

데이터 전처리(1)

숙명여자대학교 경영학부 오중산

데이터 전처리의 정의와 함수 소개

[?]데이터 전처리란?

[?]분석에 적합하도록 원자료(raw data)를
가공하는 작업

[?]실제 데이터 분석 과정에서 가장 많은 시
간이 소요되기도 함

[?]데이터 전처리에 자주 사용되는 dplyr 함수

[?]dplyr 함수들은 파이프연산자로 서로 연
결됨

dplyr 함수	기능
rename()	변수 이름 바꾸기
filter()	행 추출
select()	열(변수) 추출
arrange()	정렬
mutate()	변수 추가
summarise()	기초 통계량 계산
group_by()	집단별로 나눔
left_join()	열 데이터 합치기
bind_rows()	행 데이터 합치기

데이터 전처리: filter 함수

[?] filter 함수 소개

[?] 조건에 부합하는 사례(들)를 추출할 때 사용

[?] 예: gender 변수에서 남학생만 추출

[?] 파이프연산자를 이용해서 filter 함수를 연속 사용 가능: `filter() %>% filter()`

[?] exam 데이터 프레임을 활용한 filter 함수 실습

[?] exam 데이터 프레임에서 1반 학생만 추출해서 exam_c1 데이터프레임 만들기

[?] `exam_c1 <- exam %>% filter(class == 1)`

[?] 남학생들의 영어시험 평균 구하기

[?] 1단계(데이터프레임 만들기): `exam_male <- exam %>% filter(gender == "Male")`

[?] 2단계(평균 구하기): `mean(exam_male$english)`

↳ 내장 함수

[?] 주의! `exam %>% filter(gender == "Male") %>% mean(english)`는 오류

데이터 전처리: filter 함수

56.05

[?] exam 데이터 프레임을 활용한 filter 함수 실습

① `exam_123 <- exam %>% filter(class %in% c(1,2,3))`

[?] 문제1: 1반, 2반, 3반 학생들의 수학시험 평균은 얼마인가? ② `mean(exam_123$math)`

셋째 자리까지 구하고 싶다면 `round(mean(exam_123$math), digits = 3)`

[?] 문제2: 4반이 아닌 학생들 중에서 수학시험이 90점 이상이거나, 역사시험이 95점 이

상인 학생들을 추출하여 새로운 데이터 프레임(exam_n4)을 만드시오.

`exam_n4 <- exam %>% filter(class != 4) %>% filter(math >= 90 | history >= 95)`

[?] 문제3: 영어시험 성적이 상위 10%인 학생들만 추출하시오.

[?] 내장함수 quantile 기본 명령문: `quantile(df$var, probs = c(비율))`

`quantile(exam$english, probs = c(0.9))`

`exam %>% filter(exam >= quantile(exam$english, probs = c(0.9)))`

데이터 전처리: select 함수

[?] select 함수 소개

[?] 원하는 변수(들)만 추출할 때 사용(예: 반, 수학점수, 영어점수만 추출하기)

[?] exam 데이터 프레임을 활용한 select 함수 실습

[?] 반, 수학점수, 영어점수만 추출하기

[?] `exam %>% select(class, math, english)` : 여러 변수를 심표로 연결함

[?] 주소를 제외한 다른 변수 추출하기

[?] `exam %>% select(-address)` : -표시를 하면 해당 변수를 제외한다는 의미 => to see more rows : `%>% print(n=Inf)`

[?] 특정 단어가 포함된 변수 추출하기

[?] 기본명령문: `df %>% select(contains("특정 단어"))`

[?] 문제4: 1반 학생들만을 대상으로 성별과 수학점수를 추출 `exam %>% filter(class==1) %>% select(gender, math)`

데이터 전처리: arrange 함수

[?] arrange 함수 소개

[?] 정량적 변수에 대해 오름차순 혹은 내림차순으로 정렬할 때 사용

[?] 정렬기준이 여러 개인 경우, 우선 순위에 따라 심표로 구분하여 입력

[?] arrange 함수를 이용한 실습

[?] 수학점수를 오름차순과 내림차순으로 정렬하기

[?] `exam %>% arrange(math)`

[?] `exam %>% arrange(-math)` \Rightarrow NA는 가장 마지막에 출력

[?] 반별로 수학 최고점수 확인하기
여러 개 변수 > 두 개의 변수 심표로 연결 가능

[?] `exam %>% arrange(class, -math) %>% print(n=Inf)`

[?] 콘솔창에서 출력이 끊기면 `print(n = Inf)`를 입력

데이터 전처리: mutate 함수

?mutate 함수 소개

?기존 변수를 활용하여 새로운 변수를 만들 때 사용하는 함수

?종종 ifelse 함수와 함께 사용됨

ex) 정량적 변수 int/dbl/num
=> factor형으로



?mutate 함수를 이용한 실습

?exam에서 세 과목(수학/영어/역사) 점수 합계인 total 변수와 세 과목(수학/영어/역사) 점수 평균인 average 변수 만들기

?동시에 여러 개 변수를 만들 수 있음

```
?exam <- exam %>% mutate(total = math + english + history, average = (math + english + history)/3)
```

데이터 전처리: mutate 함수

❓ mutate 함수와 ifelse 함수 결합한 실습

❓ exam에서 test 변수 만들기

❓ total 점수가 180점 이상이면 “pass”, 그렇지 않으면 “fail”로 판정하는 파생변수 test 만들기

```
❓ exam <- exam %>% mutate(test = ifelse(total >= 180, "pass", "fail"))
```

❓ 합격자/불합격자 빈도수 확인하기

① `library(descr)` ② `freq(exam $ test == "pass")`

데이터 전처리: mutate 함수

[?] mutate 함수와 ifelse 함수 결합한 실습

[?] exam에서 grade 변수 만들기

[?] average 점수가 60점 미만이면 “fail”, 60점 이상 75점 미만이면 “middle”, 75점 이상 90점 미만이면 “good”, 90점 이상이면 “excellent”로 표기하는 파생변수 grade 만들기

[?] 범주가 n개로 구분되면 ifelse 함수를 (n-1)회 사용해야 함

```
[?] exam <- exam %>% mutate(grade = ifelse(average < 60, "fail", ifelse(average < 75, "middle", ifelse(average < 90, "good", "excellent")))))
```

데이터 전처리: mutate 함수

[?] mutate 함수와 case_when 함수 결합한 실습

[?] exam에서 test 변수와 grade 변수 만들기

[?] case_when 함수는 dplyr에 있는 함수로 ifelse와 비슷하지만, 더 간단하게 코드를 만들 수 있음

```
[?] exam <- exam %>% mutate(test = case_when(total < 180 ~ "fail", TRUE ~ "pass"))
```

[?] TRUE~는 해당 변수가 “그렇지 않으면” 이라는 의미

total NA이어도 pass, fail 나옴 ↳ 그렇지 않으면
⇒ total ≥ 180 ~ "pass"

```
[?] exam <- exam %>% mutate(grade = case_when(average < 60 ~ "fail", average < 75 ~ "middle", average < 90 ~ "good", TRUE ~ "excellent"))
```

☆ 주의! 변수 측정값에 NA가 있다면(예: total/average) TRUE~ 조건은 쓰지 않아야 함

average ≥ 90 ~ "excellent" ☆

데이터 전처리: **relocate 함수**

[?]relocate 함수 실습

[?]변수 위치를 이동시킬 때 사용하는 함수

[?]기본명령문: `df <- df %>% relocate(이동할 변수, .after(혹은 before) = 위치변수)`

[?]total을 test 앞으로 이동시키고, average는 test 뒤로 이동

[?] `exam <- exam %>% relocate(total, .before = test)`

[?] `exam <- exam %>% relocate(average, .after = test)`

[?]문자형 척도 변수를 맨 앞으로 이동하기

* \rightarrow 문자인 척도들을 맨 앞으로 이동시켜줘

[?] `exam <- exam %>% relocate(where(is.character))`

문자형 척도

[?]범주형 척도 변수를 문자형 척도 변수 앞으로 이동하기

*

[?] `exam <- exam %>% relocate(where(is.factor), .before = where(is.character))`

범주형 척도

데이터 전처리: group_by 함수와 summarise 함수

[?] **group_by 함수** 소개 : ~를 기준으로 그룹핑 한다.

[?] 사례를 어떤 변수값의 결과를 기준으로 몇 개 집단으로 구분

[?] 이때 변수의 척도는 문자형 혹은 범주형인 게 바람직함

[?] 예: 반, 성별, 주소에 따른 사례 구분

[?] **summarise 함수** 소개

[?] 어떤 변수의 기술통계량에 대한 요약결과를 보여줄 때 사용

[?] 기술통계량과 함께 빈도수를 보여줄 수 있음

[?] 일반적으로 group_by 함수와 함께 사용됨

[?] 사례를 몇 개 집단으로 구분한 후, 구분된 집단별로 관심 있는 변수의 기술통계량 제시

데이터 전처리: group_by 함수와 summarise 함수

[?] group_by 함수와 summarise 함수 실습

[?] 반별로 학생수(빈도)와 수학점수의 평균& 표준편차 제시

```
[?] exam %>% group_by(class) %>% summarise(n(), mean(math, na.rm = T), sd(math , na.rm = T))
```

```
[?] exam %>% group_by(class) %>% summarise(count = n(), mean_math = mean(math , na.rm = T), sd_math  
= sd(math , na.rm = T))
```

[?] summarise를 통해 보여주는 결과값에 대해 변수명 지정

[?] 반과 성별로 구분하여 학생수와 역사점수 평균을 요약하여 새로운 데이터 프레임 exam_new에 저장
하시오.

```
[?] exam_new <- exam %>% group_by(class, gender) %>% summarise(count = n(), mean_history =  
mean(history))
```

*참고 Sum 함수

`exam_new %>% mutate (perc = count / sum(count))`

	class	gender	count	mean_history	perc
	<fct>	<fct>	<int>	<dbl>	<dbl>
1	1	Female	3	94	0.5
2	1	Male	3	91.7	0.5
3	2	Female	4	80.2	0.5
4	2	Male	4	84.5	0.5
5	3	Female	3	98	0.5
6	3	Male	3	78.3	0.5
7	4	Female	2	77	0.4
8	4	Male	3	90.7	0.6
9	5	Female	3	67.3	0.6
10	5	Male	1	68	0.2
11	5	na	1	83	0.2

$\Rightarrow \frac{3}{30} = 0.1$ 아닌 이유

sum 전체가 아니라 1번 6명 중 3명 $\frac{3}{6} = 0.5$
↳ 반별은 sum 함

전체 sum 하려면?

`exam_new %>% mutate (perc = count / sum(exam_new$count))`

데이터 전처리(2)

숙명여자대학교 경영학부 오중산

movie 데이터프레임 전처리 실습

[?] 변수명을 소문자로 바꾸기

[?] 변수명을 다루기 편하게 하기 위해 대문자를 소문자로 변경

↔ tolower

```
[?] names(movie) <- tolower(names(movie))
```

↳ names(df) = df 안에 있는 변수

[?] 문제1: 2018년부터 2020년까지 출시된 영화의 runtime 평균 구하기

[?] 두 단계(데이터프레임 만들기과 평균 구하기)로 나누어 진행 ① library(dplyr)

② movie1 <- movie %>% filter(year %in% c(2018:2020))

[?] 유효숫자는 소수 둘째자리

③ round(mean(movie1\$runtime), digits = 2)

movie 데이터프레임 전처리 실습



[?]문제2: A등급이면서 동시에 genre에서 Drama가 포함된 영화 중에서 imdb_rating이

가장 높은 영화는 무엇인가?

```
movie 2 <- movie %>% filter(certificate == "A") %>% filter(str_detect(genre, "Drama"))  
%>% arrange(-imdb_rating)
```

[?]힌트1: Drama가 포함된 genre이므로 어떤 영화의 genre가 Drama, History and Comedy여도 해당됨

[?]힌트2: stringr 패키지에 있는 str_detect 함수를 사용해야 함

[?]기본명령문: str_detect(V1, "AAA")

[?]V1 변수에서 AAA가 포함된 사례를 파악함

[?]문제3: genre에서 Drama가 포함되고, overview에서 crime이 포함된 영화는 몇 편인가?

```
① movie 3 <- movie %>% filter(str_detect(genre, "Drama") & str_detect(overview, "crime"))
```

movie 데이터프레임 전처리 실습

[?]문제4: 문제3에서 확인된 영화 중에서 meta_score가 상위 10%에 해당되는 영화는 무엇인가?

```
quantile(movie3$meta_score, probs = c(0.9), na.rm = T)
```

```
① movie %>% filter(meta_score >= quantile(movie3$meta_score, probs = c(0.9), na.rm = T)) %>% select(title)
```

[?]문제5: star 혹은 dir 단어를 포함한 변수만 추출해서 새로운 데이터프레임을 만드시오.

[?]힌트: 논리연산자와 내장함수 contains 함께 사용

```
movie5 <- movie %>% select(contains("star") | contains("dir"))
```

[?]문제6: 문제5에서 만든 데이터프레임의 star1 변수에서 빈도수가 가장 높은 배우 세 명은

누구인가?

```
movie6 <- freq(movie5 $ star1)
```

movie 데이터프레임 전처리 실습

`movie <- movie %>% mutate(score = 10 * imdb_rating + meta_score)`

[?]문제7: 새로운 변수 score(= 10×imdb_rating + meta_score)를 만드시오.

[?]문제8: 다음과 같은 기준으로 새로운 변수 class를 만드시오.

[?]주의! case_when에서 TRUE~ 조건은 NA가 있을 경우 사용하지 말아야 함

기준	class 변수값
score ≤ 120 100	D
100 120 < score ≤ 130	C
130 < score ≤ 160	B
160 < score ≤ 180	A
180 < score ≤ 200	S

`movie <- movie %>% mutate(class = case_when
 (score ≤ 100 ~ "D",
 score ≤ 130 ~ "C",
 score ≤ 160 ~ "B",
 score ≤ 180 ~ "A",
 score > 180 ~ "S"))`

movie 데이터프레임 전처리 실습

[?]문제9: class 변수 척도를 출력순서가 S, A, B, C, D인 범주형 척도로 변경하시오.

```
movie $ class <- factor (movie $ class , levels = c ("S", "A", "B", "C", "D"))
```

[?]주의! NA는 levels에서 따로 지정하지 않음

[?]문제10: class별로 빈도수와 gross 평균을 구하시오. `movie %>% group_by(class) %>% summarise(n(), mean(gross))`

[?]문제11: 감독 중에서 빈도수가 가장 많은 10명은 누구인가?

```
movie %>% group_by(director) %>% summarise(number = n()) %>% arrange (-number) %>% head (10)
```

[?]힌트: group_by, summarise, arrange, head 함수를 순차적으로 사용

[?]참고: n_distinct 함수를 이용한 변수 측정값 개수 확인하기

[?]어떤 변수에 대한 중복된 측정값을 제외한 고유의 값의 개수 확인

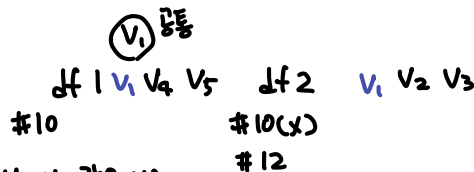
`n_distinct (movie $ director)` => 변수의 척도와 상관없이 다 쓸 수 있음

: 서로 다른 감독 몇명이나

데이터 전처리(3)

숙명여자대학교 경영학부 오중산

데이터 전처리: 세 가지 join 함수



[?] 두 개의 데이터프레임을 합칠 때 사용하는 세 가지 join 함수

↳ V2, V3 값은 NA

↳ 두 개의 df를 통합할 때 기준이 되는 변수

[?] left_join 함수 기본 명령문: `df1 <- left_join(df1, df2, by = "공통변수")`

dplyr 패키지

[?] 공통변수 기준으로 df1에 df2를 통합하되, df1\$공통변수 측정값이 df2\$공통변수 측정값에 없는 case의 경우 통합 후 새로운 변수 측정값은 NA로 처리됨 V2, V3 NA 처리 => df 1에 있는 case만 살아남음

[?] inner_join 함수 기본 명령문: `df1 <- inner_join(df1, df2, by = "공통변수")`

[?] 공통변수 기준으로 df1과 df2를 통합하되, df1\$공통변수 측정값과 df2\$공통변수 측정값이 일치하지 않는 case는 삭제됨 #10 삭제 => df1, df2 모두 있는 케이스만 살아남음

[?] full_join 함수 기본 명령문: `df1 <- full_join(df1, df2, by = "공통변수")`

[?] 공통변수 기준으로 df1과 df2를 통합하되, df1과 df2의 모든 case를 포괄하고 공통변수 측정값이 df1과 df2에 모두 존재하지 않으면 새로운 변수 측정값은 NA로 처리됨 => 둘 중 하나라도 있으면 살아남음
=> 상대 df에 존재하지 않는 케이스는 상대 df 변수에 대해 NA로 처리

데이터 전처리: 세 가지 join 함수

[?] 세 가지 join 함수 예시

ex) ID, day => 변수명은 다르지만 내용상 동일

기준이 되는 변수가 거의 비슷한 상황에서 기존의 케이스에 대해

새로운 변수 측정 값 갖다 붙임

df 1		df 2	
ID	X1	ID	X2
1	a1	2	b1
2	a2	3	b2

* left_join *		
ID	X1	X2
1	a1	NA
2	a2	b1

df 1에 있는 것만 살아남음

inner_join		
ID	X1	X2
2	a2	b1

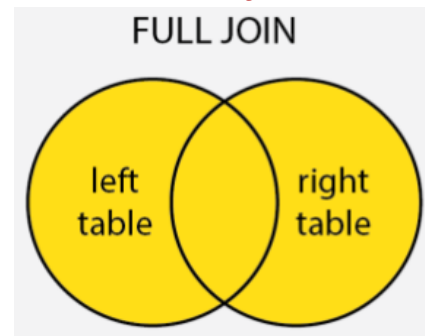
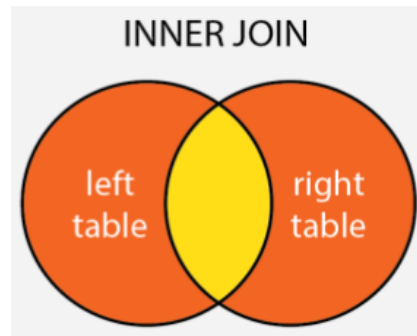
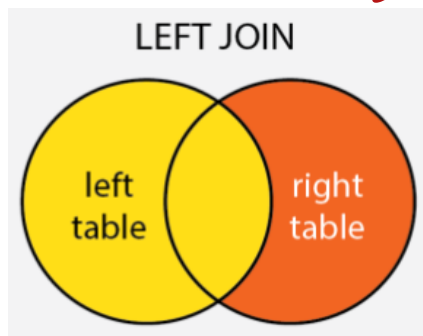
공통된 것만 살아남음

full_join		
ID	X1	X2
1	a1	NA
2	a2	b1
3	NA	b2

합집합

ex) day

데이터 달라도 날짜 동일 => 갖다 붙일 수 있음



데이터 전처리: left_join 함수

① 변수 배열

② 척도

[?] exam 데이터프레임 정비

[?] 10개의 변수 배열 순서 정리

```
exam <- exam %>% relocate(test, after = total)
```

[?] relocate 함수를 이용해서 address, gender, class, math, history, english, total, test, average, grade 순서로 배열

char f f dbl. num. int c or f
f f

```
exam <- exam %>% mutate(id = c(1:30)) %>% relocate(id)
```

[?] exam 데이터프레임에 새로운 변수 id 추가

[?] ~30까지 값을 부여한 id 변수를 새로 만들어 exam 데이터프레임 제일 앞에 배열

↳ 정의가 이렇게 된다

```
exam <- exam %>% mutate(id = c(1:30)) %>% relocate(id)
```

연속된 정수

데이터 전처리: left_join 함수

[?] left_join 함수 실습

[?] exam_science.csv 파일 불러와서 같은 이름의 데이터프레임 생성

[?] exam과 exam_science 합치기

```
[?] exam <- left_join(exam, exam_science, by = "id")
```

[?] 공통변수의 변수명이 일치하지 않을 경우에도 통합 가능

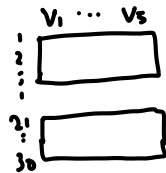
```
[?] exam <- left_join(exam, exam_science, by = c("ID" = "id"))
```

(Note: In the original image, 'ID' is crossed out with a red 'X' and labeled 'df1', and 'id' is crossed out with a red 'X' and labeled 'df2'. A red wavy line is drawn under the equals sign.)

[?] ID는 exam, id는 exam_science에 속한 내용상 동일한 변수이며, 입력 순서 주의

데이터 전처리: bind_rows 함수

↳ 행을 붙인다



[?] bind_rows 함수 소개

[?] 두 개 데이터프레임을 통합하되, 새로운 사례를 추가할 때 사용하는 함수


[?] 두 개 데이터프레임의 변수가 모두 일치할 필요는 없음

[?] 통합시 한쪽 데이터프레임에만 존재하는 변수의 경우, 해당 변수가 존재하지 않은 데이터프레임 사례에 대해서는 NA로

처리됨



[?] 두 개 데이터프레임에 모두 존재하는 내용상 동일한 변수의 경우 변수명이 일치해야 하고, 변수의 척도

도 동일해야 함  ex) ID, id => 통일해야 함 * 대문자 주의

glimpse, str

↳ df2 척도 df1과 맞추기

[?] 내용상 동일한 변수지만 변수명이 다르면 다른 변수로 인식

[?] 변수명이 일치하더라도 변수 척도가 다르면 오류 발생

데이터 전처리: bind_rows 함수

[?] bind_rows 함수 실습

```
exam_add <- read_csv("exam_add.csv", locale = locale("ko", encoding = "euc-kr"))
```

[?] exam_add.csv를 불러와서 동일한 명칭의 데이터 프레임을 만든 후, exam과 동일한

변수의 경우 척도를 일치시킴

① glimpse () ⇒ 척도 확인

② exam_add \$ gender <- as.factor(exam_add \$ gender)

[?] exam과 exam_add 간의 통합

```
[?] exam <- bind_rows(exam, exam_add)
```

* exam에서 NA 변경하기

[?] exam에서 science를 english 다음으로 이동

① average

```
exam $ average <- ifelse(is.na(exam $ average), exam $ total / 3,  
                        exam $ average)
```

② grade

```
exam <- exam %>% mutate(grade = case_when(average < 60 ~ "fail",  
                                           average < 75 ~ "middle", average < 90 ~ "good",  
                                           average >= 90 ~ "excellent"))
```

```
exam <- exam %>% relocate(science, . after = english)
```

데이터 전처리: bind_rows 함수

[?] 중복된 id가 존재하는지 확인

`exam %>% group_by(id) %>% summarise(count = n()) %>% arrange(-count)`

[?] group_by, summarise, arrange 함수 이용하여 중복 id 확인

[?] 중복 id가 존재할 경우 동일한 학생인지, id 측정값에 오류가 있는지 검토

[?] 만약 전자의 경우라면 하나의 사례만 남기고 나머지 사례는 삭제

[?] 중복된 id에 대해 하나만 남기고 나머지 사례를 삭제하는 방법

↳ 나머지 사례는 그대로

* `n_distinct()`: 고유한 값 확인

[?] `exam <- exam %>% distinct(id, .keep_all = T)`

↳ id를 기준으로 동일한 케이스 있다면 제거해줘

[?] d 변수의 측정값이 동일할 경우, 하나만 남기고 나머지 사례는 제거하라는 의미

* `n_distinct(exam$id)`

=> 35 : 뭔가 중복 있구나

[?] 주의! `.keep_all = T` 조건이 있어야 나머지 변수가 지워지지 않음

데이터 전처리: 이상치(outlier) 처리

❓ 이상치의 정의와 처리 방법

❓ 어떤 변수 측정값이 예상 범주를 벗어나면 이상치로 판정함

❓ 범주를 벗어난 측정값은 입력 오류에 기인한 경우가 많음

❓ 이런 경우 측정값을 이상치로 판정하고, 이상치를 NA로 대체함

❓ 통합된 exam에서 네 개 과목에 대해 100점 초과 사례 파악

❓ 통합된 exam에서 id = 33의 science 점수(122점)는 이론적 최대값인 100점보다 크게 측정되었으므로 이상치로 판정하고, 이를 NA로 대체해야 함

↳ 과목별로 뭐가 이상치인지 확인

1. 이상치 유무 확인 ① `table(exam[5:8] > 100)` ② `table(exam$math > 100) ...`

2. 있으면 NA 대체 `exam$science <- ifelse(exam$science > 100, NA, exam$science)`

데이터 전처리: 이상치(outlier) 처리

[?] 이상치의 정의와 처리 방법

[?] 어떤 변수의 측정값이 예상 범주 안에 있더라도 지나치게 크거나, 지나치게 작은 경우 이상치로 판정할 수 있음

↳ 사전 기준

상한, 하한 0~100 사이 아닐 수도 있음

[?] 그럴 경우에는 기준(상한과 하한값)을 설정하여 이상치로 판정해야 함

↳ 4를 확률 1%

[?] (기준 예시) 변수 측정치가 정규분포를 따른다는 가정하에, 상·하위 0.5%에 속하는 측정값을 이상치로 판정: '표본평균 ± 2.57583 × 표준편차'를 벗어나면 이상치로 판정하여 NA로 처리

$\bar{x} \pm 2.57583 \times s$

$\bar{x} \pm 2.57583 \times \text{표준편차}$ 를 벗어나면 이상치로 판정하여 NA로 처리 * `descr <- descr %>% mutate (low = mean - 2.57583 * sd , upper = mean + 2.57583 * sd)`

$\alpha = 0.05$

[?] 만약에 사전에 설정된 기준을 넘어간 경우라도, 측정값에 대한 논리적 설명이 가능하다면 이상치로

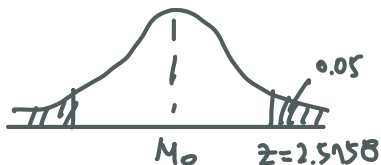
판정하는 것에 신중해야 함

① `descr <- descr (exam [5:8])`

`library(psych)`

② 상한 : `descr$mean * 2.57583 * 표준편차` ⇒ 상한, 하한 구하기

③ 상한은 최대, 하한은 최소와 비교하면 됨 (각 과목별)



* mutate로 새 변수 만들기

```
descr<- descr%>% mutate (low=mean - 2.57583 * sd ,
                        upper = mean + 2.57583 * sd )
```

```
> # 상한 #
> descr$mean + 2.57583 *descr$sd
[1] 116.8855 117.1112 122.2779 108.4332
> # 하한 #
> descr$mean - 2.57583 *descr$sd
[1] 5.526270 52.088792 9.264955 52.272690
```

=> 우리가 정한 기준보다 크거나 작을 수도 있음

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se	low	upper
math	1	34	61.20588	21.61618	59	61.78571	20.7564	20	98	78	-0.07347185	-0.91239636	3.707145	5.526270	116.8855
history	2	35	84.60000	12.62164	87	85.86207	14.8260	56	98	42	-0.70838201	-0.53335668	2.133447	52.088792	117.1112
english	3	35	65.77143	21.93719	65	67.55172	19.2738	12	98	86	-0.68896287	-0.05021166	3.708062	9.264955	122.2779
science	4	34	80.35294	10.90144	80	80.75000	14.0847	59	98	39	-0.25272081	-1.02414908	1.869581	52.272690	108.4332

min. max가 안에 있으면 ok