fasteR

Hyeongchan Bae

April 2022

Contents

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 무관환 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 폴더 명은 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)


.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_dectect() 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] -0.636095138  0.636442689  1.255968570 -1.487311713 -0.004260042
```

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

```
## [1]  0.6010475  2.5523689  1.5378670 -0.5917254 -1.9816716
```

- 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 = '둠칫') # 사실 sep = ' ' 인자가 숨어있던 것. 바꿀 수도 있음.
```

```
## [1] "이제와둠칫뒤늦게둠칫무엇을 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) # 얄짤없이 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',
```

```
##    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.

```
윤하
```

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

```
class(윤하)
```

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

- 데이터프레임은 row = observation, column = variable 개념이라, 열 이름을 웬만하면 채우려고 하는데, 여기선 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 Pyhton에서 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.

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>:6:42: 예상하지 못한 기호(symbol)입니다.
## 5:                  'Why are you cranky, boy? 뭘 그리 찡그려 너',
## 6:                  '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)],
                     LYLICS = IDLE_LYLICS)

TOMBOY
```

```
##   MEMBERS                                  LYLICS
## 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

1. What is tidyverse