PR7

PR7\_201821479\_황혜린

2018년 11월 1일

# apply함수

* 복수의 데이터에 함수를 일괄 적용할 때 사용
* apply,lapply,sapply,tapply,mapply 등이 있음
* 각 apply함수는 입력받는 데이터의 형태와 출력하는 데이터의 형태에 따라 다르게 적용함

## 1)apply 함수

* 형식 : apply(data,margin(1또는2),function)
* margin 인수를 1또는 2로 사용하며 1은 행,2는 열을 적용
* 행이나 열의 합계, 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)

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

apply(mtcars[,1:3], 2, FUN = mean)

## mpg cyl disp   
## 20.09062 6.18750 230.72188

## 2)lapply(list apply)

* 형식 : apply(data,function)
* 리스트형의 데이터를 받아 리스트로 결과를 반환
* 데이터프레임의 각 열은 리스트로 구성되어 있음

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

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

## 3)sapply(simple apply)

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

x = 1:5 ; y = 11:14  
z = list(x,y)  
sapply(x,function(x){x+1})

## [1] 2 3 4 5 6

sapply(z,function(x){x+1})

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

sapply(mtcars[1:3,],function(x){x+1})

## 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)

## $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

## 4)tapply(table apply)

* 그룹으로 묶은 후 함수를 적용, 적용값을 벡터나 행렬로 반환

patient <- read.table("simple\_data.txt", header = TRUE)  
factor(patient$type)

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

tapply(patient$type,patient$type,length)

## Type1 Type2   
## 3 3

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

## Type1 Type2   
## 35 37

## 5)mapply(multi simple apply)

* 여러개의 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)

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

mapply(sum, m1$a, m1$b, m2$c, m2$d)

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

## 6)aggregating

* 예제데이터

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

## 6.1)mean, sd 통계함수

attach(seg.df)  
mean(income[Segment == "Moving up"])

## [1] 53090.97

mean(income[Segment == "Moving up" & subscribe == "subNo"])

## [1] 53633.73

## 6.2)apply 함수

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

## 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)

## age income kids   
## 36.331144 53090.965253 1.914286

## 6.3)table 함수

table(kids)

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

table(ownHome, subscribe)

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

table(Segment, kids, subscribe)

## , , 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

## 6.4)함수

* 사용방식 : by(목표변수, 기준변수, 함수)
* by함수는 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

## 6.5)aggregate 함수

* 사용방식 : aggregate(목표변수,기준변수,함수)
* 결과값을 데이터프레임으로 출력해주는 것이 가장 큰 장점임
* 기준변수가 list입력되어야한다

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

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

## 6.6)cut함수

* cut함수는 연속형 변수를 특정 구간으로 구분하여 명목형 변수로 반환한다
* cut(데이터,breaks=구간수, lable=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

## 6.7)grep 함수

grep("ap", c("apple", "Apple", "apple2", "bbapple"))

## [1] 1 3 4

grep("ap", c("apple", "Apple", "apple2", "bbapple"), value = TRUE)

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

grep("[1-3]", c("apple1", "apple2", "apple3", "apple4", "Apple1"))

## [1] 1 2 3 5

grepl("ap", c("apple", "Apple", "apple2", "bbapple"))

## [1] TRUE FALSE TRUE TRUE

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  
## [18] 26 33 37 39 40 41 43 47 50 51 52 53 55 57 68 72 73  
## [35] 75 79 80 81 83 84 87 90 91 92 95 96 97 99 108 118 120  
## [52] 122 125 130 139 144 145 150 151 152 153 155 156 157 159 160 161 162  
## [69] 163 164 165 166 167 168 169 170 171 175 176 177 178 179 180 181 182  
## [86] 183 184 185 187 188 189 190 192 194 195 196 197 198 199 200 201 203  
## [103] 204 207 208 210 211 213 214 215 216 220 221 223 224 225 226 228 231  
## [120] 232 236 238 240 241 247 252 258 261 264 265 269 271 273 274 276 279  
## [137] 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

## 6.8)gsub 함수

* 현재데이터의 Segment컬럼에 한 칸 띄어쓰기를 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

## 6.9)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

# package

* R에서 기본적으로 제공하는 함수 외에 다른 프로그래머들이 필요하다고 생각해서 만들어놓은 함수들의 모음 -이것을 보통 library 또는 API라고 칭함
* R에서는 보통 CRAN에서 R재단이 인정한 패키지들을 공유
* 비공식적이지만 개인이 만들어서 공유하는 패키지들이 있는데 보통 github에 업로드 되어 있음

## 설치

* 설치 명령어:install.packages()
* 다음 시간 사용할 크롤링 관련 패키지 설치

#install.packages("rvest")  
#install.packages("httr")  
#install.packages("Rselenium")

## 불러오기

* library(package 이름): 설치되어 있는 패키지를 불러옴
* require(package 이름): 불러오려시 시도를 하고 logical한 return값을 반환

x <- library(rvest)

## Loading required package: xml2

x

## [1] "rvest" "xml2" "stats" "graphics" "grDevices" "utils"   
## [7] "datasets" "methods" "base"

x <- require(rvest)  
x

## [1] TRUE

## 패키지 업데이트

* 패키지 개발자가 자신이 만든 패키지의 기능을 보완하거나, R버전이 업데이트되어서 호환이 되게 수정하거나 오류사항을 수정
* 명령어는 update.package(패키지이름)
* R studio를 쓰시는 경우에는 우측 하단에 pakages라는 탭이 보이는데, 해당 탭 바로 밑에 update라는 항목이 보일겁니다
* 이걸 누르고 원하는 패키지 또는 전체 패키지를 선택하고 업데이트 눌러주시면 됩니다

#update.packages("rvest")

## 비공식 패키지 설치

#install.packages("devtools")  
library(devtools)  
#install\_github("패키지")

## 불러온 패키지 사용중지

#detach("package:ggplot2",unload = T)

## 연습문제1

setwd("C:/Users/hyere/Desktop/PR")  
pubg <- read.csv("PUBG\_Player\_Statistics.csv")   
  
pubg[,5:6] <- sapply(pubg[,5:6], function(x){gsub(",","",x)}) #sapply안에 gsub함수를 사용해서 ,를 없앱니다   
pubg[,6] <- sapply(pubg[,6], function(x){gsub(" ","",x)})  
#같은 방법으로 공백을 없앱니다   
head(pubg)

## player\_name solo\_RoundsPlayed solo\_AvgDamage solo\_AvgKills  
## 1 BreakNeck 17 255.36 2.59  
## 2 Blackwalk 33 393.04 3.61  
## 3 mercedes\_benz 5 329.76 3.60  
## 4 DORA 8 751.95 7.00  
## 5 n2tstar 6 637.48 7.00  
## 6 coldoxygen 16 411.95 3.94  
## solo\_AvgSurvivalTime solo\_MoveDistance play\_region  
## 1 1263 56462 2  
## 2 1221 165707 3  
## 3 1133 14024 2  
## 4 1678 47895 1  
## 5 1640 39162 1  
## 6 1375 91395 4

## 연습문제2

#PR4\_문제3번  
#변수에 국가별 평균 데미지와 킬수를 각각 한번에 할당할 수 있는 tapply를 적용하면 쉽게 풀 수 있습니다.  
  
setwd("C:/Users/hyere/Desktop/PR")  
pubg1 <- read.csv("PUBG\_Player\_Statistics\_fix.csv")  
pubg1[pubg1$play\_region == "1", "Country"] = "KOREA"  
pubg1[pubg1$play\_region == "2", "Country"] = "CHINA"  
pubg1[pubg1$play\_region == "3", "Country"] = "JAPAN"  
pubg1[pubg1$play\_region == "4", "Country"] = "USA"  
pubg1[pubg1$play\_region == "5", "Country"] = "OTHERS"  
  
Avgdmg <- tapply(pubg1$solo\_AvgDamage, pubg1$Country, mean)  
#Avgdmg 라는 변수에 국가별 평균 데미지를 할당합니다  
  
Avgkills <- tapply(pubg1$solo\_AvgKills, pubg1$Country, mean)  
#Avgkills라는 변수에 국가별 평균 킬수를 할당합니다  
  
ranking <- Avgkills \* 0.4 + Avgdmg \* 0.6  
#Avgdmg, Avgkills에 각각 가중치를 두어 ranking이란 변수에 할당합니다  
  
sort(ranking, decreasing=T)

## KOREA USA OTHERS CHINA JAPAN   
## 206.2068 184.3741 177.6293 173.9978 173.6385

#ranking이 높은 순서대로 출력합니다