

fasteR

Hyeongchan Bae

May 2022

Contents

Day 1	1
1. Download	1
2. Something to Know	2
3. Basic Function	4
4. Others	9
5. Quiz	9
Day 2	11
1. What is Tidyverse	11
2. Exercise : 보도자료 만들기	12
3. Advanced : 미국 수출데이터 가공하기	22
4. Quiz	26
Day 3	28

Day 1

R과 통계분석 (Tidyverse 활용) p.4~60

1. Download

R 프로그래밍 언어. 제일 먼저 설치

- <https://cran.r-project.org/bin/windows/base>

RStudio R을 활용하기 위한 통합개발환경(IDE, Integrated Development Environment)

- 달리 말하면 RStudio 외에도 다양한 프로그램에서 R을 사용할 수 있음. 여러 언어를 사용하는 개발자는 VS Code 같은 단일 IDE를 활용하기도.
- <https://www.rstudio.com/products/rstudio/download>

Rtools 패키지를 설치하다 보면 필요한 경우(Compile)가 더러 있음

- <https://cran.r-project.org/bin/windows/Rtools>

Chrome Browser Selenium 파트에서 사용할 예정

- https://www.google.com/intl/ko_kr/chrome

Extra Files 인턴의 깃허브 페이지. 강의 노트, 작성 코드 등 추가적인 파일을 업로드 해둠

- <https://github.com/Rlearnchan/fasteR>

2. Something to Know

1) “모두 다 같은 아마추어야”

- 익숙지 않고, 오래 걸리는 게 당연해요.
- 모든 과정을 R로 수행할 필요는 없으니, 잘 안 되면 데이터를 엑셀, STATA 등으로 옮겨 처리해오셔도 좋습니다.

2) R 파일(확장자)

.R 작성한 코드

- 좌상단 Script에서 코드 작성 → 필요한 부분 실행 → 좌하단 Console 창에서 실행 결과를 확인하는 게 일반적이라, 작업 첫 파트를 저장한 것이라 볼 수 있음.
- 코드를 적은 메모장쯤 되니, **비슷한 환경**이라면 타인에게 받은 코드를 실행하기만 해도 같은 결과를 시현함.

.Rdata 작업공간 이미지

- 우상단 Environment에 기록된, 작업하며 생성된 객체, 함수, 기타 데이터들의 총체.
- 임시로 만든, 코드 실행과 무관한 object 들도 저장.

.Rhistory 작업 기록

- RStudio를 종료했다가, 다시 실행하면 이전 작업 상태가 비교적 온전히 남아있는데, 이를 위한 파일이라 하겠음.

.Rmd 마크다운 파일

- html, pdf, word 등을 만들기 위해 Markdown 문법으로 작성한, 코드 친구쯤 되는 녀석.
- 본 문서도 마크다운으로 작성. 소위 ‘R로 논문 쓴다’ 할 때 등장.

3) 한글에 유독 취약한 R

UTF-8 인코딩 방식을 변경해주세요.

- Tools → Global Options → Code → Saving 경로.

- 타인에게 받은, 혹은 건네준 코드 파일에서 한글이 깨져 보인다면 대체로 이 문제.
- File → Reopen with Encoding 기능을 활용해 대처하는 방법도 있음.

Library Path 패키지 설치 경로에 한글 네이밍이 없도록 해주세요.

```
.libPaths() # 첫 번째가 default. 개인 폴더가 설정돼 두 개 나오기도 한다.
```

```
## [1] "C:/Program Files/R/R-4.1.3/library"
```

- (윈도우 기준 예시) 만일 [2] “C:/Users/**사용자 이름**/Documents/R/win-library/4.1.3” 경로에 한글이 포함된다면, 패키지를 다룰 때마다 오류 사인을 접할 공산이 큼.
- 새로 계정을 만들지 않는 이상, **사용자 이름** 구간 폴더명은 변경하기도 어려움.
- 다음과 같이 **사용자 이름**이 없는 기본(공용) 라이브러리를 default로 설정하길 권장.

```
Sys.setenv('R_LIBS_USER' = 'C:/Program Files/R/R-4.1.3/library')
```

```
# R의 '개인' 세팅을 앞서 발견한 '기본(공용)' 경로로 덮어쓰기.
```

```
.libPaths('R_LIBS_USER') # 바뀐 '개인 라이브러리'를 패키지 설치 경로로 설정.
```

```
.libPaths() # 하나의 경로로 잘 세팅되고,
```

```
## [1] "C:/Program Files/R/R-4.1.3/library"
```

```
.libPaths() == Sys.getenv('R_LIBS_USER') # 개인 라이브러리 경로와도 일치
```

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

4) ?, ?? 사용법

? 모르는 함수 검색하기

```
?print
```

- 대부분의 함수는 R Documentation 이라 해서 정의와 기능, 인자, 간단한 사용 예시 등을 요약해둔 페이지를 가지고 있음.
- 예컨대 print() 함수를 자세히 알고 싶다면, 위에서 처럼 ? 하나 붙여서 실행하면 됨.

?? 모르는 개념, 워딩 검색하기

```
??print
```

- 하지만 함수 이름조차 모르거나, 기능을 연상할 키워드 정도만 간신히 아는 경우도 많음.
- ??는 모든 R Documentation 에서 해당 단어가 포함된 것을 모두 골라 보여줌.

- 두 가지를 적절히 섞어 사용하는 게 좋음.

5) 구글링

- 사실 구글은 모든 걸 알고 있음.
- `str_detect()` 식으로 함수 이름 자체를 검색하면 국내외 사용자들이 포스팅한 글을 찾아보기 편함.
- warning 혹은 error 사인은 해당 문구를 적당히 복사해 구글에 그대로 쳐보는 게 좋음.
- stackoverflow 같은 개발자 커뮤니티 게시물이 주로 나올 텐데, 같은 문제로 고민한 사람들이 꽤 많았기 때문.

3. Basic Function

1) 숫자 계산

```
3+4-7/3 # 달리 명령어가 필요하진 않으나,
```

```
## [1] 4.666667
```

```
print(3+4-7/3) # print() 함수를 사용할 수도 있음
```

```
## [1] 4.666667
```

```
print(3+4-7/3, digits = 3) # 세 번째 자리에서 반올림
```

```
## [1] 4.67
```

```
rnorm(n = 5, mean = 0, sd = 1) # n(0, 1) 분포에서 5개 난수 생성
```

```
## [1] -0.67705998 -1.20495775 0.04327926 0.69073683 -0.16271359
```

```
stats::rnorm(n = 5, mean = 0, sd = 1) # stats 패키지의 rnorm() 함수
```

```
## [1] -0.4131008 1.5567304 -0.3170105 -0.9852751 -2.2111526
```

- 기본 패키지, 혹은 `library()` 로 장착한 패키지의 함수는 `::` 표기를 사용하지 않아도 됨.
- 여러 패키지를 동시에 사용하는 경우, 유사한 이름의 함수 간 혼동을 피하기 위해 `::` 방식을 사용하기도 함.

```
set.seed(seed = 10)
```

- reproducibility 위해서 난수 생성 규칙을 `set.seed()`로 부여.

```
rnorm(5, 0, 1)
```

```
## [1] 0.01874617 -0.18425254 -1.37133055 -0.59916772 0.29454513
```

2) 텍스트

```
'banana' # 작은 따옴표
```

```
## [1] "banana"
```

```
"banana" # 큰 따옴표 모두 사용 가능
```

```
## [1] "banana"
```

```
class('banana') # 문자 클래스
```

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

- class() 함수는 자주 쓰니 기억해 둘 필요가 있음.

```
paste0('이제와', '뒤늦게', '무엇을 더 보태려하나') # 문자열 붙여서 하나로 만듦
```

```
## [1] "이제와뒤늦게무엇을 더 보태려하나"
```

```
paste0('이제와 ', '뒤늦게 ', '무엇을 더 보태려하나') # 띄어쓰기를 포함해서 붙이는 게 요령
```

```
## [1] "이제와 뒤늦게 무엇을 더 보태려하나"
```

```
paste('이제와', '뒤늦게', '무엇을 더 보태려하나') # 한 칸씩 띄어쓰는 게 default인 함수
```

```
## [1] "이제와 뒤늦게 무엇을 더 보태려하나"
```

```
paste('이제와', '뒤늦게', '무엇을 더 보태려하나', sep = '듬칫') # 사실 sep = ' ' 인자가 숨어있던 것. 바꿀 수도 있음.
```

```
## [1] "이제와듬칫뒤늦게듬칫무엇을 더 보태려하나"
```

3) 객체

```
BR31 = 'Alien Mom' # 텍스트를 BR31 객체에 저장
```

```
br31 = 'Mint Choco' # 텍스트를 br31 객체에 저장
```

```
BR31
```

```
## [1] "Alien Mom"
```

```
br31
```

```
## [1] "Mint Choco"
```

```
paste('Which do you prefer', BR31, 'or', br31) # 객체명을 입력하면 담긴 것을 가져다 씀.
```

```
## [1] "Which do you prefer Alien Mom or Mint Choco"
```

- 객체 명을 지을 땐 **대소문자 구별**, 그리고 **첫 글자엔 숫자 및 기호 불가** 특성을 고려해야 함.

4) 벡터

```
Yunha = c(4, 8, 6) # 숫자 세 개를 벡터로 묶어 저장
```

```
Yunha
```

```
## [1] 4 8 6
```

```
class(Yunha) # 숫자 속성이 그대로 남아 있음
```

```
## [1] "numeric"
```

```
Yunha = c('Password', 4, 8, 6) # 원소가 하나라도 character가 섞이면
```

```
Yunha # 따옴표 찍힌 것부터 느낌이 다르고,
```

```
## [1] "Password" "4"      "8"      "6"
```

```
class(Yunha) # 알짬없이 전부 character로 저장
```

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

5) 행렬

```
matrix(data = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12), nrow = 3, ncol = 4)
```

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

```
## [1,]  1   4   7  10
```

```
## [2,]  2   5   8  11
```

```
## [3,]  3   6   9  12
```

```
matrix(data = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12), nrow = 3, ncol = 4, dimnames = list(c('가', '나', '다'), c('A', 'B', 'C')))
```

```
##   A B C D
```

```
## 가 1 4 7 10
```

```
## 나 2 5 8 11
```

```
## 다 3 6 9 12
```

```
matrix(data = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12),
       dimnames = list(c('가', '나', '다'), c('A', 'B', 'C', 'D')),
       nrow = 3, ncol = 4) # 코드가 슬슬 길어지니, 엔터를 적극 활용.
```

```
##  A B C D
## 가 1 4 7 10
## 나 2 5 8 11
## 다 3 6 9 12
```

- 마지막 코드에선 인자들의 순서가 조금 다른데, 이는 인자명을 지정해줬기 때문에 가능한 것.
- 지정만 잘 돼있으면 섞여도 상관 없음.

```
mat <- matrix(1:12, 3, 4) # matrix() 함수의 처음 세 인자가 data, nrow, ncol 이므로 필요한 값만 입력.
```

```
colnames(mat) <- c('A', 'B', 'C', 'D') # 열이름 덮어쓰기
```

```
rownames(mat) <- c('가', '나', '다') # 행이름 덮어쓰기
```

- 하다 보면 쉬운 길을 찾아가게 됨.
- 1:12는 seq(from = 1, to = 12, by = 1) 과 같음.
- <-는 =과 같음.

```
mat
```

```
##  A B C D
## 가 1 4 7 10
## 나 2 5 8 11
## 다 3 6 9 12
```

```
class(mat)
```

```
## [1] "matrix" "array"
```

6) 데이터프레임

```
윤하 <- as.data.frame(Yunha)
```

```
matthew <- as.data.frame(mat)
```

- as.data.frame() 함수는 벡터, 행렬 등을 인자로 받음.

```
yunha
```

```
## Yunha
## 1 Password
## 2 4
## 3 8
## 4 6
```

```
class(yunha)
```

```
## [1] "data.frame"
```

- 데이터프레임은 row = observation, column = variable 개념이라, 열 이름을 웬만하면 채우려고 하는데, 여기서 character vector 이름인 Yunha를 차용한 모습.
- 벡터 이름 : 벡터 내용 = 변수 이름 : 관측치 느낌으로 해석한 듯.

```
matthew
```

```
## A B C D
## 가 1 4 7 10
## 나 2 5 8 11
## 다 3 6 9 12
```

```
class(matthew)
```

```
## [1] "data.frame"
```

7) 인덱싱

```
Yunha[2] # Yunha 벡터의 두 번째 요소
```

```
## [1] "4"
```

```
mat[, 1] # mat 행렬의 첫 열
```

```
## 가 나 다
## 1 2 3
```

```
mat[2, 2] # mat 행렬의 (2, 2) 요소
```

```
## [1] 5
```



```
matthew$A # matthew 데이터프레임의 변수 A에 속한 값
```

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

```
matthew[2, 2] # matthew 데이터프레임의 (2, 2) 요소
```

```
## [1] 5
```

- 데이터프레임의 인덱싱이 조금 더 깔끔한 느낌.
- 실제 작업에도 용이한바, R의 데이터분석은 대개 데이터프레임 형태를 사용.
- 이를 Python에서 구현하기 위해 Pandas 라이브러리를 만듦.

4. Others

주석처리

- 코드 설명 작성 : `print(a)` # a를 출력한다
- 해당 부분을 실행에서 제외 : `#print(a)`

업데이트 R, RStudio 모두 꾸준히 새 버전이 나오지만, 구 버전을 사용해도 큰 문제 없음

- R은 새로 설치하고 구 버전을 삭제하는 게 제일 간편. RStudio는 알아서 최신의 R을 인식해 사용.
- RStudio는 Help → Check for Updates 기능을 활용.
- 패키지는 Tools → Check for Package Updates

5. Quiz

초급 다음의 행렬을 만들어보자

```
##      var1 var2 var3
## Case #1  12  21  32
## Case #2  17  22  34
## Case #3  19  25  35
```

중급 `datasets::iris` 데이터를 가져와 다음을 해결해보자

- 1) iris 변수의 이름을 `names()` 함수로 확인하라.
- 2) iris 관찰값, 변수의 개수를 `dim()`, `nrow()`, `length()` 함수로 확인하라.
- 3) iris 처음 세 줄과 마지막 세 줄을 `head()`, `tail()` 함수로 출력하라.

고급 다음 코드의 문제점을 지적해보자. 수정본을 참고해도 좋다.

```
I-DLE_MEMBERS <- c('소연', '미연', '민니', '우기', '슈화')
```

```
I-DLE_LYLICS <- c('Look at you 넌 못 감당해 날',  
  'I got to drink up now 네가 싫다 해도 좋아',  
  'Why are you cranky, boy? 뭘 그리 찡그려 너',  
  '미친 연이라 말해 What's the loss to me ya',  
  '사랑 그깟 거 따위 내 몸에 상처 하나도 어림없지',  
  'Ye I'm a Tomboy (Umm ah umm)',  
  'Ye I'll be the Tomboy (Umm ah)')
```

```
TOMBOY <- data.frame(I-DLE_MEMBERS, I-DLE_LYLICS)
```

```
## Error: <text>:7:42: 예상하지 못한 기호(symbol)입니다.
```

```
## 6:      'Why are you cranky, boy? 뭘 그리 찡그려 너',
```

```
## 7:      '미친 연이라 말해 What's
```

```
##      ^
```

```
IDLE_MEMBERS <- c('소연', '미연', '민니', '우기', '슈화')
```

```
IDLE_LYLICS <- c('Look at you 넌 못 감당해 날',  
  'I got to drink up now 네가 싫다 해도 좋아',  
  'Why are you cranky, boy? 뭘 그리 찡그려 너',  
  "미친 연이라 말해 What's the loss to me ya",  
  '사랑 그깟 거 따위 내 몸에 상처 하나도 어림없지',  
  "Ye I'm a Tomboy (Umm ah umm)",  
  "Ye I'll be the Tomboy (Umm ah)")
```

```
TOMBOY <- data.frame(MEMBERS = IDLE_MEMBERS[c(3, 4, 1, 2, 4, 3, 5)],  
  LYRICS = IDLE_LYLICS)
```

```
TOMBOY
```

```
##  MEMBERS      LYRICS  
## 1  민니      Look at you 넌 못 감당해 날  
## 2  우기      I got to drink up now 네가 싫다 해도 좋아  
## 3  소연      Why are you cranky, boy? 뭘 그리 찡그려 너  
## 4  미연      미친 연이라 말해 What's the loss to me ya
```

```
## 5 우기 사랑 그깟 거 따위 내 몸에 상처 하나도 어림없지
## 6 민니          Ye I'm a Tomboy (Umm ah umm)
## 7 슈화          Ye I'll be the Tomboy (Umm ah)
```

Day 2

R과 통계분석 (Tidyverse 활용) p.102~146 파이썬 머신러닝 판다스 데이터 분석 p.2~55

1. What is Tidyverse

Package for Packages 데이터 분석을 위한 패키지의 모음집

```
install.packages('tidyverse') # library()와는 달리 따옴표를 붙여줘야 함, 이하에선 인스톨 생략
```

```
library(tidyverse) # tidyverse 패키지 장착
```

```
## -- Attaching packages ----- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5    v purrr  0.3.4
```

```
## v tibble  3.1.6    v dplyr  1.0.8
```

```
## v tidyr   1.2.0    v stringr 1.4.0
```

```
## v readr   2.1.2    v forcats 0.5.1
```

```
## -- Conflicts ----- tidyverse_conflicts() --
```

```
## x dplyr::filter() masks stats::filter()
```

```
## x dplyr::lag()    masks stats::lag()
```

- ggplot2(시각화), dplyr(조작), stringr(텍스트), tibble(데이터프레임) 등의 패키지를 한 번에 로드, 장착할 수 있음.
- Conflicts란에 제시되는 건 이미 라이브러리를 장착해 사용 중인 함수와 이름이 겹치는 케이스.
- 이제 filter() 함수는 stats::filter()가 아니라 dplyr::mutate()를 우선 선택.
- stats::filter() 식으로 풀네임을 써야 해당 함수 사용 가능.

And then, and then 직관적인 연결 작업

- tidyverse 계열 함수들은 첫 번째 입력값을 data로 통일. function(data, actions) 개념.
- pipe operator %>%는 앞선 작업 결과물을 다음 함수의 첫 요소로 대입하는 기능
- x %>% f(y) → f(x, y) 식으로 결합 사용하는 게 일반적.

2. Exercise : 보도자료 만들기

2022년 2월 PSI https://www.kiet.re.kr/kiet_web/?sub_num=1503&state=view&idx=59127&ord=0

1) 데이터 로드

```
library(openxlsx) # openxlsx 패키지 장착
```

```
read.xlsx('PSI 연습용.xlsx')
```

```
## 연도 월      구분 응답수 경기현황 경기전망 시장판매현황
## 1 2022 2      00_전체  235 96.17021 110.63830 102.55319
## 2 2022 2      01_ICT  86 94.18605 110.46512 100.00000
## 3 2022 2      02_장비  76 102.63158 110.52632 107.89474
## 4 2022 2      03_소재  52 82.69231 111.53846 92.30769
## 5 2022 2      04_전자  42 109.52381 135.71429 109.52381
## 6 2022 2      05_가전  16 106.25000 143.75000 100.00000
## 7 2022 2      06_핸드폰 26 111.53846 130.76923 115.38462
## 8 2022 2      07_디스플레이 21 85.71429 90.47619 76.19048
## 9 2022 2      08_반도체 23 73.91304 82.60870 104.34783
## 10 2022 2      09_기계  22 100.00000 100.00000 100.00000
## 11 2022 2      10_자동차 34 102.94118 120.58824 114.70588
## 12 2022 2      11_조선  20 105.00000 105.00000 105.00000
## 13 2022 2      12_섬유  16 75.00000 112.50000 81.25000
## 14 2022 2      13_철강  14 100.00000 121.42857 100.00000
## 15 2022 2      14_화학  22 77.27273 104.54545 95.45455
## 16 2022 2      15_바이오헬스 21 114.28571 109.52381 119.04762
## 17 2022 2      16_애널리스트 105 93.33333 119.04762 103.80952
## 18 2022 2      17_공공기관및기타 130 98.46154 103.84615 101.53846
## 19 2022 2      18_애널리스트 105 93.33333 119.04762 103.80952
## 20 2022 2      19_공공기관  85 97.64706 104.70588 102.35294
## 21 2022 2      20_기타  45 100.00000 102.22222 100.00000
## 시장판매전망 수출현황 수출전망 생산수준현황 생산수준전망 재고수준현황
## 1 114.46809 106.38298 122.9787 105.95745 122.5532 107.65957
## 2 112.79070 110.46512 127.9070 112.79070 125.5814 96.51163
## 3 113.15789 118.42105 125.0000 107.89474 127.6316 128.94737
## 4 121.15385 82.69231 119.2308 88.46154 115.3846 96.15385
## 5 128.57143 133.33333 138.0952 123.80952 123.8095 102.38095
```

## 6	118.75000	112.50000	125.0000	112.50000	112.5000	100.00000
## 7	134.61538	146.15385	146.1538	130.76923	130.7692	103.84615
## 8	90.47619	80.95238	109.5238	80.95238	123.8095	80.95238
## 9	104.34783	95.65217	126.0870	121.73913	130.4348	100.00000
## 10	100.00000	113.63636	118.1818	95.45455	113.6364	104.54545
## 11	126.47059	120.58824	129.4118	105.88235	126.4706	167.64706
## 12	105.00000	120.00000	125.0000	125.00000	145.0000	90.00000
## 13	112.50000	87.50000	125.0000	87.50000	118.7500	93.75000
## 14	135.71429	85.71429	128.5714	71.42857	107.1429	100.00000
## 15	118.18182	77.27273	109.0909	100.00000	118.1818	95.45455
## 16	109.52381	104.76190	104.7619	114.28571	109.5238	104.76190
## 17	121.90476	107.61905	132.3810	110.47619	137.1429	107.61905
## 18	108.46154	105.38462	115.3846	102.30769	110.7692	107.69231
## 19	121.90476	107.61905	132.3810	110.47619	137.1429	107.61905
## 20	110.58824	108.23529	118.8235	101.17647	111.7647	110.58824
## 21	104.44444	100.00000	108.8889	104.44444	108.8889	102.22222
##	재고수준전망 신규수주현황 신규수주전망 투자액현황 투자액전망 채산성현황					
## 1	109.78723	104.87805	100.00000	108.93617	115.7447	86.38298
## 2	98.83721	NA	NA	101.16279	112.7907	87.20930
## 3	131.57895	90.00000	100.00000	115.78947	122.3684	98.68421
## 4	98.07692	NA	NA	105.76923	109.6154	61.53846
## 5	102.38095	NA	NA	102.38095	102.3810	107.14286
## 6	93.75000	NA	NA	100.00000	100.0000	112.50000
## 7	107.69231	NA	NA	103.84615	103.8462	103.84615
## 8	85.71429	NA	NA	90.47619	114.2857	76.19048
## 9	104.34783	NA	NA	108.69565	130.4348	60.86957
## 10	104.54545	NA	NA	95.45455	100.0000	77.27273
## 11	167.64706	NA	NA	132.35294	141.1765	108.82353
## 12	100.00000	90.00000	100.00000	110.00000	115.0000	105.00000
## 13	93.75000	NA	NA	87.50000	93.7500	62.50000
## 14	100.00000	NA	NA	128.57143	128.5714	78.57143
## 15	100.00000	NA	NA	104.54545	109.0909	50.00000
## 16	104.76190	119.04762	100.00000	123.80952	119.0476	100.00000
## 17	114.28571	93.33333	100.00000	109.52381	123.8095	81.90476
## 18	106.15385	111.53846	100.00000	108.46154	109.2308	90.00000
## 19	114.28571	93.33333	100.00000	109.52381	123.8095	81.90476

```
## 20 110.58824 113.33333 106.66667 107.05882 109.4118 88.23529
## 21 97.77778 109.09091 90.90909 111.11111 108.8889 93.33333
## 채산성전망
## 1 97.44681
## 2 101.16279
## 3 101.31579
## 4 82.69231
## 5 126.19048
## 6 125.00000
## 7 126.92308
## 8 76.19048
## 9 78.26087
## 10 77.27273
## 11 105.88235
## 12 120.00000
## 13 62.50000
## 14 92.85714
## 15 90.90909
## 16 104.76190
## 17 103.80952
## 18 92.30769
## 19 103.80952
## 20 89.41176
## 21 97.77778
```

```
PSI_ORIGINAL <- read.xlsx('PSI 연습용.xlsx') %>%
```

```
tibble() # 엑셀 파일을 읽어서 PSI_ORIGINAL 이라 명명.
```

```
PSI_ORIGINAL # tibble을 사용하니, 직전보단 깔끔하게 저장된다. class 표시는 덤.
```

```
## # A tibble: 21 x 20
```

```
## 연도 월 구분 응답수 경기현황 경기전망 시장판매현황 시장판매전망 수출현황
```

```
## <dbl> <dbl> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
```

```
## 1 2022 2 00_~ 235 96.2 111. 103. 114. 106.
```

```
## 2 2022 2 01_~ 86 94.2 110. 100 113. 110.
```

```
## 3 2022 2 02_~ 76 103. 111. 108. 113. 118.
```

```
## 4 2022 2 03_~ 52 82.7 112. 92.3 121. 82.7
```

```
## 5 2022 2 04_~ 42 110. 136. 110. 129. 133.
## 6 2022 2 05_~ 16 106. 144. 100 119. 112.
## 7 2022 2 06_~ 26 112. 131. 115. 135. 146.
## 8 2022 2 07_~ 21 85.7 90.5 76.2 90.5 81.0
## 9 2022 2 08_~ 23 73.9 82.6 104. 104. 95.7
## 10 2022 2 09_~ 22 100 100 100 100 114.
## # ... with 11 more rows, and 11 more variables: 수출전망 <dbl>,
## # 생산수준현황 <dbl>, 생산수준전망 <dbl>, 재고수준현황 <dbl>,
## # 재고수준전망 <dbl>, 신규수주현황 <dbl>, 신규수주전망 <dbl>,
## # 투자액현황 <dbl>, 투자액전망 <dbl>, 채산성현황 <dbl>, 채산성전망 <dbl>
```

- read.xlsx() 함수의 첫 인자는 xlsxFilE 으로, 해당 파일의 경로를 입력해야 함.
- 정확한 경로와 이름을 적으면서, 확장자 명을 꼭 포함할 것.
- 두 번째 인자는 sheetName, 미지정 시 첫 번째 시트를 로드한다. 다중 시트로 이뤄진 엑셀 파일이라면 시트 이름을 지정해주는 게 유용.
- 데이터 프레임을 가져올 땐 tibble() 함수를 거쳐서 단정하게 만들자.
- 여담으로 xlsxFilE, sheetName 처럼 영문 가운데 capital을 섞어주는 걸 camel 표기법이라 함.

2) 훑어보기

```
PSI_ORIGINAL %>% head() # 상위 순번 관측값만 표시, default = 6개
```

```
## # A tibble: 6 x 20
## 연도 월 구분 응답수 경기현황 경기전망 시장판매현황 시장판매전망 수출현황
## <dbl> <dbl> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 2022 2 00_전~ 235 96.2 111. 103. 114. 106.
## 2 2022 2 01_ICT 86 94.2 110. 100 113. 110.
## 3 2022 2 02_장~ 76 103. 111. 108. 113. 118.
## 4 2022 2 03_소~ 52 82.7 112. 92.3 121. 82.7
## 5 2022 2 04_전~ 42 110. 136. 110. 129. 133.
## 6 2022 2 05_가~ 16 106. 144. 100 119. 112.
## # ... with 11 more variables: 수출전망 <dbl>, 생산수준현황 <dbl>,
## # 생산수준전망 <dbl>, 재고수준현황 <dbl>, 재고수준전망 <dbl>,
## # 신규수주현황 <dbl>, 신규수주전망 <dbl>, 투자액현황 <dbl>, 투자액전망 <dbl>,
## # 채산성현황 <dbl>, 채산성전망 <dbl>
```

```
PSI_ORIGINAL %>% names() # 변수 이름
```

```
## [1] "연도"      "월"        "구분"      "응답수"    "경기현황"  
## [6] "경기전망"  "시장판매현황" "시장판매전망" "수출현황"  "수출전망"  
## [11] "생산수준현황" "생산수준전망" "재고수준현황" "재고수준전망" "신규수주현황"  
## [16] "신규수주전망" "투자액현황"  "투자액전망"  "채산성현황" "채산성전망"
```

```
PSI_ORIGINAL %>% dim() # dimension
```

```
## [1] 21 20
```

```
PSI_ORIGINAL$구분 # 데이터 '구분' 열에 담긴 관측값들
```

```
## [1] "00_전체"      "01_ICT"      "02_장비"  
## [4] "03_소재"      "04_전자"      "05_가전"  
## [7] "06_핸드폰"    "07_디스플레이" "08_반도체"  
## [10] "09_기계"      "10_자동차"    "11_조선"  
## [13] "12_섬유"      "13_철강"      "14_화학"  
## [16] "15_바이오헬스" "16_애널리스트" "17_공공기관및기타"  
## [19] "18_애널리스트" "19_공공기관"  "20_기타"
```

```
PSI_ORIGINAL[2, ] # 데이터 2열 관측값
```

```
## # A tibble: 1 x 20  
##   연도 월 구분 응답수 경기현황 경기전망 시장판매현황 시장판매전망 수출현황  
##   <dbl> <dbl> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 2022 2 01_ICT 86 94.2 110. 100 113. 110.  
## # ... with 11 more variables: 수출전망 <dbl>, 생산수준현황 <dbl>,  
## # 생산수준전망 <dbl>, 재고수준현황 <dbl>, 재고수준전망 <dbl>,  
## # 신규수주현황 <dbl>, 신규수주전망 <dbl>, 투자액현황 <dbl>, 투자액전망 <dbl>,  
## # 채산성현황 <dbl>, 채산성전망 <dbl>
```

3) 응답자 비율 구하기

```
PSI_ORIGINAL %>% # 원본 데이터에서  
  select(구분, 응답수) %>% # 구분, 응답수 열을 고른 다음  
  mutate(비율 = 응답수/235*100) # 새로운 열을 추가하는데, 이는 응답수/235*100 계산값임.
```

```
## # A tibble: 21 x 3  
##   구분      응답수 비율
```



```
##   <chr>      <dbl> <dbl>
## 1 00_전체      235 100
## 2 01_ICT       86 36.6
## 3 02_장비      76 32.3
## 4 03_소재      52 22.1
## 5 04_전자      42 17.9
## 6 05_가전      16  6.81
## 7 06_핸드폰    26 11.1
## 8 07_디스플레이 21  8.94
## 9 08_반도체    23  9.79
## 10 09_기계     22  9.36
## # ... with 11 more rows
```

- `select(data_table, columns, ...)` 원하는 column 고르기. 위치 인덱스나 변수 명을 적으면 됨.
- `select(-1)` 식으로 쓰면, 첫 번째 열을 제외하고 모두 선택.
- 변수 명에 띄어쓰기가 있다면, 해당 이름 전체를 따옴표나 accent 기호로 감싸야 함.
- `mutate(data_table, var_name = new_data, ...)` 새로운 열 만들기(덮어쓰기 가능).

```
PSI_RATE.1 <- PSI_ORIGINAL %>%
  select(구분, 응답수) %>%
  mutate(비율 = 응답수/235*100) # 앞서 만든 데이터를 하나의 객체로 저장해두자.

PSI_RATE.2 <- PSI_RATE.1 %>% # 아까 하던 거에서
  mutate(비율 = round(비율, digits = 1)) %>% # 비율 값은 반올림해서 덮어씌우고
  slice(1:18) # 필요한 row만 선택하자.

PSI_RATE.2 # 좋은 한데, 보도자료는 업종 순서가 달라서 붙여넣기가 애매.
```

```
## # A tibble: 18 x 3
##   구분      응답수 비율
##   <chr>      <dbl> <dbl>
## 1 00_전체      235 100
## 2 01_ICT       86 36.6
## 3 02_장비      76 32.3
## 4 03_소재      52 22.1
## 5 04_전자      42 17.9
## 6 05_가전      16  6.8
```

```
## 7 06_핸드폰      26 11.1
## 8 07_디스플레이   21  8.9
## 9 08_반도체       23  9.8
## 10 09_기계        22  9.4
## 11 10_자동차      34 14.5
## 12 11_조선        20  8.5
## 13 12_섬유        16  6.8
## 14 13_철강        14  6
## 15 14_화학        22  9.4
## 16 15_바이오헬스  21  8.9
## 17 16_애널리스트 105 44.7
## 18 17_공공기관및기타 130 55.3
```

- slice(data_table, rows, ...) 원하는 row 고르기.

```
PSI_RATE.2[c(2, 3, 4, 9, 8, 5, 7, 6, 11, 12, 10, 15, 14, 13, 16, 17, 18), ] # 순서를 바꾸면 되지
```

```
## # A tibble: 17 x 3
##   구분      응답수 비율
##   <chr>      <dbl> <dbl>
## 1 01_ICT      86 36.6
## 2 02_장비     76 32.3
## 3 03_소재     52 22.1
## 4 08_반도체   23  9.8
## 5 07_디스플레이  21  8.9
## 6 04_전자    42 17.9
## 7 06_핸드폰   26 11.1
## 8 05_가전    16  6.8
## 9 10_자동차   34 14.5
## 10 11_조선    20  8.5
## 11 09_기계    22  9.4
## 12 14_화학    22  9.4
## 13 13_철강    14  6
## 14 12_섬유    16  6.8
## 15 15_바이오헬스  21  8.9
## 16 16_애널리스트 105 44.7
## 17 17_공공기관및기타 130 55.3
```

```
APPENDIX.1 <- PSI_RATE.2[c(2, 3, 4, 9, 8, 5, 7, 6, 11, 12, 10, 15, 14, 13, 17, 18), ] # 완성품 저장
```

- 완성품, 중간 작업물은 네이밍 규칙을 달리 가져가는 게 편하다.
- 객체를 왕창 만들다보면 헛갈릴 일이 생기기 마련.

4) 기상도 만들기

```
PSI_WEATHER.1 <- PSI_ORIGINAL %>%
  select(3, '경기현황', '시장판매현황', 수출현황,
    생산수준현황, 투자액현황, 채산성현황) # 인덱스, '변수명', '변수명', 변수명 모두 가능
```

```
PSI_WEATHER.1
```

```
## # A tibble: 21 x 7
##   구분   경기현황 시장판매현황 수출현황 생산수준현황 투자액현황 채산성현황
##   <chr>   <dbl>   <dbl> <dbl>   <dbl>   <dbl>   <dbl>
## 1 00_전체    96.2    103.   106.    106.    109.    86.4
## 2 01_ICT     94.2    100.   110.    113.    101.    87.2
## 3 02_장비    103.    108.   118.    108.    116.    98.7
## 4 03_소재    82.7    92.3   82.7    88.5    106.    61.5
## 5 04_전자    110.    110.   133.    124.    102.    107.
## 6 05_가전    106.    100.   112.    112.    100.    112.
## 7 06_핸드폰  112.    115.   146.    131.    104.    104.
## 8 07_디스플레이 85.7    76.2   81.0    81.0    90.5    76.2
## 9 08_반도체    73.9    104.   95.7    122.    109.    60.9
## 10 09_기계    100     100   114.    95.5    95.5    77.3
## # ... with 11 more rows
```

```
PSI_WEATHER.2 <- PSI_WEATHER.1 %>%
  mutate_at(vars = 2:7, .funs = round) # 2:7열 관측값에 반올림 적용
```

```
PSI_WEATHER.2
```

```
## # A tibble: 21 x 7
##   구분   경기현황 시장판매현황 수출현황 생산수준현황 투자액현황 채산성현황
##   <chr>   <dbl>   <dbl> <dbl>   <dbl>   <dbl>   <dbl>
## 1 00_전체    96     103   106     106     109     86
## 2 01_ICT     94     100   110     113     101     87
```

```
## 3 02_장비      103      108      118      108      116      99
## 4 03_소재      83       92      83       88      106      62
## 5 04_전자      110      110      133      124      102      107
## 6 05_가전      106      100      112      112      100      112
## 7 06_핸드폰    112      115      146      131      104      104
## 8 07_디스플레이 86       76      81       81      90       76
## 9 08_반도체     74      104      96      122      109      61
## 10 09_기계     100      100      114      95      95      77
## # ... with 11 more rows
```

- mutate_at(data_table, .vars, .funs, ...) 는 mutate()의 고급 버전.
- .funs에 입력한 함수를 .vars에 입력한 변수에 일괄 적용하고 덮어씌움.

```
PSI_WEATHER.3 <- PSI_WEATHER.2[c(1, 9, 8, 5, 7, 6, 11, 12, 10, 15, 14, 13, 16, 2, 3, 4), ]
```

```
colnames(PSI_WEATHER.3) <- c('구분', '업황', '내수', '수출', '생산수준', '투자액', '채산성')
```

```
PSI_WEATHER.3 # 갖다 붙이려 했는데, 숫자가 묘하게 다름
```

```
## # A tibble: 16 x 7
##   구분      업황 내수 수출 생산수준 투자액 채산성
##   <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 00_전체      96  103  106    106  109    86
## 2 08_반도체     74  104   96    122  109    61
## 3 07_디스플레이 86   76   81     81   90    76
## 4 04_전자     110  110  133    124  102   107
## 5 06_핸드폰    112  115  146    131  104   104
## 6 05_가전     106  100  112    112  100   112
## 7 10_자동차     103  115  121    106  132   109
## 8 11_조선     105  105  120    125  110   105
## 9 09_기계     100  100  114     95   95    77
## 10 14_화학      77   95   77    100  105    50
## 11 13_철강     100  100   86     71  129    79
## 12 12_섬유      75   81   88     88   88    62
## 13 15_바이오헬스 114  119  105    114  124   100
## 14 01_ICT      94  100  110    113  101    87
## 15 02_장비     103  108  118    108  116    99
## 16 03_소재      83   92   83     88  106    62
```

- R의 round() 함수는 round(0.5) = 0 으로 계산해버림.
- 그럼 어떻게 해야할까. 뭐 구글 가야죠.

```
round2 = function(x, n = 0) {
  posneg = sign(x)
  z = abs(x)*10^n
  z = z + 0.5 + sqrt(.Machine$double.eps)
  z = trunc(z)
  z = z/10^n
  z*posneg
} # 구글 멧쟁이가 만들어둔 함수를 그대로 긁어오자

round(0.5) ; round2(0.5, 0) # 두 번째 인자 n은 digits
```

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

```
## [1] 1
```

- 아래 mutate_at() 함수에서 오류가 나서, n의 default 값만 function(x, n = 0) 으로 수정하자.
- tidyverse 함수에선 직전 작업 값 '.' 으로 표현한다. .\$구분 으로 적어도 해당 흐름 내에선 '작업 중인' 데이터프레임의 구분 열' 줌으로 인식하는 셈.
- .vars, .funs도 유사한 맥락의 네이밍인데, 설명하는 건 강의 레벨을 벗어나므로 스킵.

```
PSI_WEATHER.2 <- PSI_WEATHER.1 %>%
  mutate_at(.vars = 2:7, .funs = round2) # 반올림 함수 교체

PSI_WEATHER.3 <- PSI_WEATHER.2[c(1, 9, 8, 5, 7, 6, 11, 12, 10, 15, 14, 13, 16, 2, 3, 4), ]

colnames(PSI_WEATHER.3) <- c('구분', '업황', '내수', '수출', '생산수준', '투자액', '채산성')

PSI_WEATHER.3
```

```
## # A tibble: 16 x 7
##   구분      업황 내수 수출 생산수준 투자액 채산성
##   <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 00_전체      96  103  106    106  109    86
## 2 08_반도체      74  104   96    122  109    61
## 3 07_디스플레이  86   76   81     81   90    76
## 4 04_전자     110  110  133    124  102   107
```

```
## 5 06_핸드폰      112 115 146   131 104 104
## 6 05_가전       106 100 113   113 100 113
## 7 10_자동차     103 115 121   106 132 109
## 8 11_조선       105 105 120   125 110 105
## 9 09_기계       100 100 114    95  95  77
## 10 14_화학      77  95  77   100 105  50
## 11 13_철강     100 100  86    71 129  79
## 12 12_섬유      75  81  88    88  88  63
## 13 15_바이오헬스 114 119 105   114 124 100
## 14 01_ICT      94 100 110   113 101  87
## 15 02_장비     103 108 118   108 116  99
## 16 03_소재      83  92  83    88 106  62
```

APPENDIX.2 <- PSI_WEATHER.3 # 완성품 저장

- PSI_WEATER.2, PSI_WEATHER.3 만드는 코드를 고치자.
- 객체 네이밍을 순차적으로 했어서, 수정할 부분을 빠르게 찾고, 해당 파트만 교체하는 게 어렵지 않다.

5) 엑셀 출력하기

```
write.xlsx(x = list(APPENDIX.1, APPENDIX.2), # 완성품 두 개를
  sheetName = c('응답자 비율', '기상도'), # 각각의 시트로 갖는
  file = 'PSI 보도자료.xlsx') # 엑셀 파일 생성
```

- 작업 중인 폴더에 엑셀 파일이 생성된다.
- list 클래스는 최상위 레벨로, 앞서 다룬 벡터, 행렬, 데이터프레임 등을 원소로 가질 수 있음.
- 가끔 list를 입력 인자로 요구하는 함수가 있으니 알아두자.

3. Advanced : 미국 수출데이터 가공하기

```
library(tidyverse)

USTRADE <- read_csv('미국수출 연습용.csv')
```

- UN Commodity Trade 자료의 일부.

USTRADE # read_csv() 사용하면 tibble 형태로 저장

```
## # A tibble: 82,487 x 35
```

```
## Classification Year Period `Period Desc.` `Aggregate Level` `Is Leaf Code`
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 H5 2020 2020 2020 6 1
## 2 H5 2020 2020 2020 6 1
## 3 H5 2020 2020 2020 6 1
## 4 H5 2020 2020 2020 6 1
## 5 H5 2020 2020 2020 6 1
## 6 H5 2020 2020 2020 6 1
## 7 H5 2020 2020 2020 6 1
## 8 H5 2020 2020 2020 6 1
## 9 H5 2020 2020 2020 6 1
## 10 H5 2020 2020 2020 6 1
## # ... with 82,477 more rows, and 29 more variables: `Trade Flow Code` <dbl>,
## # `Trade Flow` <chr>, `Reporter Code` <dbl>, Reporter <chr>,
## # `Reporter ISO` <chr>, `Partner Code` <dbl>, Partner <chr>,
## # `Partner ISO` <chr>, `2nd Partner Code` <lgl>, `2nd Partner` <lgl>,
## # `2nd Partner ISO` <lgl>, `Customs Proc. Code` <lgl>, Customs <lgl>,
## # `Mode of Transport Code` <lgl>, `Mode of Transport` <lgl>,
## # `Commodity Code` <dbl>, Commodity <chr>, `Qty Unit Code` <dbl>, ...
```

```
USTRADE.1 <- USTRADE %>%
```

```
  select(Year, `Trade Flow Code`, `Trade Flow`, `Reporter Code`,
         Reporter, `Partner Code`, Partner, `Partner ISO`,
         `Commodity Code`, Commodity, `Qty Unit Code`, `Qty Unit`,
         Qty, `Trade Value (US$)`) # 필요 변수 선택
```

```
USTRADE.1 %>% dim() # 변수 35개 -> 14개
```

```
## [1] 82487 14
```

- 띄어쓰기가 포함된 변수 명은 앞 뒤를 '' 내지 "로 감싸야 함.

```
library(haven) # dta 로드 패키지
```

```
LINK <- read_dta('연계표 연습용.dta') %>%
```

```
  select(hsc, ksic5) # 국제무역코드(hscode), 한국표준산업분류(ksic5)만 선택
```

- STATA를 사용하면 dta 파일로 작업물을 공유하는 경우가 있음.
- R에서는 해당 파일을 haven 패키지를 사용해 입출력.

- 비슷한 방식으로 타 프로그램 사용자와 co-work 가능.

LINK

```
## # A tibble: 6,564 x 2
##   hsc ksic5
##   <dbl> <dbl>
## 1  223   NA
## 2  284   NA
## 3  285   NA
## 4  286   NA
## 5  287   NA
## 6  288   NA
## 7  289   NA
## 8  290   NA
## 9 10110 1291
## 10 10111 1291
## # ... with 6,554 more rows
```

LINK.1 <- LINK %>%

filter(!is.na(ksic5)) # 한국 분류로의 매칭이 목적이니, ksic5 공란 케이스는 제외

LINK.1 # number of rows가 6564 -> 6552로 축소

```
## # A tibble: 6,552 x 2
##   hsc ksic5
##   <dbl> <dbl>
## 1 10110 1291
## 2 10111 1291
## 3 10119 1291
## 4 10120 1291
## 5 10121 1291
## 6 10129 1291
## 7 10130 1291
## 8 10190 1291
## 9 10210 1211
## 10 10221 1211
## # ... with 6,542 more rows
```


- filter(data_table, condition, ...) 함수는 조건에 맞는 observation만 선택.
- is.na() 함수는 결측치, 즉 NA인 경우 TRUE를 출력.
- ! 연산자는 부정, not을 의미.
- 따라서 observation인데, ksic5 변수가 NA인 경우는 배제하고 LINK.1에 저장하겠다는 것.

```
MANUFACTURE_KOREA <- seq(10, 34) # ksic5에서 제조업 파트는 10~34로 시작

temp1 <- rep(0, nrow(LINK.1)) # 영벡터

for (i in seq_along(MANUFACTURE_KOREA)) { # i = 1, 2, ..., 25에 대해 다음 작업 반복

  temp2 <- str_starts(string = as.character(LINK.1$ksic5), # ksic5가
                      pattern = as.character(MANUFACTURE_KOREA[i])) # 제조업 파트로 시작하면 TRUE

  temp1 <- temp1 + temp2 # 사칙연산에서 TRUE = 1이므로, 영벡터에 1 남기게 됨.

}

temp1 %>% head() # temp1은 제조업인 경우 1이 채워진 벡터
```

```
## [1] 1 1 1 1 1 1
```

```
temp1 %>% sum() # 6413개 observation이 제조업에 해당
```

```
## [1] 6413
```

```
TESTER <- temp1 %>% as.logical() # 논리값으로 변경
```

```
TESTER %>% head() # 1은 다시 TRUE로 변환
```

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

- seq_along(vector) 함수는 1:length(vector) 수열을 출력.
- 논리값은 숫자로 변환하면 0, 1이므로 sum()과 궁합이 좋음.

```
MANUFACTURE_WORLD <- LINK.1[TESTER, ]$hsc # 논리값을 인덱싱에 활용
```

```
MANUFACTURE_WORLD %>% head() # 한국 분류로 제조업에 해당하는 hscode
```

```
## [1] 10110 10111 10119 10120 10121 10129
```

```

USTRADE.2 <- USTRADE.1 %>%
  filter(`Commodity Code` %in% MANUFACTURE_WORLD) %>% # 제조업 데이터만 선택
  arrange(Year, `Commodity Code`) %>% # 연도 순, HS code 순으로 정리
  relocate(Year, `Commodity Code`,
            Commodity, `Trade Value (US$)`) # 주요 변수 앞으로 이동

write_csv(USTRADE.2, '미국수출 연계작업.csv') # 정리한 데이터셋을 csv로 저장

```

- `x %in% y`는 `y` vector 내에 `x`라는 인수가 있다면 TRUE를 반환.
- 변수, 객체 이름을 잘 지어야 헛갈리지 않음.

4. Quiz

초급 datasets::mtcars 데이터를 가져와 다음을 해결해보자.

- 1) 차 이름을 새 변수로 설정해 표시하자. `rownames_to_column(var = 'car')` 함수를 사용하라.
- 2) 방금 만든 `car`, 그리고 `mpg(miles per gallon)`, `cyl(number of cylinders)`, `hp(gross horsepower)` 변수만을 선택하라.
- 3) 실린더 당 마력을 계산해 새 변수로 나타내라.

1, 2, 3)

```

CAR.1 <- datasets::mtcars %>%
  rownames_to_column(var = 'car') %>%
  tibble() %>%
  select(car, mpg, cyl, hp) %>%
  mutate(hp_per_cyl = hp/cyl)

```

CAR.1

```

## # A tibble: 32 x 5
##   car      mpg  cyl  hp hp_per_cyl
##   <chr>    <dbl> <dbl> <dbl>    <dbl>
## 1 Mazda RX4      21    6  110     18.3
## 2 Mazda RX4 Wag  21    6  110     18.3
## 3 Datsun 710     22.8   4   93     23.2
## 4 Hornet 4 Drive  21.4   6  110     18.3
## 5 Hornet Sportabout 18.7   8  175     21.9

```

```
## 6 Valiant      18.1  6 105   17.5
## 7 Duster 360   14.3  8 245   30.6
## 8 Merc 240D    24.4  4  62   15.5
## 9 Merc 230     22.8  4  95   23.8
## 10 Merc 280    19.2  6 123   20.5
## # ... with 22 more rows
```

중급 다음 작업을 이어서 해보자.

- 1) 1974년 데이터인데, 이 때도 차는 벤츠가 짝이었나보다. 차 이름에 Merc 문자열을 포함한 관측값은 몇 개인가. `str_detect(pattern = 'Merc')` 함수를 사용하라.
- 2) 벤츠 차량 데이터만을 추려보자. `str_detect()` 의 결과값(논리값)을 인덱싱에 사용하라.
- 3) 벤츠 차량의 평균 마력은 얼마인가.

```
# 1)

CAR.1$car %>%
  str_detect('Merc') %>%
  sum() # 논리값은 단순 덧셈하면 TRUE = 1, FALSE = 0 으로 계산된다.
```

```
## [1] 7
```

```
# 2)

CAR.2 <- CAR.1[CAR.1$car %>% str_detect('Merc'), ]

CAR.2
```

```
## # A tibble: 7 x 5
##   car      mpg  cyl  hp hp_per_cyl
##   <chr>   <dbl> <dbl> <dbl>   <dbl>
## 1 Merc 240D  24.4   4   62    15.5
## 2 Merc 230   22.8   4   95    23.8
## 3 Merc 280   19.2   6  123    20.5
## 4 Merc 280C  17.8   6  123    20.5
## 5 Merc 450SE  16.4   8  180    22.5
## 6 Merc 450SL  17.3   8  180    22.5
## 7 Merc 450SLC 15.2   8  180    22.5
```

```
# 3)
```

```
sum(CAR.2$hp) / 7
```

```
## [1] 134.7143
```

고급 다음 작업을 이어서 해보자.

1) 벤츠 차량의 수, 평균 마력을 구하라. filter()와 summarise() 함수를 사용하라.

```
# 1)
```

```
CAR.3 <- CAR.1 %>%
```

```
  filter(str_detect($car, 'Merc')) %>%
```

```
  summarise(number = n(), hp_avg = (sum(hp)/n()))
```

```
CAR.3
```

```
## # A tibble: 1 x 2
```

```
##   number hp_avg
```

```
##   <int> <dbl>
```

```
## 1     7  135.
```

Day 3