아무나를 위한 R 튜토리얼

imkdoug@gmail.com

인사말

안녕하세요. **고려대학교 통계학과 06학번 임경덕**입니다. 저는 학부와 대학원에서 통계학을 전공했고, 카드사에서 2년 일하다가 나와서 지금은 Fastcampus 등에서 R을 활용한 데이터 분석 강의를 진행하고 있습니다.

많은 회사들이 R, Python과 같은 오픈소스 분석도구에 관심이 많고 임직원 대상 교육도 많이 진행하는데요, 이번 세미나의 주제와 실습내용들은 대부분 실제 제가 교육에서 쓰는 자료들입니다. 학교에서 배우는 R보다는 좀 더 현실적인 내용일텐데요, 다만 저도 힘들게 쌓아 만든 자료니까 소중히 다뤄 주세요.

짧은 세미나를 통해서 **데이터 분석**이라는 큰 주제를 모두 살펴보는 것은 불가능하지만 데이터 요약부터 기계학습(Machine Learning)까지 다양한 주제를 압축해서 전달드리려고 합니다.

당연히 어렵겠지만 천천히 하나씩 살펴봅시다.

이 튜토리얼에서는 **R**과 **RStudio**의 설치부터 간단한 **R** 명령어와 함수의 사용법을 다룹니다. 새로운 분석 도구를 배우는 것은 마치 새로운 외국어를 배우는 것과 같기 때문에 어려운게 당연합니다. 아래의 내용을 전부 이해하면 정말 대단한 것이고, 절반만 이해해도 수업을 따라가는 데는 전혀 지장이 없습니다. 설명을 따라 함수들을 하나씩 실행해보시고, 혹시라도 궁금한 점이 있다면 **slido**(https://app.sli.do/event/1svsaqvb)에 질문을 올려주세요 그럼 천천히 하나씩 살펴보겠습니다.

R과 RStudio 설치

먼저 이메일로 전달된 강의자료 공유폴더로 접속하고, **0_tutorial** 폴더에 있는 **0_tutorial.zip** 파일을 다운받습니다.

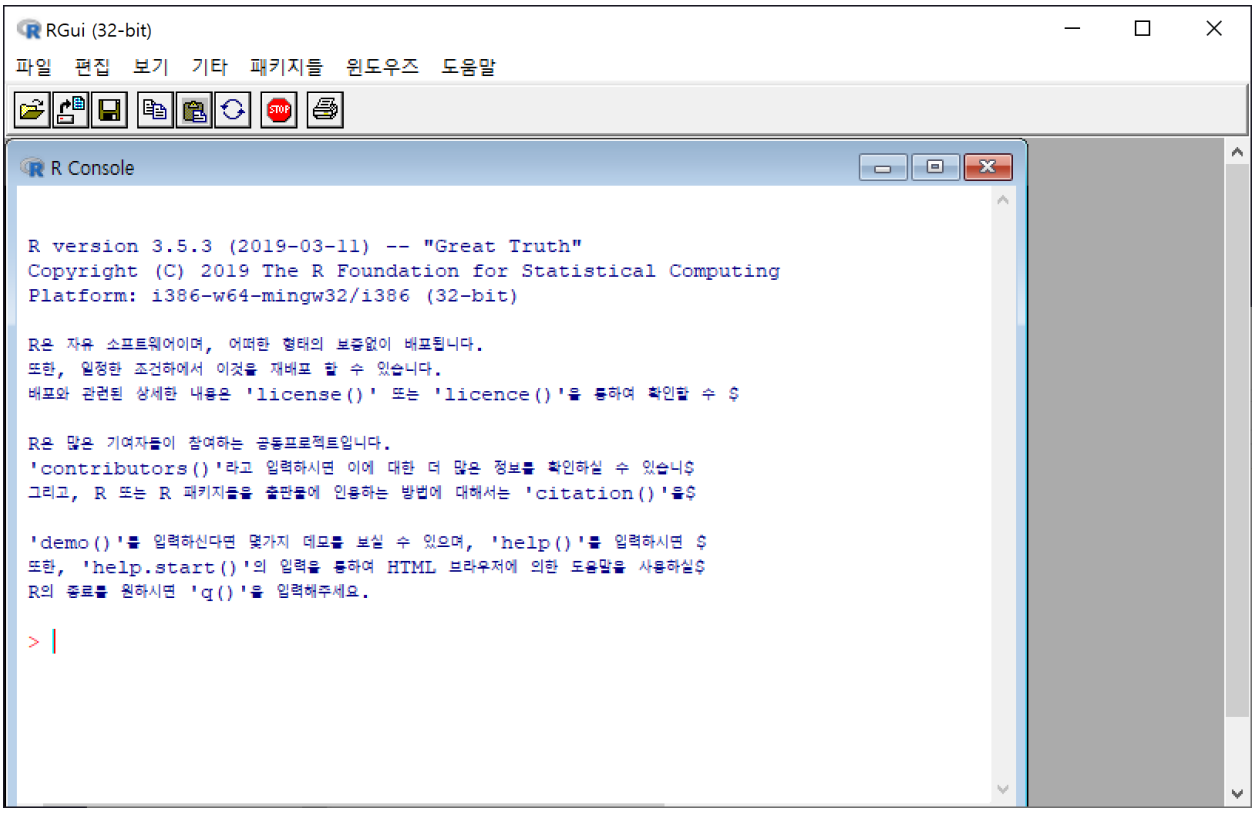
- **강의자료 다운받기(클릭) : http://bit.ly/rforyou**

(참고로 앞으로 모든 자료는 위의 공유폴더에 업로드될 예정이니 즐겨찾기에 추가하시면 편합니다!)

다운 받은 압축파일 바탕화면이나 적당한 폴더에 두시고, 반드시! 꼭! 압축을 풀어주세요.

압축을 푼 폴더로 들어가면 **"R설치파일링크"** 폴더가 있습니다. 애플의 맥(mac)사용자는 **"mac"**으로 시작하는 두개의 링크를, 윈도우(Windows) 사용자는 **"Windows"**로 시작하는 두개의 링크를 차례대로 클릭해서 **R**과 **RStudio** 설치파일을 차례로 다운 받고 설치합니다. 설치 과정에서는 무조건 "OK","Next"를 누르시면 됩니다.

우리는 **R**이라는 분석도구를 활용하는데, 이 **R**은 호주의 훌륭한 두 통계학과 교수님들께서 만든 오픈소스 분석도구입니다. 오픈소스는 곧 무료라는 말인데요, 그래서 **R**을 실행해보면 깜짝 놀랄만큼 UI/UX, 디자인과 구성이 구식입니다. 굳이 외적인 것을 신경쓸 필요가 없는 것이죠.



아무리 R이 훌륭한 도구라고 해도, 이런 실행 화면을 보면 쓰기가 싫어집니다. 그래서 보통 **RStudio**를 추가로 설치합니다. **RStudio**는 역시 훌륭한 사람들이 **R**을 더 효과적으로 사용할 수 있도록 만든 프로그램입니다. **RStudio** 또한 기본적으로는 무료라서 집에서든 학교에서든 회사에서든 마음대로 설치하고 사용할 수 있습니다. **RStudio**를 실행하면 **R**은 자동으로 백그라운드에서 실행됩니다.

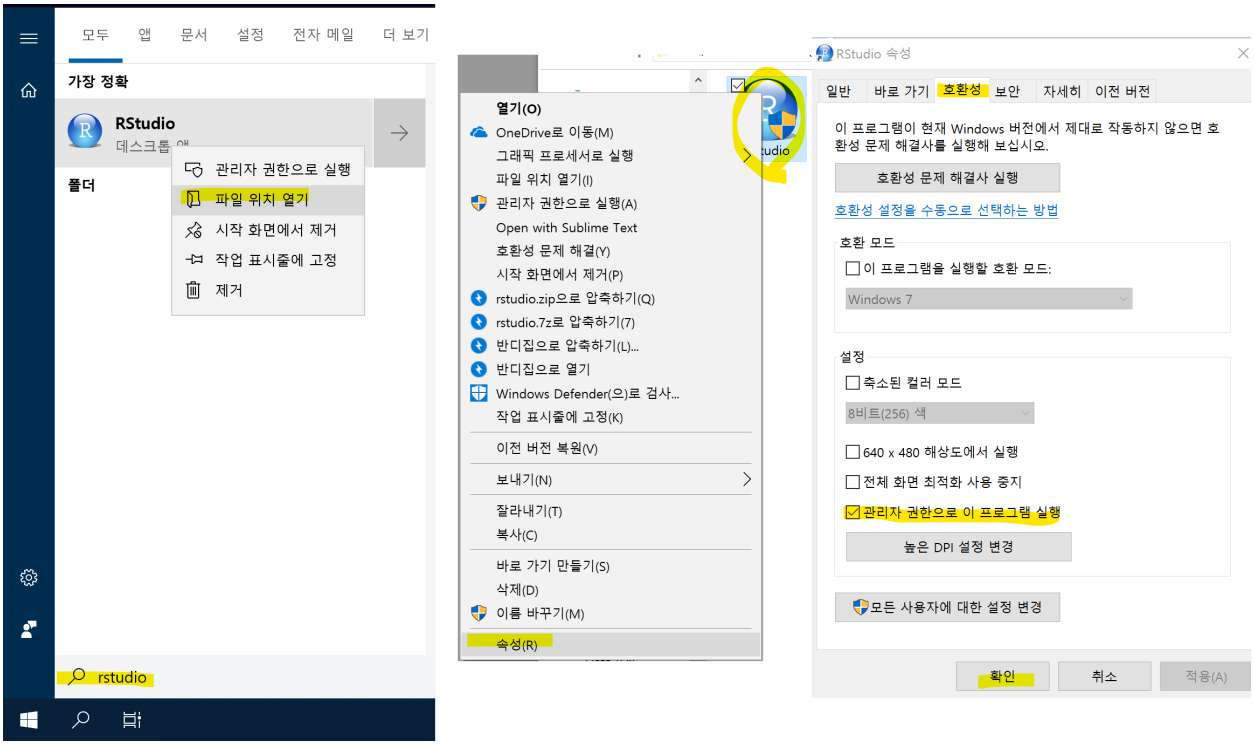
(Windows 사용자만)RStudio 설정

RStudio를 실행하기 전에 Windows 사용자 분들은 한가지 설정이 필요합니다. **RStudio**가 관리자 권한으로 실행될 수 있도록 설정을 해줘야합니다. (mac 사용자는 이 내용은 무시하셔도 됩니다.)

Windows 버전에 따라 조금 다르긴 하지만 아래의 순서로 관리자 권한으로 실행될 수 있도록 설정합니다.

---

1. 키보드의 윈도우키를 누르거나 시작 버튼을 클릭하고 검색창에 'RStudio'를 입력합니다.
2. 설치된 RStudio가 보이면 마우스로 "오른쪽"을 클릭해서 메뉴를 띄웁니다.
3. 버전에 따라 차이가 있지만 "파일 위치 열기"를 클릭합니다.
4. 실행된 탐색기에서 'RStudio'가 선택되어 있을 텐데요, 한번 더 마우스 오른쪽 클릭합니다.
5. 메뉴 제일 아래 '속성'을 클릭합니다.
6. '호환성'탭을 클릭합니다.
7. 제일 아래 '관리자 권한으로 이 프로그램 실행'에 체크하고 '확인'을 누릅니다.

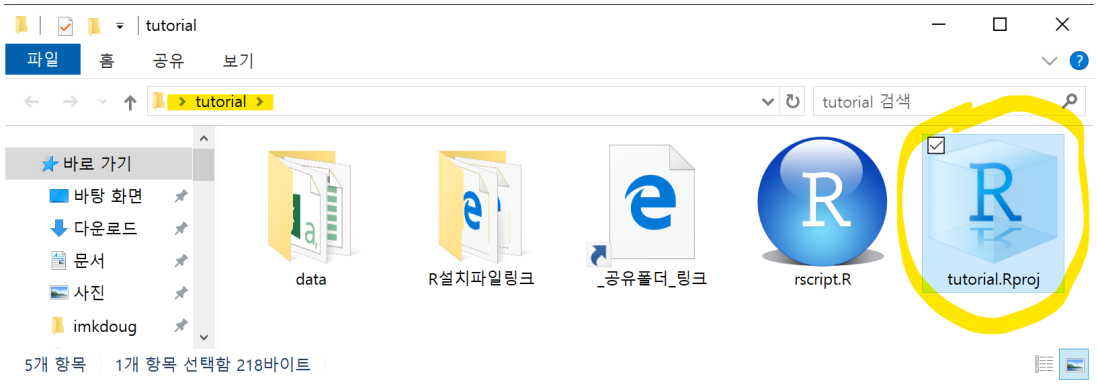


프로젝트 파일을 활용한 RStudio 실행

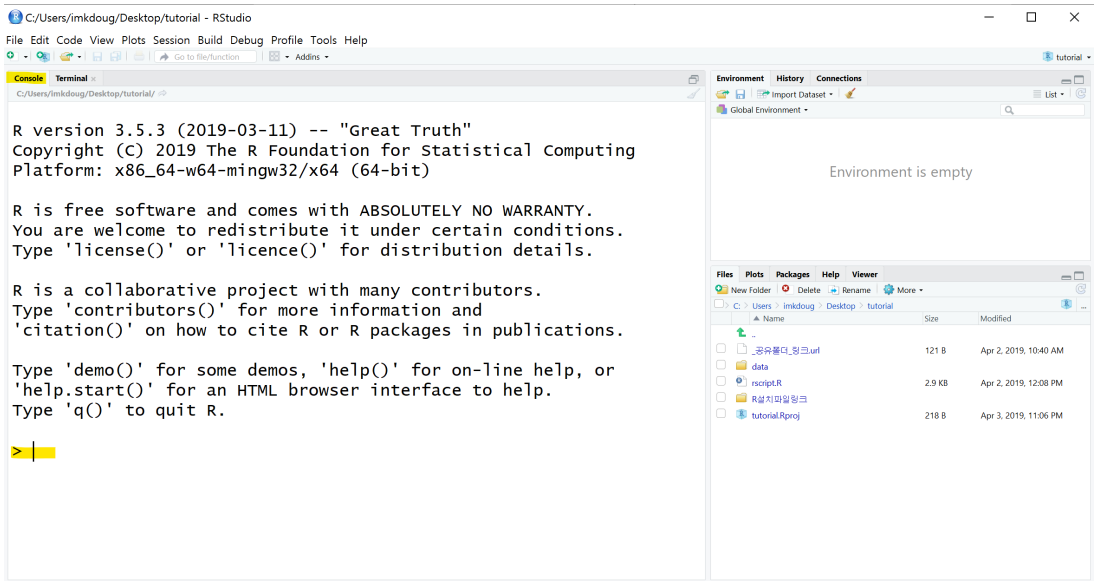
이제 설치와 설정은 모두 끝났습니다. 그럼 이제 실습을 할텐데요, 직접 RStudio를 실행하지말고 꼭 아래의 방법을 따라해 주세요. (그 이유는 첫 수업시간에 설명드리겠습니다.)

---

1. 압축을 푼 강의자료 폴더로 이동합니다.
2. 하늘색 큐브에 파란색 R이 그려진 tutorial.Rproj 파일을 실행합니다.



3. RStudio가 실행되는지 확인합니다.
4. 왼쪽 **Console**창에 커서가 깜빡이고 있는지 확인합니다. 커서가 안보이면 콘솔창을 마우스 왼쪽으로 클릭합니다.

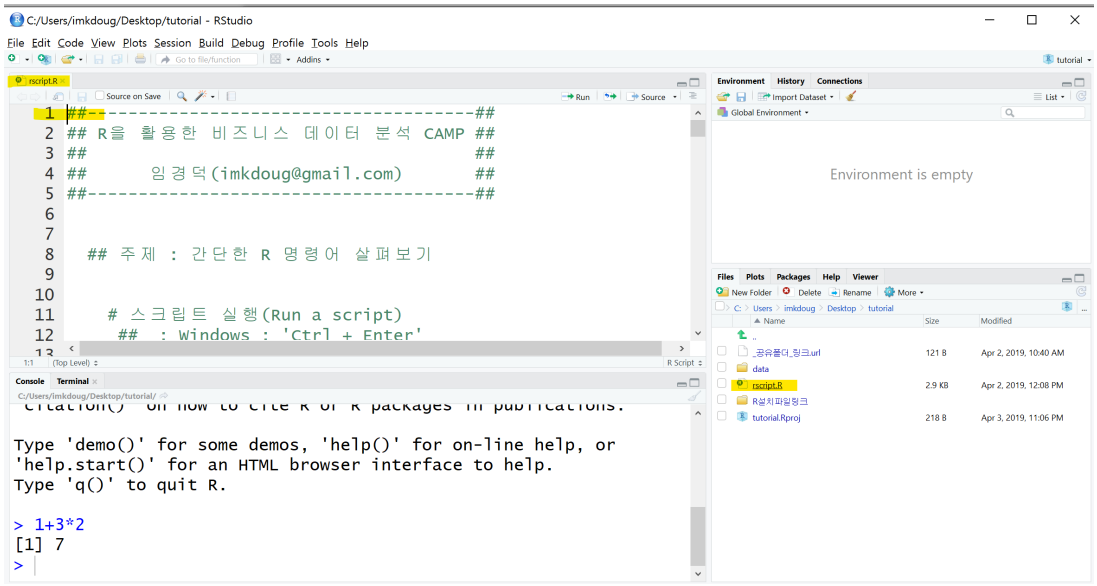


5. 콘솔창에 1+3*2를 입력하고 엔터(Enter)를 누릅니다. 그럼 그 답인 7이 출력되는 것을 볼 수 있습니다. 이처럼 콘솔창에서는 실제 명령어가 실행되고 결과를 확인할 수 있습니다. 그런데 우리가 모든 명령어를 그때 그때 입력할 수는 없기 때문에 보통 실행할 명령어를 스크립트로 작성합니다.

```
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 1+3*2
[1] 7
> |
```

6. **RStudio**화면의 오른쪽 아래에서 **rscript.R**을 클릭하면, 제가 미리 작성해둔 스크립트가 열립니다.



7. 그럼 이번에는 커서가 스크립트창에서 깜빡이고 있을 겁니다. Windows 사용자는 `Ctrl+Enter`, mac 사용자는 `Command+Enter`를 누르면 스크립트 창에 있는 명령어가 콘솔창에 복사/붙여넣기 되면서 한줄씩 차례로 실행됩니다. 그리고 결과도 콘솔창에 출력됩니다.

이제 모든 준비가 끝났습니다. 그럼 본격적으로 실습을 시작해보겠습니다.

간단한 R 명령어 설명

정상적으로 RStudio가 실행되고, 스크립트가 불러와졌다면 이제 한줄씩 명령어를 실행하면서 그 의미를 살펴보겠습니다.

숫자와 글자

어떤 분석도구든 숫자와 사칙연산의 활용방법은 다르지 않습니다. R에서도 키보드에 있는 숫자와 +, -, *, / 같은 연산자를 활용해서 원하는 사칙연산을 실행하고 값을 계산할 수 있습니다.

```
1+(2*3)/4-5
```

```
## [1] -2.5
```

그런데 글자는 조금 다릅니다. R에서 글자는 반드시 따옴표로 감싸야합니다. 큰따옴표("")와 작은따옴표('') 중에 편한 것으로 사용하면 됩니다. (글자하나는 계산할 것도, 작업할 것도 없기 때문에 입력한 그대로 출력되는 것을 볼 수 있습니다.)

```
"d"
```

```
## [1] "d"
```

```
'd'
```

```
## [1] "d"
```

만약 따옴표를 감싸지 않으면 어떻게 될까요?

```
d
```

```
## Error in eval(expr, envir, enclos):   'd'
```

콘솔창을 잘보면 "Error"라는 문구가 있는 것을 볼 수 있습니다. 뭔가 문제가 생겼습니다. 따옴표없이 글자를 적으면 그 이름을 가진 데이터나 함수, 객체(object)를 찾습니다. 그런데 아무리 찾아봐도 d라는 이름을 가진 무언가가 없기 때문에 에러가 난 것을 볼 수 있습니다. 객체의 이름이 아닌 진짜 눈에 보이는 그대로 글자를 표현할 때는 꼭 따옴표를 붙여야 합니다.

숫자열과 문자열

데이터 분석을 하다보면 숫자와 문자를 여러개를 한꺼번에 붙여서 표현해야할 때가 많습니다. 예를 들어 우리가 관심있는 연령대가 20, 30대라고 하면 나이를 기준으로 20과 39라는 두 숫자로 나이의 범위를 지정해야하고, 관심있는 지역이 따로 있다면 '서울특별시'와 '부산광역시'처럼 여러개 글자를 한 묶음으로 표현해야할 수 있어야 합니다. 이렇게 숫자나 글자의 묶음을 만들어 주는 함수가 바로 c( )입니다. c( )라는 함수에 콤마(,)를 활용해서 여러개의 값을 넣으면 하나로 묶입니다.

```r
c(1, 3, 10)
```

```
## [1]  1  3 10
```

```r
c('A', 'B', 'DDD', '123')
```

```
## [1] "A"   "B"   "DDD" "123"
```

숫자의 경우, 좀 더 다양한 방법으로 나열할 수도 있습니다. 1부터 12까지, 2011부터 2019까지 1씩 커지는 수열의 경우 콜론(:)을 활용할 수 있습니다.

```r
11:15
```

```
## [1] 11 12 13 14 15
```

```r
2011:2019
```

```
## [1] 2011 2012 2013 2014 2015 2016 2017 2018 2019
```

```r
10:1
```

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

:앞에 있는 숫자부터 :뒤에 있는 숫자까지 1씩 커지거나 작아지는 수열이 만들어집니다.

rep( )이나 seq( )같은 함수를 활용할 수도 있습니다. rep( )은 반복(repeat)을 해주는 함수인데요, 콤마를 기준으로 먼저 들어간 값이 **반복할 값**, 두번째로 들어간 값이 **반복할 횟수**가 됩니다. 출력된 결과를 보면 3이 10번 반복되는 것을 볼 수 있습니다.

```r
rep(3, 10)
```

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

seq( )는 등차수열을 만들어 줍니다. seq( )에는 세개의 입력값(input)이 필요한데요, 순서대로 시작값(from), 끝값(to), 간격(by)입니다. 예를 들어 seq(0, 10, 3)을 실행하면 0부터 10까지 3 간격으로 등차수열이 만들어집니다.

```
## [1] 0 3 6 9
```

이렇게 다양한 함수를 활용해서 좀 더 효율적으로 숫자열과 문자열을 만들 수 있습니다.

대괄호를 활용한 인덱스

따옴표 없이 LETTERS를 실행하면 오류가 뜰 것 같지만 무언가 출력이 됩니다. 우리가 만들진 않았지만 R이 내부적으로 저장하고 있던 A부터 Z까지 알파벳 대문자 26개가 출력이 됩니다.

```r
LETTERS
```

```
##  [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q"
## [18] "R" "S" "T" "U" "V" "W" "X" "Y" "Z"
```

그런데 다음 명령어를 살펴보면 이 LETTERS뒤에 대괄호([])를 붙여놨습니다.

```
LETTERS[]
```

이 명령어를 그냥 실행하지말고 대괄호 안에다가 숫자 하나 혹은 숫자열을 넣고 실행해봅시다.

```
LETTERS[17]
```

```
## [1] "Q"
```

```
LETTERS[c(5,9,13)]
```

```
## [1] "E" "I" "M"
```

대괄호에 17을 넣고 실행하면 26개 알파벳 중에 17번째인 "Q"가, c( )로 묶어 5, 9, 13을 넣고 실행하면 5번째, 9번째, 13번째 알파벳인 "E", "I", "M"이 출력되는 것을 볼 수 있습니다. 이렇게 대괄호는 인덱스(index)에 활용되고 대괄호에 적절한 값을 넣어 전체 중에서 부분을 추출할 수 있습니다.

다양한 논리연산

다음은 논리연산입니다. R에서 어떤 명령어를 실행했을 때 TRUE 혹은 FALSE의 결과를 주는 것들을 논리연산이라고 부르는데요, TRUE는 참, 조건과 일치함을 의미하고 FALSE는 반대로 거짓, 조건과 일치하지 않음을 의미합니다. 예제를 살펴보겠습니다.

1:10은 1부터 10까지 열 개 숫자를 만듭니다. 그 뒤에 다양한 연산자를 붙여 논리연산을 할 수 있습니다. 예를 들어 ==은 **일치**하는지 아닌지 논리연산을 합니다. 10개의 숫자 중에서 7과 일치하는 것은 당연히 7밖에 없습니다. 그래서 10개의 결과값 중 7번째만 TRUE이고 나머지는 FALSE가 나옵니다.

```
1:10 == 7
```

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

!=는 반대로 **불일치**하는지 아닌지 논리연산을 합니다. 이번엔 7번째만 FALSE, 나머지는 TRUE가 출력됩니다. >=과 같이 부등호를 활용해서 7이상인지 아닌지 논리연산을 할 수도 있습니다.

```
1:10 != 7
```

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

```
1:10 >= 7
```

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

등호와 부등호 말고 다양한 논리연산자가 있습니다. 예를 들어 %in%는 뒤에 나오는 목록에 **포함**되는지 아닌지 논리연산을 합니다. 10개의 숫자 중에서 뒤에 나오는 5개 숫자 목록에 포함되는 것은 TRUE 아닌 것은 FALSE의 결과가 나오는 것을 확인할 수 있습니다.

```
1:10 %in% c(1,3,5,7,9)
```

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

이런 논리연산은 나중에 조건과 일치하는 관측치 부분을 선택할 때 유용하게 활용됩니다.

=을 활용한 저장

데이터 분석을 위해서는 R로 데이터를 불러와서 저장하고, 중간 과정이나 결과도 저장을 해야합니다. R에서는 =을 써서 특정한 이름으로 값을 저장하고, 객체를 만들 있습니다. 말은 어렵지만 예제는 간단합니다.

```
x=1:5
```

x라는 이름으로 1:5, 1부터 5까지 숫자 다섯개를 저장합니다. 우리가 실행한 명령어가 "저장해라"기 때문에 별다른 반응은 없지만,

```
x
```

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

이후 따옴표없이 x를 실행하면 x라는 이름으로 저장된 5개의 숫자가 출력됩니다.

```
y = c('A', 'B', 'B', 'C', 'B')
y
```

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

당연히 글자도 저장할 수 있고, 분석과정에서의 모든 것을 저장할 수 있습니다.

범주형 변수의 활용

데이터는 보통 변수로 구성되어 있는데, 변수는 크게 수치형(numerical) 변수와 범주형(categorical) 변수로 나뉩니다. 예를 들어 성별이라는 변수가 있다고 하면, 사람이 10명이든 100명이든 1000000명이든 상관없이 "남자"와 "여자" 둘중 하나의 값을 가지게 될텐데요, 이렇게 관측치들이 한정적인 몇개의 값을 가지는 변수가 범주형 변수입니다. R에서는 **Factor**라는 형식으로 범주형 변수를 다룹니다.

```
factor(y)
```

```
## [1] A B B C B
## Levels: A B C
```

위에서 만든 y라는 값을 `factor( )` 함수에 넣으면 범주형 변수 형식, Factor 형식으로 바뀌는데요, 콘솔창에 출력된 결과를 보면 **Levels**라는 부분이 보입니다. y에는 5개의 값이 있지만 모두 값이 다른 것은 아니고 A, B, C 중 하나의 값을 가지죠. 그럼 이 A, B, C가 y의 수준(levels)이 됩니다.

한가지 조심해야할 것은 `factor(y)`를 실행했다고 y가 범주형 변수로 바뀌어 업데이트되는 것은 아니라는 점입니다. 우리가 실행한 명령은 "y를 Factor 형식으로 바꿔서 콘솔창에 출력해줘"기 때문인데요, 실제로 y를 업데이트를 하는 것도 가능합니다.

```
y = factor(y)
y
```

```
## [1] A B B C B
## Levels: A B C
```

y를 factor( )에 넣어 형식을 바꾼다음 =을 써서 y로 저장하면, 원래 있던 y를 업데이트 할 수 있습니다. 그럼 이제 y는 단순한 5개 글자의 묶음이라기 보다는 어떤 범주형 변수라고 볼 수 있죠.

이 Factor 형식의 변수를 다룰 때는 levels( )라는 함수를 많이 활용합니다. levels( )에 Factor 형식을 넣고 실행하면, 수준만 따로 확인 가능합니다.

```
levels(y)
```

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

그런데 하나 더 특징은, 이 수준도 업데이트가 가능하다는 것입니다. 역시 =을 써서 새로운 수준 값을 지정해주면 모든 관측치가 한꺼번에 새로운 수준값으로 바뀌는 것을 볼수 있습니다.

```
## [1] 4.0 3.0 3.0 2.0 3.0
## Levels: 4.0 3.0 2.0
```

데이터 불러오기

이제 조금 더 복잡한 내용인데요, 바로 데이터 불러와서 저장하기 입니다. 여러 종류의 데이터가 있고, 데이터마다 저장된 형식에 따라 확장자는 다를 수 있는데요, 일반적으로 어느 프로그램에서든 활용하기 편한 **csv** 파일을 많이 활용합니다.

우리의 실습 폴더에는 data라는 하위 폴더가 있고, 그 안에 **test.csv**라는 파일이 있습니다. **csv** 파일은 read.csv( )라는 함수로 불러올 수 있는데요, 함수안에 파일의 경로와 이름을 잘 적어줍니다. 폴더는 슬래시(/)로 구분합니다.

```
read.csv('data/test.csv')
```

```
##    age    sex    bmi children smoker    region   charges
## 1   19 female 27.900        0    yes southwest 16884.924
## 2   18   male 33.770        1     no southeast  1725.552
## 3   28   male 33.000        3     no southeast  4449.462
## 4   33   male 22.705        0     no northwest 21984.471
## 5   32   male 28.880        0     no northwest  3866.855
## 6   31 female 25.740        0     no southeast  3756.622
## 7   46 female 33.440        1     no southeast  8240.590
## 8   37 female 27.740        3     no northwest  7281.506
## 9   37   male 29.830        2     no northeast  6406.411
## 10  60 female 25.840        0     no northwest 28923.137
```

```
## 11  25   male 26.220        0     no northeast  2721.321
## 12  62 female 26.290        0    yes southeast 27808.725
## 13  23   male 34.400        0     no southwest  1826.843
## 14  56 female 39.820        0     no southeast 11090.718
## 15  27   male 42.130        0    yes southeast 39611.758
## 16  19   male 24.600        1     no southwest  1837.237
## 17  52 female 30.780        1     no northeast 10797.336
## 18  23   male 23.845        0     no northeast  2395.172
## 19  56   male 40.300        0     no southwest 10602.385
## 20  30   male 35.300        0    yes southwest 36837.467
```

`read.csv( )`를 실행해보면 데이터를 잘 불러오는 것을 볼 수 있는데요, 지금은 콘솔창에 데이터를 바로 출력하고 끝났습니다. 그런데 우리는 이런 데이터로 분석을 해야하기 때문에 적당한 이름으로 저장을 합니다.

```
test_data = read.csv('data/test.csv')
```

예를 들어 방금의 **test.csv** 데이터는 test_data라는 이름으로 저장했는데요, 그럼 이후에는 `test_data`를 실행할 때마다 저장된 데이터가 출력됩니다.

```
test_data
```

```
##    age    sex    bmi children smoker    region   charges
## 1   19 female 27.900        0    yes southwest 16884.924
## 2   18   male 33.770        1     no southeast  1725.552
## 3   28   male 33.000        3     no southeast  4449.462
## 4   33   male 22.705        0     no northwest 21984.471
## 5   32   male 28.880        0     no northwest  3866.855
## 6   31 female 25.740        0     no southeast  3756.622
## 7   46 female 33.440        1     no southeast  8240.590
## 8   37 female 27.740        3     no northwest  7281.506
## 9   37   male 29.830        2     no northeast  6406.411
## 10  60 female 25.840        0     no northwest 28923.137
## 11  25   male 26.220        0     no northeast  2721.321
## 12  62 female 26.290        0    yes southeast 27808.725
## 13  23   male 34.400        0     no southwest  1826.843
## 14  56 female 39.820        0     no southeast 11090.718
## 15  27   male 42.130        0    yes southeast 39611.758
## 16  19   male 24.600        1     no southwest  1837.237
## 17  52 female 30.780        1     no northeast 10797.336
## 18  23   male 23.845        0     no northeast  2395.172
## 19  56   male 40.300        0     no southwest 10602.385
## 20  30   male 35.300        0    yes southwest 36837.467
```

데이터 살펴보기

이렇게 불러와서 저장한 데이터는 여러 함수에 집어넣어서 특징을 살펴보게 됩니다. 먼저 head( )와 tail( )을 활용해서 첫 n개 관측치나 마지막 n개 관측치를 확인할 수 있습니다. n의 기본값은 6이라서 따로 지정하지 않으면 6개 관측치가 콘솔창에 출력됩니다.

```
head(test_data)
```

```
##   age    sex    bmi children smoker    region   charges
## 1  19 female 27.900        0    yes southwest 16884.924
## 2  18   male 33.770        1     no southeast  1725.552
## 3  28   male 33.000        3     no southeast  4449.462
## 4  33   male 22.705        0     no northwest 21984.471
## 5  32   male 28.880        0     no northwest  3866.855
## 6  31 female 25.740        0     no southeast  3756.622
```

```
tail(test_data, n=10)
```

```
##    age    sex    bmi children smoker    region   charges
## 11  25   male 26.220        0     no northeast  2721.321
## 12  62 female 26.290        0    yes southeast 27808.725
## 13  23   male 34.400        0     no southwest  1826.843
## 14  56 female 39.820        0     no southeast 11090.718
## 15  27   male 42.130        0    yes southeast 39611.758
## 16  19   male 24.600        1     no southwest  1837.237
## 17  52 female 30.780        1     no northeast 10797.336
## 18  23   male 23.845        0     no northeast  2395.172
## 19  56   male 40.300        0     no southwest 10602.385
## 20  30   male 35.300        0    yes southwest 36837.467
```

names( )는 데이터에 있는 변수의 이름을 알려줍니다. 데이터 분석은 일반적으로 변수 단위로 작업하는 경우가 많기 때문에 데이터 속 변수의 이름은 알아두는 것이 좋습니다.

```
names(test_data)
```

```
## [1] "age"      "sex"      "bmi"      "children" "smoker"   "region"
## [7] "charges"
```

이것말고도 nrow( ), ncol( )과 같은 함수에 데이터를 넣어 관측치 수, 변수의 수를 확인하는 등 다양한 함수를 활용해서 데이터의 특징을 살펴볼 수 있습니다.

```
nrow(test_data)
```

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

```
ncol(test_data)
```

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

데이터와 인덱스

앞에서 대괄호([])는 무언가 저장된 객체에서 부분을 선택할 때 활용한다는 것을 살펴봤습니다. 이 대괄호는 데이터에도 붙여서 부분 데이터 선택에 활용할 수 있습니다.

단, LETTERS' test_data'같은 일반적인 데이터는 가로와 세로, 관측치와 변수 형태로 구성되어 있습니다. 2차원 형태인 것이죠. 그래서 대괄호에 숫자하나만 넣어서는 위치를 지정하는데 어려움이 있습니다.

아래의 스크립트를 살펴보면 대괄호 사이에 콤마가 하나 포함되어 있습니다. 그럼 대괄호 안에 뭔가 두개를 집어넣을 수 있을텐데요, 일단은 콤마 앞과 뒤 모두 비어 있습니다. 비어 있으면 전체를 의미합니다. 그래서 모든 관측치와 변수가 선택됩니다.

```
test_data[ , ]
```

```
##    age    sex    bmi children smoker    region   charges
## 1   19 female 27.900        0    yes southwest 16884.924
## 2   18   male 33.770        1     no southeast  1725.552
## 3   28   male 33.000        3     no southeast  4449.462
## 4   33   male 22.705        0     no northwest 21984.471
## 5   32   male 28.880        0     no northwest  3866.855
## 6   31 female 25.740        0     no southeast  3756.622
## 7   46 female 33.440        1     no southeast  8240.590
## 8   37 female 27.740        3     no northwest  7281.506
## 9   37   male 29.830        2     no northeast  6406.411
## 10  60 female 25.840        0     no northwest 28923.137
## 11  25   male 26.220        0     no northeast  2721.321
## 12  62 female 26.290        0    yes southeast 27808.725
## 13  23   male 34.400        0     no southwest  1826.843
## 14  56 female 39.820        0     no southeast 11090.718
## 15  27   male 42.130        0    yes southeast 39611.758
## 16  19   male 24.600        1     no southwest  1837.237
## 17  52 female 30.780        1     no northeast 10797.336
## 18  23   male 23.845        0     no northeast  2395.172
## 19  56   male 40.300        0     no southwest 10602.385
## 20  30   male 35.300        0    yes southwest 36837.467
```

이번에는 콤마앞에 1을 넣었습니다. 콤마 앞은 관측치를 의미하는데요, 그럼 아래의 명령어는 "1번 관측치만 보자"는 의미가 됩니다. 만약 c(1,3,5)를 집어 넣는다면 1,3,5번째 관측치만 보자는 의미가 되구요.

```
test_data[1, ]
```

```
##   age    sex  bmi children smoker    region  charges
## 1  19 female 27.9        0    yes southwest 16884.92
```

```
test_data[c(1,3,5), ]
```

```
##   age    sex   bmi children smoker    region   charges
## 1  19 female 27.90        0    yes southwest 16884.924
## 3  28   male 33.00        3     no southeast  4449.462
## 5  32   male 28.88        0     no northwest  3866.855
```

콤마 뒤에 숫자를 넣을 수도 있습니다. 콤마 뒤는 변수를 의미하는데요, 아래의 명령어는 1부터 3까지 세 변수를 선택한다는 의미가 됩니다.

```
test_data[,1:3]
```

```
##    age    sex    bmi
## 1   19 female 27.900
## 2   18   male 33.770
## 3   28   male 33.000
## 4   33   male 22.705
## 5   32   male 28.880
## 6   31 female 25.740
## 7   46 female 33.440
## 8   37 female 27.740
## 9   37   male 29.830
## 10  60 female 25.840
## 11  25   male 26.220
## 12  62 female 26.290
## 13  23   male 34.400
## 14  56 female 39.820
## 15  27   male 42.130
## 16  19   male 24.600
## 17  52 female 30.780
## 18  23   male 23.845
## 19  56   male 40.300
## 20  30   male 35.300
```

$를 활용한 변수 선택과 요약함수의 활용

그런데 보통 변수를 하나만 선택할 때가 많고, 변수의 순서보다는 이름을 많이 활용합니다. 그래서 $를 활용해서 변수를 선택하는 경우가 많습니다.

```
test_data$age
```

```
##  [1] 19 18 28 33 32 31 46 37 37 60 25 62 23 56 27 19 52 23 56 30
```

test_data$age에서 test_data는 데이터 이름, age는 변수이름인 것을 볼 수 있습니다. 이렇게 **데이터이름$변수이름**의 형태로 특정한 변수를 선택 가능합니다.

곧 하나씩 배우겠지만, 이렇게 선택한 변수는 요약해주는 함수에 집어 넣을 수 있습니다. 선택한 변수 age를 mean( )에 넣으면 평균 나이가 계산되고, summary( )에 넣으면 최댓값, 최솟값, 중앙값 등 다양한 요약값이 한번에 계산됩니다. hist( )에 넣으면 변수 age의 분포, 나이 분포를 살펴볼수 있는 히스토그램을 그리는 것도 가능합니다.

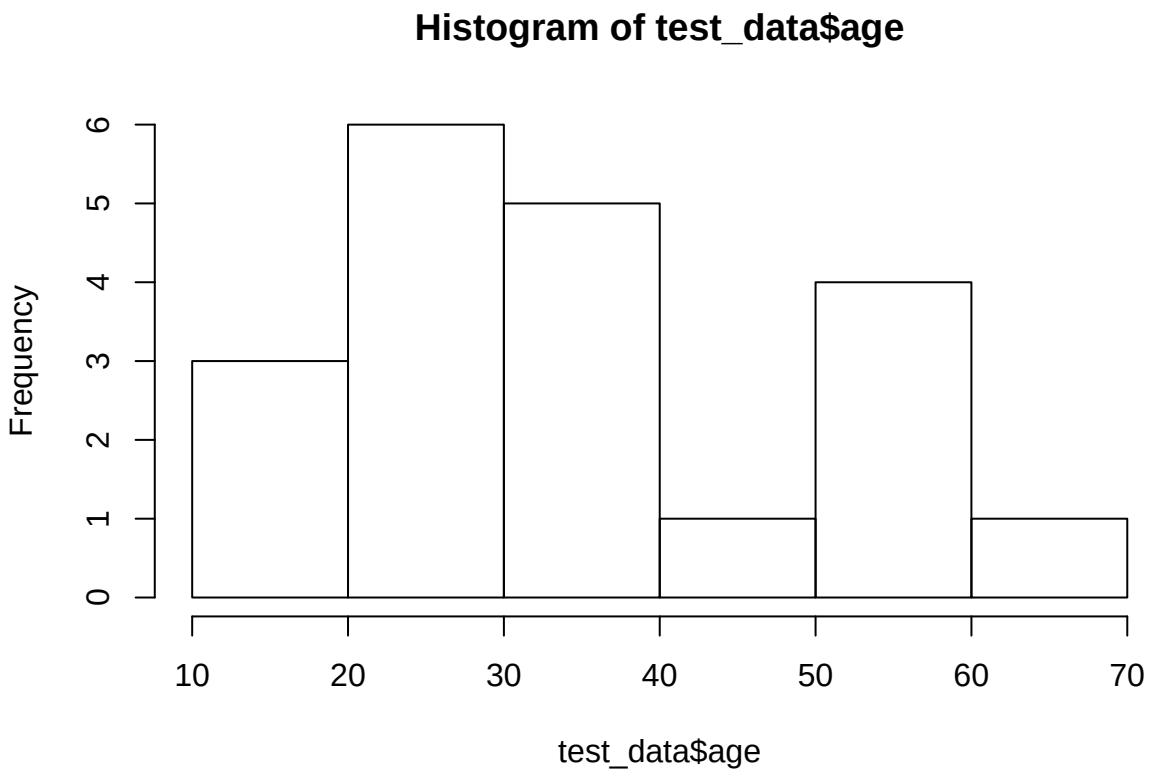
```
mean(test_data$age)
```

```
## [1] 35.7
```

```
summary(test_data$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    18.0    24.5    31.5    35.7    47.5    62.0
```

```
hist(test_data$age)
```



변수를 추가하고 업데이트 하는 것도 $를 활용합니다. 예를 들어 test_data$count를 실행해보면 오류가 뜹니다. 우리가 불러온 데이터에는 count라는 변수가 없기 때문이죠.

```
test_data$count
```

```
## NULL
```

그런데 =을 붙이고 뒤에 새로운 값을 넣고 실행하면 변수가 추가가 됩니다. 원래 있던 변수말고 count라는 이름으로 새 변수가 만들어 집니다.

```
test_data$count = 1:20
test_data
```

```
##    age    sex    bmi children smoker    region   charges count
## 1   19 female 27.900        0    yes southwest 16884.924     1
## 2   18   male 33.770        1     no southeast  1725.552     2
## 3   28   male 33.000        3     no southeast  4449.462     3
## 4   33   male 22.705        0     no northwest 21984.471     4
## 5   32   male 28.880        0     no northwest  3866.855     5
## 6   31 female 25.740        0     no southeast  3756.622     6
## 7   46 female 33.440        1     no southeast  8240.590     7
## 8   37 female 27.740        3     no northwest  7281.506     8
## 9   37   male 29.830        2     no northeast  6406.411     9
## 10  60 female 25.840        0     no northwest 28923.137    10
## 11  25   male 26.220        0     no northeast  2721.321    11
## 12  62 female 26.290        0    yes southeast 27808.725    12
## 13  23   male 34.400        0     no southwest  1826.843    13
## 14  56 female 39.820        0     no southeast 11090.718    14
## 15  27   male 42.130        0    yes southeast 39611.758    15
## 16  19   male 24.600        1     no southwest  1837.237    16
## 17  52 female 30.780        1     no northeast 10797.336    17
## 18  23   male 23.845        0     no northeast  2395.172    18
## 19  56   male 40.300        0     no southwest 10602.385    19
## 20  30   male 35.300        0    yes southwest 36837.467    20
```

기존 변수의 값 업데이트도 가능합니다.

```
test_data$count = test_data$count + 100
test_data
```

```
##    age    sex    bmi children smoker    region   charges count
## 1   19 female 27.900        0    yes southwest 16884.924   101
## 2   18   male 33.770        1     no southeast  1725.552   102
## 3   28   male 33.000        3     no southeast  4449.462   103
## 4   33   male 22.705        0     no northwest 21984.471   104
## 5   32   male 28.880        0     no northwest  3866.855   105
## 6   31 female 25.740        0     no southeast  3756.622   106
## 7   46 female 33.440        1     no southeast  8240.590   107
## 8   37 female 27.740        3     no northwest  7281.506   108
## 9   37   male 29.830        2     no northeast  6406.411   109
## 10  60 female 25.840        0     no northwest 28923.137   110
## 11  25   male 26.220        0     no northeast  2721.321   111
## 12  62 female 26.290        0    yes southeast 27808.725   112
## 13  23   male 34.400        0     no southwest  1826.843   113
## 14  56 female 39.820        0     no southeast 11090.718   114
## 15  27   male 42.130        0    yes southeast 39611.758   115
```

```
## 16  19   male 24.600        1     no southwest  1837.237   116
## 17  52 female 30.780        1     no northeast 10797.336   117
## 18  23   male 23.845        0     no northeast  2395.172   118
## 19  56   male 40.300        0     no southwest 10602.385   119
## 20  30   male 35.300        0    yes southwest 36837.467   120
```

count변수에 일괄적으로 100을 더해서 다시 count 변수로 저장하면, 기존의 값이 모두 업데이트가 됩니다. 이렇게 $와 =는 데이터를 처리할 때 너무나 중요한 역할을 합니다.

패키지(package) 설치

R 설치 용량은 채 100Mb가 되지 않습니다. 우리가 R을 설치하면 아주 기본적인 함수와 기능만을 활용할 수 있는데요, 패키지를 추가로 설치하면 뭔거 더 새롭거나 효율적인 작업을 할 수 있습니다. 예를 들어 openxlsx이라는 패키지를 설치하면 내 컴퓨터에 **Excel** 설치가 안되어 있어도 R로 .xlsx의 Excel 데이터를 R로 불러올 수 있습니다.

패키지를 설치할 때는 install.packages( ) 함수를 활용합니다.

```
install.packages('openxlsx')
```

install.packages( ) 함수안에 설치하고 싶은 패키지 이름을 넣고 실행하면, 서버에서 해당 패키지 압축파일을 다운로드하고 내 컴퓨터에 압축을 풀어 기능이 추가됩니다.

단, 패키지를 설치했다고 해서 기능을 바로 활용할 수 있는 것은 아니고 library( ) 함수로 패키지를 한번 불러와야합니다.

```
library(openxlsx)
read.xlsx('data/test_xlsx.xlsx')
```

```
##   age    sex region
## 1  19   Male  Seoul
## 2  18 Female  Busan
## 3  28   Male  Seoul
## 4  33   Male  Busan
## 5  32 Female  Seoul
```

예를 들어 library(openxlsx)로 방금 설치한 openxlsx패키지를 불러오면 기존에는 없었던 read.xlsx( ) 함수를 활용할 수 있고, 역시 함수안에 불러올 Excel 파일의 경로와 이름을 지정해주면 Excel 파일도 불러올 수 있습니다. 이 함수에 대해서는 수업시간에 한번더 다루도록 하겠습니다.

간단한 데이터 분석 예제

그럼 이번에는 R을 활용한 간단한 데이터 분석 예제를 살펴보겠습니다. 데이터를 불러와서 요약하고 시각화하는 과정을 하나씩 함께 실행해봅시다.

데이터 불러오기

보통 대부분의 비즈니스 데이터는 csv 형식이나 xlsx 형식으로 저장되어 있습니다. 그럼 앞에서 살펴본 `read.csv( )`나 `openxlsx` 패키지의 `read.xlsx( )`을 활용해서 R로 불러올 수 있습니다. 아래의 명령어로 `data` 폴더에 있는 `StudentsPerformance.csv` 파일을 불러와서 `scores`라는 이름으로 저장합니다.

```
scores = read.csv('data/StudentsPerformance.csv')
```

데이터 확인하기

이렇게 불러온 데이터는 `head( )`나 `tail( )`로 제일 앞이나 뒤의 몇개 관측치를 확인해봅니다.

```
head(scores)
```

```
##   gender race.ethnicity parental.level.of.education        lunch
## 1 female        group B           bachelor's degree     standard
## 2 female        group C                some college     standard
## 3 female        group B             master's degree     standard
## 4   male        group A          associate's degree free/reduced
## 5   male        group C                some college     standard
## 6 female        group B          associate's degree     standard
##   test.preparation.course math.score reading.score writing.score
## 1                    none         72            72            74
## 2               completed         69            90            88
## 3                    none         90            95            93
## 4                    none         47            57            44
## 5                    none         76            78            75
## 6                    none         71            83            78
```

```
tail(scores)
```

```
##      gender race.ethnicity parental.level.of.education        lunch
## 995    male        group A                high school     standard
## 996  female        group E             master's degree     standard
```

```
## 997    male        group C                high school free/reduced
## 998  female        group C                high school free/reduced
## 999  female        group D              some college     standard
## 1000 female        group D              some college free/reduced
##      test.preparation.course math.score reading.score writing.score
## 995                     none         63            63            62
## 996                completed         88            99            95
## 997                     none         62            55            55
## 998                completed         59            71            65
## 999                completed         68            78            77
## 1000                    none         77            86            86
```

공간이 좁아서 줄바꿈이 되긴 했지만 몇 개의 관측치들을 포함한 부분 데이터를 살펴보면, 데이터의 구조와 특성을 살펴볼 수 있습니다.

```
nrow(scores)
```

```
## [1] 1000
```

```
ncol(scores)
```

```
## [1] 8
```

```
names(scores)
```

```
## [1] "gender"                       "race.ethnicity"
## [3] "parental.level.of.education" "lunch"
## [5] "test.preparation.course"     "math.score"
## [7] "reading.score"               "writing.score"
```

이외에도 `nrow( )`로 1000개 관측치가 있다는 것과 `ncol( )`로 8개의 변수가 있다는 걸 확인할 수 있고, `names( )`로 이 변수들의 이름을 확인할 수 있습니다.

이 데이터는 1000명 학생들의 수학(`math.score`), 읽기(`reading.score`), 쓰기(`writing.score`) 점수를 정리한 것인데요, 학생의 성별(`gender`), 인종/민족(`race.ethnicity`), 부모의 교육수준(`parental.level.of.ed`
가계 소득과 관련이 있는 점심 지원 여부(`lunch`) 그리고 시험 준비 여부(`test.preparation.course`)에 따라 각 성적이 어떻게 달라지는지 분석해볼 수 있습니다. 그럼 **R**의 기본 함수들을 활용해서 이 데이터를 요약하고 인사이트를 찾아봅시다.

한 수치형 변수의 요약과 시각화

scores뒤에 $를 붙이고 수학성적 math.score 변수를 선택합니다.그리고 min( ), max( ), mean( ), median( ) 같은 함수들을 활용해서 최솟값, 최댓값, 평균, 중앙값 같은 다양한 요약값을 계산할 수 있습니다. 수학 점수는 0점부터 100점까지 넓게 흩어져 있고 평균과 중앙값은 66점 정도네요.

```
min(scores$math.score)
```

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

```
max(scores$math.score)
```

```
## [1] 100
```

```
mean(scores$math.score)
```

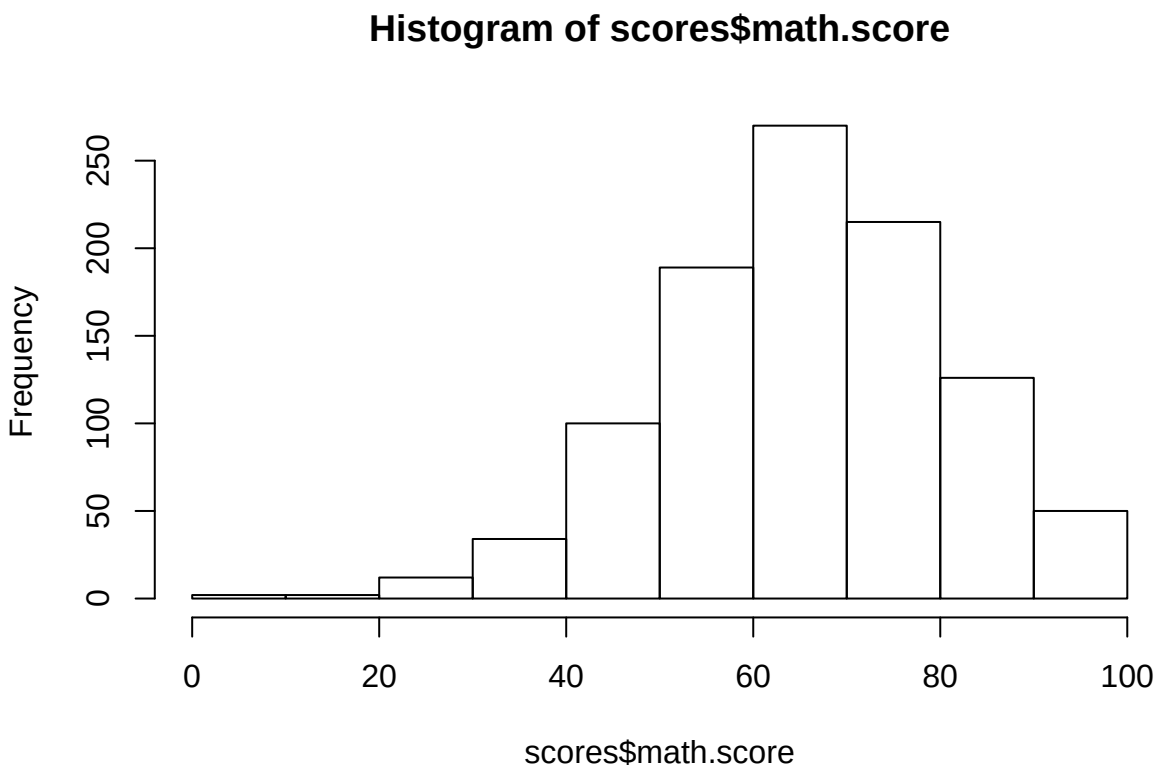
```
## [1] 66.089
```

```
median(scores$math.score)
```

```
## [1] 66
```

그런데 이런 숫자들보다 그래프를 그리는 것이 변수에 대해 더 많은 정보를 줄 때가 많습니다.

```
hist(scores$math.score)
```



hist( )로 수학성적의 히스토그램을 그려보면 60점 대가 가장 많고, 다음으로 70, 50, 80, 40점대 순 인것을 확인할 수 있습니다. 우리가 확인한 평균/중앙값인 66점을 기준으로 양쪽으로 펼쳐진 모양이네요.

히스토그램 말고 상자그림을 그리기도 합니다. 상자그림을 그리기 전에 먼저 5개의 요약값을 계산해 볼텐데요,

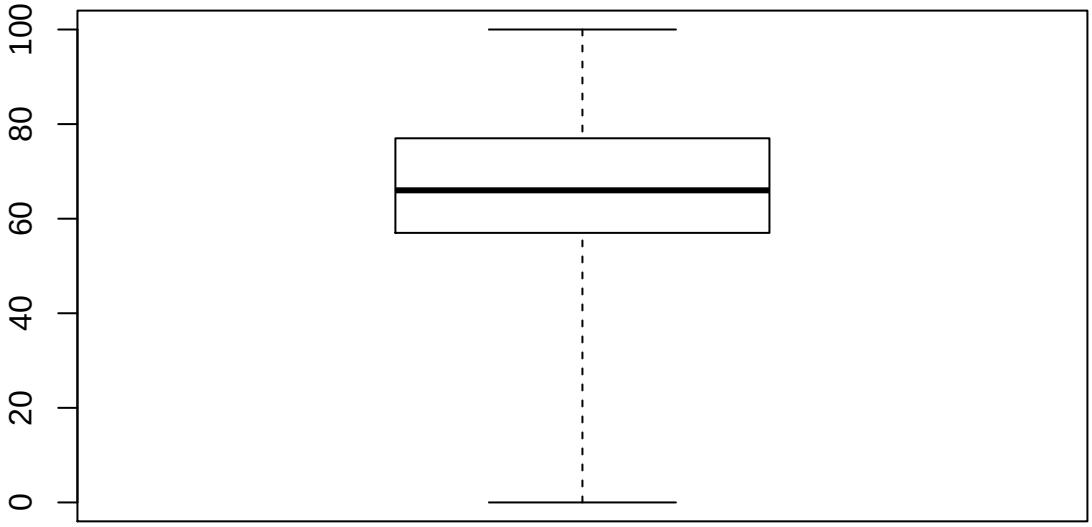
0:4/4는 0부터 4까지 5개 숫자를 4로 나눈 0부터 1까지 0.25간격의 5개 소수를 의미합니다. quantile( )은 분위수를 계산해줍니다.

```
quantile(scores$math.score, 0:4/4)
```

```
##   0%  25%  50%  75% 100%
##    0   57   66   77  100
```

계산된 결과를 보면 이미 우리가 알고 있는 세개의 값이 보입니다. 0%는 최솟값, 50%는 중앙값, 100%는 최댓값을 의미합니다. 그리고 그 사이에는 25%, 75%에 해당하는 57, 77 두 값이 보입니다. 1000개 관측치들을 크기에 따라 줄을 세운 다음 제일 먼저 등장하는 관측치 값을 확인하면 최솟값 0이 되고, 25%, 250번째쯤 나오는 관측치의 값이 첫번째 사분위수(1st Quartile, Q1)가 됩니다. 이런식으로 50%, 75%, 100%까지 총 다섯개의 사분위수를 확인하는 것을 다섯숫자요약이라고 부르는데요, 이걸 그대로 그래프로 표현한 것이 바로 상자그림입니다.

```
boxplot(scores$math.score, range=100)
```



boxplot( )에 수치형 변수를 넣으면 상자그림이 그려집니다. 상자그림은 다섯숫자요약의 5개 값을 점으로 찍고 25%의 첫번째 사분위수와 75%의 세번째 사분위수(3rd Quartile, Q3)를 상자 모양으로 만들고 중앙값은 상자의 중간에 굵은 선으로 표시합니다. 상자그림은 히스토그램에 비해 단순한 편이지만, 비율이 25%로 동일한 4개 구간의 간격을 살펴보면 한 수치형 변수의 분포를 간단히 확인할 수 있어서 유용합니다. 이 상자그림은 과거에 이상치 탐색(outlier detection)에 많이 활용되었는데요, 수업시간에 함수에 들어가 있는 range=이라는 옵션의 의미와 함께 따로 설명 드리겠습니다.

한 범주형 변수의 요약과 시각화

하나의 범주형 변수는 빈도표를 계산합니다.

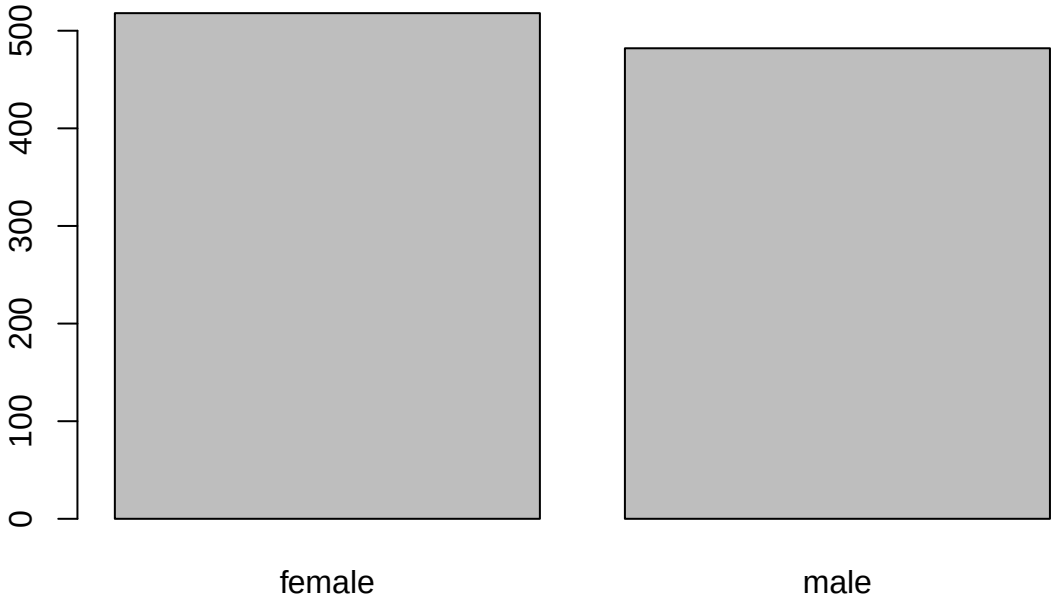
```
table(scores$gender)
```

```
##
## female   male
##    518    482
```

변수 gender를 선택한 다음 table( )로 빈도표를 만듭니다. 1000명 중에서 여자는 518명, 남자는 482명인 것을 확인할 수 있습니다.

이렇게 만든 빈도표는 보통 막대그래프로 표현하는데요, 빈도표를 적당한 이름으로 저장한 다음 barplot( )에 넣습니다.

```
t_gender = table(scores$gender)
barplot(t_gender)
```

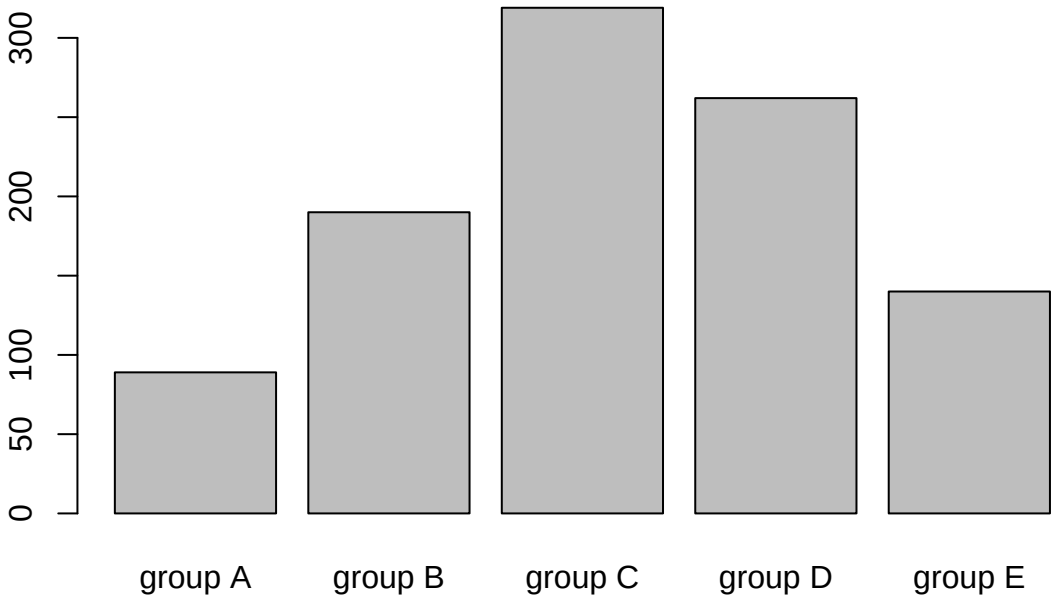


막대그래프는 빈도표의 숫자를 막대의 높이로 표현하는데요, 숫자에 비해 훨씬 비교하기 편합니다. 성별보다 조금 복잡한 범주형 변수로 살펴보면 그 뜻이 더 와 닿습니다.

```
t_race = table(scores$race.ethnicity)
t_race
```

```
##
## group A group B group C group D group E
##      89     190     319     262     140
```

```
barplot(t_race)
```


0
50
100
200
300
group A
group B
group C
group D
group E

수치형과 범주형 변수의 관계 요약과 시각화

지금부터는 두 변수의 관계를 살펴보려고 합니다. 그 중에서 먼저 한 수치형 변수와 한 범주형 변수의 관계를 살펴볼텐데요, 그룹별 평균을 계산해보겠습니다.

한 수치형 변수의 평균은 mean( )을 활용하지만 그룹별 평균은 aggregate( )라는 조금 복잡한 함수를 활용합니다.

```
aggregate(math.score ~ gender, data=scores, mean)
```

```
##   gender math.score
## 1 female   63.63320
## 2   male   68.72822
```

계산된 결과를 보면 여자의 수학 성적 평균은 63.6점인데 반해 남자는 68.7로 약 5점의 평균차이가 있는 것을 볼 수 있습니다. 이렇게 그룹별 평균 차이를 계산하면 흔히 수준(levels)이라고 부르는 그룹에 따라 수치형 변수가 어떻게 다른지를 확인할 수 있습니다.

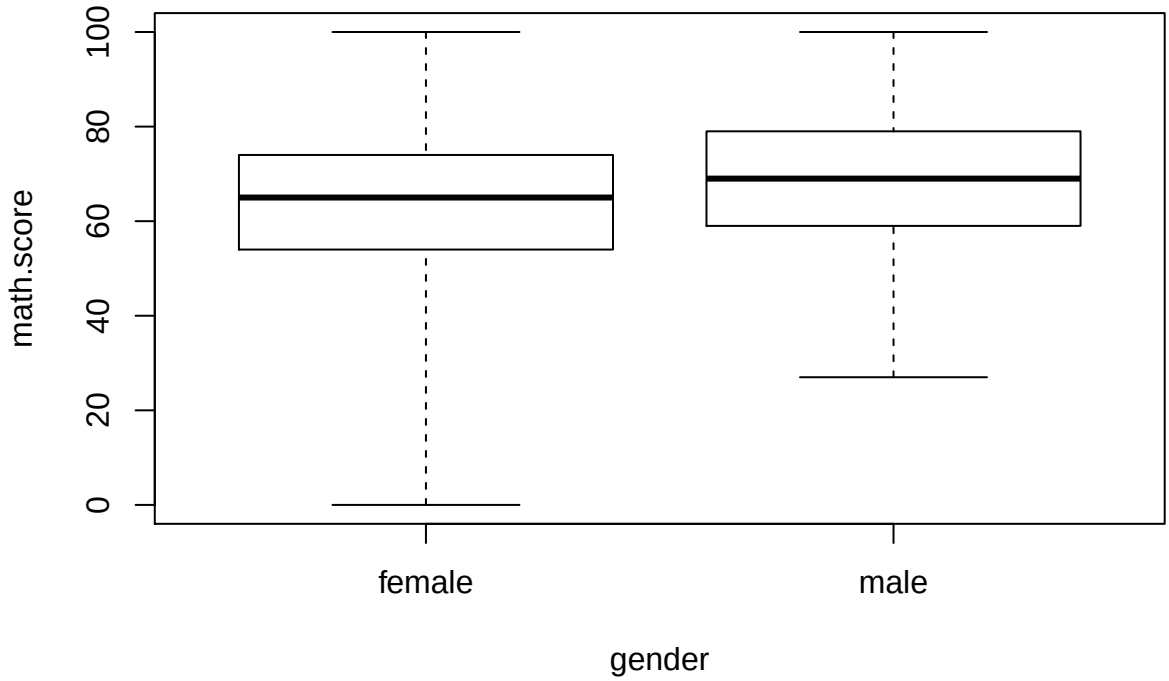
aggregate( ) 안을 자세히 살펴보면 크게 세개의 입력 값이 있는 것을 볼 수 있습니다. 중간에 data=이라는 부분에서 데이터 이름인 scores를 넣어 주고, 첫번째 부분에서 math.score ~ gender라는 표현으로 수치형 변수 math.score를 범주형 변수인 gender에 따라 나눠 요약값을 계산하겠다고 알려줍니다. 그리고 세번째부분에서 평균을 계산한다는 의미로 평균을 계산해주는 함수 mean을 넣었습니다. 조금 복잡해보이지만 R에서 자주 활용될 모형식(formula)라고 부르는 명령어 조합입니다. 비슷한 방법으로 부모의 교육 수준에 따른 읽기 점수의 중앙값은 다음과 같이 계산할 수 있습니다.

```
aggregate(reading.score ~ parental.level.of.education, data=scores, median)
```

```
##   parental.level.of.education reading.score
## 1          associate's degree          72.5
## 2           bachelor's degree          73.0
## 3                 high school          66.0
## 4             master's degree          76.0
## 5                some college          70.5
## 6            some high school          67.0
```

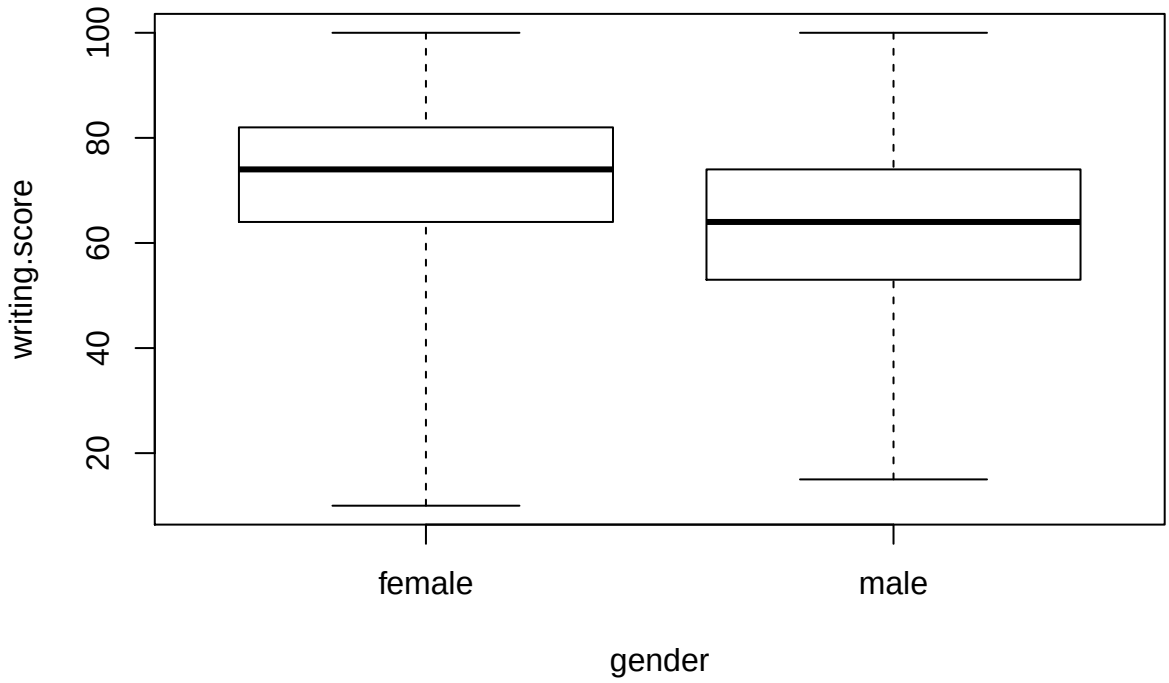
두변수의 관계를 확인하기 위해서 이렇게 그룹별 요약값을 계산할 수도 있지만, 아무래도 그래프가 유용합니다. 그룹별 상자그림을 그려 보겠습니다.

```
boxplot(math.score ~ gender, data=scores, range=100)
```



하나의 수치형 변수의 상자그림을 그릴 때 사용한 boxplot( )을 그대로 쓰지만 함수안에는 앞에서 나온 모형식을 그대로 집어 넣었습니다. 관측치를 gender별로 나눠 math.score의 상자그림을 그리라는 의미입니다. 그룹별 평균은 그룹의 개수만큼의 숫자로 두 변수의 관계를 설명하지만, 그룹별 상자그림에는 그룹별로 다섯개의 요약값이 표현되어 있기 때문에 당연히 더 자세한 비교가 가능합니다. 남자와 여자의 상자그림을 살펴보면 두 그룹 모두 최댓값은 비슷하지만 최솟값에 차이가 있고, 무엇보다도 중간에 있는 상자를 보면 남자가 전반적으로 높은 곳에 있는 것을 볼 수 있습니다. 결론적으로 수학 점수는 남자가 전반적으로 높은 경향이 있다고 볼 수 있죠.

```
boxplot(writing.score ~ gender, data=scores, range=100)
```



읽기 점수에 대해 gender에 따라 상자그림을 그려보면 반대로 여자의 점수가 전반적으로 높을 뿐만 아니라 상자 자체의 세로 길이가 남자에 비해 더 짧은 것을 보면, 여자의 읽기 점수가 좀 더 높은 쪽에 몰려 있다는 것도 확인할 수 있습니다.

두 범주형 변수의 관계 요약과 시각화

두 범주형 변수의 관계를 살펴보기 위해 먼저 교차표(분할표, contingency table)을 계산합니다. 앞서 하나의 범주형 변수의 빈도표를 만들어준 table( )함수에 두 범주형 변수를 콤마(,)로 구분해서 넣어주면 2차원 빈도표, 교차표가 만들어집니다.

```
table(scores$parental.level.of.education, scores$lunch)
```

```
##
##                      free/reduced standard
##   associate's degree           77      145
##   bachelor's degree            44       74
##   high school                  70      126
##   master's degree              24       35
##   some college                 79      147
##   some high school             61      118
```

그럼 두 변수의 수준 조합에 대한 빈도가 계산이 되어 나오는데요, 역시 적당한 이름으로 저장해서 활용합니다. 부모의 학력과 점심비용 지원여부의 교차표를 적당히 t_2로 저장한 다음 prop.table( )을 활용해서 행백분율을 계산할 수 있습니다.

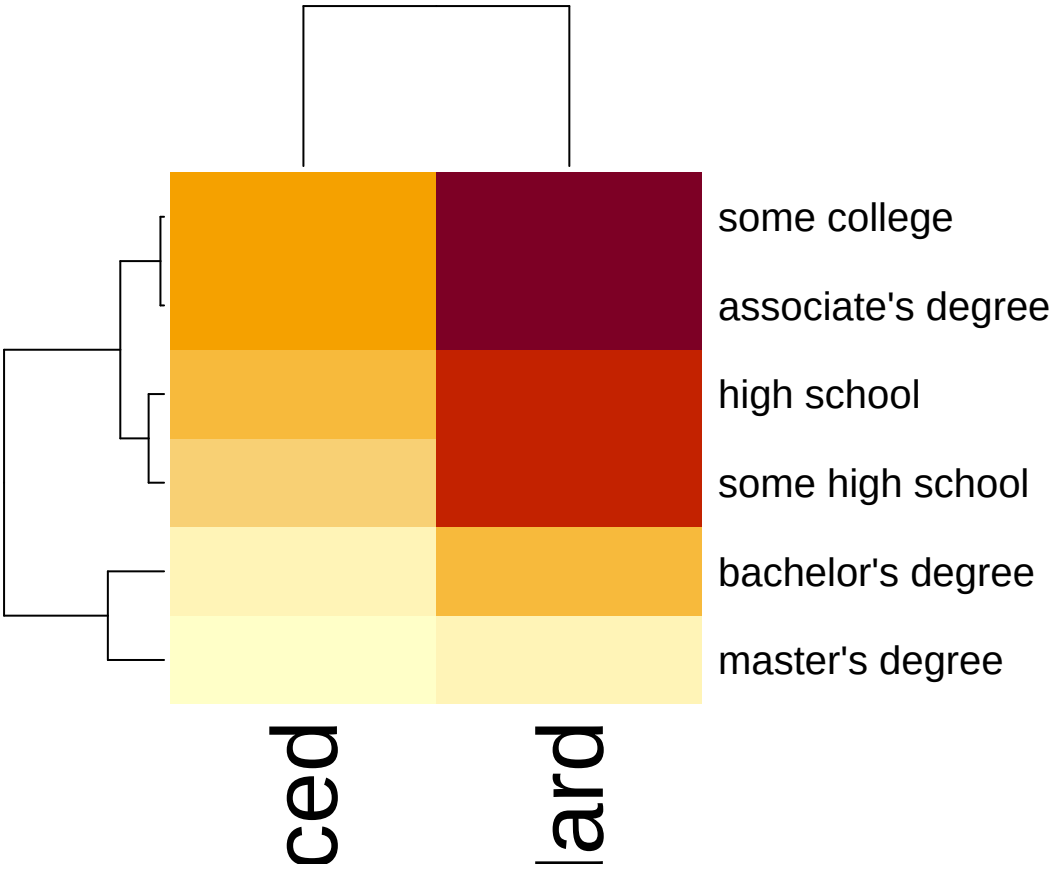
```
t_2 = table(scores$parental.level.of.education, scores$lunch)
prop.table(t_2, 1)
```

```
##
##                      free/reduced  standard
##   associate's degree    0.3468468 0.6531532
##   bachelor's degree     0.3728814 0.6271186
##   high school           0.3571429 0.6428571
##   master's degree       0.4067797 0.5932203
##   some college          0.3495575 0.6504425
##   some high school      0.3407821 0.6592179
```

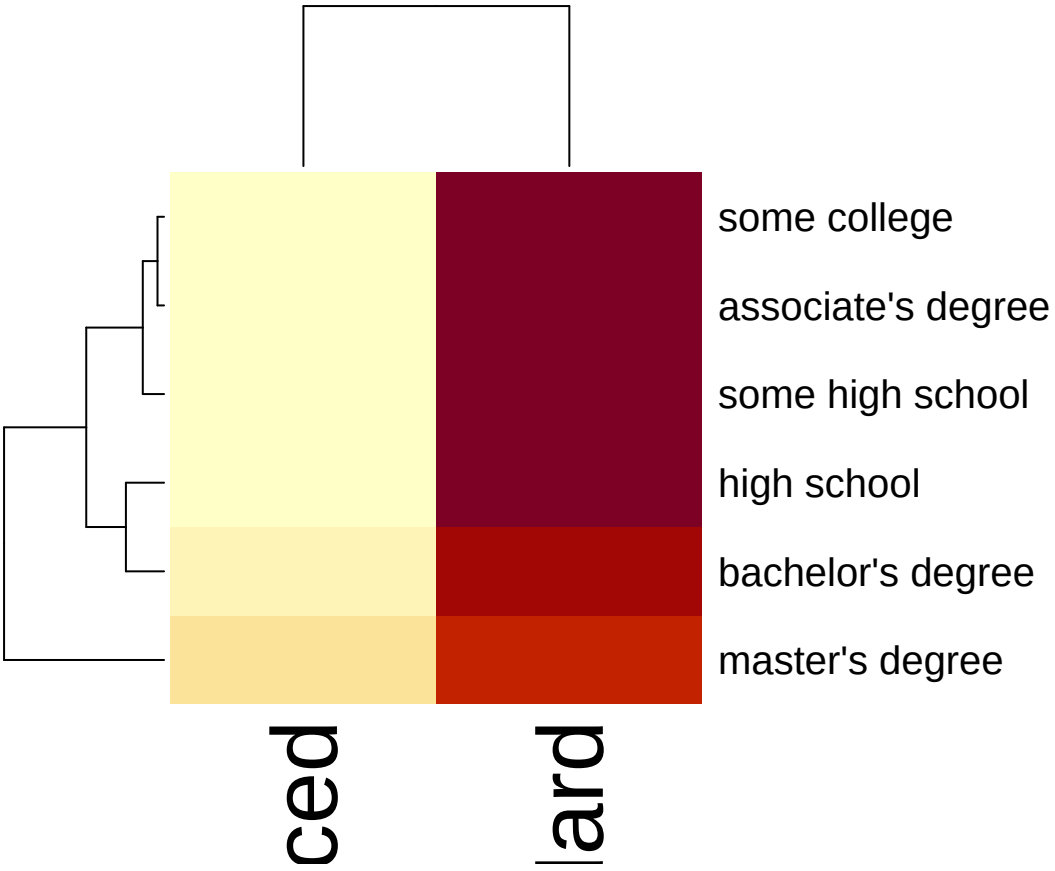
t_2는 단순히 개수를 센 빈도표지만, prop.table( )로 만든 행백분율은 각 부모의 학력안에서 점심비용 지원여부의 비율을 계산했기 때문에 상대적인 비교가 가능합니다. 숫자를 살펴보면 master's degree의 free/reduced 비율이 약 40%로 다른 수준에 비해서 가장 높은 것을 확인할 수 있습니다. 위의 prop.table( )에서 숫자 1 대신 2를 넣으면 열 백분율도 계산할 수 있습니다.

그런데 이렇게 숫자를 직접 살펴보는 것보다 그래프를 그리는 것이 인식하기가 편합니다. 교차표는 열지도(heatmap)로 표현하는데요, R에는 heatmap( )이라는 함수가 있습니다.

```
heatmap(t_2, scale='none')
```




```
heatmap(prop.table(t_2, 1), scale='none')
```

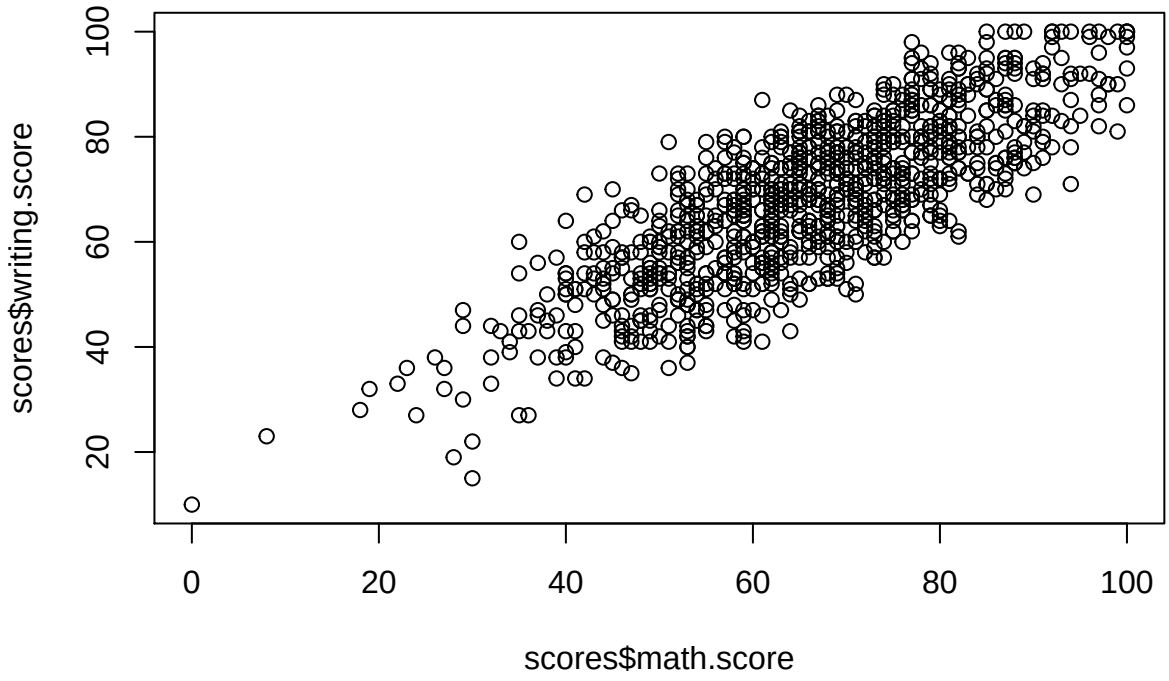


`heatmap( )`에 `t_2`와 같은 원본 교차표를 집어 넣거나 `prop.table( )`로 계산된 교차표의 행/열 백분율을 집어넣으면 숫자대신 색깔의 진하기로 표현한 열지도가 그려집니다. 숫자 여러개를 비교하는 것보다 색의 진하기를 비교하는 것이 훨씬 직관적이기 때문에 많은 사람들이 자주 활용하는 그래프 입니다.

두 수치형 변수의 관계 요약과 시각화

두 수치형 변수는 산점도(scatter plot)를 그려서 관계를 살펴봅니다. `plot( )`은 다양한 기능을 가진 그래프 함수인데요, 기본적으로는 산점도를 그려줍니다.

```
plot(scores$math.score, scores$writing.score)
```

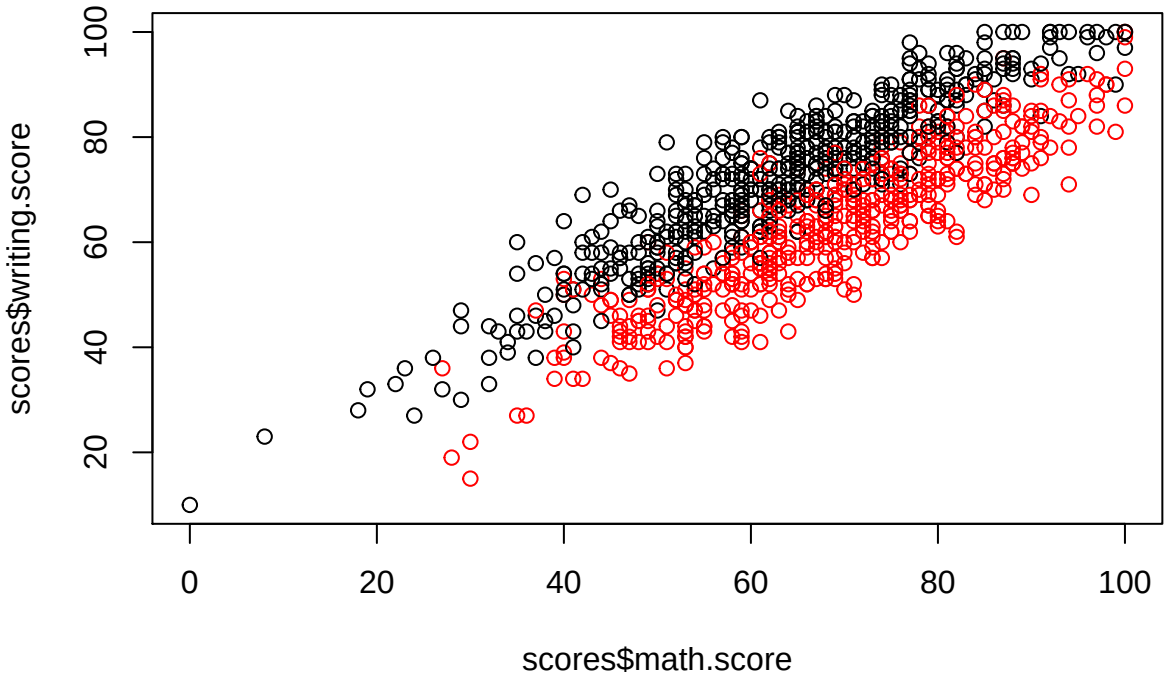


수학 점수와 쓰기 점수의 관계를 살펴보기 위해 산점도를 그렸습니다. 산점도에는 1000개 관측치가 1000개의 점으로 표시되어 있는데요, 먼저 넣은 수학 점수를 x축, 두번째로 들어간 쓰기 점수(writing.score)를 y축 좌표로 활용해서 각 관측치를 점으로 표현합니다.

이렇게 그린 산점도를 살펴보면 두 수치형 변수의 관계를 살펴볼 수 있는데요, 수학 점수가 높으면 쓰기 점수도 높고, 수학 점수 낮으면 쓰기 점수도 낮은 경향이 있는 것이 보입니다.

`plot( )`에는 다양한 옵션이 있는데, 그 중 `col=`을 활용하면 한 범주형 변수에 따라 색을 다르게 한 산점도를 그릴 수 있습니다. 예를들어 위의 명령어에 `col=scores$gender`를 추가하면 성별에 따라 색이 다른 산점도가 그려집니다.

```
plot(scores$math.score, scores$writing.score, col=scores$gender)
```



수준의 순서에 따라 여자(female)는 첫번째 색인 검은 색으로, 남자(male)는 두번째 색인 빨간 색으로 표시된 산점도가 그려집니다. 특이한 것은 관측치들의 위치가 색깔별로 뭉쳐 있다는 것 인데요, 의미를 파악해 보면 학생들의 수학 점수가 비슷하더라도 쓰기 점수는 전반적으로 여자가 더 높고, 반대로 쓰기 점수가 비슷하더라도 수학 점수는 전반적으로 남자가 더 높습니다. 수준이 비슷하더라도 성별에 따라 더 잘하는 과목이 있네요.

두 수치형 변수의 관계는 -1부터 1사이의 값을 가지는 상관계수로 계산할 수 있습니다. `cor( )`을 활용해 두 변수의 상관계수를 계산할 수는 있지만, 숫자 하나로 두 변수의 관계를 표현하기에는 잃어버리는 정보가 너무 많기 때문에 꼭 산점도를 그려보는 습관이 중요합니다.

```
cor(scores$math.score, scores$writing.score)
```

```
## [1] 0.802642
```

지금까지 예제 데이터를 활용해서 간단한 분석 예제들을 살펴봤습니다. 위에서 살펴본 명령어에서 변수 이름을 바꿔 보면서 다양한 숫자를 계산하고 그래프를 그려보면서 의미를 해석해보시면 좋겠습니다.

맺음말

지금까지 R과 RStudio를 설치하고 간단한 R 명령어와 기본 함수를 활용한 데이터 분석 예제를 살펴봤습니다.

먼저 살펴본 기본 명령어와 함수들은 아주 기초적인 내용이긴 하지만 꼭 암기를 할 필요는 없습니다. 중요한 함수들은 공부를 하다보면 또 등장할테니까요.

다음으로 살펴본 분석예제도 마찬가지입니다. R의 기본 함수를 활용해서 데이터를 충분히 요약하고 시각화할 수 있지만, 실제 데이터 분석에서는 조금 더 유용하고 효율적인 다른 함수들을 활용합니다. 첫 세미나 주제에서는 dplyr, ggplot2 같은 패키지를 설치하고 그 속에 포함된 다양한 함수들을 활용해서 데이터를 조금 더 효율적, 직관적으로 요약하고 더 이쁘고 활용도가 높은 그래프를 그리는 방법들을 살펴볼 예정입니다. 굳이 외우려 하지말고 그냥 한번 훑어본다고 생각하면 편합니다.

다시 한번 말씀 드리지만 이 튜토리얼은 R을 이해하기 위한 것이지 암기해야할 것은 아닙니다. 데이터를 R로 불러와 저장하고, 여러 함수들을 활용해서 요약하고 그래프를 그리는 과정을 느낌적인 느낌으로 살펴본 것만으로 충분합니다.

첫 세미나에서는 데이터를 불러오는 것부터 살펴볼 예정이니까 너무 걱정하지 말고 마음만 열고 오시면 됩니다. 그래도 설명이 필요하거나 궁금한 점이 있다면 수업시간에 질문하시거나 slido(https://app.sli.do/event/1svsaqvb)에 올려주세요.

앞으로 세미나 진행을 위해서 간단한 설문을 진행합니다. 아직 모두 확정된 건 없지만, 설문 바탕으로 세미나 일정을 잡고 다시 연락드릴게요!

- **설문조사 참여하기(클릭) : https://forms.gle/nBkfm7DsT7gTxbXF6**

그럼 다음에 뵙겠습니다. 감사합니다:D

임경덕(imkdoug@gmail.com)