PR8 - Apply, Aggregate

김서준

2023년 11월 03일

1. apply

- 복수의 데이터에 함수를 일괄 적용할때 사용함
- apply, lapply, sapply, vapply, tapply, mapply 등이 있음
- 각 apply함수는 입력받는 데이터의 형태와 출력하는 데이터의 형태에 따라 다르게 적용함

1.1. apply 함수

- 형식 : apply(data, margin(1또는2),function)
- margin 인수를 1 또는 2로 사용하며 1은 행, 2는 열을 적용
- 행이나 열의 함계, 평균등을 일괄적으로 구할수 있음

```
head(mtcars, 1)
```

```
##           mpg cyl disp  hp drat   wt  qsec vs am gear carb
## Mazda RX4  21   6  160 110  3.9 2.62 16.46  0  1    4    4
```

```
apply(mtcars[1:3,], 1, FUN = mean)#1~3 행의 평균
```

```
##     Mazda RX4 Mazda RX4 Wag    Datsun 710
##      29.90727      29.98136      23.59818
```

```
apply(mtcars[,1:3], 2, FUN = mean)#1~3 열의 평균
```

```
##       mpg       cyl      disp
##  20.09062   6.18750 230.72188
```

1.2. lapply(list apply)

- 형식 : lapply(data, function)
- 리스트형의 데이터를 받아 리스트로 결과를 반환
- 데이터프레임의 각 열은 리스트로 구성되어있음

```
lapply(mtcars[,1:3],mean)
```

```
## $mpg
## [1] 20.09062
##
## $cyl
## [1] 6.1875
##
## $disp
## [1] 230.7219
```

1.3. sapply (simple apply)

- 형식 : function(data, function, simplify=F)
- 입력값 : 벡터, 리스트, 데이터프레임 가능
- 출력값 : 벡터, 리스트, 매트릭스 형태로 결과를 반환
- 인수 simplify = F 이면 리스트로 결과 반환

```
x <- 1:5 ; y <- 11:14
z <-list(x,y)
sapply(x, function(x){x + 1}) # 벡터입력, 벡터 출력
```

```
## [1] 2 3 4 5 6
```

```
sapply(z, function(x){x + 1}) # 리스트 입력, 리스트 출력
```

```
## [[1]]
## [1] 2 3 4 5 6
##
## [[2]]
## [1] 12 13 14 15
```

```
sapply(mtcars[1:3,], function(x){x+1}) #데이터프레임 입력, 매트릭스 출력
```

```
##       mpg cyl disp  hp drat    wt  qsec vs am gear carb
## [1,] 22.0   7  161 111 4.90 3.620 17.46  1  2    5    5
## [2,] 22.0   7  161 111 4.90 3.875 18.02  1  2    5    5
## [3,] 23.8   5  109  94 4.85 3.320 19.61  2  2    5    2
```

```
sapply(mtcars[1:3,], function(x){x+1}, simplify = F) #데이터프레임 입력, 리스트 출력
```

```
## $mpg
## [1] 22.0 22.0 23.8
##
## $cyl
## [1] 7 7 5
##
## $disp
## [1] 161 161 109
##
## $hp
## [1] 111 111  94
##
## $drat
## [1] 4.90 4.90 4.85
##
## $wt
## [1] 3.620 3.875 3.320
##
## $qsec
## [1] 17.46 18.02 19.61
##
## $vs
## [1] 1 1 2
##
## $am
## [1] 2 2 2
##
## $gear
## [1] 5 5 5
##
## $carb
## [1] 5 5 2
```

1.4. tapply(table apply)

- 그룹으로 묶은 후 함수를 적용, 적용 값을 벡터나 행렬로 반환

```
patient <- read.table("sample_data.txt", header = TRUE)
factor(patient$type)
```

```
## [1] Type1 Type2 Type1 Type1 Type2 Type2
## Levels: Type1 Type2
```

```
tapply(patient$type, patient$type, length) #type에 따른 그룹별 환자의 수
```

```
## Type1 Type2
##     3     3
```

```
tapply(patient$age, patient$type, mean)
```

```
## Type1 Type2
##    35    37
```

```
getwd()
```

```
## [1] "C:/UnivStudy/UnivLectures/23-2/R-programming/Works/PR8(apply, aggregrate)"
```

1.5. ‘mapply’ (multi simple apply)

- 여러 개의 리스트에 함수를 적용

```
m1 <- list(a=c(1:10), b= c(11:20))
m2 <- list(c=c(21:30), d=c(31:40))
mapply(sum, m1$a, m2$d) # 2개의 리스트에 적용
```

```
##  [1] 32 34 36 38 40 42 44 46 48 50
```

```
mapply(sum, m1$a, m1$b, m2$c, m2$d) # 4개의 리스트에 적용
```

```
##  [1]  64  68  72  76  80  84  88  92  96 100
```

2. aggregating

- 예제데이터

```
seg.df <- read.csv("rintro-chapter5.csv")
head(seg.df)
```

```
##        age gender   income kids ownHome subscribe    Segment
## 1 47.31613   Male 49482.81    2   ownNo     subNo Suburb mix
## 2 31.38684   Male 35546.29    1  ownYes     subNo Suburb mix
## 3 43.20034   Male 44169.19    0  ownYes     subNo Suburb mix
## 4 37.31700 Female 81041.99    1   ownNo     subNo Suburb mix
## 5 40.95439 Female 79353.01    3  ownYes     subNo Suburb mix
## 6 43.03387   Male 58143.36    4  ownYes     subNo Suburb mix
```

2.1. mean, sd 통계함수

```
attach(seg.df)
mean(income[Segment == "Moving up"]) #Moving up 세 그먼트집단의소득평균
```

```
## [1] 53090.97
```

```
mean(income[Segment == "Moving up" & subscribe == "subNo"]) #Moving up 세그먼트 + 서비스 미사용자의 소득평균
```

```
## [1] 53633.73
```

2.2. apply 함수

```
apply(seg.df[,c(1,3,4)], 2, mean) #나이, 수입, 자녀수평균
```

```
##         age      income        kids
##    41.19965 50936.53618     1.27000
```

```
str(apply(seg.df [, c(1, 3,4)], 2, mean))
```

```
##  Named num [1:3] 41.2 50936.54 1.27
##  - attr(*, "names")= chr [1:3] "age" "income" "kids"
```

```
apply(seg.df [Segment == "Moving up", c(1,3,4)], 2, mean) #Moving up 세그먼트 + 서비스미사용자 의소득평균
```

```
##          age       income         kids
##    36.331144 53090.965253     1.914286
```

2.3. table 함수

```
table(kids) # 자녀수 현황
```

```
## kids
##   0   1   2   3   4   5   6   7
## 121  70  51  36  13   6   2   1
```

```
table(ownHome, subscribe) # 이용자기준, 주거형태현황
```

```
##         subscribe
## ownHome  subNo subYes
##   ownNo    137     22
##   ownYes   123     18
```

```
table(Segment, kids, subscribe) # 세그먼트, 구독여부, 자녀수
```

```
## , , subscribe = subNo
##
##             kids
## Segment       0  1  2  3  4  5  6  7
##   Moving up  12  9 15 11  5  3  0  1
##   Suburb mix 11 35 20 17  7  3  1  0
##   Travelers  70  0  0  0  0  0  0  0
##   Urban hip  16 12  7  4  1  0  0  0
##
## , , subscribe = subYes
##
##             kids
## Segment       0  1  2  3  4  5  6  7
##   Moving up   1  8  3  2  0  0  0  0
##   Suburb mix  0  1  2  2  0  0  1  0
##   Travelers  10  0  0  0  0  0  0  0
##   Urban hip   1  5  4  0  0  0  0  0
```

2.4. by 함수

- 사용방식: by(목표변수, 기준변수, 함수)
- by 함수는 결과 값을 리스트로 반환한다.

```
by(income, Segment, mean)
```

```
## Segment: Moving up
## [1] 53090.97
## ------------------------------------------------------------
## Segment: Suburb mix
## [1] 55033.82
## ------------------------------------------------------------
## Segment: Travelers
## [1] 62213.94
## ------------------------------------------------------------
## Segment: Urban hip
## [1] 21681.93
```

```
by(income, list(Segment, subscribe), mean)
```

```
## : Moving up
## : subNo
## [1] 53633.73
## ------------------------------------------------------------
## : Suburb mix
## : subNo
## [1] 54942.69
## ------------------------------------------------------------
## : Travelers
## : subNo
## [1] 62746.11
## ------------------------------------------------------------
## : Urban hip
## : subNo
## [1] 22082.11
## ------------------------------------------------------------
## : Moving up
## : subYes
## [1] 50919.89
## ------------------------------------------------------------
## : Suburb mix
## : subYes
## [1] 56461.41
## ------------------------------------------------------------
## : Travelers
## : subYes
## [1] 58488.77
## ------------------------------------------------------------
## : Urban hip
## : subYes
## [1] 20081.19
```

2.5. aggregate 함수

- 사용방식 : aggregate(목표변수, 기준변수, 함수)
- 결과값을 데이터프레임으로 출력해 주는 것이 가장 큰 장점임.
- 기준변수가list로 입력되어야 한다.

```
aggregate(income, list(Segment), mean)
```

```
##      Group.1        x
## 1  Moving up 53090.97
## 2 Suburb mix 55033.82
## 3  Travelers 62213.94
## 4  Urban hip 21681.93
```

```
str(aggregate(income, list(Segment), mean))
```

```
## 'data.frame':    4 obs. of  2 variables:
##  $ Group.1: chr  "Moving up" "Suburb mix" "Travelers" "Urban hip"
##  $ x      : num  53091 55034 62214 21682
```

- 포뮬러를 사용하면 효과적이다 (변수명지정, 리스트변환)

```
aggregate(income ~ Segment, data=seg.df, mean)
```

```
##      Segment   income
## 1  Moving up 53090.97
## 2 Suburb mix 55033.82
## 3  Travelers 62213.94
## 4  Urban hip 21681.93
```

```
aggregate(income~Segment+ownHome+subscribe, data=seg.df, mean)
```

```
##       Segment ownHome subscribe   income
## 1   Moving up   ownNo     subNo 55402.89
## 2  Suburb mix   ownNo     subNo 54579.99
## 3   Travelers   ownNo     subNo 65852.54
## 4   Urban hip   ownNo     subNo 21604.16
## 5   Moving up  ownYes     subNo 49898.85
## 6  Suburb mix  ownYes     subNo 55354.86
## 7   Travelers  ownYes     subNo 61749.71
## 8   Urban hip  ownYes     subNo 23993.93
## 9   Moving up   ownNo    subYes 50675.70
## 10 Suburb mix   ownNo    subYes 63753.97
## 11  Travelers   ownNo    subYes 48091.75
## 12  Urban hip   ownNo    subYes 20271.33
## 13  Moving up  ownYes    subYes 51359.44
## 14 Suburb mix  ownYes    subYes 52815.13
## 15  Travelers  ownYes    subYes 62944.64
## 16  Urban hip  ownYes    subYes 19320.64
```

2.6. cut 함수

- cut함수는 연속형 변수를 특정 구간으로 구분하여 명목형 변수로 변환한다.
- cut(데이터, breaks=구간수, labels = 구간이름)

```
cut.data = aggregate(income ~ Segment + ownHome + subscribe, data = seg.df, mean)
cut.data$income2 = cut(cut.data$income, breaks = seq(0, 70000, 10000))
cut.data$income2 = cut(cut.data$income, breaks = c(0, 20000, 30000,40000, 50000, 60000, 70000),
labels = c('2만이하', '2만~3만', '3만~4만', '4만~5만', '5만~6만', '6만이상'))
cut.data
```

```
##       Segment ownHome subscribe   income income2
## 1   Moving up   ownNo     subNo 55402.89 5만~6만
## 2  Suburb mix   ownNo     subNo 54579.99 5만~6만
## 3   Travelers   ownNo     subNo 65852.54 6만이상
## 4   Urban hip   ownNo     subNo 21604.16 2만~3만
## 5   Moving up  ownYes     subNo 49898.85 4만~5만
## 6  Suburb mix  ownYes     subNo 55354.86 5만~6만
## 7   Travelers  ownYes     subNo 61749.71 6만이상
## 8   Urban hip  ownYes     subNo 23993.93 2만~3만
## 9   Moving up   ownNo    subYes 50675.70 5만~6만
## 10 Suburb mix   ownNo    subYes 63753.97 6만이상
## 11  Travelers   ownNo    subYes 48091.75 4만~5만
## 12  Urban hip   ownNo    subYes 20271.33 2만~3만
## 13  Moving up  ownYes    subYes 51359.44 5만~6만
## 14 Suburb mix  ownYes    subYes 52815.13 5만~6만
## 15  Travelers  ownYes    subYes 62944.64 6만이상
## 16  Urban hip  ownYes    subYes 19320.64 2만이하
```

2.7. grep 함수

```
grep("ap", c("apple", "Apple", "apple2", "bbapple")) #ap를 포함하는 원소들의 위치
```

```
## [1] 1 3 4
```

```
grep ("ap", c("apple", "Apple", "apple2", "bbapple"), value=TRUE) #ap를 포함하는 원소
```

```
## [1] "apple"   "apple2"  "bbapple"
```

```
grep("[1-3]", c("apple1", "apple2", "apple3", "apple4", "Apple1")) #1,2,3을 포함하는 원소위치
```

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

```
grepl("ap", c("apple", "Apple", "apple2", "bbapple")) #ap를 포함하는 원소들의위치
```

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

- 공통된 패턴을 가진 자료들의 위치를 찾아서 위치 값을 활용해 데이터를 일괄 변환할 매 사용한다

```
seg.df $ownHome = as.character (seg.df$ownHome)
grep('Yes', seg.df$ownHome)
```

```
##   [1]   2   3   5   6  10  11  14  15  16  17  18  19  20  21  22  24  25  26
##  [19]  33  37  39  40  41  43  47  50  51  52  53  55  57  68  72  73  75  79
##  [37]  80  81  83  84  87  90  91  92  95  96  97  99 108 118 120 122 125 130
##  [55] 139 144 145 150 151 152 153 155 156 157 159 160 161 162 163 164 165 166
##  [73] 167 168 169 170 171 175 176 177 178 179 180 181 182 183 184 185 187 188
##  [91] 189 190 192 194 195 196 197 198 199 200 201 203 204 207 208 210 211 213
## [109] 214 215 216 220 221 223 224 225 226 228 231 232 236 238 240 241 247 252
## [127] 258 261 264 265 269 271 273 274 276 279 286 293 295 296 300
```

```
head(seg.df)
```

```
##        age gender   income kids ownHome subscribe    Segment
## 1 47.31613   Male 49482.81    2   ownNo     subNo Suburb mix
## 2 31.38684   Male 35546.29    1  ownYes     subNo Suburb mix
## 3 43.20034   Male 44169.19    0  ownYes     subNo Suburb mix
## 4 37.31700 Female 81041.99    1   ownNo     subNo Suburb mix
## 5 40.95439 Female 79353.01    3  ownYes     subNo Suburb mix
## 6 43.03387   Male 58143.36    4  ownYes     subNo Suburb mix
```

```
seg.df$ownHome [grep('Yes', seg.df$ownHome) ] = 'Yes'
head(seg.df)
```

```
##        age gender   income kids ownHome subscribe    Segment
## 1 47.31613   Male 49482.81    2   ownNo     subNo Suburb mix
## 2 31.38684   Male 35546.29    1     Yes     subNo Suburb mix
## 3 43.20034   Male 44169.19    0     Yes     subNo Suburb mix
## 4 37.31700 Female 81041.99    1   ownNo     subNo Suburb mix
## 5 40.95439 Female 79353.01    3     Yes     subNo Suburb mix
## 6 43.03387   Male 58143.36    4     Yes     subNo Suburb mix
```

2.8. ‘gsub’ 함수

- 현재데이터의 Segment 컬럼에 한칸 띄워쓰기를 없애고 싶을때, 다음과 같이 사용한다.

```
seg.df$Segment <- gsub(" ", "", seg.df$Segment)
head(seg.df)
```

```
##        age gender   income kids ownHome subscribe   Segment
## 1 47.31613   Male 49482.81    2   ownNo     subNo Suburbmix
## 2 31.38684   Male 35546.29    1     Yes     subNo Suburbmix
## 3 43.20034   Male 44169.19    0     Yes     subNo Suburbmix
## 4 37.31700 Female 81041.99    1   ownNo     subNo Suburbmix
## 5 40.95439 Female 79353.01    3     Yes     subNo Suburbmix
## 6 43.03387   Male 58143.36    4     Yes     subNo Suburbmix
```

PR8 연습문제

다음은 노트북 기종 빛 제조사별 가격 종류 등에 대한 데이터셋이다. 해당 데이터셋을 lapptop이라는 변수에 저장하고 아래의 문제들을 해결하시오.

```
laptop <- read.csv("Cleaned_Laptop_data.csv")
```

문제1. 노트북 제조사별(brand)로 리뷰의 개수(reviews)를 구하시오

조건1. answer1이라는 변수에 저장하시오. 조건2. head(answer1)을 통해 출력하시오.

```
answer1 <- aggregate(reviews ~ brand, data = laptop, FUN = length)

head(answer1)
```

```
##       brand reviews
## 1      acer      58
## 2 ALIENWARE       4
## 3     APPLE      28
## 4      ASUS     254
## 5     Avita      18
## 6      DELL     154
```

문제2. 6,8,9 번째 열에는 'GB'가 포함되어 있다. 'GB'를 삭제하고 숫자형으로 바꾸시오.

조건1. sapply 사용하시오. 조건2. gsub 사용하시오. 조건3. answer2이라는 변수에 저장하시오. 조건4. head(answer2)을 통해 출력하시오.

```
answer2 <- laptop

answer2[, c(6, 8, 9)] <- sapply(answer2[, c(6, 8, 9)], function(x) as.numeric(gsub("GB", "",
x)))

head(answer2)
```

```
##    brand   model processor_brand    processor_name processor_gnrtn ram_gb
## 1 Lenovo A6-9225             AMD A6-9225 Processor            10th      4
## 2 Lenovo Ideapad             AMD          APU Dual            10th      4
## 3  Avita    PURA             AMD          APU Dual            10th      4
## 4  Avita    PURA             AMD          APU Dual            10th      4
## 5  Avita    PURA             AMD          APU Dual            10th      4
## 6  Avita    PURA             AMD          APU Dual            10th      8
##   ram_type ssd  hdd      os os_bit graphic_card_gb     weight display_size
## 1     DDR4   0 1024 Windows 64-bit              0 ThinNlight      Missing
## 2     DDR4   0  512 Windows 64-bit              0     Casual      Missing
## 3     DDR4 128    0 Windows 64-bit              0 ThinNlight      Missing
## 4     DDR4 128    0 Windows 64-bit              0 ThinNlight      Missing
## 5     DDR4 256    0 Windows 64-bit              0 ThinNlight      Missing
## 6     DDR4 256    0 Windows 64-bit              0 ThinNlight           14
##   warranty Touchscreen msoffice latest_price old_price discount star_rating
## 1        0          No       No        24990     32790       23         3.7
## 2        0          No       No        19590     21325        8         3.6
## 3        0          No       No        19990     27990       28         3.7
## 4        0          No       No        21490     27990       23         3.7
## 5        0          No       No        24990     33490       25         3.7
## 6        0          No       No        24990     33490       25         3.7
##   ratings reviews
## 1      63      12
## 2    1894     256
## 3    1153     159
## 4    1153     159
## 5    1657     234
## 6    1657     234
```

문제3. 이번 수업 시간에 배웠던 함수를 사용하여 5개 이상의 코드를 작성하고, 주석을 상세하게 추가하시오.

조건1. apply계열 함수, aggregate 함수는 반드시 각각 1개 이상씩 포함되어야 함. 조건2. 주석은 코드 개수에 포함되지 않음.

```
answer2 <- laptop
answer2[, c(6, 8, 9)] <- sapply(answer2[, c(6, 8, 9)], function(x) as.numeric(gsub("GB", "", x)))
# 1. RAM 용량에 따른 가격 평균 및 표준 편차 계산
ram_mean <- mean(answer2$ram_gb)# RAM 용량의 평균 계산
ram_sd <- sd(answer2$ram_gb) # RAM 용량의 표준 편차 계산
ram_price_mean <- aggregate(latest_price ~ ram_gb, data = answer2, FUN = mean)  # RAM 용량별 가격 평균 계산
head(ram_price_mean)  # 결과 출력
```

```
##   ram_gb latest_price
## 1      4     59468.00
## 2      8     68397.81
## 3     16    118955.19
## 4     32    168959.67
```

```
# 2. 브랜드 별 별점 평균 계산
brand_rate_mean <- aggregate(star_rating ~ brand, data = answer2, FUN = mean)  # 브랜드 별 별점 평균 계산
head(brand_rate_mean) # 결과 출력
```

```
##       brand star_rating
## 1      acer    3.055172
## 2 ALIENWARE    4.400000
## 3     APPLE    4.717857
## 4      ASUS    2.681496
## 5     Avita    1.805556
## 6      DELL    2.863636
```

```
# 3. 운영체제(OS) 별 별점 평균 계산
os_rate_mean <- tapply(answer2$star_rating, answer2$os, mean)  # 운영체제(OS) 별 별점 평균 계산
head(os_rate_mean) # 결과 출력
```

```
##      DOS      Mac  Windows
## 3.647222 4.717857 2.893149
```

```
# 4. 운영체제 비트(os_bit)에 따른 가격 평균 계산
os_bit_price_mean <- aggregate(latest_price ~ os_bit, data = answer2, FUN = mean) # 운영체제 비트별 가격 평균 계산
head(os_bit_price_mean) # 결과 출력
```

```
##   os_bit latest_price
## 1 32-bit     82768.27
## 2 64-bit     75164.15
```

```
# 5. 브랜드 별 리뷰 수 합계 계산
brand_rev_cnt <- aggregate(reviews ~ brand, data = answer2, FUN = sum) # 브랜드 별 리뷰 수 합계 계산
head(brand_rev_cnt)  # 결과 출력
```

```
##       brand reviews
## 1      acer    3700
## 2 ALIENWARE       9
## 3     APPLE    3291
## 4      ASUS   12050
## 5     Avita    1040
## 6      DELL    2184
```