문공 A0015

R 프로그래밍

김 태 완

kimtwan21@dongduk.ac.kr

데이터 전처리

- 결측값 (missing value)
- 특이값 (outlier)
- 데이터 정렬 (order / sort)
- 데이터 분리와 선택 (split / select)
- 데이터 샘플링과 조합 (sampling / combination)
- 데이터 집계와 병합 (aggregate/ merge)

데이터 전처리

- **결측값 (missing value)**
- 특이값 (outlier)
- 데이터 정렬 (order / sort)
- 데이터 분리와 선택 (split / select)
- 데이터 샘플링과 조합 (sampling / combination)
- 데이터 집계와 병합 (aggregate/ merge)

데이터 전처리

데이터 전처리(data preprocessing)

데이터 분석을 제대로 하기 위해서 초기에 확보한 데이터를 정제하고 가공하여
분석에 적합한 데이터를 확보하기 위한 과정

데이터 전처리를 하지 않으면 결측값 또는 특이값 등으로 인하여
통계 결과가 왜곡될 수 있음

- 결측값
  - 결측값의 개념

결측값(missing value)

데이터를 수집하고 저장하는 과정에서 저장할 값을 얻지 못하는 경우 발생

ex) 통계 조사 응답자가 어떤 문항에 대해 응답을 하지 않았다면
그 문항의 데이터 값은 결측 값이 됨

결측값의 처리

- 결측값을 제거하거나 제외한 다음 데이터를 분석
- 결측값을 추정하여 적당한 값으로 치환한 후 데이터를 분석

- 결측값
  - 벡터의 결측값 처리 : NA는 숫자형, 문자형, 논리형 데이터 어디에서나 결측값을 나타낼 수 있음
    - 결측값의 특성과 존재 여부 확인 : is.na( ) 함수 이용

```
z ← c(1,2,3,NA,5,NA,8)          # 결측값이 포함된 벡터 z
sum(z)                          # 정상 계산이 안 됨
is.na(z)                        # NA 여부 확인
sum(is.na(z))                   # NA의 개수 확인
sum(z, na.rm=TRUE)              # NA를 제외하고 합계를 계산
```

데이터 전처리

- 결측값
  - 벡터의 결측값 처리 : NA는 숫자형, 문자형, 논리형 데이터 어디에서나 결측값을 나타낼 수 있음
    - 결측값 대체 및 제거

```
z1 ← c(1,2,3,NA,5,NA,8)          # 결측값이 포함된 벡터 z1
z2 ← c(5,8,1,NA,3,NA,7)          # 결측값이 포함된 벡터 z2
z1[is.na(z1)] ← 0                # NA를 0으로 치환
z1 # [1] 1 2 3 0 5 0 8
z3 ← as.vector(na.omit(z2))      # NA를 제거하고 새로운 벡터 생성
z3 # [1] 5 8 1 3 7
```

데이터 전처리

- 결측값
  - 매트릭스와 데이터프레임의 결측값 처리
    - 결측값이 포함된 데이터프레임 생성

```
x ← iris
x[1,2]← NA; x[1,3]← NA
x[2,3]← NA; x[3,4]← NA
head(x)
```

```
> head(x)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1          NA           NA         0.2  setosa
2          4.9         3.0           NA         0.2  setosa
3          4.7         3.2          1.3          NA  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

- 결측값
  - 데이터프레임의 열별 결측값 확인 1 : for문을 이용한 방법

```
for (i in 1:ncol(x)) {
 this.na ← is.na(x[,i])
 cat(colnames(x)[i], "₩t", sum(this.na), "₩n")
}
```

```
Sepal.Length  0
Sepal.Width   1
Petal.Length  2
Petal.Width   1
Species       0
```

- 결측값
  - 데이터프레임의 열별 결측값 확인 2 : apply( ) 함수를 이용한 방법

```
col_na ← function(y) {
 return(sum(is.na(y)))
}

na_count ←apply(x, 2, FUN=col_na)
na_count
```

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
           0           1            2            1            0
```

- 결측값
  - 데이터프레임의 행별 결측값 확인

```
rowSums(is.na(x))                    # 행별 NA의 개수
sum(rowSums(is.na(x))>0)             # NA가 포함된 행의 개수


sum(is.na(x))                  # 데이터셋 전체에서 NA 개수
```

- 결측값
  - 데이터프레임의 행별 결측값 확인 : **complete.cases( ) 함수** 이용
    - 어떤 데이터셋에서 NA를 포함하지 않은 완전한 (complete 행들을 찾아줌)

```
head(x)
x[!complete.cases(x),]                # NA가 포함된 행들 출력
y ← x[complete.cases(x),]             # NA가 포함된 행들 제거
head(y)
```

- 결측값
  - 데이터프레임의 행별 결측값 확인 : **complete.cases( ) 함수** 이용
    - 어떤 데이터셋에서 NA를 포함하지 않은 완전한 (complete 행들을 찾아줌)

```
x[!complete.cases(x),]                    # NA가 포함된 행들 출력
```

```
> x[!complete.cases(x),]                  # NA가 포함된 행들 출력
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1          NA           NA         0.2  setosa
2          4.9         3.0           NA         0.2  setosa
3          4.7         3.2          1.3          NA  setosa
```

- 결측값
  - 데이터프레임의 행별 결측값 확인 : **complete.cases( ) 함수** 이용
    - 어떤 데이터셋에서 NA를 포함하지 않은 완전한 (complete 행들을 찾아줌)

```
y ← x[complete.cases(x),]          # NA가 포함된 행들 제거
head(y)
```

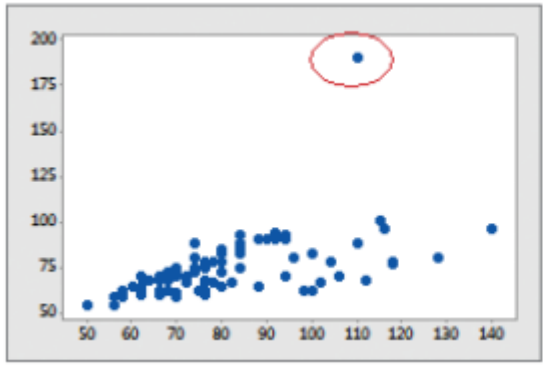
```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
7          4.6         3.4          1.4         0.3  setosa
8          5.0         3.4          1.5         0.2  setosa
9          4.4         2.9          1.4         0.2  setosa
```

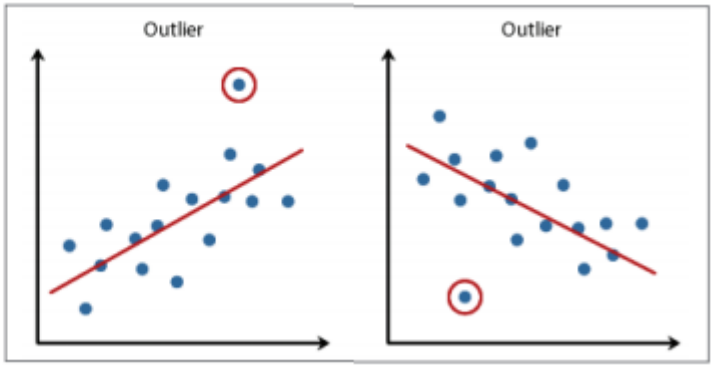
데이터 전처리

- 결측값 (missing value)
- **특이값 (outlier)**
- 데이터 정렬 (order / sort)
- 데이터 분리와 선택 (split / select)
- 데이터 샘플링과 조합 (sampling / combination)
- 데이터 집계와 병합 (aggregate/ merge)

데이터 전처리

- 특이값
  - 특이값의 개념





특이값(outlier)

정상적이라고 생각되는 데이터의 분포 범위 밖에 위치하는 값들

특이값 찾기

논리적으로 있을 수 없는 값이 있는지 찾아본다
ex) 몸무게에 마이너스 값이 있음

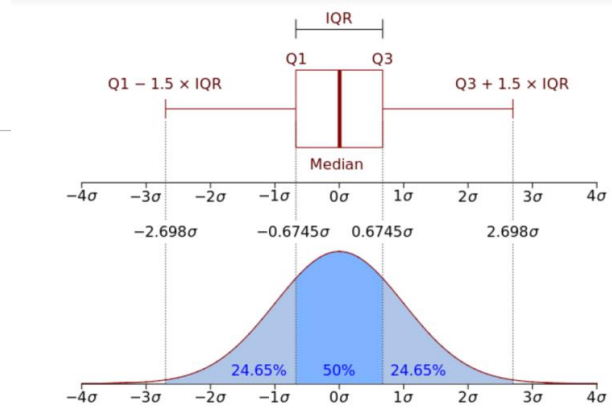
상식을 벗어난 값이 있는지 찾아본다
ex) 나이가 120살 이상인 사람

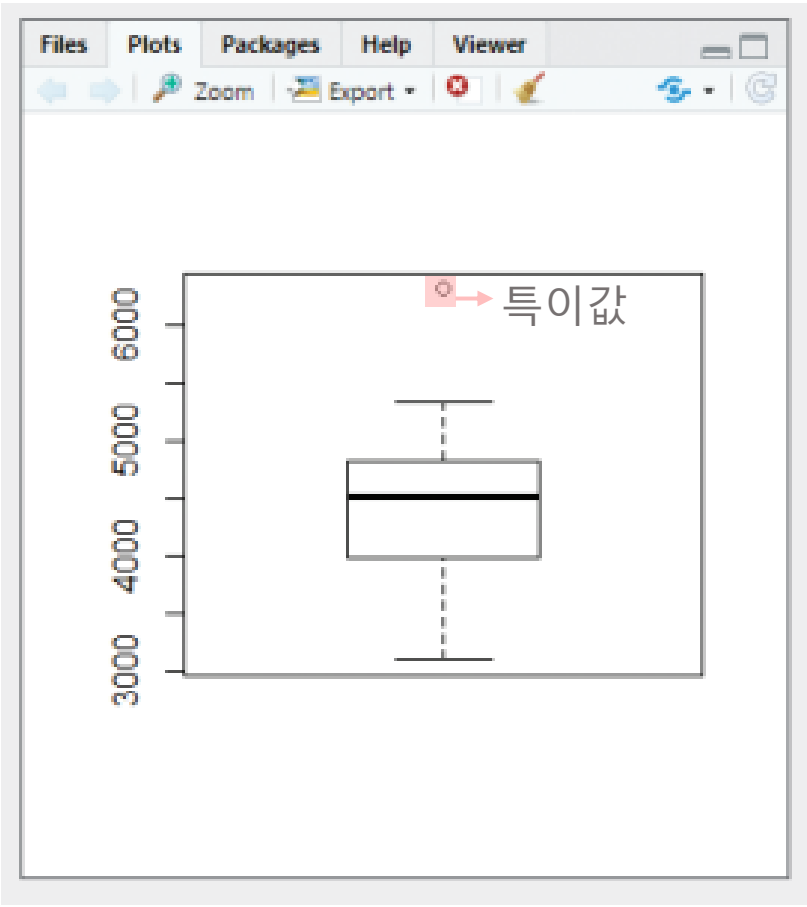
상자그림(boxplot)을 통해 찾아본다
ex) 정상범위 밖에 동그라미 표시가 있으면 특이값을 의미

데이터 전처리

- 특이값
  - 특이값 추출 및 제거 : 상자그림을 통한 특이값 확인
  - boxplot.stats( )
    - 리스트 형태로 여러 개의 결과값 반환
    - 그 중에서 out은 특이값을 의미

```
st ← data.frame(state.x77)
boxplot(st$Income)
boxplot.stats(st$Income)$out
```





- 특이값
  - 특이값을 포함한 행 제거

```
out.val ← boxplot.stats(st$Income)$out          # 특이값 추출
st$Income[st$Income %in% out.val] ← NA          # 특이값을 NA로 대체

newdata ← st[complete.cases(st),]               # NA가 포함된 행 제거
```

  - Income 열에서 Income 값이 out.val에 포함되면 Income 값을 NA로 대체
  - st$Income == out.val로 명령을 내리지 않는 이유
    - out.val이 하나의 값이 아니라 여러 개의 특이값을 포함한 벡터일 수도 있기 때문에
    - == 안 쓰고 %in% 사용

데이터 전처리

- 결측값 (missing value)
- 특이값 (outlier)
- **데이터 정렬 (order / sort)**
- 데이터 분리와 선택 (split / select)
- 데이터 샘플링과 조합 (sampling / combination)
- 데이터 집계와 병합 (aggregate/ merge)

데이터 전처리

- 데이터 정렬
  - **벡터의 정렬** : sort( )함수와 order( )함수 이용
  - order 함수
    - 벡터 및 데이터 프레임의 원소를 크기대로 정렬하여 위치를 추적하는 함수

```
D ← c('c','o','d','e','r')
A ← c(1, 2, 5, 2)

order(D)        # 알파벳 순서에 따른 순번 반환
[1] 1 3 4 2 5
order(A)        # 같은 숫자가 있을 경우 앞에 있는 숫자가 앞의 순서 차지
[1] 1 2 4 3
order(c('c','o','d','e','r', 1, 2, 5, 2))
# 영어과 숫자가 동시에 있을 경우엔 '숫자 - 영어' 순으로 정렬
[1] 6 7 9 8 1 3 4 2 5
```

- 데이터 정렬
  - **벡터의 정렬** : sort( )함수와 order( )함수 이용

```
v1 ← c(1,7,6,8,4,2,3)
v1 ← sort(v1)                    # 오름차순
v2 ← sort(v1, decreasing=T)      # 내림차순
```

- 데이터 정렬
  - **매트릭스와 데이터프레임의 정렬**

    - 특정 열의 값들을 기준으로 행들을 재배열하는 형태로 정렬

```
order(iris$Sepal.Length)
iris[order(iris$Sepal.Length),]                    # 오름차순으로 정렬
iris[order(iris$Sepal.Length, decreasing=T),]      # 내림차순으로 정렬
iris.new ← iris[order(iris$Sepal.Length),]         # 정렬된 데이터를 저장
iris[order(iris$Species, -iris$Petal.Length, decreasing=T),] # 정렬 기준이 2개
```

데이터 전처리

- 데이터 정렬
  - **매트릭스와 데이터프레임의 정렬**

    - 특정 열의 값들을 기준으로 행들을 재배열하는 형태로 정렬

```
>   head(iris)
Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
...
>   order(iris$Sepal.Length)                  # iris의 Sepal.Length열의 값들을 정렬한 인덱스
                                                ex) 14번째 행의 값이 최솟값
[1]   14   9  39  43  42   4   7  23  48   3  30  12  13  25  31
[16]  46   2  10  35  38  58 107   5   8  26  27  36  41  44  50
...
```

- 데이터 정렬
  - **매트릭스와 데이터프레임의 정렬**

    - 특정 열의 값들을 기준으로 행들을 재배열하는 형태로 정렬

```
>   iris[order(iris$Sepal.Length),]                    # Sepal.Length를 기준으로 오름차순 정렬
    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
14           4.3         3.0          1.1         0.1  setosa
9            4.4         2.9          1.4         0.2  setosa
39           4.4         3.0          1.3         0.2  setosa
...
>   iris[order(iris$Sepal.Length, decreasing = T),]    # Sepal.Length를 기준으로 내림차순 정렬
    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
132          7.9         3.8          6.4         2.0 virginica
118          7.7         3.8          6.7         2.2 virginica
119          7.7         2.6          6.9         2.3 virginica
...
```

- 데이터 정렬
  - **매트릭스와 데이터프레임의 정렬**

    - 특정 열의 값들을 기준으로 행들을 재배열하는 형태로 정렬

```
>   iris.new ← iris[order(iris$Sepal.Length),]                    # 정렬된 데이터를 iris.new에 저장

# order(  ) 함수를 적용하면 정렬된 값을 출력할 뿐, 실제 iris 데이터셋의 값은 변경되지 않음
# 정렬된 값을 저장하고 싶다면 새로운 변수에 저장 필요
```

품종별로 내림차순 정렬

```
>   iris[order(iris$Species, -iris$Petal.Length, decreasing=T),]   # 정렬 기준이 2개
                                                                   #  '-'iris$Petal.Length : decreasing에서
```

같은 품종 내에서는 꽃잎의 길이를 기준으로 오름차순 정렬

선언한 순서와 반대 -> 오름차순

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
107	4.9	2.5	4.5	1.7	virginica
127	6.2	2.8	4.8	1.8	virginica
139	6.0	3.0	4.8	1.8	virginica
…					

데이터 전처리

- 결측값 (missing value)
- 특이값 (outlier)
- 데이터 정렬 (order / sort)
- **데이터 분리와 선택 (split / select)**
- 데이터 샘플링과 조합 (sampling / combination)
- 데이터 집계와 병합 (aggregate/ merge)

데이터 전처리

- 데이터 분리와 선택
  - 데이터 분리 : 열의 값을 기준으로 분리 : split( ) 함수 이용
  - sp : 벡터가 아닌 리스트

```
sp ← split(iris, iris$Species)        # 품종별로 데이터 분리
summary(sp)                           # 분리 결과 요약
sp$setosa                             # setosa 품종의 데이터 확인
```

데이터 전처리

- 데이터 분리와 선택
  - 데이터 분리 : 열의 값을 기준으로 분리 : split( ) 함수 이용

```
>   summary(sp)
            Length  Class      Mode                  # sp는 3개의 데이터셋으로 분리되어 있는 '리스트'
setosa      5   data.frame   list                    # Length : 분리된 데이터에서 열의 개수
versicolor  5   data.frame   list
virginica   5   data.frame   list
>   sp$setosa                                        # sp 리스트의 setosa 품종의 데이터만 추출
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
```

데이터 전처리

- 데이터 분리와 선택
  - 데이터 선택 : 조건에 맞는 행들을 추출 : subset( ) 함수 이용

```
>   subset(iris, Species == 'setosa')          # iris 데이터셋에서 품종이 setosa인 행들만 추출
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
...
> subset(iris, Sepal.Length > 7.6,           # iris 데이터셋에서 꽃받침의 길이가 7.6보다 큰 행들만
+        select = c(Petal.Length, Petal.Width))  추출한 뒤, Petal.Length 열과 Petal.Width 열의 값들만
    Petal.Length Petal.Width                    추출
118          6.7         2.2
119          6.9         2.3                  # select 매개변수 : 추출할 열을 지정
123          6.7         2.0
...
```

데이터 전처리

- 데이터 분리와 선택
  - 데이터 선택 : 조건에 맞는 행들을 추출 : subset( ) 함수 이용

```
subset(iris, Species == "setosa")
subset(iris, Sepal.Length > 7.5)
subset(iris, Sepal.Length > 5.1 & Sepal.Width > 3.9)
subset(iris, Sepal.Length > 7.6, select=c(Petal.Length,Petal.Width))
```

  - select 매개변수
    - 추출할 열을 지정

데이터 전처리

- 결측값 (missing value)
- 특이값 (outlier)
- 데이터 정렬 (order / sort)
- 데이터 분리와 선택 (split / select)
- **데이터 샘플링과 조합 (sample / combination)**
- 데이터 집계와 병합 (aggregate/ merge)

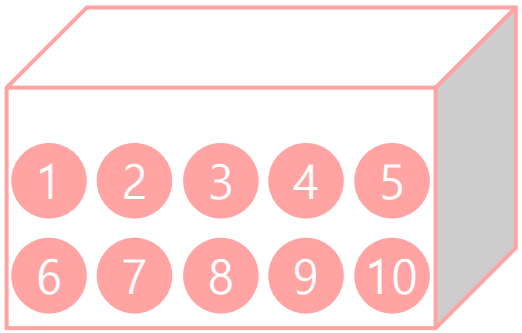
데이터 전처리

- 데이터 샘플링과 조합
  - 데이터 샘플링

샘플링(sampling)

주어진 값들이 있을 때 그 중에서 임의의 개수의 값들을 추출하는 작업

데이터셋의 크기가 너무 커서 데이터 분석에 시간이 많이 걸릴 때 일부의 데이터만 샘플링하여 대략의 결과를 미리 확인하고자 할 때 사용



한 번 추출한 쪽지는 다시 추첨함에 넣지 않는 방식으로 추출 → 비복원추출

추출한 쪽지의 내용을 확인하고 다시 추첨함에 넣은 후 새로운 쪽지를 추출하는 방식 → 복원추출

- 데이터 샘플링과 조합
  - 데이터 샘플링

    - 숫자를 임의로 추출하기 : sample( ) 함수 이용

```
x ← 1:100
y ← sample(x, size=10, replace = FALSE)        # 비복원추출
```

- 데이터 샘플링과 조합
  - 데이터 샘플링
    - 행을 임의로 추출하기 : sample( ) 함수 이용

```
idx ← sample(1:nrow(iris), size=50,  replace = FALSE)
iris.50 ← iris[idx,]                    # 50개의 행 추출
dim(iris.50)                            # 행과 열의 개수 확인
head(iris.50)
```

- 데이터 샘플링과 조합
  - 데이터 샘플링

    - set.seed( ) 함수 이해하기

```
sample(1:20, size=5)
sample(1:20, size=5)
sample(1:20, size=5)

# sample( ) 함수는 임의로 샘플을 추출하는 방식이기 때문에
  함수를 실행할 때마다 매번 결과가 다름

set.seed(100)
sample(1:20, size=5)
set.seed(100)
sample(1:20, size=5)
set.seed(100)
sample(1:20, size=5)

# set.seed( ) 함수를 sample( ) 함수 실행 전에 먼저 실행하여
  set.seed( ) 함수의 매개변수 값이 같으면
  sample( )함수의 결과도 동일함
# set.seed( ) 함수의 매개변수 값이 변함
-> sample( ) 함수의 결과가 변함
```

- 데이터 샘플링과 조합
  - 데이터 조합 : combn( ) 함수 이용

    - 조합(combination) : 주어진 데이터값들 중에서 몇 개씩 짝을 지어 추출하는 작업

```
combn(1:5,3)                          # 1~5에서 3개를 뽑는 조합

x = c("red","green","blue","black","white")
com ← combn(x, 2)                     # x의 원소를 2개씩 뽑는 조합
com

for(i in 1:ncol(com)) {               # 1부터 10(com의 개수)까지
 cat(com[ , i], "₩n")                    com의 i열과 Enter를 출력
}                                        -> 조합이 열별로 출력됨
```

데이터 전처리

- 결측값 (missing value)
- 특이값 (outlier)
- 데이터 정렬 (order / sort)
- 데이터 분리와 선택 (split / select)
- 데이터 샘플링과 조합 (sampling / combination)
- **데이터 집계와 병합 (aggregate/ merge)**

- 데이터 집계와 병합
  - 2차원 데이터는 데이터 그룹에 대해서 합계나 평균을 계산해야 하는 일이 많음
  - 데이터 집계 : aggregate( ) 함수 이용
    - 집계 : 데이터 그룹에 대해서 합계나 평균 등을 계산하는 작업
    - iris 데이터셋에서 각 변수의 품종별 평균 출력

iris의 5열(품종)을 제외한 데이터를 품종을 기준으로 평균을 구함

```
agg ← aggregate(iris[,-5], by=list(iris$Species), FUN=mean)
```

```
> agg <- aggregate(iris[,-5], by=list(iris$Species),
+                  FUN=mean)
> agg
     Group.1 Sepal.Length Sepal.Width Petal.Length Petal.Width
1     setosa        5.006       3.428        1.462       0.246
2 versicolor        5.936       2.770        4.260       1.326
3  virginica        6.588       2.974        5.552       2.026
```

데이터 전처리

- 데이터 집계와 병합
  - 2차원 데이터는 데이터 그룹에 대해서 합계나 평균을 계산해야 하는 일이 많음
  - 데이터 집계 : aggregate( ) 함수 이용
    - 집계 : 데이터 그룹에 대해서 합계나 평균 등을 계산하는 작업
    - iris 데이터셋에서 각 변수의 품종별 평균 출력

집계의 기준이 되는 열의 이름을 '품종 ' 으로 변경

```
agg ← aggregate(iris[,-5], by = list(품종=iris$Species), FUN=mean)
```

```
> agg <- aggregate(iris[,-5], by=list(품종=iris$Species),
+                  FUN=mean)
> agg
        품종 Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
1     setosa        5.006        3.428         1.462        0.246
2 versicolor        5.936        2.770         4.260        1.326
3  virginica        6.588        2.974         5.552        2.026
```

데이터 전처리

- 데이터 집계와 병합
  - 2차원 데이터는 데이터 그룹에 대해서 합계나 평균을 계산해야 하는 일이 많음
  - 데이터 집계 : aggregate( ) 함수 이용

    - 집계 : 데이터 그룹에 대해서 합계나 평균 등을 계산하는 작업

    - iris 데이터셋에서 각 변수의 품종별 표준편차 출력

```
agg ← aggregate(iris[,-5], by=list(표준편차=iris$Species), FUN=sd)
```

```
     표준편차 Sepal.Length Sepal.Width Petal.Length Petal.Width
1     setosa    0.3524897   0.3790644    0.1736640   0.1053856
2 versicolor    0.5161711   0.3137983    0.4699110   0.1977527
3  virginica    0.6358796   0.3224966    0.5518947   0.2746501
```

- 데이터 집계와 병합
  - 데이터 집계 : aggregate( ) 함수 이용
    - mtcars 데이터셋에서 각 변수의 최댓값 출력

cyl과 vs를 기준으로 다른 열들의 최댓값을 출력

```
agg ← aggregate(mtcars, by=list(cyl=mtcars$cyl, vs=mtcars$vs), FUN=max)
```

```
  cyl vs  mpg cyl  disp  hp drat    wt  qsec vs am gear carb
1   4  0 26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
2   6  0 21.0   6 160.0 175 3.90 2.875 17.02  0  1    5    6
3   8  0 19.2   8 472.0 335 4.22 5.424 18.00  0  1    5    8
4   4  1 33.9   4 146.7 113 4.93 3.190 22.90  1  1    5    2
5   6  1 21.4   6 258.0 123 3.92 3.460 20.22  1  0    4    4
```

데이터 전처리

- 데이터 집계와 병합
  - 데이터 병합

name	math
a	90
b	80
c	40

name	Korean
a	75
b	60
d	90

name을 공통열로 가지고 있음
-> name을 중심으로 병합

```
>   x ← data.frame(name = c('a','b','c'), math = c(90,80,40))
>   y ← data.frame(name = c('a','b','d'), korean = c(75,60,90))
```

데이터 전처리

- 데이터 집계와 병합
  - 데이터 병합 : merge( ) 함수 이용

```
x ← data.frame(name=c("a","b","c"), math=c(90,80,40))
y ← data.frame(name=c("a","b","d"), korean=c(75,60,90))
z ← merge(x,y, by=c("name"))
```

```
> z <- merge(x,y, by=c("name"))
> z
  name math korean
1    a   90     75
2    b   80     60
```

#'name'열을 기준으로 x, y를 병합
name이 c, d인 경우 상대방의 데이터셋에 대응하는 값이 없기 때문에 병합에서 제외

- 데이터 집계와 병합
  - 데이터 병합 : merge( ) 함수 이용

```
merge(x,y, all.x=T)   # 첫 번째 데이터셋의 행들은 모두 표시되도록
merge(x,y, all.y=T)   # 두 번째 데이터셋의 행들은 모두 표시되도록
merge(x,y, all=T)     # 두 데이터셋의 모든 행들이 표시되도록
```

- 데이터 집계와 병합
  - 데이터 병합 : merge( ) 함수 이용

```
>   merge(x, y, all.x=T)   # x의 행들은 모두 출력
 name math korean          -> 대응되는 행이 y에 있으면 병합해서 출력하고, 없으면 NA로 출력
1    a   90     75
2    b   80     60
3    c   40     NA
>   merge(x, y, all.y=T)   # y의 행들은 모두 출력
 name math korean          -> 대응되는 행이 x에 있으면 병합해서 출력하고, 없으면 NA로 출력
1    a   90     75
2    b   80     60
3    d   NA     90
>   merge(x, y, all=T)     # x,y의 공통 열의 값들이 어느 쪽에 있더라도 모두 출력하고,
 name math korean          대응되는 행들이 있으면 병합하고 없으면 NA로 출력
1    a   90     75
2    b   80     60
3    c   40     NA
4    d   NA     90
```

- 데이터 집계와 병합
  - 데이터 병합 : merge( ) 함수 이용

```
x <- data.frame(name=c("a","b","c"), math=c(90,80,40))
y <- data.frame(sname=c("a","b","d"), korean=c(75,60,90))

merge(x,y, by.x=c("name"), by.y=c("sname"))
```

```
> merge(x,y, by.x=c("name"), by.y=c("sname"))
  name math korean
1    a   90     75
2    b   80     60
```

by.x : x의 병합 기준 열의 이름 지정
by.y : y의 병합 기준 열의 이름 지정

예시

- 예시 1 : state.x77의 결측값 분석

```
1   ds← state.x77
2   ds[2,3] ← NA; ds[3,1] ← NA; ds[2,4] ← NA; ds[4,3] ← NA
# cording here #
```

- 실행 결과

```
>   ds← state.x77
>   ds[2,3] ← NA; ds[3,1] ← NA; ds[2,4] ← NA; ds[4,3] ← NA
> for                  {                                    # ds의 열별 결측값
+
+   cat(colnames(ds)[i], "₩t",              , "₩n")
+ }
Population    1
Income        0
Illiteracy    2
Life Exp      1
...
```

예시

- 예시 2 : state.x77의 결측값 분석

```
1   ds← state.x77
2   ds[2,3] ← NA; ds[3,1] ← NA; ds[2,4] ← NA; ds[4,3] ← NA
# cording here #
```

- 실행 결과

```
>   ds← state.x77
>   ds[2,3] ← NA; ds[3,1] ← NA; ds[2,4] ← NA; ds[4,3] ← NA
>                                  # 결측값이 포함된 행의 개수
[1]   3
>   ds.new ←                       # 결측값이 포함된 행들을 제외하고 새로운 데이터셋
>   head(ds.new)                     ds.new 생성
 Population Income Illiteracy Life Exp Murder HS Grad
Alabama          3615   3624        2.1    69.05   15.1    41.3
California      21198   5114        1.1    71.71   10.3    62.6
Colorado         2541   4884        0.7    72.06    6.8    63.9
...
```

예시

- 예시 3 : state.x77의 Population열의 특이값 분석

```
1   df ← data.frame(state.x77)
# cording here #
```

- 실행 결과

```
>   df ← data.frame(state.x77)                    # 결측값이 포함된 행의 개수
>   out_val ←                                     # df의 특이값을 out_val에 저장
>                                  ← NA           # df의 특이값을 NA값으로 대체
>   df_2 ← df[complete.cases(df),]                # NA값을 제거한 df를 df_2에 저장
>   rownames(df)
[1]   "Alabama"       "Alaska"        "Arizona"       "Arkansas"
[5]   "California"    "Colorado"      "Connecticut"   "Delaware"
...
>   rownames(df_2)
[1]   "Alabama"       "Alaska"        "Arizona"       "Arkansas"
[5]   "Colorado"      "Connecticut"   "Delaware"      "Florida"
...
```

예시

- 예시 4 : state.x77 정렬

```
# cording here #
```

- 실행 결과

```
>    state.x77                                          # state.x77을 Population을
          Population Income Illiteracy Life Exp Murder HS Grad Frost Area 기준으로 오름차순 정렬
Alaska       365      6315       1.5      69.31     11.3       66.7     152   566432
Wyoming      376      4566       0.6      70.29      6.9       62.9     173    97203
...
>    state.x77                                          # state.x77을 Income을
           Population Income Illiteracy Life Exp Murder HS Grad Frost Area 기준으로 내림차순 정렬
Alaska        365      6315       1.5       69.31     11.3       66.7     152   566432
Connecticut  3100      5348       1.1       72.48      3.1       56.0     139     4862
...
```

예시

- 예시 5 : mtcars 데이터 분리

```
# cording here #
```

- 실행 결과

```
>   mt_g ← s                                  # mtcars의 gear 열을 기준으로 데이터를
>   mt_g                                        분리하여 mt_g에 저장
$`3`
                     mpg  cyl  disp  hp  drat  wt    qsec vs am gear carb
Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
...
>   mt_wt ← s                                 # mtcars의 wt가 1.5보다 크고 3.0보다 작은
>   mt_wt                                       행을 추출하여 mt_wt에 저장
                   mpg cyl disp  hp  drat  wt    qsec vs am gear carb
Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
...
```

예시

- 예시 6 : state.x77 데이터 샘플링

```
1    set.seed(100)
# cording here #
```

- 실행 결과

```
>    set.seed(100)
>    st_sam ←                                  # state.x77의 20개의 행 번호를 임의로 추출
>    st_other ←                                # 추출한 행이 아닌 나머지 행을
>    st_other                                    st_other에 저장
           Population Income Illiteracy Life Exp Murder HS Grad Frost   Area
Alabama      3615      3624       2.1     69.05   15.1    41.3    20   50708
Arizona      2212      4530       1.8     70.55    7.8    58.1    15  113417
...
```

예시

- 예시 7 : state.x77 데이터 샘플링

```
authors ← data.frame(
  surname = c('Twein', 'Venables', 'Tierney', 'McNeil'),
  nationality = c('US', 'Australia', 'US', 'UK')
)
books ← data.frame(
  name = c('Venables', 'Tierney', 'Ripley', 'McNeil'),
  title = c('Modern Applied Statistics ...', 'LISP-STAT', 'Spatial Statistics',
            'Interactive Data Analysis')
)
# cording here #
```

- 실행 결과

```
> 
  surname nationality                    title
1  McNeil          UK Interactive Data Analysis
2 Tierney          US                LISP-STAT
3 Venables   Australia Modern Applied Statistics ...
```

authors와 books를 각각 surname, name을 기준으로 병합 단, 공통 열의 값이 일치하는 것들만 병합

감사합니다

kimtwan21@dongduk.ac.kr

김 태 완