

Instagram @ba_eb_ae



늦었다고 생각할 때가
진짜 너무 늦었다

GitHub @Rlearnchan



이름 : _____

소속 OR 연락처 : _____

안녕하세요. 이하 질문에 SO MUCH 자유롭게 답해주세요:)

- 체크박스 여러 개 골라도 됩니다.
- 선지가 마땅찮으면 근처에 수기로 남겨주세요.

1. 인턴을 찾아온 이유가 무엇인가요?

☐ 교재 받기 ☐ 강의 듣기 ☐ 작업 상담 ☐ R 코드 질문

1-1. (강의 듣기 체크) 참석 가능한 시간대

☐ 엘리버드 ☐ 도시락회의 ☐ 일과시간 ☐ 직장인반

2. R에서 알고 싶은 것

3. R이 해주면 좋겠다 싶은 것

4. (필수) 두 개 이상 동의해주세요.

☐ all 무료 ☐ 테스트지 인턴 소장 ☐ 끝날 때 단체 사진

☐ 부끄럼 없이 질문 ☐ 댄스동아리 가입상담

(초급) 한 번쯤 들어 본 개념이면 체크

☐ 객체 ☐ 클래스 ☐ 인덱싱 ☐ 패키지
☐ Tidyverse ☐ Pipe Operator ☐ 텍스트마이닝 ☐ loop
☐ API ☐ Selenium ☐ html ☐ GitHub

(중급) 앞서 고른 개념에서, 두 개 골라 자유롭게 설명해주세요.

1. () : _____

2. () : _____

(고급) 코드의 실행 결과와, 간단한 설명을 적어주세요.

```
KYT <- c('신지', '종민', '백가')  
  
for (i in seq_along(KYT)) {  
  KYT[i] <- paste0(KYT, '_', 화이팅)  
}  
  
print(KYT)
```

프로그램 계획서

프로그램명		fasteR				
담당	이름	배형찬				
	소속	산업연구원 동향분석실				
	연락처	010-3251-5754 instagram @ba_eb_ae	이메일	fragrant@kiet.re.kr		
목표	데이터 분석을 위한 R 기초를 배우고, 실제 코드를 작성·활용해본다.					
권장 대상	STATA 등 경제 분석 소프트웨어는 다뤄봤지만, 프로그래밍 언어엔 익숙지 않으신 분		수강 인원	3~6명		
상세 일정	회기	일시	내용		장소	비고
	A		프로그램 수요 조사, 레벨 테스트			10분 내외
	1		R 세팅 및 기초 개념			
	2		세련된 데이터 분석 : Tidyverse			
	3		엑셀 다운로드는 그만 : API			
	4		크롬 자동화 : Selenium			
	B		각자 원하는 데이터로 실습			분석·발표
참고 도서	1. R과 통계분석 (tidyverse 활용) // 박동련 // 9791158082819 2. 파이썬 머신러닝 판다스 데이터 분석 // 오승환 // 9788956748337					
공지 사항	1. 개인 노트북, 무선 네트워크(핫스팟) 연결을 준비해주세요. 2. 미리 프로그램을 설치해주시길 권장합니다. - R, RStudio, Chrome browser 최신 버전 - 기타 파일, 코드 : https://github.com/Rlearnchan/fasteR → Code → Download ZIP 3. 동일 주제의 Python 사용법을 서브 노트로 제공할 예정입니다. 4. 일정상 A, B회기는 축소·삭제될 수 있습니다.					
Plan B	1. 이미 작성한 작업 코드를 소재로 간단 설명, 함수 활용법을 익힌다.					

fasteR

Hyeongchan Bae

June 2022

Contents

Day 1	2
1. Download	2
2. Something to Know	2
3. Basic Function	5
4. Others	12
5. Quiz	13
Day 2	15
1. What is Tidyverse	15
2. Exercise : PSI 보도자료	16
3. Advanced : 미국 수출데이터 가공	30
4. Quiz	35
Day 3	38
1. What is API	38
2. Exercise : 서비스업생산지수 업데이트	38
3. Advanced : 다중 데이터 로드	44
Day 4	49
1. What is Selenium	49
2. Exercise : 업무포털 로그인	50
3. Advanced : 아이유 Top 100 수록곡	52

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> (강의 관련)
- https://github.com/Rlearnchan/KIET_Public (업무 관련)
- Download zip 기능을 쓰거나, R 파일을 열어 코드만 체크하는 식으로 이용.
- html 버전 교재는 중요 부분에 밑줄, 긴 코드도 잘림없이 표시되니 pdf 버전과 함께 보길 권장.

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.
- 인턴도 R이 익숙해지면서, 여러 프로그램에 걸쳐있던 작업을 하나씩 구현해보자 마인드로 배우고 있습니다.

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.
- 동일한 패키지 설치, 중간 작업물(.Rdata, .Rhistory) 보유 등이 **비슷한 환경**을 만들.

.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. tex 문법 지원.

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-4.2.0/library"
```

- (윈도우 기준 예시) 만일 [2] "C:/Users/**사용자 이름**/Documents/R/win-library/4.2.0" 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-4.2.0/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-4.2.0/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
```

- print() 함수엔 digits 라는 인자(설정 옵션)가 있음.

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

```
## [1] 0.0001952151 0.9883824685 -0.8379969080 -0.9520213021 -1.4598149913
```

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

```
## [1] 0.4655239 -0.8249507 0.1812973 1.1381168 -0.2570590
```

- 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.
- model 성능을 보려 같은 숫자를 input할 경우나, 지금처럼 같은 결과물을 공유할 때.

```
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"
```

- 그냥 banana라고 치면 banana 이름을 가진 객체, 그 속의 값을 호출하란 명령.
- 해당 객체가 부재해서, R과 내가 그런 약속을 한 바 없으므로 에러가 발생할 것.
- class() 함수는 자주 쓰니 기억해 둘 필요가 있음.
- 함수는 인자마다 특정 클래스를 요구하곤 함.
- 예컨대 숫자(double) 자리인데, 텍스트(string)을 입력한다든지. 매우 흔한 오류 케이스.

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

```
## [1] "이제와뒤늦게무엇을 더 보태려하나"
```

```
paste0('이제와 ', '뒤늦게 ', '무엇을 더 보태려하나') # 띄어쓰기를 추가해보자
```

```
## [1] "이제와 뒤늦게 무엇을 더 보태려하나"
```

- 띄어쓰기를 포함한 문구를 paste0() 함수로 잇는 게 직관적이면서 유용.

```
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"
```

- ‘너 BR31 뭔지 알지? 우리 약속했잖아...’

```
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.
- 괄호나 슬래시처럼 함수 구성, 연산에 쓰일 법한 기호는 특히 오류가 빈번.

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
```

- data = what 해야되는데, data = 1, 2, 3, ... 식으로는 정확한 제시가 안 됨.
- R은 data = 1 이고, 2, 3, ...은 matrix() 함수의 다른 input 쪽으로 인식.
- 따라서 1, 2, 3, ...을 묶어줘야 하고, 여기선 vector를 쓴 것.

```
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
```

- 마찬가지로 dimnames = what 위해서, 벡터 두 개를 상위 개념인 list로 묶음.
- scala, vector, matrix(or dataframe), ..., list
- 요거트, 요거트 4개 묶음, 플레인 4개 + 딸기맛 4개 세일상품, ..., 장바구니
- list 아래는 이것저것 다 들어감. 다른 list도 원소로 가질 수 있음.

```
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) # 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 섞어도 상관 없음.
- 코드가 길어지면 한 번에 보기 어려우니, 적당한 단위마다 엔터를 쳐주자.

```
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
```

- 1:12는 seq(from = 1, to = 12, by = 1) 과 같음.
- <-는 =과 같음.
- 자주쓰는 함수라면 인자들 위치를 대개 아니까, input 값만 순서에 맞춰 넣어도 좋다.
- 다만, 타인과 공유할 때, 혹은 미래의 내가 볼 땐 다소 헷갈릴 수도.
- 코드를 가장 잘 아는 건 코드 쓸 때의 나 자신.
- 같은 작업이라도 여러 route가 있음.

```
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

```
yunha <- 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"
```

- 경제, 통계에서 다루는 데이터는 대개 2차원 + 변수 이름 확실 + observation 여러 개인 형태.
- R은 데이터프레임 형태를 기본으로 두고 있으며, Python에선 이를 구현하기 위해 pandas 라이브러리가 존재.
- pandas는 panel data analysis의 약자. 물론 마스코트는 팬더.

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
```

- dataframe \$ variable 인덱싱은 정말 자주 사용.
- 하나의 열벡터로 추출하는 거라, matthew\$A[3] 식으로 다시 한 번 인덱싱 가능.

```
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.

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 최근 설치한 R을 인식해 사용.
- RStudio는 Help → Check for Updates 기능을 활용.
- 패키지는 Tools → Check for Package Updates

Working Directory 작업 공간(폴더)을 지정하자

- 외부 파일을 불러오거나, 외부로 저장할 일이 많음.
- R은 기본적으로 C:/Users/Document, 즉 내 문서 폴더를 기본 작업 공간으로 인식.
- 매번 내 문서 폴더에서 작업을 할 순 없는 노릇.
- `setwd(folder)` 함수의 folder 인자에 원하는 경로를 입력.

```
setwd('C:₩Users₩KIET₩Documents₩GitHub') # 그대로 붙여넣으면 안 되고,
```

```
setwd('C:/Users/KIET/Documents/GitHub') # ₩를 /로 교체해줘야 함.
```

- R에서 ₩ 문자는 정규표현식의 하나로, 슬래시 표현 외에 약속된 다른 기능을 담당함.
- 그런 차에 /를 경로 기술에 대신 사용하는 듯.

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.
- 4) iris 데이터를 간단히 소개하라.

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>:7:35: 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 따워 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 따워 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

- <https://www.tidyverse.org/packages/>
- Tidyverse 생태계라 봐도 괜찮음.

```
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 3.3.6    v purrr  0.3.4
```

```
## v tibble  3.1.6    v dplyr  1.0.9
```

```
## 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.
- Conflicts 란에 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.

Key Setting 두 가지 단축키 설정 권장

- Tools → Modify Keyboard Shortcuts 에서 단축키를 확인, 변경할 수 있음.

- 가장 자주 쓰는 <-, %>% 정도는 단축키를 활용하는 게 좋음.
- 인턴은 Insert Assign Operator를 home, Insert Pipe Operator를 end에 두고 사용.

2. Exercise : PSI 보도자료

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

```

- setwd() 함수로 작업 경로를 지정했고, 해당 폴더 내 excel 파일이 있음을 전제.

```
PSI_ORIGINAL <- read.xlsx('PSI 연습용.xlsx') %>%
  tibble() # 엑셀 파일을 읽어서 PSI_ORIGINAL 이라 명명.
```

PSI_ORIGINAL # tibble을 사용하니, 직전보다 깔끔하게 저장된다. class 표시는 됨.

```
## # 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 xlsxFFile 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 xlsxFFile, 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>   <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>
```

- head(n = 10) 해서 상위 10개를 볼 수도 있음.
- tidyverse 함수가 아녀도 pipe operator를 사용할 수 있음. 시너지가 좋은 거지, 개별로도 powerful.

```
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, 얘는 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.
- `mutate(data_table, var_name = new_data, ...)` 새로운 열 만들기(덮어쓰기 가능).
- 인턴의 함수 설명이, 실제 인자값 네이밍을 적은 건 아님에 유의. 뉘앙스로 이해.


```
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.
```

- 객체를 나누면, 작업 과정을 분기할 수 있음. 일종의 save point.

```
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 헷갈릴 일이 생기기 마련.
- 인턴의 네이밍 규칙은 이렇다.
- 임시 : temp1, temp2, ...
- 중간작업 : 대문자_파트.버전
- 완성품은 다른 이름으로 저장.

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. 특히 가전 파트.
```

```
## # 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 멋쟁이가 만들어둔 함수를 그대로 긁어오자

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

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

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

- `function(factors, ...) { what to do }` 식으로 사용자 함수를 정의할 수 있음.
- 아래 `mutate_at()` 함수에서 오류가 나서, `n`의 default 값만 `function(x, n = 0)` 으로 수정하자.
- tidyverse 함수에선 직전 작업 값 ‘.’ 으로 표현한다. `.$구분` 으로 적어도 해당 흐름 내에선 ‘작업 중이던 데이터프레임의 구분 열’ 줌으로 인식하는 셈.
- `.vars`, `.funs`도 유사한 맥락의 네이밍인데, 설명하는 건 강의 레벨을 벗어나므로 스킵.
- 궁금한 사람은 purrr 패키지 참고.

```
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_WEATHER.2, PSI_WEATHER.3 만드는 코드를 고치자.
- 객체명을 따로 안 바꿔서, new version 으로 덮어씌우게 됨.
- 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 가공

```
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
```

- 코드가 조금 난잡해서, 인쇄물로 보고 있다면 html 버전 교재도 참고하기 바람.

```
library(haven) # dta 로드 패키지
```

```

LINK <- read_dta('연계표 연습용.dta') %>%
  select(hsc, ksic5) # 국제무역코드(hscode), 한국표준산업분류의 세세분류(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.

}

```

- seq_along(vector) 함수는 1:length(vector) 수열을 출력.
- 1:6552 체크리스트(영벡터)에 v 표시(1) 채워가는 것.
- 10으로 시작하면 체크, 11로 시작하면 체크, ...
- 손으로 하던 작업을 코드로 구현한 셈.

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

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

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

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

- 논리값은 숫자로 변환하면 0, 1이므로 sum()과 궁합이 좋음.

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

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

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

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

- LINK.1인데, TESTER = TRUE인 row만 선택하고, hsc column만 추출.

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

```
## [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 연산은 y vector 내에 x라는 인수가 있다면 TRUE를 반환.

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 짱이었나보다. 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

1. What is API

Computers' Talk 컴퓨터나 컴퓨터 프로그램 사이의 연결

- language for computer 라서, 사람인 우리가 보고 쓰려면 일정 부분 배워야.
- 사람 보라고 만든 인터페이스는 UI(User Interface), 예컨대 삼성 갤럭시는 One UI 4.1
- API(Application Programming Interface)에 대한 짧은 클립을 추천.
- 노마드코더, API 기초개념 <https://youtu.be/iyFHfzCRHA8>

Kosis Open API 국가통계포털 공유서비스

- https://kosis.kr/openapi/introduce/introduce_01List.jsp
- 통계 데이터를 프로그램 상에서 바로 로드, 작업할 수 있음.
- 호출키(+인증키), 항목 설정 등으로 이뤄진 URL을 통해 데이터를 받아옴.
- 통계청뿐만 아니라 한은, 공공데이터포털, DART 등 대부분의 정보처는 API 서비스를 제공.

2. Exercise : 서비스업생산지수 업데이트

서비스업 생산지수 https://kosis.kr/statHtml/statHtml.do?orgId=101&tblId=DT_1KS2015

ISTANS 산업동향지수 <https://istans.or.kr/su/newSuTab.do?scode=S359>

- kosis 원본(86개) → grc 매칭, 재분류(20개) → istans 업로드
- 여기서 첫 번째 단계를 API 활용으로 대체해봄.

최초 사용자가, URL 생성 방식으로 진행함을 가정

- 서비스이용 → 통계자료 탭을 눌러서 간단한 신청 폼을 작성.

- 자료등록 메뉴로 이동, 통계조사명 란에 ‘서비스업동향조사’ 입력하고 검색.
- 서비스업동향조사 내 통계표가 여럿 나오는데, 2페이지의 산업별 서비스업생산지수의 사용여부 체크하고 통계표등록.

2) 상세 조건 설정

- URL생성 메뉴로 이동, URL 생성 조건을 설정하자.
- 통계청 포털에서 체크박스 누르던 파트를 옮겨놓은 것.
- 필요한 데이터는 **불변/계절조정지수**의 최근값인데, **1레벨의 총지수**와 **3레벨 모든 항목**이 담겨있어야 함.

KOSIS
공공서비스

소개 및 이용방법

서비스이용

개발가이드

커뮤니티

이용현황

통계목록

통계자료

대용량통계자료

통계설명

통계표설명

KOSIS통합검색

통계자료

통계자료 설명을 제공합니다. [계통정보 확인하기](#)

활동신청

신청현황

자료등록

URL생성

URL생성 조건 설정

통계표명	산업업 서비스업생산지수 (2015=100.0)	통계표ID	DT_LKS2015
활용 자료명	서비스업생산지수_불변		
조회구분 (?)	<input checked="" type="radio"/> 시계열(time series) <input type="radio"/> 횡단면(cross sectional)		

분류

항목

업종별

☒ 개별 ☐ 집합법 1 / 5

☐ 경상지수
☒ 불변지수
☐ 계절조정지수

분류/항목선택

업종별

☒ T 총지수
☐ E 수도, 하수 및 폐기물 처리, 원료 재생업
☐ E36 수도업
☐ E37 하수, 폐수 및 폐기물 처리업
☒ E370 하수, 폐수 및 폐기물 처리업
☐ E38 폐기물 수집, 운반, 처리 및 원료 재생업
☐ G 도매 및 소매업
☐ G45 자동차 및 부품 판매업
☒ G451 자동차 판매업

선택

닫기

URL생성

취소

- 조회구분 = 시계열 선택
- 분류는 조금 길어서 다음을 순차적으로 진행
- 개별 옆의 선택 버튼 클릭
- 업종별 옆의 부등호를 등호로 교체
- 1을 3으로 교체하고 이동 버튼 클릭 (3레벨 분기로 전환)
- 체크박스 일괄 선택하고, 선택 버튼 눌러서 설정 저장
- 다시 개별 옆의 선택 버튼 클릭
- 총지수도 체크해 선택 버튼 클릭
- 항목의 불변지수, 계절조정지수 체크
- 활용 자료명에 '서비스업생산지수 업데이트' 입력
- URL생성 버튼 클릭
- 너무 많이 선택했다고 에러 사인. 3레벨 산업 분기를 전부 체크한 탓으로 보임.
- 원래는 대용량 데이터로 개별 신청해야 하나, 나름 대응 방안이 있음. 아래서 자세히 설명.

- KOSIS

공유서비스

소개 및 이용방법

서비스이용

개발가이드

커뮤니티

이용현황

개발한 년도

로그아웃

마이페이지

국가통계포럼

공공데이터포럼

통계목록

통계자료

대용량통계자료

통계설명

통계표설명

KOSIS통계입력

통계주요자료

통계자료

통계자료로 제공합니다.

재공표본 확인하기

Home

서비스 이용

로그인

회원

통계설명

통계현황

자료등록

URL생성

URL생성

URL생성 조건 설정

통계표명

산업별 서비스업생산자수 (2015~2020)

통계표ID

DT_1K32015

조회구분

시계열

URL생성 상세조건 정보

출력형태 설정

데이터 포맷

JSON

SDMX

조회기간 설정

기간설정

최근시점기준

최근시점부터

주기

월

회근

3

개 자료

수출시점간격

최근시점 조절에 사용할 수 있는 항목으로 현재 시점부터 사용할 기준시점까지의 간격을 입력 ex) 2019, 2017, 2015 등 2개 시점 간격으로 추출시 [2] 입력

활동자요명

URL 상세 정보

조회구분

URL기/결과값보기

서비스업생산자수_불변_1

불변지수 + 수도업

시계열

URL보기

결과값보기

서비스업생산자수_불변_2

불변지수 + 학수, 폐수 및 분뇨 처리업

시계열

URL보기

결과값보기

서비스업생산자수_불변_3

불변지수 + 자동차 판매업

시계열

URL보기

결과값보기

서비스업생산자수_불변_4

불변지수 + 자동차 부품 및 이륜차 판매업

시계열

URL보기

결과값보기

서비스업생산자수_불변_5

불변지수 + 산업용 건축산물 도매업

시계열

URL보기

결과값보기

서비스업생산자수_불변_6

불변지수 + 금·석유 및 광물 도매업

시계열

URL보기

결과값보기

서비스업생산자수_불변_7

불변지수 + 생활용품 도매업

시계열

URL보기

결과값보기

서비스업생산자수_불변_8

불변지수 + 기계장비 및 관련 물품 도매업

시계열

URL보기

결과값보기

서비스업생산자수_불변_9

불변지수 + 건축자의 도매업

시계열

URL보기

결과값보기

서비스업생산자수_불변_10

불변지수 + 기타 건물 도매업

시계열

URL보기

결과값보기

서비스업생산자수_불변_11

불변지수 + 종합 소매업

시계열

URL보기

결과값보기
- 조금 기다리면 항목별(산업별)로 총 86개의 URL이 만들어짐.
- 3) 단일 데이터 로드

서비스업생산자수_불변_67

불변지수 + 개인 및 가정용품 임대업

시계열

URL보기

결과값보기

서비스업생산자수_불변_68

불변지수 + 상업용 기계 및 장비 임대업

시계열

URL보기

결과값보기

서비스업생산자수_불변_69

불변지수 + 조등 조류기관

시계열

URL보기

결과값보기

서비스업생산자수_불변_70

불변지수 + 증등 조류기관

시계열

URL보기

결과값보기

서비스업생산자수_불변_71

불변지수 + 교통 조류기관

시계열

URL보기

결과값보기

서비스업생산자수_불변_72

불변지수 + 항공

시계열

URL보기

결과값보기

서비스업생산자수_불변_73

불변지수 + 항항

시계열

URL보기

결과값보기

서비스업생산자수_불변_74

불변지수 + 위항

시계열

URL보기

결과값보기

서비스업생산자수_불변_75

불변지수 + 거주 목적시설 운영업

시계열

URL보기

결과값보기

서비스업생산자수_불변_76

불변지수 + 비거주 목적시설 운영업

시계열

URL보기

결과값보기

서비스업생산자수_불변_77

불변지수 + 창작 및 예술관련 서비스업

시계열

URL보기

결과값보기

서비스업생산자수_불변_78

불변지수 + 도서관, 사적지 및 유사 여가관련 서비스업

시계열

URL보기

결과값보기

서비스업생산자수_불변_79

불변지수 + 스포츠 서비스업

시계열

URL보기

결과값보기

서비스업생산자수_불변_80

불변지수 + 유점기 및 기타 오락관련 서비스업

시계열

URL보기

결과값보기

서비스업생산자수_불변_81

불변지수 + 산업 및 전문가 단체

시계열

URL보기

결과값보기

서비스업생산자수_불변_82

불변지수 + 자동차 수리업

시계열

URL보기

결과값보기

서비스업생산자수_불변_83

불변지수 + 개인 및 가정용품을 수리업

시계열

URL보기

결과값보기

서비스업생산자수_불변_84

불변지수 + 미용, 화장 및 유사 서비스업

시계열

URL보기

결과값보기

서비스업생산자수_불변_85

불변지수 + 그 외 기타 개인 서비스업

시계열

URL보기

결과값보기

서비스업생산자수_불변_86

불변지수 + 송지수

시계열

URL보기

결과값보기

생성된 URL

https://kosis.kr/openapi/statisticData.do?method=getList&apiKey=OTY3MmY1NTk0MjM4MDU4dXNlZWVCEWNT0000&format=json&D=8UserStatId=4b4554/101/DT_1K32015/2/3/202020191261911&ServiceMethod=SDMX-C

URL복사

뒤돌아

통계서비스기록과

- 41

- 해당 URL을 복사해 다음과 같이 로드.

```
library(tidyverse) # 데이터 핸들링
```

```
library(jsonlite) # JSON 파일 로드
```

```
URL <- 'https://kosis.kr/openapi/statisticsData.do?method=getList&apiKey=<인증키>=&format=json&jsonVD=Y&userS
```

```
SERVICE <- fromJSON(URL) %>% tibble()
```

```
SERVICE
```

```
## # A tibble: 3 x 16
```

```
## TBL_NM PRD_DE TBL_ID ITM_NM ITM_NM_ENG ITM_ID UNIT_NM ORG_ID UNIT_NM_ENG
```

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

```
## 1 산업별 서비~ 202201 DT_1K~ 불변~ Volume T2 2015=~ 101 2015=100
```

```
## 2 산업별 서비~ 202202 DT_1K~ 불변~ Volume T2 2015=~ 101 2015=100
```

```
## 3 산업별 서비~ 202203 DT_1K~ 불변~ Volume T2 2015=~ 101 2015=100
```

```
## # ... with 7 more variables: C1_OBJ_NM <chr>, C1_OBJ_NM_ENG <chr>, DT <chr>,
```

```
## # PRD_SE <chr>, C1 <chr>, C1_NM <chr>, C1_NM_ENG <chr>
```

- fromJSON(source, ...) 함수는 URL, 로컬 JSON파일 등을 읽어서 데이터 프레임으로 출력.
- 최근 3개월 자료를 default로 설정해둔 바, 3개의 row가 로드됨.
- 16개의 column이 생성되었는데, 네이밍 요령은 **개발 가이드**란 이름으로 다음에서 제공.
- https://kosis.kr/openapi/devGuide/devGuide_0201List.jsp
- **통계표선택 방법의 출력결과** 항목을 보면 해당 column에 어떤 variable이 담겼는지 알 수 있음.

```
temp <- SERVICE %>%
  select(TBL_NM, # 통계표 이름
         PRD_DE, # 시점
         ITM_NM, # 지수(항목)
         C1_NM, # 산업(분류)
         C1, # 산업 코드
         DT) %>% # 지수값
  mutate(DT = as.double(DT)) # 값은 숫자 클래스로 변경
```

- 주석을 달아주고자 엔터로 분리.

```
colnames(temp) <- c('통계', '시점', '지수', '산업', '코드', '값') # 직관적인 네이밍으로 변경

SERVICE.1 <- temp

rm(temp) # 임시 객체 삭제

SERVICE.1 # 필요한 값만 정리
```

```
## # A tibble: 3 x 6
##   통계      시점 지수   산업 코드   값
##   <chr>      <chr> <chr>  <chr> <chr> <dbl>
## 1 산업별 서비스업생산지수 (2015=100.0) 202201 불변지수 수도업 E360 108.
## 2 산업별 서비스업생산지수 (2015=100.0) 202202 불변지수 수도업 E360 98.2
## 3 산업별 서비스업생산지수 (2015=100.0) 202203 불변지수 수도업 E360 104.
```

- 다루는 객체가 많아질수록 헛갈리기 마련. 임시 객체를 활용하는 것도 방법.

3. Advanced : 다중 데이터 로드

- 현대, 이걸 86번 반복할 순 없는 노릇.
- 다시 URL생성 코너로 돌아가, 불변지수x하수, 폐수 및 분뇨 처리업 의 URL을 확인하자.
- 자세히 보면 \$prdSe 앞의 숫자가 2로 변경돼있다.
- 같은 조건에서 URL을 다중 생성하다보니, 다른 부분은 동일하더라도, 구별 차원에서 숫자를 달리 매겨놓은 것.
- 해당 숫자를 매개로 for() 구문을 사용하면 반복 작업을 할 수 있겠다.
- Day 2의 Advanced에 있던 내용과 유사하다. 아래 코드를 차용해도 좋지만, 가급적 대용량 통계자료 신청 기능을 이용하자.
- 왜냐면, 이하 내용은 대처법에 가깝고, 아마 통계청에서 요정도는 가공해서 단일 URL로 제공해줄 터라.
- 전체 코드를 작성하고, 설명을 달아둔 것이니, 한 번 overview 하고서 부분별로 읽어보기 바람.

0. what do you need

KEY <- '개인 인증키'

ID <- 'Kosis ID'

INDEX.1 <- '불변지수 URL 고유번호'

INDEX.2 <- '계절조정지수 URL 고유번호'

- #을 이용한 주석은 코드 실행되지 않는 바, 소제목 대용으로도 좋음.
- 비슷한 내용의 코드라면 덩어리 지어 직관적으로 보이게 하는 것.
- 코드 구성은 같지만, 개인화된 정보를 기입해야 되는 경우가 있음.
- parameter를 앞으로 빼두면 이하 코드에 자동 적용되니 편함.
- 인턴은 이를 what do you need 코너로 만들어 작성하는 편.

1. setting

library(tidyverse) # 데이터 핸들링

library(jsonlite) # JSON 파일 로드

library(openxlsx) # 엑셀 입출력

- 필요한 패키지 로드는 가급적 코드 상단에 작성.
- 물론 코드를 짜면서 패키지를 하나씩 추가하기 마련.
- 다만 정리를 위쪽에 해두자는 것.

2. url

```
BASE <- paste0('https://kosis.kr/openapi/statisticsData.do?method=getList', # 요청
               '&apiKey=', KEY, # 인증키
               '&format=json&jsonVD=Y', # 포맷 : JSON
               '&userStatsId=', ID) # 방식 : 사용자가 기등록한 자료 로드
```

- 개인 URL은 여러 파트로 분기될 수 있음.
- 본 예시에선 &을 기준으로 인자가 나뉘짐.
- paste0() 함수로 어차피 합쳐질 테니 코드 상에선 쉼표로 분절하고 엔터, 주석 표시.
- KEY, ID 객체는 앞서 적어둔 값을 받아와 입력.

```
CORE <- c(INDEX.1, INDEX.2) # 불변지수, 계절조정지수 URL 고유번호
```

```
NUMBER <- 1:86 # URL 개수
```

- 산업별 86개 자료를 로드하는 데, 그걸 불변/계절조정지수에서 각각 해야함.
- for(k = 1:2, for(i = 1:86, load)) 식으로 loop를 짜야겠다는 생각.
- 물론 실제 코드 작성은 반대로 이뤄짐. like 역진귀납.
- 구체적으로 URL 1개 → 그 코드를 for로 감싸서 URL 86개 → 그걸 또 감싸서 86x2개
- 따라서 내부 코드만 잘 짜두면 반복은 쉬움.
- 작업을 말로 설명할 수 있다면, 그걸 하나씩 코드로 바꿔가면 되는 것.

```
DATA <- tibble() # 데이터 담을 빈 그릇
```

- 아무 인자도 없는 tibble()은 0x0 data frame을 생성.
- Day 2에서 영벡터 만들었던 것과 같은 맥락.

3. data load

```
for (k in seq_along(CORE)) { # k = 1, 2에 대해서 다음 작업 실행
```

```
  temp1 <- tibble() # 데이터 담을 빈 그릇
```

```
  for (i in seq_along(NUMBER)) { # i = 1, 2, ..., 86에 대해서 다음 작업 실행
```

```
    URL <- paste0(BASE, # URL 앞 부분
```

```
      '101/', # 통계청
```

```
      'DT_1KS2015/', # 산업별 서비스업생산지수(2015=100)
```

```
      '2/', # 시계열
```

```
      '1/', # 간격 : 1
```

```
      CORE[k], '_', NUMBER[i], # URL 나열
```

```
      '&prdSe=', 'M', # 주기 : Month
```

```
      '&newEstPrdCnt=', '1') # 최근 1개 자료
```

```
    temp2 <- tryCatch(expr = fromJSON(URL) %>% # fromJSON ~ select 함수를 실행하되,  
      tibble() %>%  
      mutate(번호 = i) %>%  
      select(번호, PRD_DE, ITM_NM, C1_NM, C1, DT),  
      error = function(e) tibble(NULL)) # 오류(데이터 부재) 발생하면 스킵
```

```
    temp1 <- rbind(temp2, temp1) # stacking
```

```
  }
```

```
temp3 <- temp1 %>%
```

```
  arrange(nchar(C1), C1) %>% # 분류값 순으로 정렬
```

```
  mutate_at(vars(DT), as.double) # 수치값 class 숫자로 변경
```

```
DATA <- rbind(temp3, DATA) # stacking
```

```
}
```

- 용례가 어렵지만, `rbind(a, b)`는 자주 등장하는 기본 함수.
- 같은 column으로 구성된 a, b 데이터를 row 연장해서 붙임.
- 5x10, 10x10 데이터 테이블을 `rbind()`로 묶는다면, 15x10 데이터 테이블이 됨.
- 여기서 0x0, 1x6 데이터를 묶어서 저장하고, 그 1x6 데이터에 새로운 1x6을 더하고...
- $(n+1) \times 6 \leftarrow n \times 6 + 1 \times 6$ 식으로 지정해 쌓아가는 것.
- `tryCatch(function, 에러대처, ...)` 함수는 간혹 통계 데이터가 없어서 에러, loop 정지되는 경우가 있기에 작성.
- 이런 게 역진으로 올라가면서 생기는 문제고, generalize 과정이겠음.
- `arrange(condition1, condition2, ...)`는 row 순서를 해당 조건에 맞춰 정렬해줌.
- condition1을 맞추고, condition2를 이어 맞추는 식.
- 여기서 코드의 자릿수(짧으면 1레벨, 길면 3레벨이니까)로 맞추고, 이어서 코드 문구로 맞춤.

4. finish

```
rm(list = c('temp1', 'temp2', 'temp3')) # 임시 객체 삭제

colnames(DATA)[2:6] <- c('시점', '지수', '산업', '코드', '값') # 변수명 변경
```

- `rm(list = ls())` 하면 모든 객체가 삭제된다.

5. export

```
write.xlsx(DATA, '서비스업생산지수 업데이트.xlsx') # 엑셀 파일로 저장
```

- 여기까지 덩석 이해했다면, 당신은 본 강의 레벨을 아득히 넘는다.

```
read.xlsx('서비스업생산지수 업데이트.xlsx') %>%
  tibble() %>%
  select(-1) # 첫 번째 열은 삭제
```

```
## # A tibble: 172 x 5
##   시점   지수   산업           코드   값
##   <chr> <chr>   <chr>       <chr> <dbl>
## 1 202203 계절조정지수 총지수           T    114.
## 2 202203 계절조정지수 수도업           E360 107.
## 3 202203 계절조정지수 하수 폐수 및 분뇨 처리업 E370 96.4
## 4 202203 계절조정지수 자동차 판매업           G451 123.
## 5 202203 계절조정지수 자동차 부품 및 이륜차 판매업 G452 116.
## 6 202203 계절조정지수 산업용 농축산물 도매업     G462 110.
## 7 202203 계절조정지수 음·식료품 및 담배 도매업   G463 110.
## 8 202203 계절조정지수 생활용품 도매업           G464 104.
## 9 202203 계절조정지수 기계장비 및 관련 물품 도매업 G465 122.
## 10 202203 계절조정지수 건축자재 도매업          G466 90.4
## # ... with 162 more rows
```

- 번호는 오류 체크용(오류나면 해당 번호 부재)으로 만들었던지라, 최종 결과물에 굳이 담을 필요는 없음.

Day 4

1. What is Selenium

Web Browser Automation 크롬, 엣지 등 인터넷 브라우저 내 작업을 대신해줌

- URL 접속, 텍스트 수집은 물론, 버튼 클릭, 텍스트 입력 등도 가능
- 웹 상 반복 작업을 하는 경우가 많은바, 자동화 스킬을 익혀두면 매우 편리
- R에서는 RSelenium 패키지를 로드해 코드 작성.
- Python에도 똑같은 라이브러리가 있고, 코드도 비슷해서 Python 사례도 참고 가능.

Setting 셀레늄 서버, 드라이버 2종, 크롬 브라우저, java가 필요

- (깃허브) selenium-server-standalone-4.0.0-alpha-1.jar
- (깃허브) geckodriver.exe
- (깃허브, 구글 검색) chromedriver.exe
- (구글 검색) 크롬 브라우저
- (구글 검색) java

Notice 크롬 드라이버와 크롬 브라우저는 버전을 맞춰야 함.

- 크롬브라우저가 업데이트되면, chromedriver.exe 도 같은 버전으로 대체해줘야 함
- 크롬브라우저 버전 확인 : 설정 → Chrome 정보 → 'Chrome이 최신 버전입니다.' 아래 숫자 체크
- chromedriver.exe : chromedriver.chromium.org 접속 → 같은 버전 클릭, 다운로드

2. Exercise : 업무포털 로그인

산업연구원 업무포털 <https://ep.kiet.re.kr/>

0. what do you need

ID <- '포털 아이디'

PW <- '포털 비밀번호'

- 산업연구원 업무포털 아이디, 비밀번호를 기입하자.

1. setting

library(tidyverse) # 데이터 핸들링

library(rstudioapi) # 터미널 사용

library(RSelenium) # 크롬 자동화

library(rvest) # html 해석

2. selenium

TERM_COMMAND <- 'java -Dwebdriver.gecko.driver="geckodriver.exe" -jar selenium-server-standalone-4.0.0-alpha-

terminalExecute(command = TERM_COMMAND) # RStudio Terminal 탭 열어서 상기 커맨드 입력

REMDR = remoteDriver(port = 4445, browserName = 'chrome') # 브라우저로 크롬 선택

- (윈도우 기준) 명령 프롬프트를 켜서, 'java ... port 4445' 문구를 입력하는 게 세팅 과정.
- 이를 반복하기가 번거로우니, RStudio 내장 터미널 활용하는 식으로 대체.
- REMDR는 일종의 list 객체. 기능을 사용할 땐 REMDR\$function() 형태로 작성.
- 개인 컴퓨터에서 잘 안 되는 경우 java 설치, 방화벽 개인/공용 설정 권장

3. login

```
REMDR$open() # 크롬 오픈
```

```
REMDR$navigate('https://ep.kiet.re.kr/index.do') # 업무 포털 접속해서
```

```
BUTTON_LOGIN <- REMDR$findElement('xpath', '//*[@id="f_login"]/ul/li[3]') # 로그인 버튼
```

```
TEXT_ID <- REMDR$findElement('xpath', '//*[@id="loginId"]') # 아이디 입력창
```

```
TEXT_PW <- REMDR$findElement('xpath', '//*[@id="pwd"]') # 패스워드 입력창을 찾아내고
```

```
TEXT_ID$sendKeysToElement(list(ID)) # 아이디 입력
```

```
TEXT_PW$sendKeysToElement(list(PW)) # 패스워드 입력
```

```
BUTTON_LOGIN$clickElement() # 로그인 버튼 클릭
```

- 사람이 웹 브라우저 상에서 하는 행동을 코드로 본딴 것.
- REMDR\$findElement(selector, text, ...) 함수는 실행 당시 페이지 기준으로 데이터를 수집.
- xpath는 페이지 각 요소의 고유 위치 정보.
- 우클릭 → 검사(inspect) → 해당 html 파트 우클릭 → copy xpath
- 버튼과 입력창을 xpath로 찾아 지정하고, 필요한 입력/클릭을 명령으로 대체.

3. Advanced : 아이유 Top 100 수록곡

멜론 시대별 차트 <https://www.melon.com/chart/age/index.htm?chartType=YE&chartGenre=KPOP&chartDate=2000>

1. setting

```
library(tidyverse) # 데이터 핸들링
library(rstudioapi) # 터미널 사용
library(RSelenium) # 크롬 자동화
library(rvest) # html 해석
library(openxlsx) # 엑셀 입출력
```

2. selenium

```
TERM_COMMAND <- 'java -Dwebdriver.gecko.driver="geckodriver.exe" -jar selenium-server-standalone-4.0.0-alpha-4.jar'
terminalExecute(command = TERM_COMMAND)
```

```
REMDR = remoteDriver(port = 4445, browserName = 'chrome')
REMDR$open() # 크롬 오픈
```

- 이하 내용도, for() 구문 내의 함수들을 단 건 적용해보고, 차차 확장해나가는 식으로 학습해보자.

3. crawling

```
PERIOD <- 2000:2021 # 2000~2021년 데이터 수집하려고 vector 만듦
```

```
MASTERPIECE <- tibble() # 데이터 담을 빈 그릇
```

```
for (i in seq_along(PERIOD)) { # i = 1, 2, ..., 22 마다 다음 작업 실행
```

```
  URL <- paste0('https://www.melon.com/chart/age/index.htm',
    '?chartType=YE',
    '&chartGenre=KPOP',
    '&chartDate=', PERIOD[i]) # 달라지는 마지막 부분만 입력변수로 대체
```

```
  REMDR$navigate(URL) # URL 접속
```

```

temp1 <- REMDR$getPageSource() # 페이지 소스 가져오기

TITLE <- read_html(temp1[[1]]) %>% # html 형태의 페이지 정보를 읽어서
  html_elements('form#frm') %>% # form#frm 부분의
  html_elements('div.wrap_song_info') %>% # div.wrap_song_info 부분의
  html_elements('div.ellipsis.rank01') %>% # div.ellipsis.rank01 부분의
  html_text() %>% # 텍스트를 출력하되
  str_remove_all('\n') %>% # 노이즈 '\n' 문자와
  str_remove_all('\t') %>% # 노이즈 '\t' 문자를 제거하고
  str_sub(end = -2) # 마지막에 공란이 있으니 애도 빼자

SINGER <- read_html(temp1[[1]]) %>%
  html_elements('form#frm') %>%
  html_elements('div.wrap_song_info') %>%
  html_elements('div.ellipsis.rank02') %>%
  html_text() %>%
  str_sub(end = nchar(.) / 2) # 같은 문구가 두 번 반복되니 반절만 선택

ALBUM <- read_html(temp1[[1]]) %>%
  html_elements('form#frm') %>%
  html_elements('div.wrap_song_info') %>%
  html_elements('div.ellipsis.rank03') %>%
  html_text() %>%
  str_remove_all('\n') %>%
  str_remove_all('\t')

LIKE <- read_html(temp1[[1]]) %>%
  html_elements('button.btn_icon.like') %>%
  html_text() %>%
  str_remove_all('좋아요\n\n총건수\n') %>% # 노이즈 제거
  str_remove_all(',') %>% # 노이즈 제거
  as.integer() # 이하 계산 편의를 위해 정수 integer class로 변경

temp2 <- tibble(YEAR = PERIOD[i], RANK = 1:100,
  TITLE, SINGER, ALBUM, LIKE) # 수집한 데이터를 묶어 temp2로 저장

```



```
MASTERPIECE <- rbind(MASTERPIECE, temp2) # 기존 데이터에 적층
```

```
Sys.sleep(5 + rnorm(1)) # 시스템 과부하, 차단 방지를 위해 5초 딜레이
```

```
}
```

The screenshot displays a music player interface on the left and its corresponding HTML DOM structure on the right. The music player shows a list of songs with columns for rank, album art, song title, artist, and play count. The HTML view shows the underlying DOM structure, including a table with song details and a list of song titles.

- html 파일은 수많은 부분집합으로 이뤄짐. $A \supset B, C \supset D, E, F$
- `html_elements()` 함수는 html의 부분집합을 찾아가는 역할.
- 예컨대 F를 찾고 싶다면, `html_elements('A') %>% html_elements('B') %>% html_elements('F')`
- 하지만 모든 집합명을 적어야 되는 건 아님. 루트만 잘 제시해주면 됨.
- “시청 지나 다리 건너면 큰 건물이 있어요. 거기서 F 사무관을 불러주세요.”
- “시청 지나 한누리대교를 건너면 정부청사가 있어요. 거기서 산자부 F 사무관을 불러주세요.”
- 둘 다 일정 부분 모호하지만, 아래 정도로만 설명해주면 큰 문제 없잖나.
- 문제가 생기면 어떻게 하나. `html_elements()`를 하나 더 써서, 설명을 추가해줘야겠지.
- 인턴은 상기 코드에서 `for()` 문을 가장 마지막에 작성했다.
- 단일 링크, 여기서 단일 연도 데이터로 먼저 테이블을 만들어보고, 이를 반복/확장시킨 것.

4. export(save)

```
write.xlsx(MASTERPIECE, '멜론차트.xlsx') # 차트 원본 데이터를 엑셀로 저장
```

- MASTERPIECE 객체는 매번 크롤링해서 가져오게 돼서 번거로움.
- 아예 로컬 파일로 저장해두고, 애를 로드해 작업하는 게 편함.

5. import

```
MELON <- read.xlsx('멜론차트.xlsx') %>% tibble()
```

```
MELON # 22개년도 멜론차트 TOP100 데이터
```

```
## # A tibble: 2,200 x 6
##   YEAR RANK TITLE                                SINGER ALBUM      LIKE
##   <dbl> <dbl> <chr>                                <chr> <chr>      <dbl>
## 1 2000    1 아시나요                                조성모 Let Me Love  33931
## 2 2000    2 다 줄거야 (Acoustic Ver.)              조규만 I Will Give Yo~ 12492
## 3 2000    3 Run To You                                DJ DOC The Life... DO~ 18349
## 4 2000    4 거짓말                                god Chapter 3      34488
## 5 2000    5 초련(初戀) (Techno Mix) (Feat. 윤진) 클론 New World      4492
## 6 2000    6 가시나무                                조성모 Classic      3540
## 7 2000    7 흔들린 우정                                홍경민 홍경민 3      7646
## 8 2000    8 나의 연인(我戀)                            임창정 White       16178
## 9 2000    9 영원                                    스카이 Final Fantasy 28158
## 10 2000   10 멍                                    김현정 The Third Eye 25846
## # ... with 2,190 more rows
```

```
filter(MELON, RANK == 1)$TITLE # 연도별 1위곡
```

```
## [1] "아시나요"                "벌써 일년"
## [3] "No.1"                    "Break Away"
## [5] "친구여 (Feat. 인순이)"    "죄와벌"
## [7] "내사랑 : Partner For Life" "미인 (美人)"
## [9] "So Hot"                  "Gee"
## [11] "Bad Girl Good Girl"      "Roly-Poly"
## [13] "강남스타일"              "자니 (Feat. Dynamic Duo)"
## [15] "썸 (Feat. 릴보이 Of 빅스)" "뱅뱅뱅 (BANG BANG BANG)"
```

```
## [17] "CHEER UP"           "첫눈처럼 너에게 가겠다"
## [19] "사랑을 했다 (LOVE SCENARIO)" "사랑에 연습이 있었다면 (Prod. 2soo)"
## [21] "아무노래"           "Celebrity"
```

- pipe operator를 안 쓰고, filter() 첫 자리에 data_table을 넣은 모습.

6. analysis

6-1. TOP100 수록곡이 가장 많은 가수는?

```
SUPERSTAR <- MELON %>%
  distinct(TITLE, .keep_all = TRUE) %>% # 여러 해 차트인 했다면 하나만 남김
  group_by(SINGER) %>% # 데이터를 가수명으로 그룹지어서
  summarise(NUMBER = n()) %>% # 그룹마다 원소 개수를 세고 NUMBER라 하자
  arrange(desc(NUMBER)) # summary를 NUMBER 내림차순으로 정렬
```

SUPERSTAR # 빅뱅, 아이유가 31곡으로 동점

```
## # A tibble: 651 x 2
##   SINGER      NUMBER
##   <chr>      <int>
## 1 BIGBANG (빅뱅)    31
## 2 아이유          31
## 3 다비치          25
## 4 방탄소년단       23
## 5 SG 워너비        22
## 6 2NE1             18
## 7 MC몽             18
## 8 백지영           16
## 9 케이윌           16
## 10 성시경           15
## # ... with 641 more rows
```

6-2. 아이유의 TOP100 수록곡은?

```
DLWLRMA <- MELON %>%  
  filter(SINGER %in% c('아이유', 'IU')) %>% # 가수명이 아이유 내지 IU인 관찰값  
  arrange(desc(LIKE), RANK) %>% # 좋아요 내림차순으로 정렬하고, 높은 순위대로 나열  
  distinct(TITLE, .keep_all = TRUE) # 역시 중복 케이스 삭제  
  
DLWLRMA # 밤편지 최고
```

```
## # A tibble: 31 x 6  
##   YEAR RANK TITLE                                SINGER ALBUM    LIKE  
##   <dbl> <dbl> <chr>                                <chr> <chr>    <dbl>  
## 1 2017    2 밤편지                                아이유 밤편지  440455  
## 2 2020    5 에잇(Prod.&Feat. SUGA of BTS)        아이유 에잇   377404  
## 3 2020    6 Blueming                                아이유 Love poem 335354  
## 4 2020   29 Love poem                                아이유 Love poem 325406  
## 5 2021    1 Celebrity                                아이유 IU 5th A~ 317773  
## 6 2017   67 가을 아침                                아이유 꽃갈피 둘 287467  
## 7 2018   46 뽀뽀                                아이유 뽀뽀    286851  
## 8 2017    7 팔레트 (Feat. G-DRAGON)                아이유 Palette  279988  
## 9 2014    4 금요일에 만나요 (Feat. 장이정 Of HISTORY) 아이유 Modern T~ 268366  
## 10 2021    5 라일락                                아이유 IU 5th A~ 238361  
## # ... with 21 more rows
```

- 직접 크롤링을 했다면, 교재 상의 LIKE와 조금 다른 숫자를 접했을 것.
- 당신은 멜론에서 따끈따끈한 새 데이터를 가져왔기 때문.
- 매번 최신 값으로 업데이트 해야되는 데이터가 있다면, 자동화 코드를 짜보는 게 어떨까.

6-3. 연도별 아이유 LIKE 총량은?

```
UAENA <- MELON %>%  
  filter(SINGER %in% c('아이유', 'IU')) %>% # 마찬가지로 아이유 곡인데  
  group_by(YEAR) %>% # 연도별로 그룹지어서  
  summarise(HEART = sum(LIKE)) # 해당 연도의 total like를 계산  
  
UAENA # 음반 발표가 없던 2016년, 또는 연말에서야 나온 2013년을 제외하면 준수한 차트인 성적
```

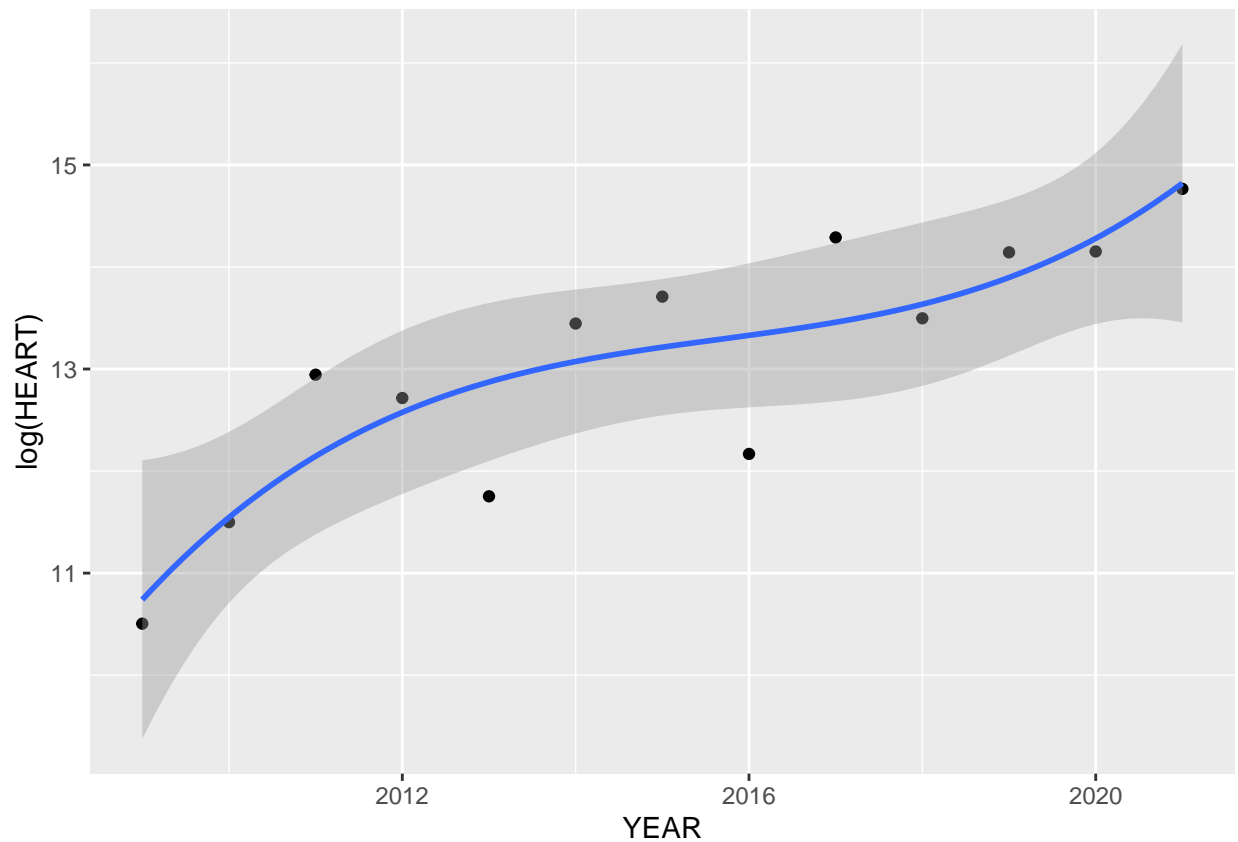
```
## # A tibble: 13 x 2  
##   YEAR HEART  
##   <dbl> <dbl>  
## 1 2009 36484  
## 2 2010 98675  
## 3 2011 418547  
## 4 2012 332969  
## 5 2013 127111  
## 6 2014 691169  
## 7 2015 899371  
## 8 2016 192534  
## 9 2017 1606544  
## 10 2018 727306  
## 11 2019 1388066  
## 12 2020 1399919  
## 13 2021 2585607
```

- %in% 대신 filter(SINGER == '아이유' | SINGER == 'IU') 써도 무방.

6-4. 아이유의 성장은 계속될까?

```
UAENA_GRAPH <- UAENA %>% # 6-3 분석 결과를
  ggplot(aes(x = YEAR, y = log(HEART))) + # 그래프로 나타내는데
  geom_point() + # 점을 찍고
  geom_smooth(formula = y ~ poly(x, 3), method = 'lm') # 3차 회귀적합도 덧붙임

UAENA_GRAPH # 라일락 대박, 다음 앨범도 기대
```



- aes는 aesthetic, 그래프 변수 및 visual에 관해 전역적으로 설정.
- log() 함수는 자연로그가 default.
- ggplot 함수, 요소는 +로 연결.