PR8- Apply,Aggregate

조성우

2020년5월8일

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. apply함수

- 형식: apply(data.margin(1또는2),function)
- margin 안수를1 또는2로 사용하며1은행, 2는 열을 적용
- 행이나 열의 합계, 평균등을 일괄적으로 구할 수 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 appl)

- 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
- 인수simplify=F이면 리스트로 결과 반환

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

## [1] 2 3 4 5 6

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

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

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) #type에 따른 그룹별 나의 평균
```

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

2. aggregating

- 예제 데이터

```
seg.df <- read.csv("http://goo.gl/qw303p")
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 세그먼트 집단의 소득평균

## [1] 53090.97

mean(income[Segment == "Moving up" & subscribe == "subNo"]) # Moving up
세그먼트 + 서비스미사용자의소득평균

## [1] 53633.73
```

2.2. apply 함수

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

##         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 +
서비스미사용자의 소득평균

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

2.3. table함수

```
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)
- by함수는 결과값을 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 주는것이 가장 큰 장점임
- 기준변수가list로 입력돼야 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: Factor w/ 4 levels "Moving up","Suburb mix",..: 1 2 3 4
##  $ 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('2만 이하',
'2만~3만','3만~4만','4만~5만','5만~6만','6만 이상'))
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 6만 이상
## 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 6만 이상
## 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 6만 이상
```

```
## 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 6만 이상
## 16  Urban hip  ownYes    subYes 19320.64 2만 이하
```

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")) #1,2,3를 포함하는 원소
```

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

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

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

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

- 현재데이터의Segment 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
```

2.9. which,which.max ,which.min

```
x <-c(2,4,6,7,10)
x%%2
```

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

```
which(x %% 2 == 0)
```

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

```
x[which(x %% 2 == 0 )]
```

```
## [1]  2  4  6 10
```

```
x <- c(2,4,6,7,10)
which.min(x)
```

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

```
x[which.min(x)]
```

```
## [1] 2
```

```
which.max(x)
```

```
## [1] 5
```

```
x[which.max(x)]
```

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

PR8 연습문제

문제1

- 다음 문제를 grade.csv 파일을 활용해 해결하세요 ###### 1.grade.csv 파일을 grade 변수에 저장합니다

```
grade <- read.csv("grade.csv") #1
```

2.aggregate 함수를 활용해 각 반별 수학점수 평균을 출력해보세요.

```
MathMbyclass<- aggregate(grade$math,list(grade$class),mean) #2
print(MathMbyclass)
```

```
##   Group.1     x
## 1       A 62.50
## 2       B 51.25
```

3.apply 함수를 활용해 수학 점수와 컴퓨터 점수의 평균을 출력해보세요.

```
apply(grade[,c(4,5)],2,FUN=mean) #3
```

```
##     math computer
## 55.00000 59.16667
```

문제2

- PR5 문제1 에서 "," 로 인해 숫자가 character 형으로 인식되는 문제가 발생했었습니다.
- 이 문제를 sapply 함수를 사용하여 해결해보세요.
- Hint: gsub 함수 이용 (특정 문자열 치환)

PR5 문제1 불러오기

```r
#불러오기 + 문제점발견
library(XML)
library(httr)
```

```
## Warning: package 'httr' was built under R version 3.6.3
```

```r
url <- "https://www.worldometers.info/coronavirus/"


html_source <- GET(url) #html 전체 소스를 받아옴
tabs <- readHTMLTable(rawToChar(html_source$content), stringsAsFactors = F)
covid_yesterday <- tabs$main_table_countries_yesterday
str(covid_yesterday)
```

```
## 'data.frame':    222 obs. of  13 variables:
##  $ Country,Other    : chr  "Asia" "North America" "Europe" "South America"
...
##  $ TotalCases       : chr  "634,831" "1,442,750" "1,581,256" "282,175" ...
##  $ NewCases         : chr  "+16,327" "+33,432" "+27,556" "+16,063" ...
##  $ TotalDeaths      : chr  "21,501" "87,090" "150,514" "14,577" ...
##  $ NewDeaths        : chr  "+296" "+2,129" "+2,053" "+994" ...
##  $ TotalRecovered   : chr  "349,156" "277,874" "628,115" "99,221" ...
##  $ ActiveCases      : chr  "264,174" "1,077,786" "802,627" "168,377" ...
##  $ Serious,Critical : chr  "5,115" "18,147" "15,176" "10,013" ...
##  $ Tot횴Cases/1M pop: chr  "" "" "" "" ...
##  $ Deaths/1M pop    : chr  "" "" "" "" ...
##  $ TotalTests       : chr  "" "" "" "" ...
##  $ Tests/1M pop     : chr  "" "" "" "" ...
##  $ Continent        : chr  "Asia" "North America" "Europe" "South America"
...
```

```r
head(covid_yesterday,20)
```

```
##    Country,Other TotalCases NewCases TotalDeaths NewDeaths TotalRecovered
## 1           Asia    634,831  +16,327      21,501      +296        349,156
## 2  North America  1,442,750  +33,432      87,090    +2,129        277,874
## 3         Europe  1,581,256  +27,556     150,514    +2,053        628,115
## 4  South America    282,175  +16,063      14,577      +994         99,221
## 5         Africa     59,050   +3,731       2,161       +78         19,839
## 6        Oceania      8,508      +19         118                     7,522
## 7                       721                  15                       645
## 8          World  4,009,291  +97,128     275,976    +5,550      1,382,372
## 9          China     82,886       +1       4,633                   77,993
## 10           USA  1,321,785  +29,162      78,615    +1,687        223,603
## 11         Spain    260,117   +3,262      26,299      +229        168,408
## 12         Italy    217,185   +1,327      30,201      +243         99,023
## 13            UK    211,364   +4,649      31,241      +626            N/A
## 14        Russia    187,859  +10,699       1,723       +98         26,608
## 15        France    176,079   +1,288      26,230      +243         55,782
## 16       Germany    170,588   +1,158       7,510      +118        141,700
```

```
## 17        Brazil    145,892  +10,199       9,992      +804         59,297
## 18        Turkey    135,569   +1,848       3,689       +48         86,396
## 19          Iran    104,691   +1,556       6,541       +55         83,837
## 20        Canada     66,434   +1,512       4,569      +161         30,406
##    ActiveCases Serious,Critical Tot혽Cases/1M pop Deaths/1M pop TotalTests
## 1      264,174            5,115
## 2    1,077,786           18,147
## 3      802,627           15,176
## 4      168,377           10,013
## 5       37,050              223
## 6          868               25
## 7           61                4
## 8    2,350,943           48,703               514          35.4
## 9          260               18                58             3
## 10   1,019,567           16,978             3,993           238  8,636,435
## 11      65,410            2,075             5,563           562  1,932,455
## 12      87,961            1,168             3,592           500  2,445,063
## 13     179,779            1,559             3,114           460  1,631,561
## 14     159,528            2,300             1,287            12  4,980,000
## 15      94,067            2,868             2,698           402  1,384,633
## 16      21,378            1,712             2,036            90  2,755,770
## 17      76,603            8,318               686            47    339,552
## 18      45,484            1,219             1,607            44  1,298,806
## 19      14,313            2,711             1,246            78    558,899
## 20      31,459              502             1,760           121  1,032,088
##    Tests/1M pop         Continent
## 1                           Asia
## 2                  North America
## 3                         Europe
## 4                  South America
## 5                         Africa
## 6                Australia/Oceania
## 7
## 8                            All
## 9                           Asia
## 10       26,092     North America
## 11       41,332            Europe
## 12       40,440            Europe
## 13       24,034            Europe
## 14       34,125            Europe
## 15       21,213            Europe
## 16       32,891            Europe
## 17        1,597     South America
## 18       15,400              Asia
## 19        6,654              Asia
## 20       27,346     North America
```

해결

```r
#해결과정

covid_yesterdayL<-
sapply(covid_yesterday,function(x){gsub(",","",x)},simplify=F) #gsub으로 ","를 ""로 치환 , simplify를 F로 줌으로써 리스트로 반환
covid_yesterdayL[c(2:9)] <-
sapply(covid_yesterdayL[c(2:9)],as.numeric,simplify=F) #리스트 [2~9] 까지 numeric으로 변환하여 다시 리스트로 반환

## Warning in lapply(X = X, FUN = FUN, ...): 강제형변환에 의해 생성된 NA 입니다

covid_yesterday <- data.frame(covid_yesterdayL,stringsAsFactors = F) #리스트로 반환한 sapply 출력값을 데이터프레임으로 재구성

str(covid_yesterday)

## 'data.frame':    222 obs. of  13 variables:
##  $ Country.Other    : chr  "Asia" "North America" "Europe" "South America"
...
##  $ TotalCases       : num  634831 1442750 1581256 282175 59050 ...
##  $ NewCases         : num  16327 33432 27556 16063 3731 ...
##  $ TotalDeaths      : num  21501 87090 150514 14577 2161 ...
##  $ NewDeaths        : num  296 2129 2053 994 78 ...
##  $ TotalRecovered   : num  349156 277874 628115 99221 19839 ...
##  $ ActiveCases      : num  264174 1077786 802627 168377 37050 ...
##  $ Serious.Critical : num  5115 18147 15176 10013 223 ...
##  $ Tot횢Cases.1M.pop: num  NA NA NA NA NA ...
##  $ Deaths.1M.pop    : chr  "" "" "" "" ...
##  $ TotalTests       : chr  "" "" "" "" ...
##  $ Tests.1M.pop     : chr  "" "" "" "" ...
##  $ Continent        : chr  "Asia" "North America" "Europe" "South America"
...
```