flights2와 airline, airports, planes 병합
planes2 <- planes %>%
  select(tailnum, model)

flights2 %>%
  left_join(airlines, by = "carrier") %>%
  left_join(airports2, by = c("origin" = "faa")) %>%
  left_join(planes2, by = "tailnum") %>%
  print

# A tibble: 336,776 x 11
  year month   day hour origin dest tailnum carrier name      airport_name model
  <int> <int> <int> <dbl> <chr>  <chr> <chr>  <chr>  <chr>    <chr>
1 2013     1     1     5 EWR    IAH    N14228  UA     United Ai~ Newark Liber~
2 2013     1     1     5 LGA    IAH    N24211  UA     United Ai~ La Guardia
3 2013     1     1     5 JFK    MIA    N619AA  AA     American ~ John F Kenne~
4 2013     1     1     5 JFK    BQN    N804JB  B6     JetBlue A~ John F Kenne~
5 2013     1     1     6 LGA    ATL    N668DN  DL     Delta Air~ La Guardia
6 2013     1     1     5 EWR    ORD    N39463  UA     United Ai~ Newark Liber~
7 2013     1     1     6 EWR    FLL    N516JB  B6     JetBlue A~ Newark Liber~
8 2013     1     1     6 LGA    IAD    N829AS  EV     ExpressJe~ La Guardia
9 2013     1     1     6 JFK    MCO    N593JB  B6     JetBlue A~ John F Kenne~
10 2013    1     1     6 LGA   ORD    N3ALAA  AA     American ~ La Guardia
# ... with 336,766 more rows

```

```

1 2013    1    1    5 EWR   IAH   N14228  UA     Unit~ Newark Libe~ 737~~
2 2013    1    1    5 LGA   IAH   N24211  UA     Unit~ La Guardia  737~~
3 2013    1    1    5 JFK   MIA   N619AA  AA     Amer~ John F Kenn~ 757~~
4 2013    1    1    5 JFK   BQN   N804JB  B6     JetB~ John F Kenn~ A320~
5 2013    1    1    6 LGA   ATL   N668DN  DL     Delt~ La Guardia  757~~
6 2013    1    1    5 EWR   ORD   N39463  UA     Unit~ Newark Libe~ 737~~
7 2013    1    1    6 EWR   FLL   N516JB  B6     JetB~ Newark Libe~ A320~
8 2013    1    1    6 LGA   IAD   N829AS  EV     Expr~ La Guardia  CL-6~
9 2013    1    1    6 JFK   MCO   N593JB  B6     JetB~ John F Kenn~ A320~
10 2013   1    1    6 LGA  ORD   N3ALAA  AA     Amer~ La Guardia <NA>
# ... with 336,766 more rows

```

```

# flights2 와 airline, airports2, planes2, weather2 병합
## weather 데이터 간소화
weather2 <- weather %>%
  select(origin:temp, wind_speed)

flights2 %>%
  left_join(airlines, by = "carrier") %>%
  left_join(airports2, by = c("origin" = "faa")) %>%
  left_join(planes2, by = "tailnum") %>%
  left_join(weather2, by = c("origin", "year",
                             "month", "day", "hour")) %>%
  glimpse

```

Observations: 336,776

Variables: 13

```

$ year      <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, ...
$ month     <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
$ day       <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
$ hour      <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6, 5, 6, 6, ...
$ origin    <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR", "LGA"...
$ dest      <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL", "IAD"...

```

```
$ tailnum      <chr> "N14228", "N24211", "N619AA", "N804JB", "N668DN", "N39...
$ carrier      <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV", "B6", ...
$ name         <chr> "United Air Lines Inc.", "United Air Lines Inc.", "Ame...
$ airport_name <chr> "Newark Liberty Intl", "La Guardia", "John F Kennedy I...
$ model        <chr> "737-824", "737-824", "757-223", "A320-232", "757-232"...
$ temp          <dbl> 39.02, 39.92, 39.02, 39.02, 39.92, 39.02, 37.94, 39.92...
$ wind_speed   <dbl> 12.65858, 14.96014, 14.96014, 14.96014, 16.11092, 12.6...
```

`dplyr *_join()` 과 `base` 패키지의 `merge()` 비교

TABLE 4.7: `dplyr join` 함수와 `merge()` 함수 비교

| <code>dplyr::*_join()</code> | <code>base::merge()</code> |
|-------------------------------|--|
| <code>inner_join(x, y)</code> | <code>merge(x, y)</code> |
| <code>left_join(x, y)</code> | <code>merge(x, y, all.x = TRUE)</code> |
| <code>right_join(x, y)</code> | <code>merge(x, y, all.y = TRUE)</code> |
| <code>full_join(x, y)</code> | <code>merge(x, y, all.x = TRUE,</code> <code>all.y = TRUE)</code> |

4.4.12 확장 예제: Gapminder



연습 데이터: Gapminder 데이터 활용. 각 대륙에 속한 국가의 인구, 경제, 보건, 교육, 환경, 노동에 대한 연도 별 국가 통계를 제공함. Gapminder⁷는 스웨덴의 비영리 통계 분석 서비스를 제공하는 웹사이트로, UN이 제공하는 데이터를 기반으로 인구 예측, 부의 이동 등에 관한 연구 논문 및 통계정보, 데이터를 공유함 (gap, 2018). R 패키지 중 `gapminder` (Bryan, 2017)은 1950 ~ 2007년 까지 5년 단위의 국가별 인구(population), 기대수명(year), 일인당 국민 총소득(gross domestic product per capita, 달러)에 대한 데이터를 제공 하지만, 본 강의에서는 현재 Gapminder 사이트에서 직접 다운로드 받은 가장 최근 데이터를 가지고 `dplyr` 패키지의 기본 동사를 학습함과 동시에 최종적으로 `gapminder` 패키지에서 제공하는 데이터와 동일한 형태의 구조를 갖는 데이터를 생성하는 것이 목직임.

해당 데이터는 github 계정⁸에서 다운로드가 가능함. `gapminder-exercise.xlsx`는 총 4 개의 시트로 구성되어 있으며, 각 시트에 대한 설명은 아래와 같음.

TABLE 4.8: `gapminder-exercise.xlsx` 설명

| 시트 이름 | 설명 |
|--------------------------|--------------------------------|
| <code>region</code> | 국가별 지역 정보 포함 |
| <code>country_pop</code> | 국가별 1800 ~ 2100년 까지 추계 인구수(명) |
| <code>gdpcap</code> | 국가별 1800 ~ 2100년 까지 국민 총소득(달러) |
| <code>lifeexp</code> | 국가별 1800 ~ 2100년 까지 기대수명(세) |

Prerequisites

gapminder 패키지 설치하기

```
install.packages("gapminder")
```



Gapminder 데이터 핸들링 실습

1. `readxl` 패키지 + `%>%` 를 이용해 Gapminder 데이터 (`gapminder-exercise.xlsx`) 불러오기

```
require(readxl)
require(gapminder)
```

필요한 패키지를 로딩중입니다: `gapminder`

```

path <- "dataset/gapminder/gapminder-exercise.xlsx"
# base R 문법 적용
# sheet_name <- excel_sheets(path)
# gapmL <- lapply(sheet_name, function(x) read_excel(path = path, sheet = x))
# names(gapmL) <- sheet_name

# pipe 연산자 이용
path %>%
  excel_sheets %>%
  set_names %>%
  map(read_excel, path = path) -> gapmL

# 개별 객체에 데이터 저장
command <- paste(names(gapmL), "<-",
                  paste0("gapmL$", names(gapmL)))
for (i in 1:length(command)) eval(parse(text = command[i]))

# check
ls()

```

```

[1] "airports2"          "audi"           "base::merge()"
[4] "by_mpg"              "codebook"        "command"
[7] "country_pop"        "covid19"         "dbp"
[10] "dbp2"                "dd"              "def.chunk.hook"
[13] "diab"                "dL"              "dL2"
[16] "dplyr::*_join()"    "flights2"       "gallon"
[19] "gapmL"               "gdpcap"         "herb"
[22] "hook_output"         "i"               "input1"
[25] "input2"               "kpl"             "lifeexp"
[28] "mile"                 "mpg"            "mpg_asc"
[31] "mpg_sortb"            "mpg_sortt"      "mpg_uniq"
[34] "mpg_uniq2"            "path"           "planes2"

```

```
[37] "plasma"          "pulse"           "R base 패키지 함수"
[40] "region"          "sheet_name"       "slice_mpg"
[43] "slice_mpg_grp"   "tab4_01"          "tab4_03"
[46] "tab4_04"          "tab4_05"          "tab4_06"
[49] "tab4_07"          "tab4_08"          "titanic"
[52] "titanic2"         "titanic3"         "varn_mpg"
[55] "varname"          "weather2"         "x"
[58] "y"                "내용"            "동사(함수)"
[61] "변수"             "변수명"          "변수설명(국문)"
[64] "변수설명(영문)"  "설명"            "시트 이름"
[67] "연비"
```

```
region %>% print
```

```
# A tibble: 234 x 3
  iso    country      region
  <chr> <chr>        <chr>
  1 AFG  Afghanistan Southern Asia
  2 ALB  Albania      Southern Europe
  3 DZA  Algeria      Northern Africa
  4 ASM  American Samoa Polynesia Oceania
  5 AND  Andorra      Southern Europe
  6 AGO  Angola       Middle Africa
  7 AIA  Anguilla     Caribbean America
  8 ATG  Antigua and Barbuda Caribbean America
  9 ARG  Argentina    South America
 10 ARM  Armenia      Western Asia
# ... with 224 more rows
```

```
country_pop %>% print
```

```
# A tibble: 59,297 x 4
  iso    country      year population
  <chr> <chr>        <dbl>     <dbl>
```

```

<chr> <chr>      <dbl>      <dbl>
1 afg   Afghanistan 1800    3280000
2 afg   Afghanistan 1801    3280000
3 afg   Afghanistan 1802    3280000
4 afg   Afghanistan 1803    3280000
5 afg   Afghanistan 1804    3280000
6 afg   Afghanistan 1805    3280000
7 afg   Afghanistan 1806    3280000
8 afg   Afghanistan 1807    3280000
9 afg   Afghanistan 1808    3280000
10 afg  Afghanistan 1809   3280000
# ... with 59,287 more rows

```

```
gdpcap %>% print
```

```

# A tibble: 46,995 x 4
  iso_code country     year gdp_total
  <chr>     <chr>     <dbl>     <dbl>
1 afg      Afghanistan 1800 1977840000
2 afg      Afghanistan 1801 1977840000
3 afg      Afghanistan 1802 1977840000
4 afg      Afghanistan 1803 1977840000
5 afg      Afghanistan 1804 1977840000
6 afg      Afghanistan 1805 1977840000
7 afg      Afghanistan 1806 1977840000
8 afg      Afghanistan 1807 1977840000
9 afg      Afghanistan 1808 1977840000
10 afg     Afghanistan 1809 1977840000
# ... with 46,985 more rows

```

```
lifeexp %>% print
```

```
# A tibble: 56,130 x 4
```

```

country      iso_code  year life_expectancy
<chr>        <chr>    <dbl>           <dbl>
1 Afghanistan afg     1800            28.2
2 Afghanistan afg     1801            28.2
3 Afghanistan afg     1802            28.2
4 Afghanistan afg     1803            28.2
5 Afghanistan afg     1804            28.2
6 Afghanistan afg     1805            28.2
7 Afghanistan afg     1806            28.2
8 Afghanistan afg     1807            28.1
9 Afghanistan afg     1808            28.1
10 Afghanistan afg    1809            28.1
# ... with 56,120 more rows

```

2. country_pop, gdpcap, lifeexp 데이터 셋 결합

```

gap_unfilter <- country_pop %>%
  left_join(gdpcap, by = c("iso" = "iso_code",
                           "country",
                           "year")) %>%
  left_join(lifeexp, by = c("iso" = "iso_code",
                           "country",
                           "year"))

gap_unfilter %>% print

```

```

# A tibble: 59,297 x 6
  iso   country      year population  gdp_total life_expectancy
  <chr> <chr>    <dbl>       <dbl>       <dbl>           <dbl>
1 afg   Afghanistan 1800     3280000 1977840000            28.2
2 afg   Afghanistan 1801     3280000 1977840000            28.2
3 afg   Afghanistan 1802     3280000 1977840000            28.2
4 afg   Afghanistan 1803     3280000 1977840000            28.2
5 afg   Afghanistan 1804     3280000 1977840000            28.2

```

```

6 afg  Afghanistan  1805    3280000 1977840000      28.2
7 afg  Afghanistan  1806    3280000 1977840000      28.2
8 afg  Afghanistan  1807    3280000 1977840000      28.1
9 afg  Afghanistan  1808    3280000 1977840000      28.1
10 afg  Afghanistan 1809    3280000 1977840000      28.1
# ... with 59,287 more rows

```

3. 인구 수 6만 이상, 1950 ~ 2020년 년도 추출

```

gap_filter <- gap_unfilter %>%
  filter(population >= 60000,
         between(year, 1950, 2020))
gap_filter %>% print

```

```

# A tibble: 13,159 x 6
  iso   country     year population   gdp_total life_expectancy
  <chr> <chr>     <dbl>      <dbl>        <dbl>            <dbl>
1 afg  Afghanistan 1950    7752117 18543063864      32.5
2 afg  Afghanistan 1951    7840151 18988845722      32.9
3 afg  Afghanistan 1952    7935996 19538422152      33.6
4 afg  Afghanistan 1953    8039684 20645908512      34.3
5 afg  Afghanistan 1954    8151316 20997790016      35.0
6 afg  Afghanistan 1955    8270992 21330888368      35.7
7 afg  Afghanistan 1956    8398873 22206620212      36.4
8 afg  Afghanistan 1957    8535157 22131662101      37.1
9 afg  Afghanistan 1958    8680097 23297380348      37.9
10 afg  Afghanistan 1959   8833947 23895826635      38.6
# ... with 13,149 more rows

```

4. iso 변수 값을 대문자로 변환하고, 1인당 국민소득 (gdp_total/population) 변수 gdp_cap 생성 후 gdp_total 제거

```

gap_filter <- gap_filter %>%
  mutate(iso = toupper(iso),
        gdp_cap = gdp_total/population) %>%
  select(-gdp_total)
gap_filter %>% print

# A tibble: 13,159 x 6
  iso   country     year population life_expectancy gdp_cap
  <chr> <chr>     <dbl>      <dbl>            <dbl>      <dbl>
1 AFG  Afghanistan 1950    7752117          32.5    2392
2 AFG  Afghanistan 1951    7840151          32.9    2422
3 AFG  Afghanistan 1952    7935996          33.6    2462
4 AFG  Afghanistan 1953    8039684          34.3    2568
5 AFG  Afghanistan 1954    8151316          35.0    2576
6 AFG  Afghanistan 1955    8270992          35.7    2579
7 AFG  Afghanistan 1956    8398873          36.4    2644
8 AFG  Afghanistan 1957    8535157          37.1    2593
9 AFG  Afghanistan 1958    8680097          37.9    2684
10 AFG  Afghanistan 1959   8833947          38.6    2705
# ... with 13,149 more rows

```

5. region 데이터셋에서 대륙(region) 변수 결합

```

gap_filter <- gap_filter %>%
  left_join(region %>% select(-country),
            by = c("iso"))
gap_filter %>% print

# A tibble: 13,159 x 7
  iso   country     year population life_expectancy gdp_cap region
  <chr> <chr>     <dbl>      <dbl>            <dbl>      <dbl> <chr>
1 AFG  Afghanistan 1950    7752117          32.5    2392 Southern Asia
2 AFG  Afghanistan 1951    7840151          32.9    2422 Southern Asia

```

```

3 AFG  Afghanistan  1952    7935996      33.6    2462 Southern Asia
4 AFG  Afghanistan  1953    8039684      34.3    2568 Southern Asia
5 AFG  Afghanistan  1954    8151316      35.0    2576 Southern Asia
6 AFG  Afghanistan  1955    8270992      35.7    2579 Southern Asia
7 AFG  Afghanistan  1956    8398873      36.4    2644 Southern Asia
8 AFG  Afghanistan  1957    8535157      37.1    2593 Southern Asia
9 AFG  Afghanistan  1958    8680097      37.9    2684 Southern Asia
10 AFG  Afghanistan  1959   8833947      38.6    2705 Southern Asia
# ... with 13,149 more rows

```

6. 변수 정렬 (iso, country, region, year:gdp_cap 순서로)

```

gap_filter <- gap_filter %>%
  select(iso:country, region, everything())

```

7. 문자형 변수를 요인형으로 변환하고, population을 정수형으로 변환

```

gap_filter <- gap_filter %>%
  mutate_if(~ is.character(.), ~factor(.)) %>%
  mutate(population = as.integer(population))

```

8. 2-7 절차를 pipe로 묶으면

```

gap_filter <- country_pop %>%
  left_join(gdpcap, by = c("iso" = "iso_code",
                           "country",
                           "year")) %>%
  left_join(lifeexp, by = c("iso" = "iso_code",
                           "country",
                           "year")) %>%
  filter(population >= 60000,
         between(year, 1950, 2020)) %>%

```

```

    mutate(iso = toupper(iso),
           gdp_cap = gdp_total/population) %>%
    select(-gdp_total) %>%
    left_join(region %>% select(-country),
              by = c("iso")) %>%
    select(iso:country, region, everything()) %>%
    mutate_if(~ is.character(.), ~factor(.)) %>%
    mutate(population = as.integer(population))

# write_csv(gap_filter, "dataset/gapminder/gapminder_filter.csv")

```

9. 2020년 현재 지역별 인구수, 평균 일인당 국민소득, 평균 기대수명 계산 후
인구 수로 내림차순 정렬

```

gap_filter %>%
  filter(year == 2020) %>%
  group_by(region) %>%
  summarise(Population = sum(population),
            `GDP/Captia` = mean(gdp_cap),
            `Life expect` = mean(life_expectancy, na.rm = TRUE)) %>%
  arrange(desc(Population))

```

```

# A tibble: 22 x 4
  region          Population `GDP/Captia` `Life expect`
  <fct>           <int>        <dbl>        <dbl>
1 Southern Asia   1940369605     8371.       73.4
2 Eastern Asia    1677440285     30459.      78.9
3 South-Eastern Asia 668619854     24279.      73.8
4 Eastern Africa  444237457      4692.       66.1
5 South America   430457607     13721.      76.4
6 Western Africa  401855184      2882.       65.6
7 Northen America 368744804     51032.      80.5

```

```

8 Eastern Europe      293013210      23683.      75.6
9 Western Asia        279636774      30616.      77.3
10 Northern Africa    245635178      10702.      74.6
# ... with 12 more rows

```

 지금까지 배운 dplyr 패키지와 관련 명령어를 포함한 주요 동사 및 함수에 대한 개괄적 사용 방법은 RStudio에서 제공하는 cheat sheet⁹을 통해 개념을 다질 수 있음.

4.5 데이터 변환



Tidy data에 대한 개념을 알아보고, tidyverse 패키지에서 제공하는 데이터 변환 함수 사용 방법에 대해 익힌다.

- 데이터 분석에서 적어도 80% 이상의 시간을 데이터 전처리에 할애
- 실제 데이터 (real world data, RWD)가 우리가 지금까지 다뤄왔던 예제 데이터처럼 깔끔하게 정리된 경우는 거의 없음.
 - 이상치 (outlier)
 - 결측 (missing data)
 - 변수 정의의 부재 (예: 여러 가지 변수 값이 혼합되어 한 열로 포함된 경우)
 - 비정형 문자열로 구성된 변수
 - 불분명한 데이터 구조
 - ...
- Tidyverse 세계에서 지저분한 데이터 (messy data)를 분석이 용이하고

(전산 처리가 쉽고) 깔끔한 데이터(tidy data)로 정제하기 위해 데이터의 구조를 변환하는 함수를 포함하고 있는 패키지가 **tidyverse**

- 여기서 “tidy”는 “organized”와 동일한 의미를 가짐
- **tidyverse**은 Hadley Wickam 이 개발한 **reshape** 와 **reshape2** 패키지 (Wickham, 2007)가 포함하고 있는 전반적인 데이터 변환 함수 중 tidy data를 만드는데 핵심적인 함수만으로 구성된 패키지

4.5.1 Tidy data



시작 전 tidyverse 패키지를 R 작업공간으로 불러오기!!

아래 강의 내용은 Wickham et al. (2014) 의 내용을 재구성함

- 데이터셋의 구성 요소
 - a. 데이터셋은 관측값(value)으로 구성
 - b. 각 관찰값은 변수(variable)와 관측(observation) 단위에 속함
 - c. 변수는 측정 속성과 단위가 동일한 값으로 구성(예: 키, 몸무게, 체온 등)
 - d. 관측(observation)은 속성(변수) 전체에서 동일한 단위(예: 사람, 가구, 지역 등)에서 측정된 모든 값



Tidy Data Principles

- 각각의 변수는 하나의 열로 구성된다(Each variable forms a column).
- 각각의 관측은 하나의 행으로 구성된다(Each observation forms a row).
- 각각의 값은 반드시 자체적인 하나의 셀을 가져야 한다(Each value must have its own cell).
- 각각의 관찰 단위는 테이블을 구성한다(Each type of observational unit forms a table).

| 변수1 | 변수2 | 변수3 | 변수4 |
|-----|-----|-----|------|
| aaa | 174 | M | 23.6 |
| bbb | 158 | F | 21.2 |
| ccc | 182 | M | 27.8 |
| ddd | 178 | M | 25.4 |
| eee | 152 | F | 19.2 |
| fff | 167 | F | 22.7 |
| ggg | 160 | M | 28.4 |

| 변수1 | 변수2 | 변수3 | 변수4 |
|-----|-----|-----|------|
| aaa | 174 | M | 23.6 |
| bbb | 158 | F | 21.2 |
| ccc | 182 | M | 27.8 |
| ddd | 178 | M | 25.4 |
| eee | 152 | F | 19.2 |
| fff | 167 | F | 22.7 |
| ggg | 160 | M | 28.4 |

| | | | |
|-----|-----|-----|------|
| 변수1 | 변수2 | 변수3 | 변수4 |
| aaa | 174 | M | 23.6 |
| bbb | 158 | F | 21.2 |
| ccc | 182 | M | 27.8 |
| ddd | 178 | M | 25.4 |
| eee | 152 | F | 19.2 |
| fff | 167 | F | 22.7 |
| ggg | 160 | M | 28.4 |

FIGURE 4.9: 데이터의 구성 요소

- 2×2 교차설계 데이터 예시: 2개의 열(column), 3개의 행(row), 각 행과 열은 이름을 갖고 있음(labelled)

TABLE 4.9: Tidy data 예시 데이터 1

| | treatmenta | treatmentb |
|-----------------|------------|------------|
| James McGill | NA | 1 |
| Kimberly Wexler | 17 | 14 |
| Lalo Salamanca | 8 | 19 |

데이터셋에서 중요한 요인인 배정군의 수준이 변수로 사용 → a와 b는 treatment의 하위 수준임

- 예시 데이터 1 전치 → 위 데이터 셋과 동일한 내용이지만 다른 레이아웃 형태

TABLE 4.10: Tidy data: 예시 데이터 1과 동일 내용, 다른 레이아웃

| | James McGill | Kimberly Wexler | Lalo Salamanca |
|------------|--------------|-----------------|----------------|
| treatmenta | NA | 17 | 8 |
| treatmentb | 1 | 14 | 19 |

관측 단위가 변수로 사용

- 위 예시 데이터 1을 재정의
 1. person: 3 개의 값(James, Kimberly, Lalo)
 2. treatment: 2 개의 값(a, b)
 3. result: 6 개의 값(결측 포함) person과 treatment의 조합

TABLE 4.11: Tidy data: 예시 데이터 1 구조 변환

| name | treatment | result |
|-----------------|-----------|--------|
| James McGill | a | NA |
| Kimberly Wexler | a | 17 |
| Lalo Salamanca | a | 8 |
| James McGill | b | 1 |
| Kimberly Wexler | b | 14 |
| Lalo Salamanca | b | 19 |

위 데이터는

- 모든 행(row)은 observation을 나타냄

- 모든 `result`에 해당하는 값(**value**)은 하나의 `treatment`과 하나의 `person`에 대응함
- 모든 열은 변수(**variable**)

→ **tidy data** 원칙을 만족

Tidy data의 장점

- 표준화된 데이터 구조로 변수 및 관측치 추출이 용이
- 일관된 데이터 구조를 유지된다면 이와 관련한 도구(함수)를 배우는 것이 보다 용이함
- R의 vectorized programming 환경에 최적화 → 동일한 관측에 대한 서로 다른 변수값이 항상 짹으로 매칭되는 것을 보장

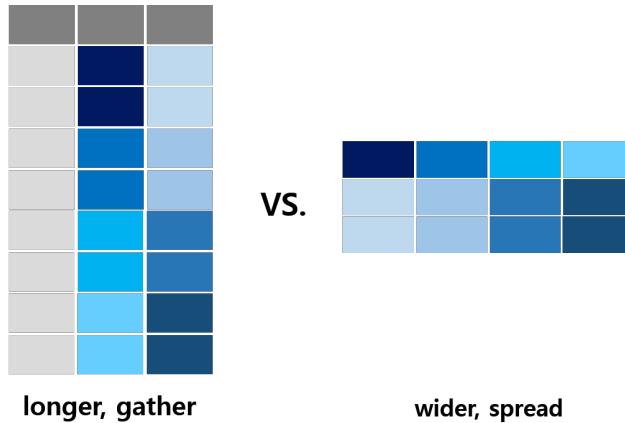
Messy data의 일반적인 문제점

- 열 이름을 변수명이 아닌 값(**value**)으로 사용
- 두 개 이상의 변수에 해당하는 값이 하나의 열에 저장

이러한 문제를 해결(데이터 정돈)하기 위해 데이터의 구조 변환은 필수적이며,

이를 위해 tidyverse 패키지에서 아래와 같은 패키지 제공

- `gather()`, `pivot_longer()`: 아래로 긴 데이터 셋(melt된 데이터셋) → **long format**
- `spread()`, `pivot_wider()`: 옆으로 긴 데이터 셋(열에 cast된 데이터셋) → **wide format**



- long format: 데이터가 적은 수의 열로 이루어지며, 각 열 보통 행의 unique 한 정보를 표현하기 위한 ID(또는 key)로 구성되어 있고 보통은 관측된 변수에 대한 한 개의 열로 구성된 데이터 형태
- wide format: 통계학에서 다루는 데이터 구조와 동일한 개념으로 한 관측 단위(사람, 가구 등)가 한 행을 이루고, 관측 단위에 대한 측정값(예: 키, 몸무게, 성별)들이 변수(열)로 표현된 데이터 형태

```

      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
Valiant        18.1   6 225 105 2.76 3.460 20.22  1  0    3    1

# A tibble: 352 x 3
  model      variable  value
  <chr>      <chr>    <dbl>
1 AMC Javelin mpg      15.2
2 AMC Javelin cyl      8
3 AMC Javelin disp    304

```

```

4 AMC Javelin hp      150
5 AMC Javelin drat     3.15
6 AMC Javelin wt       3.44
7 AMC Javelin qsec     17.3
8 AMC Javelin vs        0
9 AMC Javelin am        0
10 AMC Javelin gear      3
# ... with 342 more rows

```



dplyr 패키지와 마찬가지로 tidyr 패키지에서 제공하는 함수는 데이터 프레임 또는 티블에 서만 작동함

4.5.2 Long format

- “long” 형태의 데이터 구조는 “wide” 형태의 데이터 보다 “사람”이 이해하기에 편한 형태는 아니지만 아래와 같은 장점을 가짐
 - “컴퓨터”가 이해하기 편한 구조
 - “wide” 형태보다 유연 → 데이터의 추가 및 삭제 용이
- “wide” 형태의 데이터를 “long” 형태로 변환 해주는 tidyr 패키지 내장 함수는
 - pivot_longer()**: 데이터의 행의 길이를 늘리고 열의 개수를 줄이는 함수
 - gather()**: pivot_longer()의 이전 버전으로 보다 쉽게 사용할 수 있고, 함수 명칭도 보다 직관적이지만 함수 업데이트는 종료

“wide” 형태의 데이터를 “long” 형태로 바꾸는 것은 원래 구조의 데이터를 녹여서 (melt) 길게 만든다는 의미로도 해석할 수 있음. tidyr의 초기 버전인 reshape 패키지에서 pivot_wider() 또는 gather()와 유사한 기능을 가진

함수 이름이 `melt()` 임. 본 강의에서는 `melt()` 함수의 사용 방법에 대해서는
다루지 않음.

```
# pivot_longer()의 기본 사용 형태
pivot_longer(
  data, # 데이터 프레임
  cols, # long format으로 변경을 위해 선택한 열 이름
  # dplyr select() 함수에서 사용한 변수선정 방법 준용
  names_to, # 선택한 열 이름을 값으로 갖는 변수명칭 지정
  names_prefix, # 변수명에 처음에 붙는 접두어 패턴 제거 (예시 참조, optional)
  names_pattern, # 정규표현식의 그룹지정(())에 해당하는 패턴을 값으로 사용
  # 예시 참조(optional)
  values_to # 선택한 열에 포함되어 있는 셀 값을 저장할 변수 이름 지정
)

# gather() 기본 사용 형태
gather(
  data, # 데이터 프레임
  key, # 선택한 열 이름을 값으로 갖는 변수명칭 지정
  value, # 선택한 열에 포함되어 있는 셀 값을 저장할 변수 이름 지정
  ... # long format으로 변경할 열 선택
)
```

```
pivot_longer(cols = V1:V3, names_to = "key", values_to = "value")
```



Examples

1. 열의 이름이 변수명이 아니라 값으로 표현된 경우

```
# 데이터 불러오기: read_csv() 함수를 이용해
# tidyverse-ex01.csv 파일 불러오기
wide_01 <- read_csv("dataset/tidyverse-ex01.csv")
wide_01
```

```
# A tibble: 3 x 21
  country `2001` `2002` `2003` `2004` `2005` `2006` `2007` `2008` `2009` `2010` 
  <chr>    <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl> 
1 Germany  37325. 37262. 36977. 37418. 37704. 39143. 40474. 40989. 38784. 40429.
2 South     21530. 22997. 23549. 24606. 25517. 26697. 28014. 28588. 28643. 30352.
3 United~  45663. 46029. 46941. 48275. 49513. 50438. 50898. 50350. 48644. 49479.
# ... with 10 more variables: `2011` <dbl>, `2012` <dbl>, `2013` <dbl>,
#   `2014` <dbl>, `2015` <dbl>, `2016` <dbl>, `2017` <dbl>, `2018` <dbl>,
#   `2019` <dbl>, `2020` <dbl>
```

- 총 21개의 열과 3개의 행으로 구성된 “wide” 형태 데이터 구조
- 열 이름 2001 ~ 2020은 2001년부터 2020년 까지 년도을 의미함

- 현재 데이터 구조에서 각 셀의 값(value)은 일인당 국민소득을 나타냄
- 한 행은 국가(country)의 2001년부터 2020년 까지 년도 별 일인당 국민소득
- 여기서 관측 단위(observational unit)은 국가(country)이며, 각 국가는 2001년부터 2020년 까지 일인당 국민소득에 대한 관찰값을 가짐

위 데이터가 tidy data 원칙을 준수하려면 어떤 형태로 재구성 되야 할까?

- 열 이름은 년도에 해당하는 값(value)임 → year라는 새로운 변수에 해당 값을 저장
- 일인당 국민소득 정보를 포함한 gdp_cap이라는 변수 생성
- 대략 아래와 같은 형태의 데이터

| country | year | gdp_cap |
|-------------|------|----------|
| South Korea | 2001 | 21530.31 |
| South Korea | 2002 | 22887.09 |
| South Korea | 2003 | 23549.29 |
| South Korea | 2004 | 24605.60 |
| South Korea | 2005 | 25516.89 |
| South Korea | 2006 | 26696.97 |
| South Korea | : | : |
| South Korea | 2020 | 38016.79 |

long format 의 데이터 구조

- Unique한 각각의 관측 결과(대한민국의 2001년 일인당 국민소득이 얼마)는 하나의 행에 존재
- 데이터에서 변수로 표현할 수 있는 속성은 모두 열로 표시
- 각 변수에 해당하는 값(value)은 하나의 셀에 위치

→ **tidy data** 원칙 만족

- 예시 1: `wide_01` 데이터셋

```
# wide_01 데이터 tidyizing
## pivot_wider() 사용
tidy_ex_01 <- wide_01 %>%
  pivot_longer(`2001`:`2020`,
              names_to = "year",
              values_to = "gdp_cap")
tidy_ex_01 %>% print

# A tibble: 60 x 3
  country year   gdp_cap
  <chr>    <chr>   <dbl>
1 Germany 2001    37325.
2 Germany 2002    37262.
3 Germany 2003    36977.
4 Germany 2004    37418.
5 Germany 2005    37704.
6 Germany 2006    39143.
7 Germany 2007    40474.
8 Germany 2008    40989.
9 Germany 2009    38784.
10 Germany 2010   40429.
# ... with 50 more rows

## gather() 사용
tidy_ex_01 <- wide_01 %>%
  gather(year, gdp_cap, `2001`:`2020`)
tidy_ex_01 %>% print
```

```
# A tibble: 60 x 3
  country     year   gdp_cap
  <chr>      <chr>   <dbl>
1 Germany    2001    37325.
2 South Korea 2001    21530.
3 United States 2001    45663.
4 Germany    2002    37262.
5 South Korea 2002    22997.
6 United States 2002    46029.
7 Germany    2003    36977.
8 South Korea 2003    23549.
9 United States 2003    46941.
10 Germany   2004    37418.
# ... with 50 more rows
```

- 예시 2: `tidyverse` 패키지에 내장되어 있는 `billboard` 데이터셋
(`help(billboard)` 참고)

```
billboard %>% print
```

```
# A tibble: 317 x 79
  artist track date.entered   wk1   wk2   wk3   wk4   wk5   wk6   wk7   wk8
  <chr>  <chr> <date>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 2 Pac Baby~ 2000-02-26     87    82    72    77    87    94    99    NA
2 2Ge+h~ The ~ 2000-09-02    91    87    92    NA    NA    NA    NA    NA
3 3 Doo~ Kryp~ 2000-04-08    81    70    68    67    66    57    54    53
4 3 Doo~ Loser 2000-10-21    76    76    72    69    67    65    55    59
5 504 B~ Wobb~ 2000-04-15    57    34    25    17    17    31    36    49
6 98~0 Give~ 2000-08-19    51    39    34    26    26    19     2     2
7 A*Tee~ Danc~ 2000-07-08    97    97    96    95    100   NA    NA    NA
8 Aaliy~ I Do~ 2000-01-29    84    62    51    41    38    35    35    38
9 Aaliy~ Try ~ 2000-03-18    59    53    38    28    21    18    16    14
10 Adams~ Open~ 2000-08-26   76    76    74    69    68    67    61    58
```

```
# ... with 307 more rows, and 68 more variables: wk9 <dbl>, wk10 <dbl>,
#   wk11 <dbl>, wk12 <dbl>, wk13 <dbl>, wk14 <dbl>, wk15 <dbl>, wk16 <dbl>,
#   wk17 <dbl>, wk18 <dbl>, wk19 <dbl>, wk20 <dbl>, wk21 <dbl>, wk22 <dbl>,
#   wk23 <dbl>, wk24 <dbl>, wk25 <dbl>, wk26 <dbl>, wk27 <dbl>, wk28 <dbl>,
#   wk29 <dbl>, wk30 <dbl>, wk31 <dbl>, wk32 <dbl>, wk33 <dbl>, wk34 <dbl>,
#   wk35 <dbl>, wk36 <dbl>, wk37 <dbl>, wk38 <dbl>, wk39 <dbl>, wk40 <dbl>,
#   wk41 <dbl>, wk42 <dbl>, wk43 <dbl>, wk44 <dbl>, wk45 <dbl>, wk46 <dbl>,
#   wk47 <dbl>, wk48 <dbl>, wk49 <dbl>, wk50 <dbl>, wk51 <dbl>, wk52 <dbl>,
#   wk53 <dbl>, wk54 <dbl>, wk55 <dbl>, wk56 <dbl>, wk57 <dbl>, wk58 <dbl>,
#   wk59 <dbl>, wk60 <dbl>, wk61 <dbl>, wk62 <dbl>, wk63 <dbl>, wk64 <dbl>,
#   wk65 <dbl>, wk66 <lgl>, wk67 <lgl>, wk68 <lgl>, wk69 <lgl>, wk70 <lgl>,
#   wk71 <lgl>, wk72 <lgl>, wk73 <lgl>, wk74 <lgl>, wk75 <lgl>, wk76 <lgl>

names(billboard)

[1] "artist"      "track"        "date.entered" "wk1"          "wk2"
[6] "wk3"         "wk4"          "wk5"          "wk6"          "wk7"
[11] "wk8"         "wk9"          "wk10"         "wk11"         "wk12"
[16] "wk13"        "wk14"         "wk15"         "wk16"         "wk17"
[21] "wk18"        "wk19"         "wk20"         "wk21"         "wk22"
[26] "wk23"        "wk24"         "wk25"         "wk26"         "wk27"
[31] "wk28"        "wk29"         "wk30"         "wk31"         "wk32"
[36] "wk33"        "wk34"         "wk35"         "wk36"         "wk37"
[41] "wk38"        "wk39"         "wk40"         "wk41"         "wk42"
[46] "wk43"        "wk44"         "wk45"         "wk46"         "wk47"
[51] "wk48"        "wk49"         "wk50"         "wk51"         "wk52"
[56] "wk53"        "wk54"         "wk55"         "wk56"         "wk57"
[61] "wk58"        "wk59"         "wk60"         "wk61"         "wk62"
[66] "wk63"        "wk64"         "wk65"         "wk66"         "wk67"
[71] "wk68"        "wk69"         "wk70"         "wk71"         "wk72"
[76] "wk73"        "wk74"         "wk75"         "wk76"
```

```
# pivot_wider()을 이용해 데이터 정돈
billb_tidy <- billboard %>%
  pivot_longer(starts_with("wk"),
               names_to = "week",
               values_to = "rank")
billb_tidy %>% print

# A tibble: 24,092 x 5
  artist   track date.entered week   rank
  <chr>   <chr>    <date>     <chr> <dbl>
1 2 Pac Baby Don't Cry (Keep... 2000-02-26 wk1     87
2 2 Pac Baby Don't Cry (Keep... 2000-02-26 wk2     82
3 2 Pac Baby Don't Cry (Keep... 2000-02-26 wk3     72
4 2 Pac Baby Don't Cry (Keep... 2000-02-26 wk4     77
5 2 Pac Baby Don't Cry (Keep... 2000-02-26 wk5     87
6 2 Pac Baby Don't Cry (Keep... 2000-02-26 wk6     94
7 2 Pac Baby Don't Cry (Keep... 2000-02-26 wk7     99
8 2 Pac Baby Don't Cry (Keep... 2000-02-26 wk8     NA
9 2 Pac Baby Don't Cry (Keep... 2000-02-26 wk9     NA
10 2 Pac Baby Don't Cry (Keep... 2000-02-26 wk10    NA
# ... with 24,082 more rows
```

```
# pivot_longer() 함수의 인수 중 value_drop_na 값 조정을 통해
# 데이터 값에 포함된 결측 제거 가능
billb_tidy <- billboard %>%
  pivot_longer(starts_with("wk"),
               names_to = "week",
               values_to = "rank",
               values_drop_na = TRUE)
billb_tidy %>% print
```

```
# A tibble: 5,307 x 5
  artist   track date.entered week   rank
  <chr>   <chr>    <date>     <chr> <dbl>
```

```

<chr> <chr>           <date> <chr> <dbl>
1 2 Pac Baby Don't Cry (Keep... 2000-02-26 wk1    87
2 2 Pac Baby Don't Cry (Keep... 2000-02-26 wk2    82
3 2 Pac Baby Don't Cry (Keep... 2000-02-26 wk3    72
4 2 Pac Baby Don't Cry (Keep... 2000-02-26 wk4    77
5 2 Pac Baby Don't Cry (Keep... 2000-02-26 wk5    87
6 2 Pac Baby Don't Cry (Keep... 2000-02-26 wk6    94
7 2 Pac Baby Don't Cry (Keep... 2000-02-26 wk7    99
8 2Ge+her The Hardest Part Of ... 2000-09-02 wk1    91
9 2Ge+her The Hardest Part Of ... 2000-09-02 wk2    87
10 2Ge+her The Hardest Part Of ... 2000-09-02 wk3   92
# ... with 5,297 more rows

```

```

# pivot_longer() 함수의 인수 중 names_prefix 인수 값 설정을 통해
# 변수명에 처음에 붙는 접두어(예: V, wk 등) 제거 가능
billb_tidy <- billboard %>%
  pivot_longer(starts_with("wk"),
               names_to = "week",
               values_to = "rank",
               names_prefix = "wk",
               values_drop_na = TRUE)
billb_tidy %>% print

```

```

# A tibble: 5,307 x 5
  artist track      date.entered week rank
  <chr> <chr>       <date>     <chr> <dbl>
1 2 Pac Baby Don't Cry (Keep... 2000-02-26 1    87
2 2 Pac Baby Don't Cry (Keep... 2000-02-26 2    82
3 2 Pac Baby Don't Cry (Keep... 2000-02-26 3    72
4 2 Pac Baby Don't Cry (Keep... 2000-02-26 4    77
5 2 Pac Baby Don't Cry (Keep... 2000-02-26 5    87
6 2 Pac Baby Don't Cry (Keep... 2000-02-26 6    94
7 2 Pac Baby Don't Cry (Keep... 2000-02-26 7    99

```

```
8 2Ge+her The Hardest Part Of ... 2000-09-02 1 91
9 2Ge+her The Hardest Part Of ... 2000-09-02 2 87
10 2Ge+her The Hardest Part Of ... 2000-09-02 3 92
# ... with 5,297 more rows
```

```
# pivot_longer() 함수의 인수 중 names_ptypes 또는 values_ptypes 인수 값 설정을 통해
```

```
# 새로 생성한 변수 (name과 value에 해당하는)의 데이터 타입 변경 가능
```

```
billb_tidy <- billboard %>%
  pivot_longer(starts_with("wk"),
               names_to = "week",
               values_to = "rank",
               names_prefix = "wk",
               names_ptypes = list(week = integer()),
               values_drop_na = TRUE)
billb_tidy %>% print
```

```
# A tibble: 5,307 x 5
  artist   track           date.entered week   rank
  <chr>   <chr>          <date>        <int> <dbl>
1 2 Pac   Baby Don't Cry (Keep... 2000-02-26 1     87
2 2 Pac   Baby Don't Cry (Keep... 2000-02-26 2     82
3 2 Pac   Baby Don't Cry (Keep... 2000-02-26 3     72
4 2 Pac   Baby Don't Cry (Keep... 2000-02-26 4     77
5 2 Pac   Baby Don't Cry (Keep... 2000-02-26 5     87
6 2 Pac   Baby Don't Cry (Keep... 2000-02-26 6     94
7 2 Pac   Baby Don't Cry (Keep... 2000-02-26 7     99
8 2Ge+her The Hardest Part Of ... 2000-09-02 1     91
9 2Ge+her The Hardest Part Of ... 2000-09-02 2     87
10 2Ge+her The Hardest Part Of ... 2000-09-02 3     92
# ... with 5,297 more rows
```

```
# 연습: wide_01 데이터에서 year을 정수형으로 변환 (mutate 함수 사용하지 않고)
```

2. 두 개 이상의 변수에 해당하는 값이 하나의 열에 저장

- 예시 데이터: `tidyverse` 패키지에 내장되어 있는 `who` 데이터셋
(`help(who)`를 통해 데이터 설명 참고)

```
# A tibble: 7,240 x 60
  country iso2 iso3    year new_sp_m014 new_sp_m1524 new_sp_m2534 new_sp_m3544
  <chr>   <chr> <chr> <int>       <int>       <int>       <int>       <int>
1 Afghan~ AF    AFG    1980      NA        NA        NA        NA
2 Afghan~ AF    AFG    1981      NA        NA        NA        NA
3 Afghan~ AF    AFG    1982      NA        NA        NA        NA
4 Afghan~ AF    AFG    1983      NA        NA        NA        NA
5 Afghan~ AF    AFG    1984      NA        NA        NA        NA
6 Afghan~ AF    AFG    1985      NA        NA        NA        NA
7 Afghan~ AF    AFG    1986      NA        NA        NA        NA
8 Afghan~ AF    AFG    1987      NA        NA        NA        NA
9 Afghan~ AF    AFG    1988      NA        NA        NA        NA
10 Afghan~ AF   AFG    1989      NA        NA        NA        NA
# ... with 7,230 more rows, and 52 more variables: new_sp_m4554 <int>,
#   new_sp_m5564 <int>, new_sp_m65 <int>, new_sp_f014 <int>,
#   new_sp_f1524 <int>, new_sp_f2534 <int>, new_sp_f3544 <int>,
#   new_sp_f4554 <int>, new_sp_f5564 <int>, new_sp_f65 <int>,
#   new_sn_m014 <int>, new_sn_m1524 <int>, new_sn_m2534 <int>,
#   new_sn_m3544 <int>, new_sn_m4554 <int>, new_sn_m5564 <int>,
#   new_sn_m65 <int>, new_sn_f014 <int>, new_sn_f1524 <int>,
#   new_sn_f2534 <int>, new_sn_f3544 <int>, new_sn_f4554 <int>,
#   new_sn_f5564 <int>, new_sn_f65 <int>, new_ep_m014 <int>,
#   new_ep_m1524 <int>, new_ep_m2534 <int>, new_ep_m3544 <int>,
#   new_ep_m4554 <int>, new_ep_m5564 <int>, new_ep_m65 <int>,
#   new_ep_f014 <int>, new_ep_f1524 <int>, new_ep_f2534 <int>,
#   new_ep_f3544 <int>, new_ep_f4554 <int>, new_ep_f5564 <int>,
#   new_ep_f65 <int>, newrel_m014 <int>, newrel_m1524 <int>,
#   newrel_m2534 <int>, newrel_m3544 <int>, newrel_m4554 <int>,
#   newrel_m5564 <int>, newrel_m65 <int>, newrel_f014 <int>,
```

```
# newrel_f1524 <int>, newrel_f2534 <int>, newrel_f3544 <int>,
# newrel_f4554 <int>, newrel_f5564 <int>, newrel_f65 <int>
```

TABLE 4.12: *tidyR* 패키지 내장 데이터 *who* 코드 설명

| 변수명 | 변수설명 |
|-----------------------------|---|
| country | 국가명 |
| iso2, iso3 | 2자리 또는 3자리 국가코드 |
| year | 년도 |
| new_sp_m014 - newrel_f65 | 변수 접두사: new_ 또는 new; 진단명: sp = positive pulmonary smear, sn = negative pulmonary smear, ep = extrapulmonary, rel = relapse; 성별: m = male, f = female; 연령대: 014 = 0-14 yrs, 1524 = 14-24 yrs, 2534 = 25-34 yrs, 3544 = 35-44 yrs, 4554 = 45-54 yrs, 5564 = 55-64 yrs, 65 = 65 yrs or older |

- 데이터 정돈 전략(*pivot_longer()* 이용)
 - country, iso2, iso3, year*은 정상적인 변수 형태임 → 그대로 둔다
 - names_to* 인수에 *diagnosis, sex, age_group*로 변수명을 지정
 - names_prefix* 인수에서 접두어 제거
 - names_pattern* 인수에서 추출한 변수의 패턴을 정규표현식을 이용해 표현(각 변수는 ()으로 구분) → _를 기준으로 왼쪽에는 (소문자 알파벳이 하나 이상 존재하고), 오른쪽에는 (m 또는 f)와 (숫자가 1개 이상)인 패턴
 - names_ptypes* 인수를 이용해 생성한 변수의 데이터 타입 지정

- diagnosis: factor
- sex: factor
- age_group: factor

6. `values_to` 인수에 longer 형태로 변환 후 생성된 값을 저장한 열(변수) 이름 `count` 지정

7. `values_drop_na` 인수를 이용해 결측 제거

```
# pivot_longer()를 이용해 who 데이터셋 데이터 정돈
who_tidy <- who %>%
  pivot_longer(
    new_sp_m014:newrel_f65,
    names_to = c("diagnosis", "sex", "age_group"),
    names_prefix = "^new_?",
    names_pattern = "[a-z]+_(m|f)([0-9]+)",
    names_ptypes = list(
      diagnosis = factor(levels = c("rel", "sn", "sp", "ep")),
      sex = factor(levels = c("f", "m")),
      age_group = factor(levels = c("014", "1524", "2534", "3544",
                                    "4554", "5564", "65")),
      ordered = TRUE)
  ),
  values_to = "count",
  values_drop_na = TRUE
)

who_tidy %>% print

# A tibble: 76,046 x 8
  country   iso2   iso3   year diagnosis sex   age_group count
  <chr>     <chr>  <chr>  <int> <fct>    <fct> <ord>    <int>
  1 Afghanistan AF    AFG    1997 sp       m     014        0
  2 Afghanistan AF    AFG    1997 sp       m     1524       10
```

```

3 Afghanistan AF    AFG    1997 sp      m    2534    6
4 Afghanistan AF    AFG    1997 sp      m    3544    3
5 Afghanistan AF    AFG    1997 sp      m    4554    5
6 Afghanistan AF    AFG    1997 sp      m    5564    2
7 Afghanistan AF    AFG    1997 sp      m     65    0
8 Afghanistan AF    AFG    1997 sp      f     014    5
9 Afghanistan AF    AFG    1997 sp      f    1524   38
10 Afghanistan AF    AFG   1997 sp      f    2534   36
# ... with 76,036 more rows

```

4.5.3 Wide format

- long format의 반대가 되는 데이터 형태
- 관측 단위의 측정값(예: 다수 변수들)이 다중 행으로 구성된 경우 tidy data를 만들기 위해 wide format으로 데이터 변환 요구
- 요약표 생성 시 유용하게 사용
- “long” 형태의 데이터를 “wide” 형태로 변환 해주는 tidyverse 패키지 내장 함수
 - `pivot_wider()`: 데이터의 행을 줄이고 열의 개수를 늘리는 함수
 - `spread()`: `pivot_wider()`의 이전 버전

```

# pivot_wider()의 기본 사용 형태
pivot_wider(
  data, # 데이터 프레임
  names_from, # 출력 시 변수명으로 사용할 값을 갖고 있는 열 이름
  values_from, # 위에서 선택한 변수의 각 셀에 대응하는 측정 값을 포함하는 열 이름
  values_fill # 
)

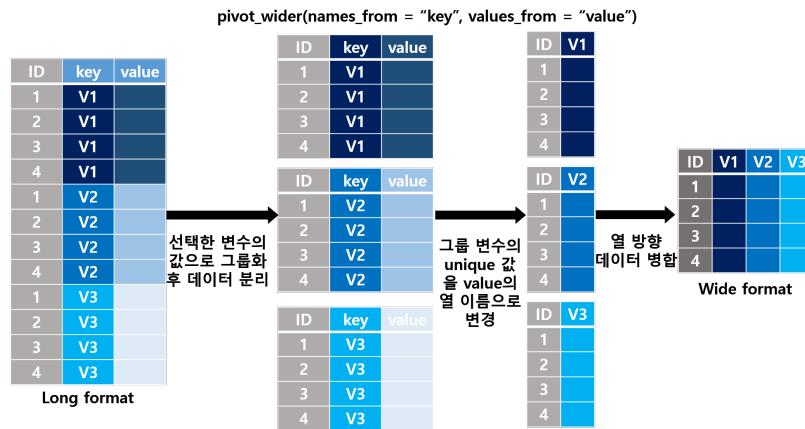
# spread() 기본 사용 형태
spread(

```

```

data, # 데이터 프레임
key, # 출력 시 변수명으로 사용할 값을 갖고 있는 열 이름
value # 위에서 선택한 변수의 각 셀에 대응하는 측정 값을 포함하는 열 이름
)

```



Examples

1. pivot_longer()와의 관계

```

# 위 예시에서 생성한 tidy_ex_01 데이터 예시
## long format으로 변환한 데이터를 다시 wide format으로 변경
## pivot_wider() 함수

wide_ex_01 <- tidy_ex_01 %>%
  pivot_wider(
    names_from = year,
    values_from = gdp_cap
  )
wide_ex_01 %>% print

# A tibble: 3 x 21
country `2001` `2002` `2003` `2004` `2005` `2006` `2007` `2008` `2009` `2010` ...

```

```
<chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 Germany 37325. 37262. 36977. 37418. 37704. 39143. 40474. 40989. 38784. 40429.
2 South ~ 21530. 22997. 23549. 24606. 25517. 26697. 28014. 28588. 28643. 30352.
3 United~ 45663. 46029. 46941. 48275. 49513. 50438. 50898. 50350. 48644. 49479.
# ... with 10 more variables: `2011` <dbl>, `2012` <dbl>, `2013` <dbl>,
#   `2014` <dbl>, `2015` <dbl>, `2016` <dbl>, `2017` <dbl>, `2018` <dbl>,
#   `2019` <dbl>, `2020` <dbl>
```

```
## 데이터 동일성 확인  
all.equal(wide_01, wide_ex_01)
```

```
[1] TRUE
```

```
## spread() 함수  
wide_ex_01 <- tidy_ex_01 %>%  
  spread(year, gdp_cap)  
  
all.equal(wide_01, wide_ex_01)
```

```
[1] TRUE
```

2. 관측 단위의 측정값(예: 다수 변수들)이 다중 행으로 구성된 데이터 구조

변환

- 예시 데이터: **tidyR** 패키지 **table2** 데이터셋

```
# table2 데이터셋 check  
table2 %>% print  
  
# A tibble: 12 x 4  
  country     year type       count  
  <chr>      <int> <chr>      <int>  
1 Afghanistan 1999 cases        745  
2 Afghanistan 1999 population 19987071
```

```

3 Afghanistan 2000 cases           2666
4 Afghanistan 2000 population    20595360
5 Brazil       1999 cases          37737
6 Brazil       1999 population    172006362
7 Brazil       2000 cases          80488
8 Brazil       2000 population   174504898
9 China        1999 cases          212258
10 China       1999 population  1272915272
11 China       2000 cases          213766
12 China       2000 population  1280428583

```

```

# type 변수의 값은 사실 관측 단위의 변수임
# type 값에 대응하는 값을 가진 변수는 count 임
## 데이터 정돈(pivot_wider() 사용)

```

```



```

```

# A tibble: 6 x 4
  country      year  cases population
  <chr>     <int> <int>     <int>
1 Afghanistan  1999    745  19987071
2 Afghanistan  2000   2666  20595360
3 Brazil       1999  37737  172006362
4 Brazil       2000  80488  174504898
5 China        1999  212258 1272915272
6 China       2000  213766 1280428583

```

3. 데이터 요약 테이블

- 예시 데이터: mtcars 데이터셋

```
# 1) mtcars 데이터셋: 행 이름을 변수로 변환 후 long format 변환
## rownames_to_column() 함수 사용
mtcars2 <- mtcars %>%
  rownames_to_column(var = "model") %>%
  pivot_longer(
    -c("model", "vs", "am"),
    names_to = "variable",
    values_to = "value"
  )

# 2) 엔진 유형 별 variable의 평균과 표준편차 계산
# "사람"이 읽기 편한 형태로 테이블 변경
mtcars2 %>%
  mutate(vs = factor(vs,
                     labels = c("V-shaped",
                               "Straight")))) %>%
  group_by(vs, variable) %>%
  summarise(Mean = mean(value),
            SD = sd(value)) %>%
  pivot_longer(
    Mean:SD,
    names_to = "stat",
    values_to = "value"
  ) %>%
  pivot_wider(
    names_from = variable,
    values_from = value
  )

# A tibble: 4 x 11
# Groups:   vs [2]
```

```

vs      stat   carb   cyl   disp   drat   gear   hp     mpg   qsec   wt
<fct>    <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 V-shaped Mean  3.61 7.44 307.  3.39  3.56  190.  16.6  16.7  3.69
2 V-shaped SD    1.54 1.15 107.  0.474 0.856 60.3  3.86  1.09  0.904
3 Straight Mean  1.79 4.57 132.  3.86  3.86  91.4  24.6  19.3  2.61
4 Straight SD    1.05 0.938 56.9  0.506 0.535 24.4  5.38  1.35  0.715

```

```

# 조금 더 응용...
## 위 Mean ± SD 형태로 위와 유사한 구조의 테이블 생성
#### tip: 한글로 "ㄷ(e) + 한자" 통해 ± 입력 가능

```

```

mtcars2 %>%
  mutate(vs = factor(vs,
                     labels = c("V-shaped",
                               "Straight")))) %>%
  group_by(vs, variable) %>%
  summarise(Mean = mean(value),
            SD = sd(value)) %>%
  mutate(res = sprintf("%.1f ± %.1f",
                      Mean, SD)) %>%
  select(-(Mean:SD)) %>%
  pivot_wider(
    names_from = variable,
    values_from = res
  )

```

```

# A tibble: 2 x 10
# Groups:   vs [2]
  vs      carb   cyl   disp   drat   gear   hp     mpg   qsec   wt
  <fct>    <chr> <chr> <chr>    <chr> <chr> <chr> <chr> <chr> <chr>
1 V-shap~ 3.6 ± ~ 7.4 ± ~ 307.1 ± ~ 3.4 ± ~ 3.6 ± ~ 189.7 ~ 16.6 ~ 16.7 ~ 3.7 ±~
2 Straig~ 1.8 ± ~ 4.6 ± ~ 132.5 ± ~ 3.9 ± ~ 3.9 ± ~ 91.4 ±~ 24.6 ~ 19.3 ~ 2.6 ±~

```

4.5.4 Separate and unite

- 하나의 열을 구성하는 값이 두 개 이상 변수가 혼합되어 한 셀에 표현된 경우 이를 분리해야 할 필요가 있음 → **separate()**
- 하나의 변수에 대한 값으로 표현할 수 있음에도 불구하고 두 개 이상의 변수로 구성된 경우(예: 날짜 변수의 경우 간혹 year, month, day와 같이 3 개의 변수로 분리), 이를 연결하여 하나의 변수로 변경 필요 → **unite()**

Separate

- separate()**: 지정한 구분 문자가 존재하는 경우 이를 쪼개서 하나의 열을 다수의 열로 분리

```
# separate() 함수 기본 사용 형태
separate(
  data, # 데이터 프레임
  col, # 분리 대상이 되는 열 이름
  into, # 분리 후 새로 생성한 열들에 대한 이름(문자형 벡터) 지정
  sep = "[[:alnum:]]+", # 구분자: 기본적으로 정규표현식 사용
  convert # 분리한 열의 데이터 타입 변환 여부
)
```

- 예시: tidyverse 패키지 table3 데이터셋

```
# table3 데이터 체크
table3 %>% print

# A tibble: 6 x 3
  country     year   rate
  <chr>      <int> <chr>
1 Afghanistan 1999 745/19987071
2 Afghanistan 2000 2666/20595360
3 Brazil      1999 37737/172006362
```

```

4 Brazil      2000 80488/174504898
5 China       1999 212258/1272915272
6 China       2000 213766/1280428583

```

```

# rate 변수를 case와 population으로 분리
table3 %>%
  separate(rate,
    into = c("case", "population"),
    sep = "/") %>%
  print

```

```

# A tibble: 6 x 4
  country     year case  population
  <chr>      <int> <chr>   <chr>
1 Afghanistan 1999  745    19987071
2 Afghanistan 2000  2666   20595360
3 Brazil      1999  37737  172006362
4 Brazil      2000  80488  174504898
5 China       1999  212258 1272915272
6 China       2000  213766 1280428583

```

```

table3 %>%
  separate(rate,
    into = c("case", "population"),
    convert = TRUE) -> table3_sep

## sep 인수값이 수치형 벡터인 경우 분리할 위치로 인식
## 양수: 문자열 맨 왼쪽에서 1부터 시작
## 음수: 문자열 맨 오른쪽에서 -1부터 시작
## sep의 길이 (length)는 into 인수의 길이보다 작아야 함

# year 변수를 century와 year로 분할
table3 %>%

```

```

separate(year,
         into = c("century", "year"),
         sep = -2) %>%
print

# A tibble: 6 x 4
  country   century year   rate
  <chr>     <chr>    <chr> <chr>
1 Afghanistan 19      99     745/19987071
2 Afghanistan 20      00     2666/20595360
3 Brazil       19      99     37737/172006362
4 Brazil       20      00     80488/174504898
5 China        19      99     212258/1272915272
6 China        20      00     213766/1280428583

```

Unite

- `unite()`: `seprate()` 함수의 반대 기능을 수행하며, 다수 변수를 결합

```

# unite() 기본 사용 형태
unite(
  data, # 데이터프레임
  ..., # 선택한 열 이름
  sep, # 연결 구분자
)

```

- 예제: tidyverse 패키지 `table5` 데이터셋

```

# table5 체크
table5 %>% print

# A tibble: 6 x 4
  country   century year   rate
  <chr>     <chr>    <chr> <chr>
1 Afghanistan 19      99     745/19987071
2 Afghanistan 20      00     2666/20595360
3 Brazil       19      99     37737/172006362
4 Brazil       20      00     80488/174504898
5 China        19      99     212258/1272915272
6 China        20      00     213766/1280428583

```

```
* <chr>      <chr>      <chr> <chr>
1 Afghanistan 19      99      745/19987071
2 Afghanistan 20      00      2666/20595360
3 Brazil       19      99      37737/172006362
4 Brazil       20      00      80488/174504898
5 China        19      99      212258/1272915272
6 China        20      00      213766/1280428583
```

```
# century와 year을 결합한 new 변수 생성
```

```
table5 %>%
  unite(new, century, year) %>%
  print
```

```
# A tibble: 6 x 3
  country     new    rate
  <chr>      <chr> <chr>
1 Afghanistan 19_99 745/19987071
2 Afghanistan 20_00 2666/20595360
3 Brazil       19_99 37737/172006362
4 Brazil       20_00 80488/174504898
5 China        19_99 212258/1272915272
6 China        20_00 213766/1280428583
```

```
# _없이 결합 후 new를 정수형으로 변환
```

```
table5 %>%
  unite(new, century, year, sep = "") %>%
  mutate(new = as.integer(new)) %>%
  print
```

```
# A tibble: 6 x 3
  country     new    rate
  <chr>      <int> <chr>
1 Afghanistan 1999 745/19987071
```

```

2 Afghanistan 2000 2666/20595360
3 Brazil      1999 37737/172006362
4 Brazil      2000 80488/174504898
5 China       1999 212258/1272915272
6 China       2000 213766/1280428583

```

```

# table5 데이터 정돈(separate(), unite() 동시 사용)
table5 %>%
  unite(new, century, year, sep = "") %>%
  mutate(new = as.integer(new)) %>%
  separate(rate, c("case", "population"),
            convert = TRUE) %>%
  print

```

```

# A tibble: 6 x 4
  country     new    case population
  <chr>     <int>  <int>      <int>
1 Afghanistan 1999     745 19987071
2 Afghanistan 2000    2666 20595360
3 Brazil      1999   37737 172006362
4 Brazil      2000   80488 174504898
5 China       1999  212258 1272915272
6 China       2000  213766 1280428583

```

- 응용 예제: mtcars 데이터셋에서 계산한 통계량 정리

```

# 기어 종류(`am`) 별 `mpg`, `cyl`, `disp`, `hp`, `drat`, `wt`, `qsec`의
# 평균과 표준편차 계산
mtcar_summ1 <- mtcars %>%
  mutate(am = factor(am,
                     labels = c("automatic", "manual"))) %>%
  group_by(am) %>%
  summarise_at(vars(mpg:qsec),

```

```

        list(mean = ~ mean(.),
             sd = ~ sd(.)))
mtcar_summ1 %>% print

# A tibble: 2 x 15
  am     mpg_mean cyl_mean disp_mean hp_mean drat_mean wt_mean qsec_mean mpg_sd
  <fct>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
1 auto~     17.1     6.95    290.     160.     3.29     3.77     18.2     3.83
2 manu~     24.4     5.08    144.     127.     4.05     2.41     17.4     6.17
# ... with 6 more variables: cyl_sd <dbl>, disp_sd <dbl>, hp_sd <dbl>,
#   drat_sd <dbl>, wt_sd <dbl>, qsec_sd <dbl>

# am을 제외한 모든 변수에 대해 long format으로 데이터 변환
mtcar_summ2 <- mtcar_summ1 %>%
  pivot_longer(
    -am,
    names_to = "stat",
    values_to = "value"
  )
mtcar_summ2 %>% print

# A tibble: 28 x 3
  am      stat      value
  <fct>    <chr>    <dbl>
1 automatic mpg_mean 17.1
2 automatic cyl_mean  6.95
3 automatic disp_mean 290.
4 automatic hp_mean  160.
5 automatic drat_mean 3.29
6 automatic wt_mean  3.77
7 automatic qsec_mean 18.2
8 automatic mpg_sd   3.83
9 automatic cyl_sd   1.54

```

```

10 automatic disp_sd    110.
# ... with 18 more rows

# stat 변수를 "variable", "statistic"으로 분리 후
# variable과 value를 wide format으로 데이터 변환
mtcar_summ3 <- mtcar_summ2 %>%
  separate(stat, c("variable", "statistic")) %>%
  pivot_wider(
    names_from = variable,
    values_from = value
  )
mtcar_summ3 %>% print

# A tibble: 4 x 9
  am      statistic   mpg   cyl  disp    hp  drat    wt  qsec
  <fct>    <chr>     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 automatic mean     17.1   6.95 290.   160.   3.29   3.77   18.2
2 automatic sd       3.83   1.54 110.   53.9  0.392  0.777  1.75
3 manual   mean     24.4   5.08 144.   127.   4.05   2.41   17.4
4 manual   sd        6.17   1.55  87.2  84.1  0.364  0.617  1.79

```

Tidy data를 만들기 위한 과정이 꼭 필요할까? → long format 데이터가
정말 필요할까?

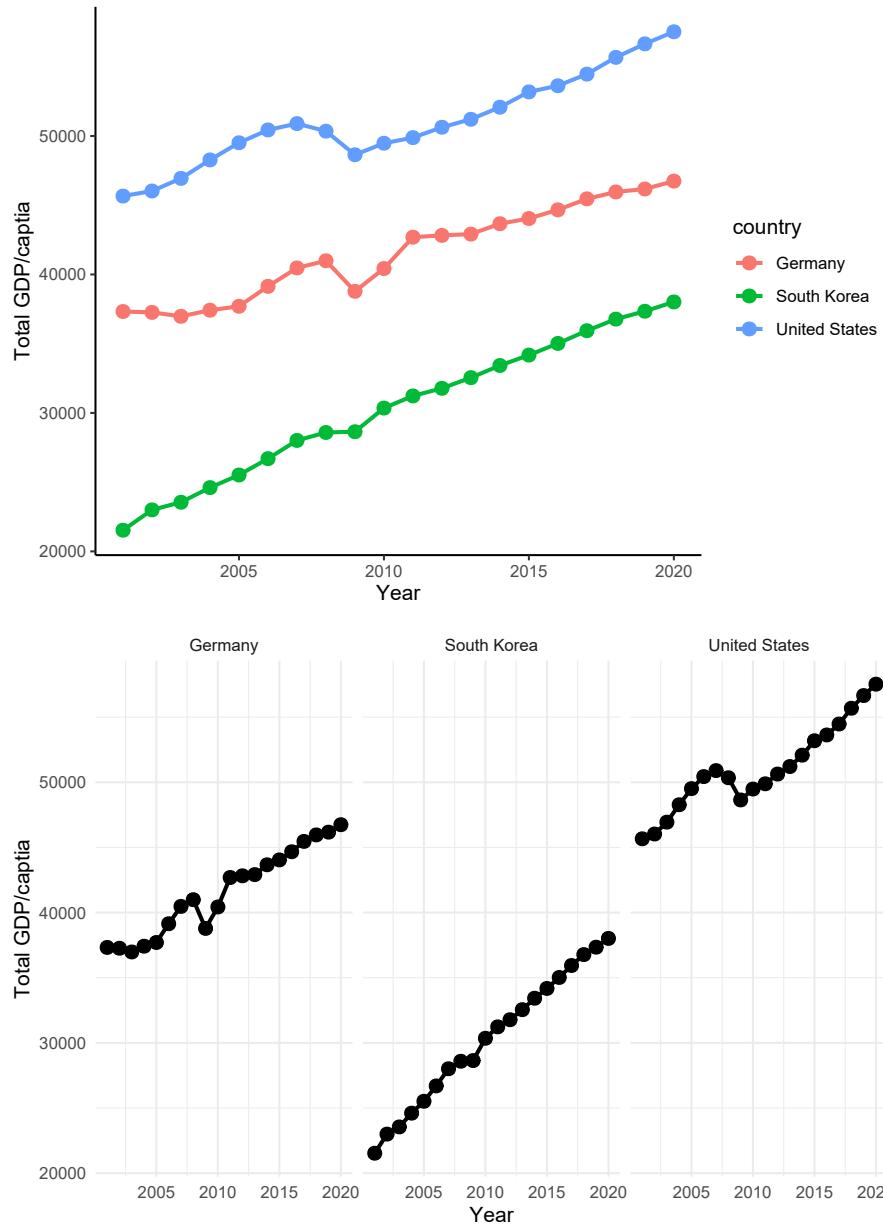
ggplot trailer: 4.5.2 절 Long format에서 예시 데이터로 활용한 wide-01
데이터셋을 이용해 국가별 연도에 따른 일인당 국민소득 추이를 시각화

Strategy

- wide-01 데이터 형태 그대로 시각화
- wide-01을 long format으로 변환한 tidy_ex_01에서 시각화

```
# ggplot trailer
tidy_ex_01 <- wide_01 %>%
  pivot_longer(~2001:`2020`,
  names_to = "year",
  values_to = "gdp_cap",
  names_ptypes = list(
    year = integer()
  ))
tidy_ex_01 %>%
  ggplot +
  aes(x = year, y = gdp_cap,
  color = country,
  group = country) +
  geom_point(size = 3) +
  geom_line(size = 1) +
  labs(x = "Year",
  y = "Total GDP/capita") +
  theme_classic()

tidy_ex_01 %>%
  ggplot +
  aes(x = year, y = gdp_cap,
  group = country) +
  geom_point(size = 3) +
  geom_line(size = 1) +
  labs(x = "Year",
  y = "Total GDP/capita") +
  facet_grid(~ country) +
  theme_minimal()
```



4.6 Homework #4

과제 제출 방식



- R Markdown 문서 (`Rmd`) 파일과 해당 문서를 컴파일 후 생성된 `html` 파일 모두 제출할 것
- 모든 문제에 대해 작성한 R 코드 및 결과가 `html` 문서에 포함되어야 함.
- 해당 과제에 대한 R Markdown 문서 템플릿은 https://github.com/zorba78/cnu-r-programming-lecture-note/blob/master/assignment/homework4_template.Rmd에서 다운로드 또는 `Ctrl + C, Ctrl + V` 가능
- 최종 파일명은 `학번-성명.Rmd`, `학번-성명.html`로 저장
- 압축파일은 `*.zip` 형태로 생성할 것

주의 사항

- 과제에 필요한 텍스트 데이터 파일은 가급적 제출 파일(`rmd` 및 `html` 파일)이 생성되는 폴더 안에 폴더를 만든 후 텍스트 파일을 저장할 것. 예를 들어 `homework4.Rmd` 파일이 `C:/my-project`에서 생성된다면 `C:/my-project/exercise` 폴더 안에 텍스트 파일 저장
- 만약 `Rmd` 파일이 작업 디렉토리 내 별도의 폴더 (예: `C:/my-project/rmd`)에 저장되어 있고 텍스트 데이터 파일이 `C:/my-project/exercise`에 존재한다면, 다음과 같은 chunk 가 `Rmd` 파일 맨 처음에 실행되어야 함.

```
```{r, eval=FALSE}
knitr::opts_knit$set(root.dir = '..')
````
```

1. 사이버 캠퍼스 자료실에 업로드된 `exercise.zip` 파일을 다운로드 후 `exercise` 폴더에 압축을 풀면 총 20개의 텍스트 파일이 저장되어 있다. 해당 파일들은 휴면상태 뇌파(resting state EEG) 신호로부터 추출한 특징 (feature)이다. 폴더에 포함된 텍스트 파일의 이름은 기기명 (`h7n1`), EEG 변

수 특징 (`beam_results`), 파일번호 (예: 009)로 구성되어 있고 `_`로 연결되어 있다.

- a. 저장된 텍스트 파일 중 하나를 열어보고 해당 텍스트 파일이 저장하고 있는 데이터의 구조에 대해 설명하고, 열과 열을 구분하기 위해 어떠한 구분자 (separator)가 사용되었는지 기술하시오.

1-a 답: 입력(입력 시 해당 문구 삭제)

- b. 다운로드한 텍스트 파일이 저장된 폴더 경로를 `path`라는 객체에 저장하고, `dir()` 함수를 이용해 해당 폴더에 저장되어 있는 파일의 이름 모두를 `filename`이라는 객체에 저장 하시오. (참고: `dir()` 함수는 인수로 받은 폴더 경로 내 존재하는 모든 파일의 이름 및 확장자를 문자형 벡터로 반환해 주는 함수임. 자세한 사용법은 `help(dir)`을 통해 확인)

```
# path <- "텍스트 파일이 저장된 폴더명"  
# filename <- dir(path)
```

- c. `filename`에서 기기명 부분만 추출 후, `file_dev` 객체에 저장 하시오.

- d. 정규표현식을 이용하여 `filename`에서 기기명에 해당하는 부분을 삭제 후 `file_id` 객체에 저장 하시오 (hint: `gsub()` 함수를 사용할 수 있으마, `file_id`에 저장되어 있는 문자열 원소 모두는 `beam_results_009.txt`와 같은 형태로 반환되어야 함).

- e. 정규표현식을 사용하여 위에서 생성한 `file_id`에서 숫자만 추출 후 `id_tmp`라는 객체를 생성 하시오. 그리고 ID 문자열과 `file_id`에 저장되어 있는 문자열과 결합해 모든 원소가 ID009와 같은 형태의 원소값을 갖는 ID 객체를 생성하시오

- f. `paste()` 또는 `paste0()` 함수를 활용해 1-a. 에서 생성한 `path`라는 객체와 `filename`을 이용해 파일경로/파일명 형태의 문자형 벡터를 `full_filename` 객체에 저장하시오.
- g. 1-f.에서 만든 `full_filename`, `lapply()`와 `read.table()` 함수를 활용하여 폴더에 저장되어 있는 모든 텍스트 파일을 리스트 형태로 저장한 `dat1` 객체를 생성 하시오.
- h. 1-g.에서 생성한 `dat1`에 저장되어 있는 20개의 데이터 프레임을 하나의 데이터 프레임으로 묶은 결과를 저장한 `dat` 객체를 생성 하시오.
- i. 1-c. 와 1-d.에서 생성한 `ID`와 `file_dev`를 이용해 두 개의 변수로 구성된 `id_info`라는 데이터 프레임을 생성 하시오. 단 두 문자형 벡터의 각 원소는 3 번씩 반복되어야 하고, 각 변수는 모두 문자형으로 저장되어야 함.
- j. 1-i.에서 생성한 데이터 프레임 `id_info` 와 1.h에서 생성한 `dat`을 하나의 데이터 프레임으로 묶은 `dat_fin`이라는 객체를 생성 하시오.
- k. 사이버 캠퍼스 자료실에 업로드된 `beam-crf-ex.rds`를 다운로드 한 후 R 작업공간에 불러온 결과를 `beam_crf` 객체에 저장하시오.
- l. tidyverse 패키지를 R 작업공간으로 읽은 후 dplyr에서 제공하는 함수를 이용해 1-k.에서 생성한 `beam_crf`의 변수 `eeg_filenam` 문자열 중 처음 5개 문자(예: ID158)만 추출한 `eeg_id`라는 변수를 `beam_crf` 데이터 프레임 내에 새로운 변수로 만드시오.
- m. 두 데이터 프레임 `beam_crf`와 `dat_fin`은 연결할 수 있는가? 연결할 수 있다면 그 이유를 설명 하시오.

1-m 답:

- n. 만약 연결할 수 있다면 `beam_crf`와 `dat_fin`을 join 하여 두 데이터 프레임에 공통으로 포함된 행으로 구성된 데이터 프레임 `beam_sub` 객체를 생성 하시오.
- o. 1.n. 에서 생성한 `beam_sub`에 대해 `dplyr`에서 제공하는 함수를 이용해 아래 기술한 내용을 수행 하시오. 단 각 단계는 파이프 연산자 (`%>%`)로 연결 하시오.
1. `usubjid`, `sex`, `age`, `literacy`, `Row`, `MDF`, `PF`, `ATR` 변수를 선택한 다음
 2. 변수 `sex`, `literacy`, `Row`를 요인형 (factor)으로 변환하고,
 3. 변수 `age`를 `floor()` 함수를 이용해 소수점 내림한 결과가 저장된 `beam_sub2` 객체를 생성 하시오.
- p. `beam_sub2`를 이용해 아래 기술한 결과를 반환하는 스크립트를 작성 후 확인 하시오.
1. `Row` 수준별 `MDF`, `PF`, `ATR`의 평균(`mean()`), 표준편차(`sd()`), 최솟값 (`min()`), 중앙값(`median()`), 최댓값(`max()`)을 출력 하시오(`dplyr` 패키지 함수 이용).
 2. `literacy`는 조사에 참여한 대상자가 문자식별(문자를 읽고 쓸수 있는지)에 대한 정보를 담고 있는 변수이다. 문자식별 변수의 수준 별 케이스 수와 `age`, `MDF`, `PF`, `ATR`의 평균 결과를 출력 하시오(`dplyr` 패키지 함수 이용).
 3. 1.p.1 과 1.p.2 와 동일한 결과를 출력하는 스크립트를 R 기본 문법을 이용해 작성해 본 후 두 방법(`dplyr` 문법 vs. R 기본 문법)에 대해 비교해 보시오.

```
# 1.
```

2.

3.

1-p-3 비교 서술:

5

데이터 시각화

학습 목표

- R에서 기본으로 제공하는 그래프 생성 개념 및 관련 함수의 의미 및 사용 방법에 대해 학습한다.
- Grammar of graphics를 기반으로 개발된 ggplot2 패키지에 대해 알아보고 사용 방법을 학습한다.

“The simple graph has brought more information to the data analyst’s mind than any other device.”

→ John Tukey

- 그래프는 생각보다 더 많은 정보를 제공
- 데이터 분석 시 통계량 만으로 데이터의 속성을 결정하는 것은 매우 위험
(예: Anscombe’s quartet 데이터 예제)

| x1 | x2 | x3 | x4 | y1 | y2 | y3 | y4 | |
|----|----|----|----|----|------|------|-------|------|
| 1 | 10 | 10 | 10 | 8 | 8.04 | 9.14 | 7.46 | 6.58 |
| 2 | 8 | 8 | 8 | 8 | 6.95 | 8.14 | 6.77 | 5.76 |
| 3 | 13 | 13 | 13 | 8 | 7.58 | 8.74 | 12.74 | 7.71 |
| 4 | 9 | 9 | 9 | 8 | 8.81 | 8.77 | 7.11 | 8.84 |
| 5 | 11 | 11 | 11 | 8 | 8.33 | 9.26 | 7.81 | 8.47 |
| 6 | 14 | 14 | 14 | 8 | 9.96 | 8.10 | 8.84 | 7.04 |

| x1 | x2 | x3 | x4 | y1 | y2 | y3 | y4 |
|----------|----------|----------|----------|----------|----------|----------|----------|
| 9.000000 | 9.000000 | 9.000000 | 9.000000 | 7.500909 | 7.500909 | 7.500000 | 7.500909 |
| 3.316625 | 3.316625 | 3.316625 | 3.316625 | 2.031568 | 2.031657 | 2.030424 | 2.030579 |

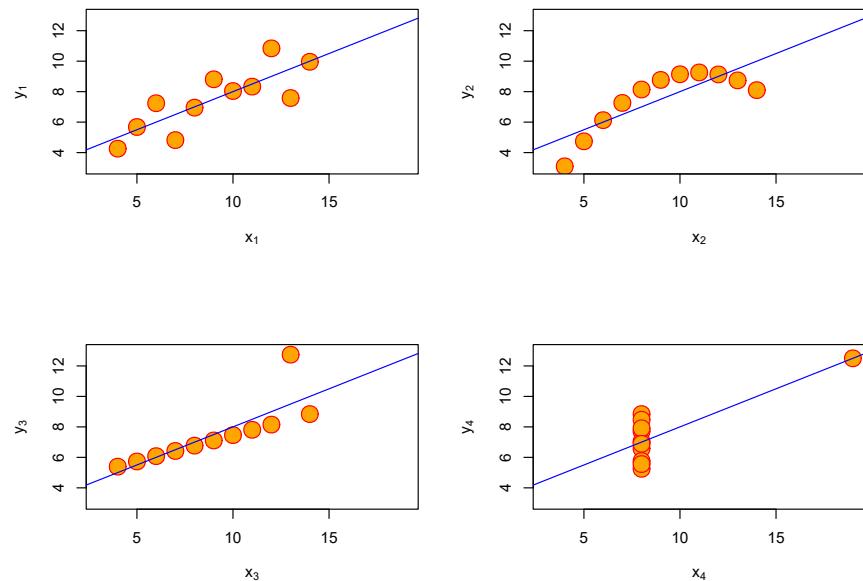
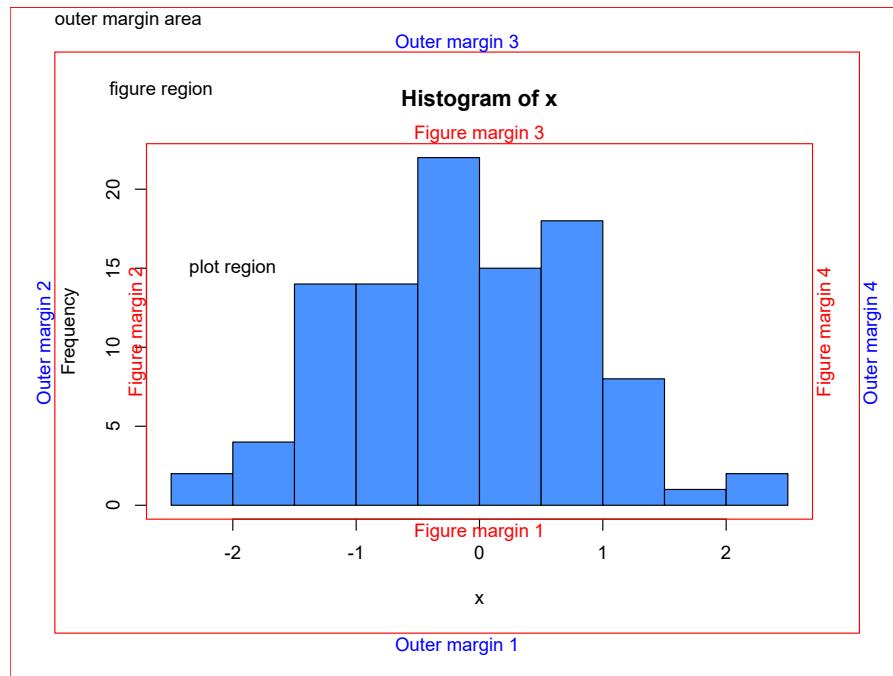


FIGURE 5.1: Anscombe's quartet: <https://goo.gl/Ugv3Cz>에서 스크립트
발췌

- 시각화는 분석에 필요한 통계량 또는 분석 방법론에 대한 가이드를 제시
- 인간의 뇌 구조 상 추상적인 숫자나 문자 보다는 그림이나 도표를 더 빨리 이해
- 다른 통계 패키지(SPSS, SAS, STATA 등)와 비교할 수 없을 정도로 월등한 성능의 그래픽 도구 및 기능 제공

5.1 R 기본 그래프 함수

- R의 그래픽은 그래픽 장치에 특정 그림(선, 점, 면 등)을 순차적으로 추가하는 명령(스크립트)을 통해 생성
- **그래픽 장치**: R에서 그래프가 출력되는 장치
 - windows: R 프로그램 내에서 출력
 - graphic files: pdf, jpeg, tiff, png, bmp 등의 확장자를 갖는 이미지 파일
- 그래프 장치를 열기 위해 사용되는 함수 - `windows()` 또는 `win.graph()`:
그래픽 장치를 열기 위해 사용하는 함수 - `dev.cur()`: 현재 활성화된 그래프
장치 확인 - `dev.set()`: 다수의 그래프 장치가 열려 있는 경우 `which = 번호`
로 변경 - `dev.list()`: 현재 열려 있는 그래픽 장치 목록 조회 - `dev.off()`:
현재 작업 중인 그래픽 장치 중지 - `graphics.off()`: 열려있는 모든 그래픽
장치 중지
- R 그래프의 구조
 - Figure region: 범례(legend), x축, y축, 도표 등을 그래프가 표현하는 모든
구성요소를 포함하는 영역(plot region 포함)
 - Plot region: 도표 부분 출력되는 영역
 - Figure margin: figure region 안에서 plot region의 여백 부분을 나타내며,
x, y 축 레이블(label), 제목(title), 각 축의 tick 및 값 등이 주로 위치하는
영역
 - Outer margin: figure region 밖의 여백 부분

**FIGURE 5.2:** R 그래프영역

R 기본 그래프 함수에 대한 강의 내용은 주로 AIMS-R-users¹에서 참고를 함

그래프의 요소: 점(point), 선(line), 면(area), 텍스트(text), 축(axis),
눈금(tick), 범례(legend) 등

- 저수준 그래프 함수(low level plotting function): 위의 그래프 요소들을 개별적으로 작업(좌표축 정의, 여백 정의)하기 위한 함수군
- 고수준 그래프 함수(high level plotting function): 그래프의 함수 기능(저수준 그래프 함수)을 모아서 하나의 완성된 도표(산점도, 막대도표, 히스토그램, 상자그림 등)를 생성할 수 있는 함수군
 - 고수준 그래프 함수를 호출할 경우 자동으로 그래픽 장치가 열려서

`win.graph()` 등을 사용할 필요가 없으나, 이미 호출된 그래프는 사용



주의: 일반적으로 R 기본 그래픽 함수로 도표 작성 시 저수준 그래프 함수는 고수준 그래프 함수로 생성한 그래프에 부가적 기능을 추가하기 위해 사용됨. 따라서 저수준 그래프 함수들은 고수준 그래프 함수를 통해 먼저 생성한 그래프(주로 아래 설명할 `plot()` 함수) 위에 적용됨.

5.2 고수준 그래프 함수

5.2.1 `plot()` 함수

- R의 가장 대표적인 2차원 고수준 그래프 출력 함수
- `plot()`의 가장 일반적인 용도는 그래프 장치를 설정(축, 값의 범위 등) 후 저수준 그래프 함수(축, 선, 점, 면 등)를 그래프 장치에 적용
- 데이터가 저장되어 있는 객체(벡터, 행렬, 데이터 프레임 등) 하나 이상을 함수의 인수(argument)로 사용
- 데이터의 클래스에 따라 출력되는 그래프 결과가 다름 → `methods(plot)`을 통해 `plot()` 함수가 적용되는 클래스 확인 가능

```
#각 클래스에 적용되는 plot() 함수 리스트
```

```
methods(plot)
```

```
[1] plot,ANY-method          plot,color-method      plot.acf*  
[4] plot.ACF*                plot.augPred*        plot.compareFits*  
[7] plot.data.frame*         plot.decomposed.ts*  plot.default  
[10] plot.dendrogram*        plot.density*       plot.ecdf  
[13] plot.factor*             plot.formula*        plot.function  
[16] plot.ggplot*             plot.gls*           plot.gtable*
```

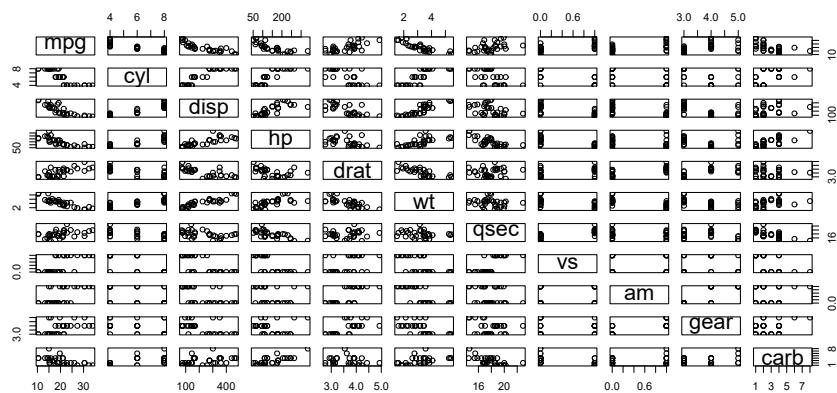
```
[19] plot.hcl_palettes*      plot.hclust*          plot.histogram*
[22] plot.HoltWinters*     plot.intervals.lmList*  plot.isoreg*
[25] plot.lm*               plot.lme*             plot.lmList*
[28] plot.medpolish*       plot.mlm*             plot.nffGroupedData*
[31] plot.nfnGroupedData*   plot.nls*             plot.nmGroupedData*
[34] plot.pdMat*            plot.ppr*             plot.prcomp*
[37] plot.princomp*         plot.profile.nls*    plot.R6*
[40] plot.ranef.lme*        plot.ranef.lmList*   plot.raster*
[43] plot.shingle*          plot.simulate.lme*  plot.spec*
[46] plot.stepfun           plot.stl*             plot.table*
[49] plot.trans*            plot.trellis*        plot.ts
[52] plot.tskernel*         plot.TukeyHSD*       plot.Variogram*
see '?methods' for accessing help and source code
```

#예시 1: 객체 클래스가 데이터 프레임인 경우

```
# mtcars 데이터 예시
class(mtcars)
```

```
[1] "data.frame"
```

```
plot(mtcars)
```



```
# 예시2: lm()으로 도출된 객체(list)
## 연비(mpg)를 종속 변수, 배기량(disp)을 독립변수로 한 회귀모형
## lm() 함수 사용 -> 객체 클래스는 lm
```

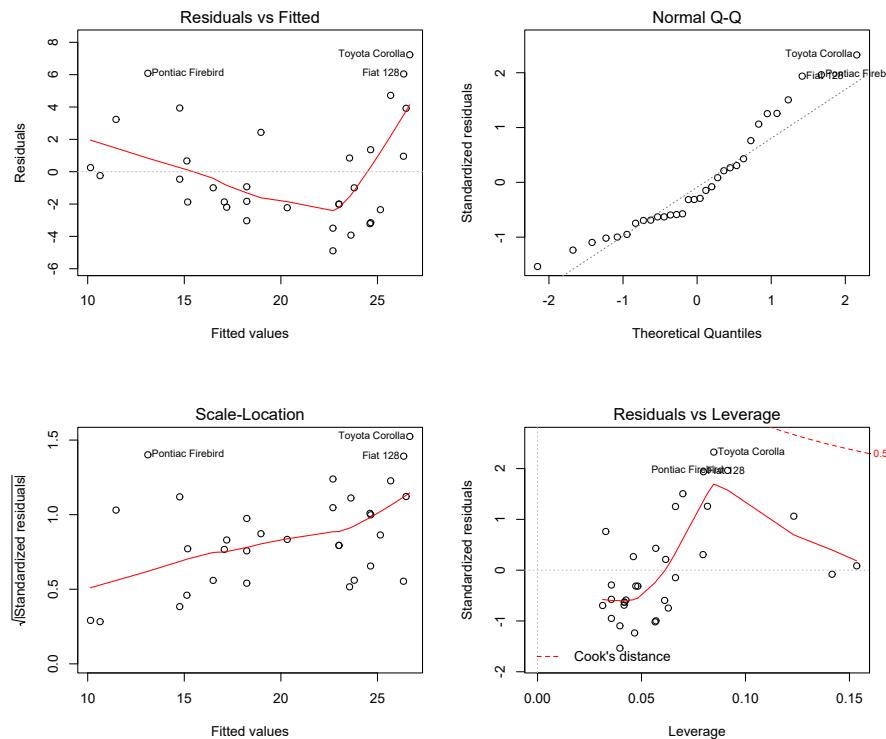
```
mod <- lm(mpg ~ disp, data = mtcars)
class(mod)
```

```
[1] "lm"
```

```
par(mfrow = c(2, 2)) # 4개 도표를 한 화면에 표시(2행, 2열)
plot(mod)
dev.off() # 활성화된 그래프 장치 닫기
```

```
null device
```

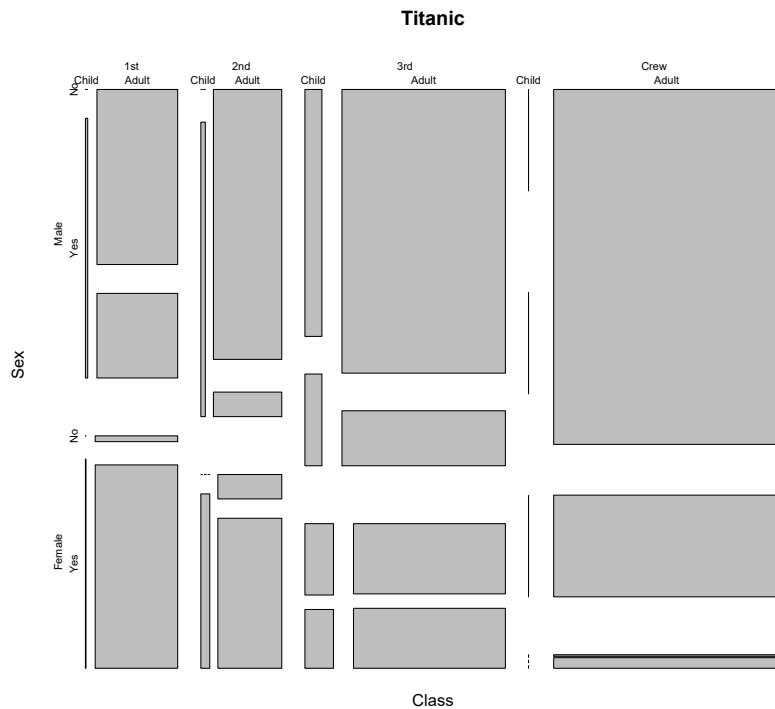
```
1
```



```
# 예시 3: 테이블 객체
class(Titanic)
```

```
[1] "table"
```

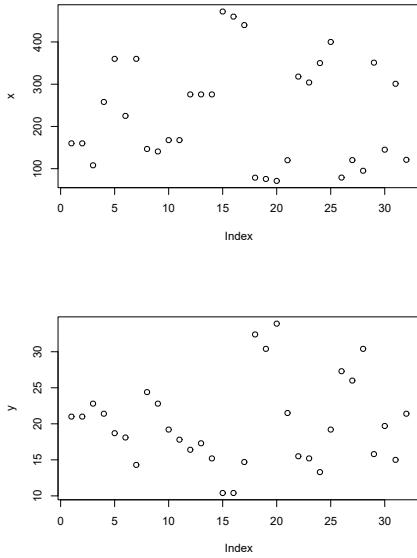
```
plot(Titanic)
```



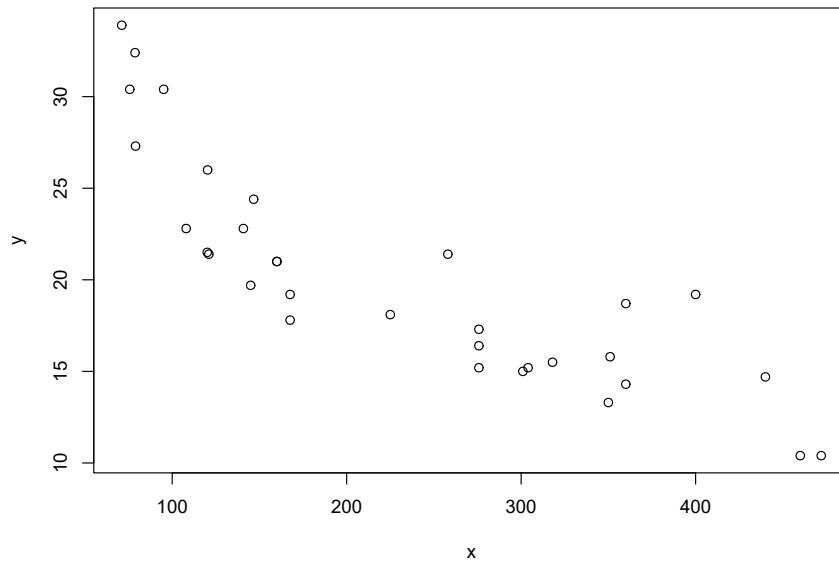
- 객체의 클래스가 벡터나 행렬인 경우, 객체에 저장된 데이터를 2차원 평면 (x-y 좌표)에 출력

```
# 예시 1: 데이터 객체를 하나만 인수로 받는 경우
# -> x축은 객체의 색인이고, x의 데이터는 y 좌표에 매핑
x <- mtcars$disp
y <- mtcars$mpg

plot(x); plot(y)
```



```
# 두개의 객체를 인수로 받은 경우  
# -> 2차원 산점도 출력  
  
plot(x, y)
```



- `plot()` 함수의 세부 옵션

```
plot(  
  x, # x 축에 대응하는 데이터 객체  
  y, # y 축에 대응하는 데이터 객체  
  type, # 그래프 타입(예시 참조)  
  main, # 제목  
  sub, # 부제목  
  xlim, ylim, # x, y 축 범위 지정  
  xlab, ylab, # x-y 축 이름  
  lty, # 선 모양  
  pch, # 점 모양  
  cex, # 점 및 텍스트 크기  
  lwd, # 선 굵기  
  col # 색상  
)
```

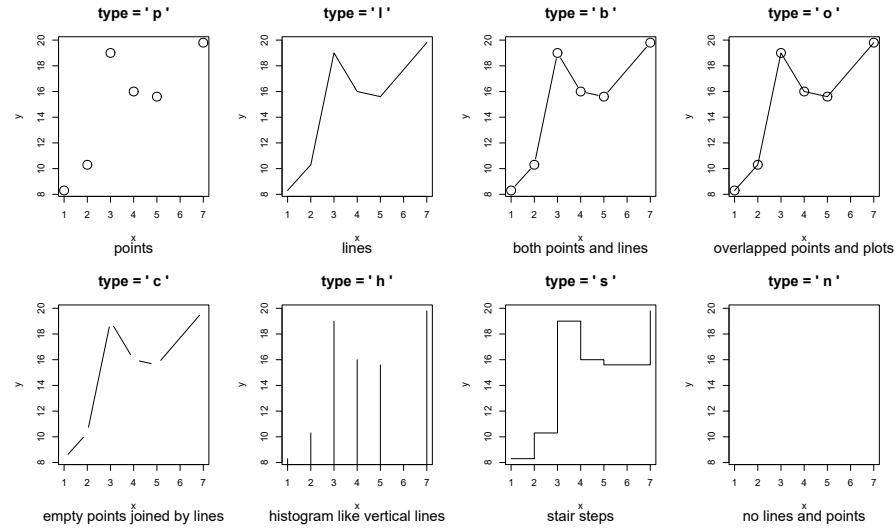
- type 인수: 그래프 타입 지정

```
# BOD 데이터셋 이용
x <- BOD$Time; y <- BOD$demand
x; y

[1] 1 2 3 4 5 7
[1] 8.3 10.3 19.0 16.0 15.6 19.8

ctype <- c("p", "l", "b", "o", "c", "h", "s", "n")
type_desc <- c("points", "lines",
              "both points and lines",
              "overlapped points and plots",
              "empty points joined by lines",
              "histogram like vertical lines",
              "stair steps",
              "no lines and points")

op <- par(mfrow = c(2, 4))
for (i in 1:length(ctype)) {
  plot(x, y,
        type = ctype[i],
        main = paste("type =", "", ctype[i], ""),
        sub = type_desc[i],
        cex.main = 1.5,
        cex.sub = 1.5,
        cex = 2)
}
```



```
par(op)
```

- **xlim, ylim** 인수: x, y 축의 범위 지정

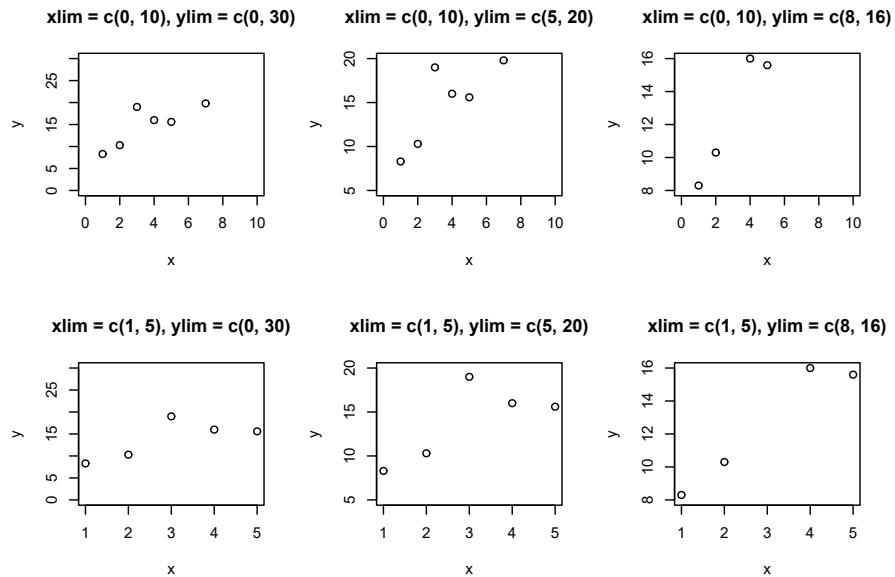
```
op <- par(mfrow = c(2, 3))
range <- data.frame(
  x1 = rep(c(0, 1), each = 3),
  x2 = rep(c(10, 5), each = 3),
  y1 = rep(c(0, 5, 8), times = 2),
  y2 = rep(c(30, 20, 16), times = 2)
)
for (i in 1:6) {
  plot(x, y,
    xlim = as.numeric(range[i, 1:2]),
    ylim = as.numeric(range[i, 3:4]),
    main = paste0("xlim = c(",
      paste(as.numeric(range[i, 1:2]),
        collapse = ", ")),
      ", ",
      "ylim = c(",
      "
```

```

    paste(as.numeric(range[i, 3:4]),
          collapse = ", ", ")"))
}

par(op)

```



- **xlab, ylab** 인수: x축과 y축 이름 지정

```

x_lab <- c(" ", "Time (days)")
y_lab <- c("Demand", "Oxygen demand (mg/l)")

op <- par(mfrow = c(2, 2))
lab_d <- expand.grid(x_lab, y_lab)

for (i in 1:4) {
  plot(x, y,
       xlab = lab_d[i, 1],

```

```

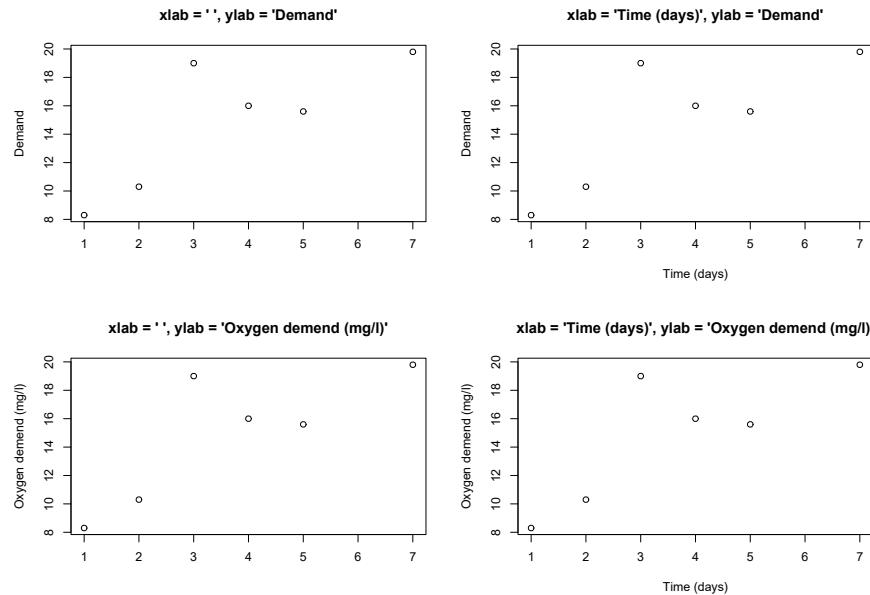
ylab = lab_d[i, 2],
main = paste0("xlab = '", "", lab_d[i, 1], "'", "", "",
              "ylab = '", "'", lab_d[i, 2], "'")
)
}

par(op); dev.off()

```

null device

1



- lty 인수: 선의 형태 지정

```

line_type <- c("blank", "solid", "dashed", "dotted",
             "dotdash", "longdash", "twodash")
plot(x = c(1:7), y = c(1:7), type="n",
      axes = FALSE,
      xlab = "",
      ylab = ""

```

```
main = "Basic Line Types",
cex.main = 1.5)

for (i in 1:length(line_type)) {
  lines(c(1, 5.2), c(i, i), lty = i - 1, lwd = 2)
  text(5.5, i,
       labels = paste0("lty = ", i - 1, " (",
                      line_type[i], ")"),
       cex = 1.2,
       adj = 0)
}
```

Basic Line Types

----- Ity = 6 (twodash)

----- Ity = 5 (longdash)

----- Ity = 4 (dotdash)

----- Ity = 3 (dotted)

----- Ity = 2 (dashed)

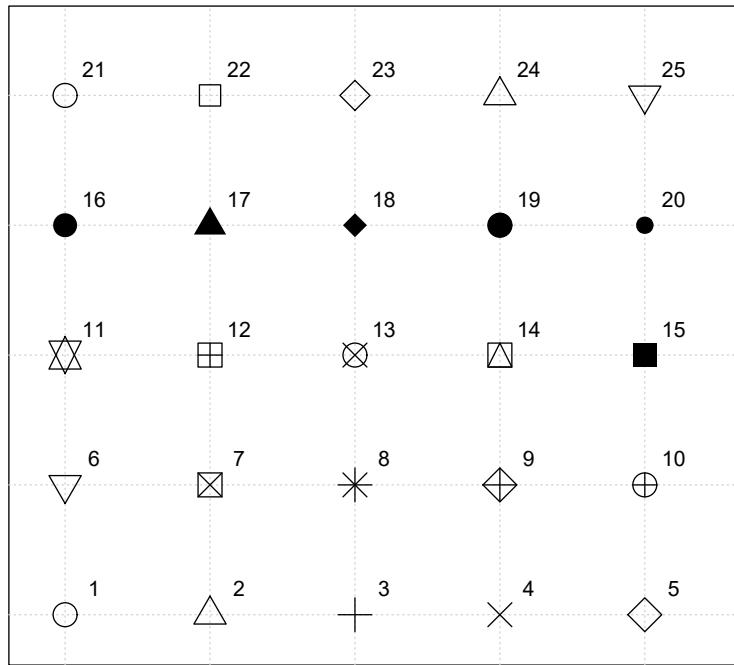
----- Ity = 1 (solid)

Ity = 0 (blank)

- pch 인수: 점(point)의 모양을 지정

```
coord <- expand.grid(x = 1:5, y = 1:5)
plot(coord, type = "n",
      xlim = c(0.8, 5.5),
      ylim = c(0.8, 5.5),
      xlab = "",
      ylab = "",
      main = "Basic plotting characters",
      xaxt = "n",
      yaxt = "n")
grid()
points(coord, pch=1:25, cex = 2.5)
text(coord + 0.2, labels = 1:25, cex = 1)
```

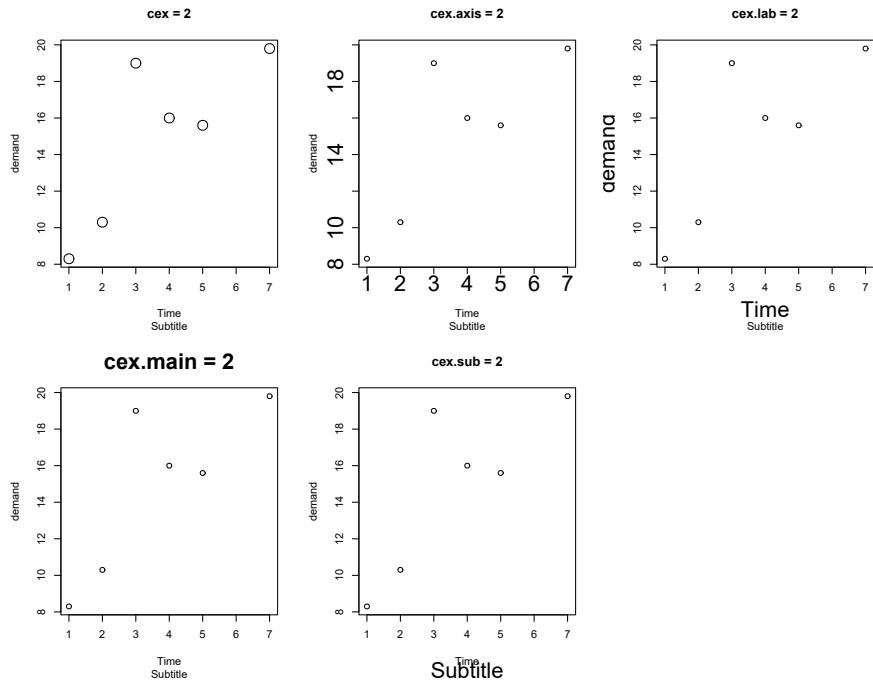
Basic plotting characters



- `cex` 인수: 텍스트 및 점의 크기 지정
 - `cex.axis`: 각 축의 눈금 레이블 크기 조정
 - `cex.lab`: x-y 축의 이름 크기 조정
 - `cex.main`: 그림 제목 크기 조정
 - `cex.sub`: 부제목 크기 조정
- 텍스트 `cex` 인수 적용 예시

```
par(mfrow = c(2, 3))
plot(BOD, type = "p", cex = 2,
     main = "cex = 2",
```

```
sub = "Subtitle")
plot(BOD, type = "p",
      cex.axis = 2,
      main = "cex.axis = 2",
      sub = "Subtitle")
plot(BOD, type = "p",
      cex.lab = 2,
      main = "cex.lab = 2",
      sub = "Subtitle")
plot(BOD, type = "p",
      cex.main = 2,
      main = "cex.main = 2",
      sub = "Subtitle")
plot(BOD, type = "p",
      cex.sub = 2,
      main = "cex.sub = 2",
      sub = "Subtitle")
```



- **lwd** 인수: 선의 두께 지정
 - 점 **cex** 크기와 **lwd** 두께

```

coord <- expand.grid(x = 1:5, y = 1:5)
plot(coord, type="n",
     xlab = "cex",
     ylab = "lwd",
     xlim = c(0.5, 5.5),
     ylim = c(0.5, 5.5),
     main = "pch and lwd size",
     cex.main = 2,
     cex.lab = 1.5)
points(coord, pch=16, cex = 1:5, col = "darkgray")
for (i in 1:5) {
  points(1:5, coord$y[coord$y == i], pch=21,
         cex = 1:5,
         lwd = 1:5)
}

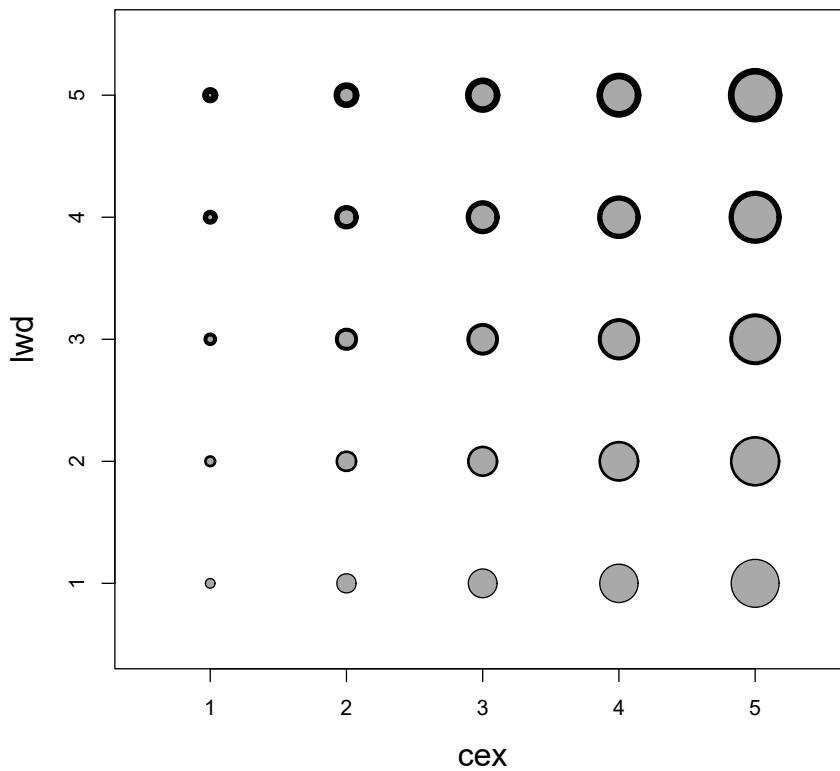
```

```

    lwd = i, col = "black")
}

```

pch and lwd size



- col 인수: 그래프의 점, 면, 선의 색상
- palette() 함수를 통해 그래픽 기본 색상 확인(총 8개)

| | | | | | | | | |
|-------------|-------------|-----------|--------------|------------|------------|---------------|--------------|------------|
| col='white' | col='black' | col='red' | col='green3' | col='blue' | col='cyan' | col='magenta' | col='yellow' | col='gray' |
| col=0 | col=1 | col=2 | col=3 | col=4 | col=5 | col=6 | col=7 | col=8 |

- colors()를 통해 R에서 기본으로 제공하는 색상 확인 가능(총 657개)

- 내장 색상 팔레트: n 개의 색상을 반환하고, 색상의 투명도는 alpha 인수를 통해 조정
 - `rainbow(n)`: Red → Violet
 - `heat.colors(n)`: White → Orange → Red
 - `terrain.colors(n)`: White → Brown → Green
 - `topo.colors(n)`: White → Brown → Green → Blue
 - `grey(n)`: White → Black
- R Color Chart² 참고

5.2.2 주요 고수준 그래픽 함수

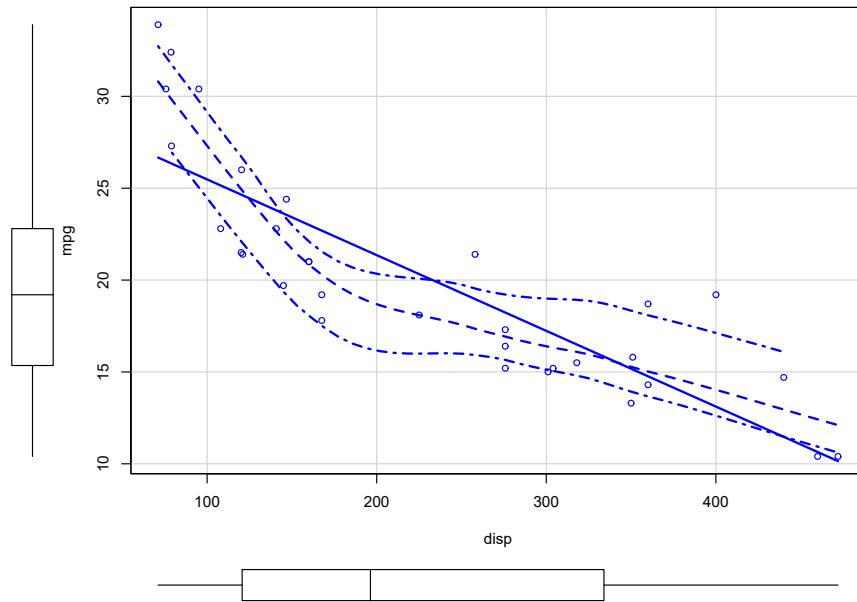
산점도

`car::scatterplot()`

- `plot(x, y)`를 통해 2차원 산점도를 그릴 수 있으나, car 패키지에 내장되어 있는 해당 함수를 이용해 보다 많은 정보(상자그림, 회귀곡선 등)를 포함

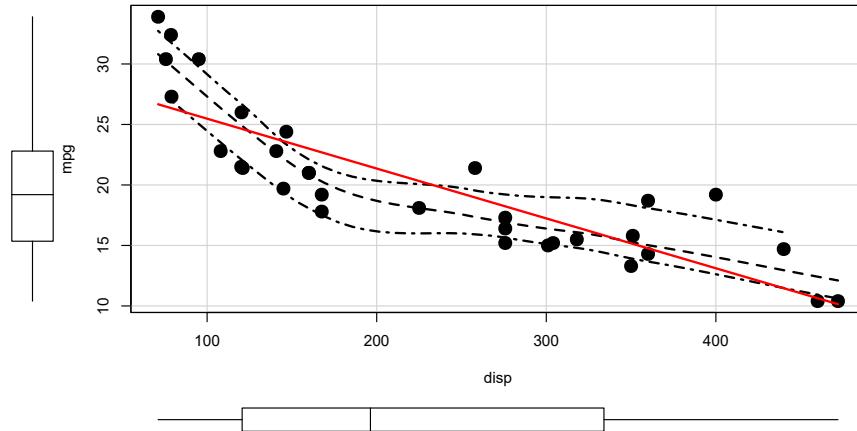
```
# car 패키지 설치  
# install.packages("car")  
# require(car)  
car::scatterplot(mpg ~ disp, data = mtcars)
```

²https://rstudio-pubs-static.s3.amazonaws.com/3486_79191ad32cf74955b4502b8530aad627.html



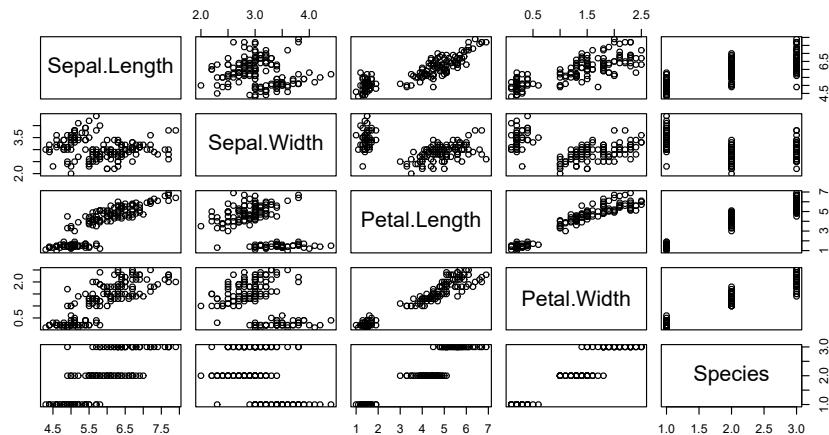
- `plot()` 함수의 인수 적용 가능

```
# help(scatterplot) 참고
car::scatterplot(mpg ~ disp, data = mtcars,
                 regLine = list(method = lm, lty = 1, col = "red"),
                 col = "black", cex = 2, pch = 16)
```



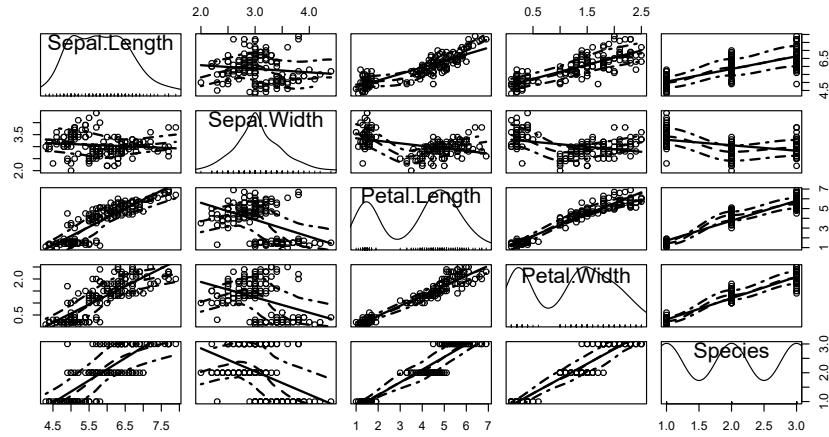
- `pairs()`: 산점도 행렬을 생성해주는 함수로, 객체의 클래스가 데이터 프레임인 경우 `plot(dat)`과 동일한 그래프를 반환

```
# iris dataset
plot(iris)
```

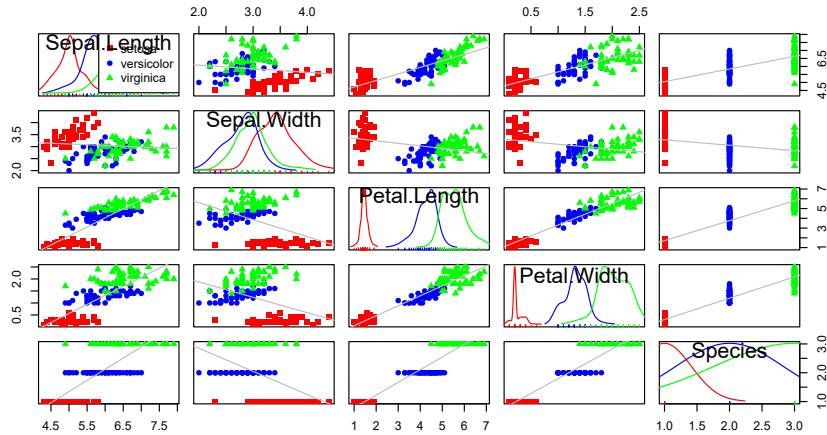


- `car:::scatterplotMatrix()`: R graphics 패키지의 `pair()`와 유사하나 각 변수 쌍별 회귀 곡선 및 분포 확인 가능

```
# iris dataset  
car:::scatterplotMatrix(iris, col = "black")
```



```
# help(scatterplotMatrix)  
car:::scatterplotMatrix(iris, col = c("red", "blue", "green"),  
                      smooth = FALSE,  
                      groups = iris$Species,  
                      by.groups = FALSE,  
                      regLine = list(method = lm, lwd = 1, col = "gray"),  
                      pch = (15:17))
```



행렬 그래프

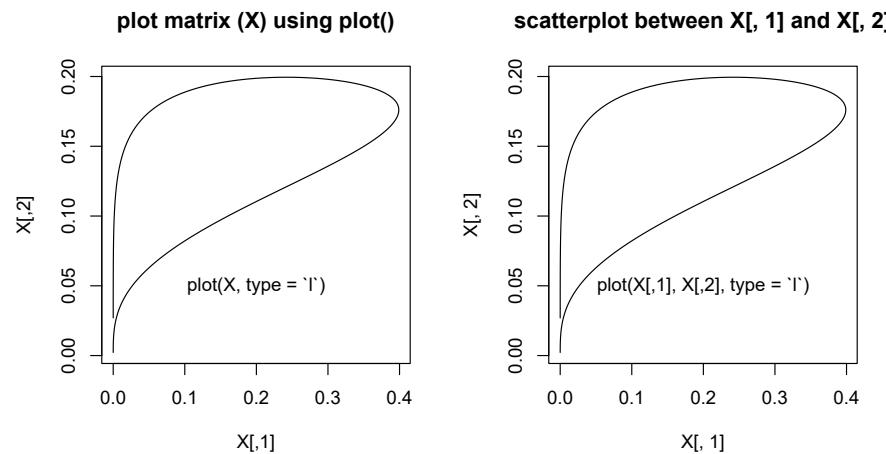
- 행렬 객체를 `plot()` 함수의 입력으로 사용한 경우 1-2 번째 열 데이터로 산점도를 출력

```
# 행렬을 plot() 함수의 입력으로 받은 경우
par(mfrow = c(1,2))
x <- seq(-5, 5, 0.01)
X <- mapply(dnorm,
             list(a = x, b = x, c = x),
             c(0, 1, 2),
             c(1, 2, 4))
X <- matrix(X, nrow = length(x), ncol = 3)
head(X)
```

| | [,1] | [,2] | [,3] |
|------|--------------|-------------|------------|
| [1,] | 1.486720e-06 | 0.002215924 | 0.02156933 |
| [2,] | 1.562867e-06 | 0.002249385 | 0.02166383 |
| [3,] | 1.642751e-06 | 0.002283295 | 0.02175862 |
| [4,] | 1.726545e-06 | 0.002317658 | 0.02185368 |
| [5,] | 1.814431e-06 | 0.002352479 | 0.02194902 |

```
[6,] 1.906601e-06 0.002387763 0.02204463
```

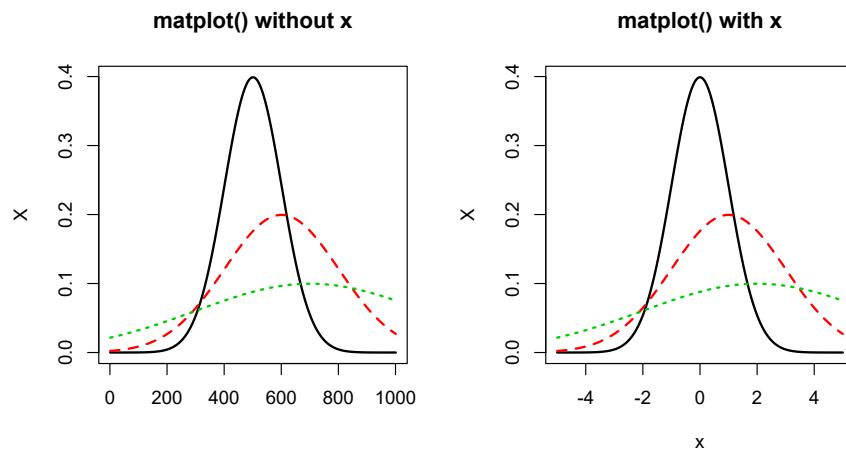
```
# plot() 함수를 이용한 행렬 그래프 출력
plot(X, type = "l", main = "plot matrix (X) using plot()")
text(0.2, 0.05, labels = "plot(X, type = 'l')")
plot(X[, 1], X[, 2], type = "l",
      main = "scatterplot between X[, 1] and X[, 2]")
text(0.2, 0.05, labels = "plot(X[,1], X[,2], type = 'l')")
```



- `matplot()`: 객체의 클래스가 행렬 (matrix) 형태로 이루어진 데이터에 대한 그래프 출력
 - 열 기준으로 그래프 출력
 - x 가 주어지지 않은 경우, 행렬의 색인을 x 축으로 사용

```
# matplot 도표
par(mfrow = c(1, 2))
matplot(X, type = "l",
        lwd = 2,
        main = "matplot() without x")
matplot(x, X, type = "l",
```

```
lwd = 2,
main = "matplotlib() with x")
```



히스토그램

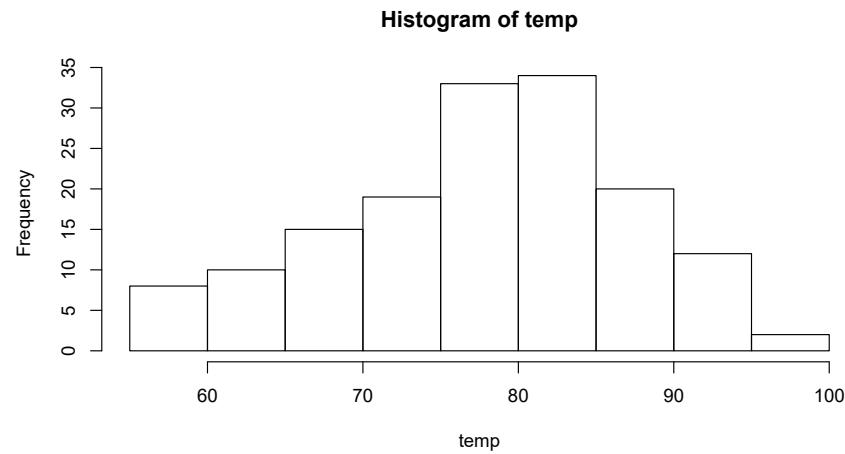
hist()

```
hist(
  x, # vector 객체
  breaks, # 빈도 계산을 위한 구간
  freq, # y축 빈도 또는 밀도 (density) 여부
  col, # 막대 색상 지정
  border, # 막대 테두리 색 지정
  labels, # 막대 위 y 값 레이블 출력 여부
  ...
)
```

```
# airquality 데이터셋
# help(airquality) 참고
glimpse(airquality)
```

```
Observations: 153
Variables: 6
$ Ozone    <int> 41, 36, 12, 18, NA, 28, 23, 19, 8, NA, 7, 16, 11, 14, 18, 1...
$ Solar.R   <int> 190, 118, 149, 313, NA, NA, 299, 99, 19, 194, NA, 256, 290, ...
$ Wind      <dbl> 7.4, 8.0, 12.6, 11.5, 14.3, 14.9, 8.6, 13.8, 20.1, 8.6, 6.9...
$ Temp      <int> 67, 72, 74, 62, 56, 66, 65, 59, 61, 69, 74, 69, 66, 68, 58, ...
$ Month     <int> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, ...
$ Day       <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ...
```

```
temp <- airquality$Temp
hist(temp)
```



- `hist()` 함수의 반환값

```
h <- hist(temp, plot = FALSE) # 그래프를 반환하지 않음
h
```

```
$breaks
[1] 55 60 65 70 75 80 85 90 95 100

$counts
```

```
[1] 8 10 15 19 33 34 20 12 2

$density
[1] 0.010457516 0.013071895 0.019607843 0.024836601 0.043137255 0.044444444
[7] 0.026143791 0.015686275 0.002614379

$mid
[1] 57.5 62.5 67.5 72.5 77.5 82.5 87.5 92.5 97.5

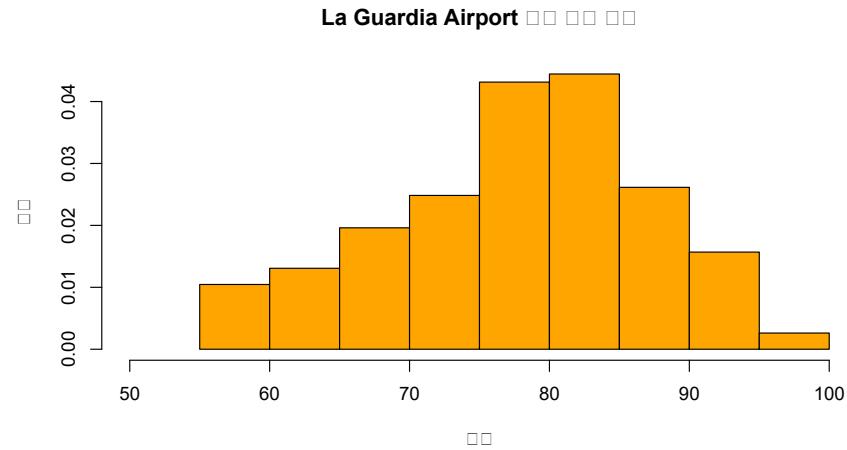
$xname
[1] "temp"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
```

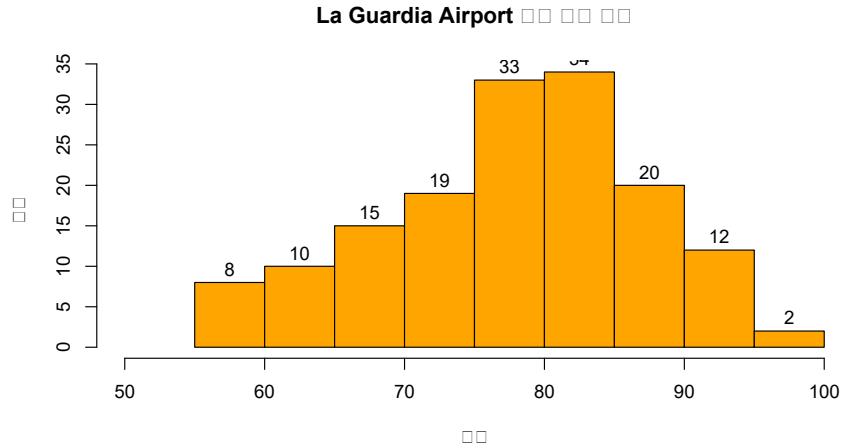
- `hist()` 함수의 인수 사용(`plot()` 함수의 인수 거의 대부분 사용 가능)

```
hist(temp,
main="La Guardia Airport 일중 최고 기온",
xlab = "온도",
ylab = "밀도",
xlim = c(50,100),
col = "orange",
freq = FALSE
)
```



- `labels` 인수를 통해 빈도값 출력

```
hist(temp,
main = "La Guardia Airport 일중 최고 기온",
xlab = "온도",
ylab = "빈도",
xlim = c(50,100),
col = "orange",
labels = TRUE
)
```

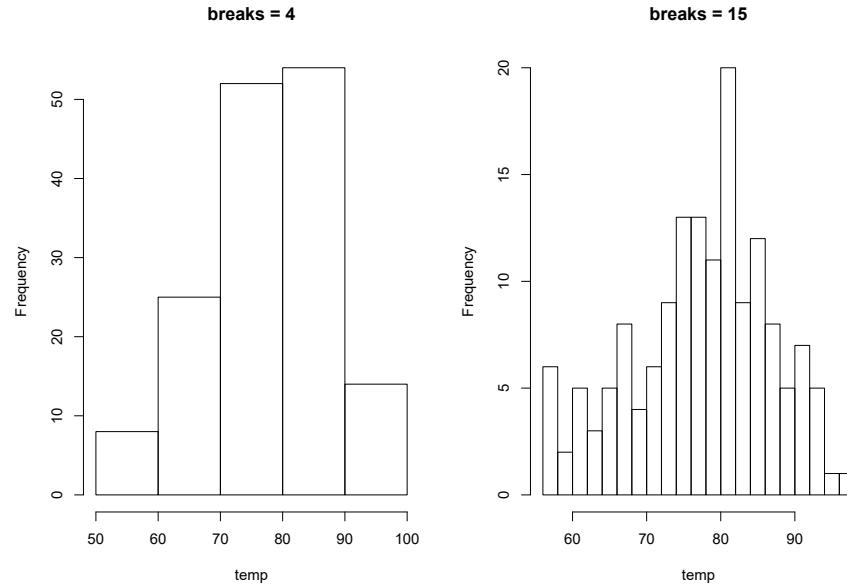


- `breaks` 인수를 통해 막대 구간 조정

```
op <- par(mfrow = c(1, 2))
hist(temp, breaks = 4, main = "breaks = 4")
hist(temp, breaks = 15, main = "breaks = 15")
par(op); dev.off()
```

```
null device
```

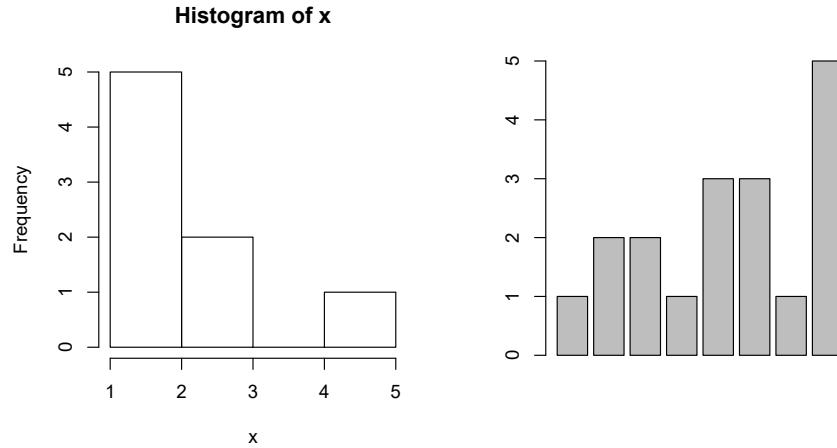
```
1
```



막대 그래프

- 히스토그램(hist())은 연속형 데이터의 구간 별 빈도 또는 밀도를 나타냄
- 막대 도표(bar plot)는 해당 좌표의 값(value)를 나타냄

```
x = c(1,2,2,1,3,3,1,5)
par(mfrow = c(1, 2))
hist(x); barplot(x)
```



barplot()

- `help(barplot)` 을 통해 함수 사용 방법 확인
- 보통 요약통계량(예: 그룹별 빈도, 평균)의 시각화를 위해 많이 사용

```
## Wool dataset: warpbreaks
## 제직 중 방적 횟수
## 직조기 당 날실 파손 횟수 데이터
head(warpbreaks)
```

```
breaks wool tension
1     26    A      L
2     30    A      L
3     54    A      L
4     25    A      L
5     70    A      L
6     52    A      L
```

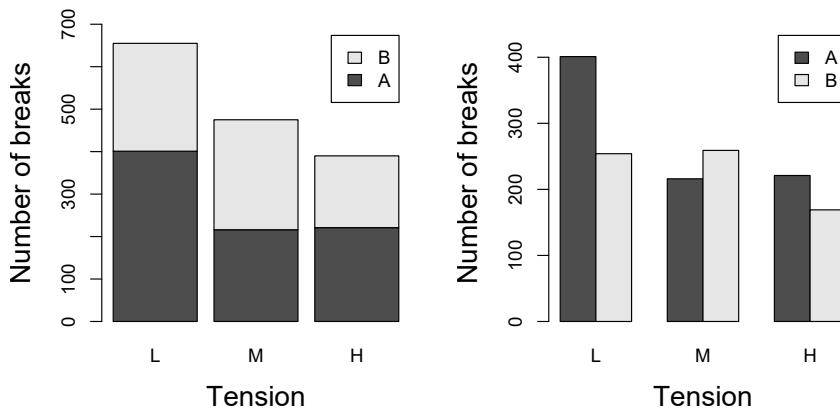
```
count <- with(warpbreaks,
  tapply(breaks, list(wool, tension),
  sum))
```

```

par(mfrow = c(1, 2))
barplot(count, legend = TRUE,
        xlab = "Tension",
        ylab = "Number of breaks",
        ylim = c(0, 700),
        cex.lab = 1.5) # stack 형태

barplot(count, legend = TRUE, beside = TRUE,
        xlab = "Tension",
        ylab = "Number of breaks",
        ylim = c(0, 450),
        cex.lab = 1.5) # 분리 형태

```



- 데이터 프레임을 대상으로 `barplot()` 실행 시 수식 표현 가능
- 막대도표 + 표준오차

```

mean_breaks <- aggregate(breaks ~ wool + tension,
                           data = warpbreaks,
                           mean)

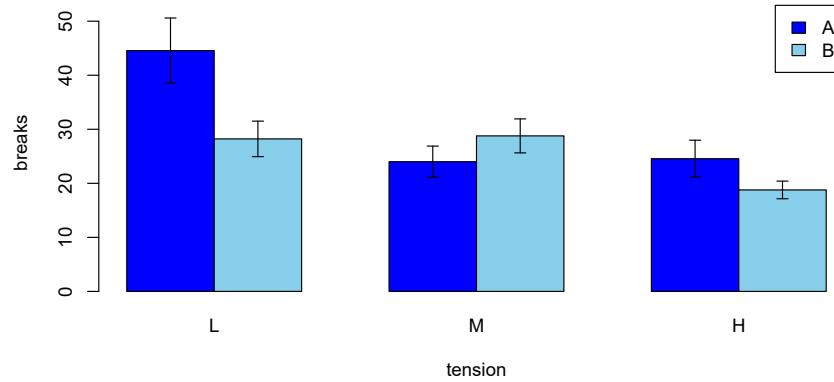
```

```
se_breaks <- aggregate(breaks ~ wool + tension,
                      data = warpbreaks,
                      FUN = function(x) sd(x)/sqrt(length(x)))

barplot(breaks ~ wool + tension,
        data = mean_breaks,
        ylim = c(0, 55),
        beside = TRUE,
        legend = TRUE, # 범례
        col = c("blue", "skyblue")
      ) -> bp

cent <- matrix(mean_breaks$breaks, 2, 3)
sem <- matrix(se_breaks$breaks, 2, 3)

arrows(bp, cent - sem, bp, cent + sem, angle = 90, code = 3, length = 0.05)
```



Bibliography

- (2018). Gapminder. <https://ko.wikipedia.org/wiki/%ED%8A%A7%EC%9D%98>.
- (2018). R 에서 원하는 키워드의 뉴스를 웹크롤링(스크래핑) 하는 방법.
<https://dr-hkim.github.io/Naver-News-Web-Scraping-using-Keywords-in-R/>.
- (2018 Accessed: 2020-04-16). POSIX. <https://ko.wikipedia.org/wiki/POSIX>.
- Bryan, J. (2017). *gapminder: Data from Gapminder*. R package version 0.3.0.
- Chen, D.-G. D. and Peace, K. E. (2010). *Clinical trial data analysis using R*. CRC Press.
- Rizzo, M. L. (2019). *Statistical computing with R*. CRC Press.
- Wickham, H. (2007). Reshaping data with the reshape package. *Journal of Statistical Software*, 21(12):1–20.
- Wickham, H. (2016). *ggplot2: elegant graphics for data analysis*. Springer.
- Wickham, H. (2019a). *feather: R Bindings to the Feather 'API'*. R package version 0.3.5.

Wickham, H. (2019b). *tidyverse: Easily Install and Load the 'Tidyverse'*. R package version 1.3.0.

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., Francois, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Muller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., Takahashi, K., Vaughan, D., Wilke, C., Woo, K., and Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, 4(43):1686.

Wickham, H. et al. (2014). Tidy data. *Journal of Statistical Software*, 59(10):1–23.

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., Allaire, J., and Grolemund, G. (2018). *R Markdown: The Definitive Guide*. Chapman and Hall/CRC, Boca Raton, Florida. ISBN 9781138359338.

권재명 (2017). 실리콘밸리 데이터 과학자가 알려주는 따라하며 배우는 데이터 과학. 제이펍, 1st edition. ISBN 979-1185890869.

매트로프, . (2012). 빅데이터 분석 도구 R 프로그래밍. 에이콘출판, 1st edition. ISBN 978-8960773332.

서민구 (2014). R을 이용한 데이터 처리 & 분석. 길벗, 1st edition. ISBN 978-8966188260.

유충현, 이상호, and 김정일 (2005). *R 그래픽스. 자유아카데미*, 1st edition.

ISBN 978-8973385539.
