[R] R에서 사용되는 Data.frame 과 Factor 에 사용되는 다양한 함수 — 나무늘보의 개발 블로그

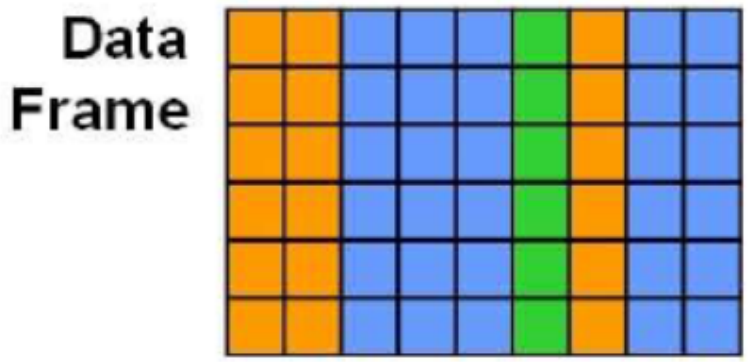
노트북:	blog
만든 날짜:	2020-10-02 오후 10:04
URL:	https://continuous-development.tistory.com/36?category=793392

R

[R] R에서 사용되는 Data.frame 과 Factor 에 사용되는 다양한 함수

2020. 7. 24. 02:03 수정 삭제 공개

#데이터 프레임(data.frame)



배열(array) 3차원 벡터로서 동일 타입의 데이터만 저장 가능하다.
2차원 구조로서 열 단위로 서로 다른 타입의 데이터들로 구성 가능하다.
또한 모든 열의 데이터 개수(행의 개수)는 동일해야 한다.

데이터 프레임 변환 :rbind(df, 백터), cbind(df, 벡터)
데이터 프레임의 구조 확인 :str(df)
인덱싱 : [행의인덱싱, 열의 인덱싱], [열의 인덱싱], df$칼럼 이름, [[열 인덱싱]]

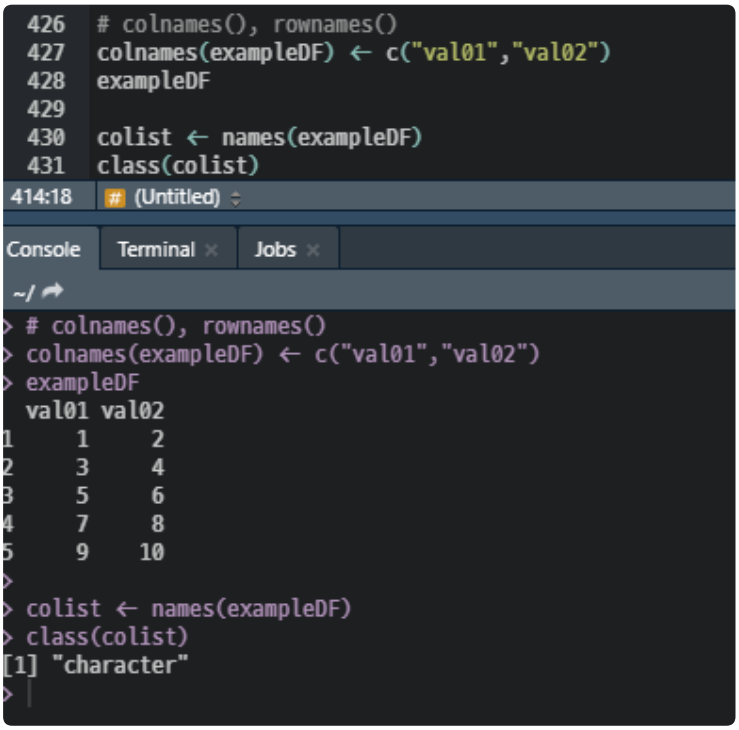
subset (df, select=컬럼명들, subset=(조건))

data.frame(value1,value2) - 데이터 프레임 생성

```r
##########
#data.frame
#행렬과 비슷하다
#다만 , 다양한 변수 (관측값이 숫자, 문자 , 범주 등) 으로 표현된다.
#각 열에 대한 접근은 $이용하여 접근할 수 있다.
# 인덱스를 활용하는 방법도 있다.

x <- c(1,3,5,7,9)
y <- c(2,4,6,8,10)

exampleDF <- data.frame(x,y)
exampleDF
str(exampleDF)

exampleDF[1,]
exampleDF[ ,c("x") ]
class(exampleDF[ ,c("x") ])
exampleDF
```

```
3 5  6
4 7  8
5 9 10
> str(exampleDF)
'data.frame':   5 obs. of  2 variables:
 $ x: num  1 3 5 7 9
 $ y: num  2 4 6 8 10
>
> exampleDF[1,]
  x y
1 1 2
> exampleDF[ ,c("x") ]
[1] 1 3 5 7 9
> class(exampleDF[ ,c("x") ])
[1] "numeric"
> exampleDF
  x  y
1 1  2
2 3  4
3 5  6
4 7  8
5 9 10
```

#colnames()/rownames() - 행 / 열 이름 변경


426 # colnames(), rownames()
427 colnames(exampleDF) ← c("val01","val02")
428 exampleDF
429
430 colist ← names(exampleDF)
431 class(colist)
414:18 (Untitled)
Console
Terminal
Jobs
~/
> # colnames(), rownames()
> colnames(exampleDF) ← c("val01","val02")
> exampleDF
val01 val02
1 1 2
2 3 4
3 5 6
4 7 8
5 9 10
>
> colist ← names(exampleDF)
> class(colist)
[1] "character"
>

예제

```r
# 문자열 벡터 , 숫자형 벡터 , 문자열 벡터
# data.frame

stuName ← c("조동균","한소연","박수진","최가은")
subject.eng ← c(100,100,100,70)
subject.math ← c(80,75,100,100)
subject.kor ← c(100,100,100,70)
score.grade ← c("A","B","A","c")

student ← data.frame(stuName,subject.eng,subject.math,subject.kor,score.grade)
student
colnames(student) ← c("이름","영어","수학","국어","성적")
student

```

422:21 (Untitled)

Console Terminal Jobs

~/

```
> stuName ← c("조동균","한소연","박수진","최가은")
> subject.eng ← c(100,100,100,70)
> subject.math ← c(80,75,100,100)
> subject.kor ← c(100,100,100,70)
> score.grade ← c("A","B","A","c")
>
> student ← data.frame(stuName,subject.eng,subject.math,subject.kor,score.grade)
> student
  stuName subject.eng subject.math subject.kor score.grade
1  조동균         100           80         100           A
2  한소연         100           75         100           B
3  박수진         100          100         100           A
4  최가은          70          100          70           c
> colnames(student) ← c("이름","영어","수학","국어","성적")
> student
    이름 영어 수학 국어 성적
1 조동균  100   80  100    A
2 한소연  100   75  100    B
3 박수진  100  100  100    A
4 최가은   70  100   70    c
>
```

#nrow(value) - 행의 갯수 출력

```
448 # nrow() - 행의 갯수
449 nrow(student)
450 ncol(student)
451
```

447:1 (Untitled)

Console Terminal Jobs

```
~/
> # nrow() - 행의 갯수
> nrow(student)
[1] 4
> ncol(student)
[1] 5
```

```
453 # 열 추가
454 # 학생의 학번의 열을
455 학번<-c("056541","995254","205477","182354")
456
457 scondschDF <- cbind(student ,학번)
458 scondschDF
459
460
```

459:1 (Untitled)

Console Terminal Jobs

```
~/
> # 열 추가
> # 학생의 학번의 열을
> 학번<-c("056541","995254","205477","182354")
> scondschDF <- cbind(student ,학번)
> scondschDF
    이름 영어 수학 국어 성적   학번
1 조동균  100   80  100    A 056541
2 한소연  100   75  100    B 995254
3 박수진  100  100  100    A 205477
4 최가은   70  100   70    c 182354
```

#cbind(value1, value2) - 열 추가 함수

#rbind(value1, value2) - 행 추가 함수

```r

# nrow() - 행의 갯수
nrow(student)
ncol(student)


# 열 추가
# 학생의 학번의 열을 추가
학번←c("056541","995254","205477","182354")

scondschDF ← cbind(student ,학번)
scondschDF
str(scondschDF)

#더미 데이터를 이용해서 행 추가
newStudent←c("라이언","100","100","100","A+","777777")

thridschDF ← rbind(scondschDF, newStudent)
thridschDF
thridschDF$이름
thridschDF$이름[5]
thridschDF[[1]][5]
thridschDF[1:4]

```

```
3 박수진  100  100  100    A 205477
4 최가은   70  100   70    c 182354
> str(scondschDF)
'data.frame':   4 obs. of  6 variables:
 $ 이름: chr  "조동균" "한소연" "박수진" "최가은"
 $ 영어: num  100 100 100 70
 $ 수학: num  80 75 100 100
 $ 국어: num  100 100 100 70
 $ 성적: chr  "A" "B" "A" "c"
 $ 학번: chr  "056541" "995254" "205477" "182354"
>
> #더미 데이터를 이용해서 행 추가
> newStudent←c("라이언","100","100","100","A+","777777")
>
> thridschDF ← rbind(scondschDF, newStudent)
> thridschDF
    이름 영어 수학 국어 성적   학번
1 조동균  100   80  100    A 056541
2 한소연  100   75  100    B 995254
3 박수진  100  100  100    A 205477
4 최가은   70  100   70    c 182354
5 라이언  100  100  100   A+ 777777
>
```

#with(data, expression) - 데이터 프레임 또는 리스트 내 필드를 필드 이름만으로 접근할 수 있게 해주는 함수

#within(data, expression) - with함수의 기능에 더해서 데이터를 수정하는 기능까지 제공

```r
# with(data, expression) - 여러가지 값을 한번에 확인  ,within(data, expression) - 값을 확인해서 다시 반영하는 용도

data(iris)
iris

mean(iris$Sepal.Length)
mean(iris$Sepal.Width )

with( # $를 쓰지 않고 바로 사용한다 왜냐하면 with안에 iris가 들어가 있어서 이미 접근한 상태이다.
  iris,
  {
    print(mean(Sepal.Length))
    print(mean(Sepal.Width))
  }
)


x <- data.frame(val=c(1,2,3,4,NA,5,NA))
x

x<-within(
  x,
  {
    val <- ifelse(is.na(val),mean(val,na.rm=T),val )
  }
)
x

```

495:29 # (Untitled)

Console Terminal Jobs

~/

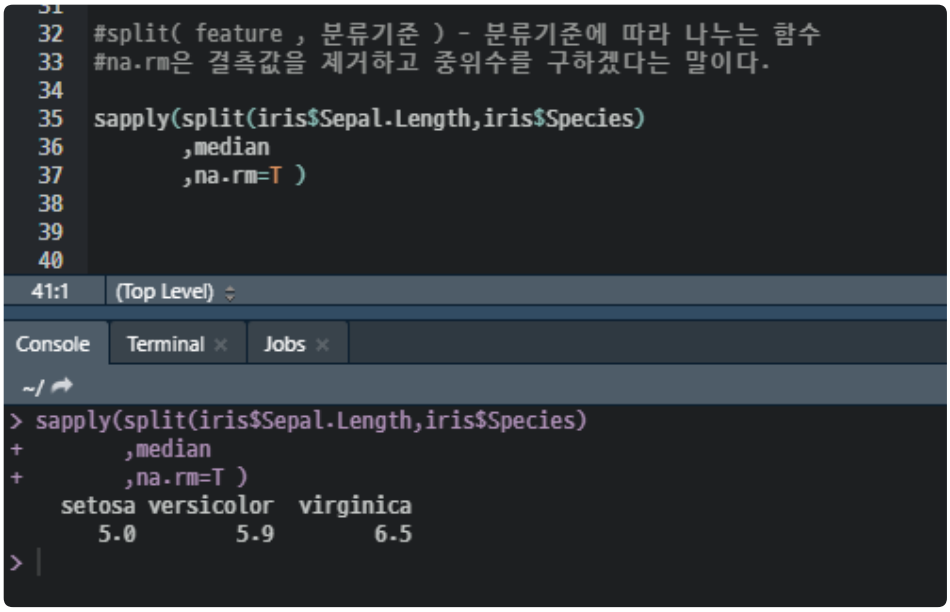
```
> mean(iris$Sepal.Length)
[1] 5.843333
> mean(iris$Sepal.Width )
[1] 3.057333
> with( # $를 쓰지 않고 바로 사용한다 왜냐하면 with안에 iris가 들어가 있어서 이미 접근한 상태이다.
+   iris,
+   {
+     print(mean(Sepal.Length))
+     print(mean(Sepal.Width))
+   }
+ )
[1] 5.843333
[1] 3.057333
> x <- data.frame(val=c(1,2,3,4,NA,5,NA))
> x
  val
1   1
2   2
3   3
4   4
5  NA
6   5
7  NA
```

```
> x<-within(
+   x,
+   {
+     val <- ifelse(is.na(val),mean(val,na.rm=T),val )
+   }
+ )
> x
  val
1   1
2   2
3   3
4   4
5   3
6   5
7   3
```

#within을 통해 결측치값을 평균으로 바꿔주는 구문

```r

irisSlMedian← sapply(split(iris$Sepal.Length,iris$Species)
                     ,median
                     ,na.rm=T )

class(irisSlMedian)
irisSlMedian[iris$Species]

iris←within(
  iris,
  {
    Sepal.Length ← ifelse(is.na(Sepal.Length),irisSlMedian[iris$Species],Sepal.Length )
  }
)
iris


iris2←within(
  iris,
  {
    Sepal.Length ← ifelse(is.na(Sepal.Length), sapply(split(iris$Sepal.Length,iris$Species)
                                                      ,median
                                                      ,na.rm=T ) ,Sepal.Length )
  }
)
iris2

```

#split(feature,분류기준,[중위값],[결측값을 중앙값으로 변환]) - 분류기준에 따라 데이터를 나누어 반환하는 함수



#subset(value, 조건, [select]) - 설정하는 조건에 맞는 벡터, 매트릭스 혹은 데이터 프레임을 반환하는 함수

```r
#subset() - 설정하는 조건에 맞는 벡터, 매트릭스 혹은 데이터 프레임을 반환
?subset

x ← 1:5
y ← 6:10

?letters

z ← letters[1:5]
z

exampleDF ← data.frame(x,y,z)
str(exampleDF)

# x의 값이 3이상인 결과를 새로운데이터 프레임으로 만들어보자
subDF01← subset(exampleDF , x ⩾ 3)
subDF01

# y의 값이 8이하인 결과를 새로운 데이터 프레임으로 만들어 보자
subDF02← subset(exampleDF , y ⩽ 8)
subDF02

# x의 값이 2이상이고 y의 값이 8이하인  결과를 새로운 데이터 프레임으로 만들어 보자
subDF03← subset(exampleDF,  x ⩾ 2  &y ⩽ 8)
subDF03

#subset()에서 select 으로 원하는 컬럼 선택 가능
subDF2 ← subset(exampleDF, x⩾3, select=c(x,y))
subDF2
```

#select 조건 - 원하는 컬럼만 가져온다.

```r
#subset()에서 select 으로 원하는 컬럼 선택 가능
subDF2 ← subset(exampleDF, x⩾3, select=c(x,y))
subDF2

iris
str(iris)
names(iris)

# Petal.Length 평균을 구하라
mean(iris$Petal.Length)
iris_df ← subset(iris, Petal.Length ⩾ mean(Petal.Length) , select=c(Sepal.Length,Petal.Length,Species ))
iris_df

str(iris_df)

```

105:1 (Top Level)

Console Terminal Jobs

~/

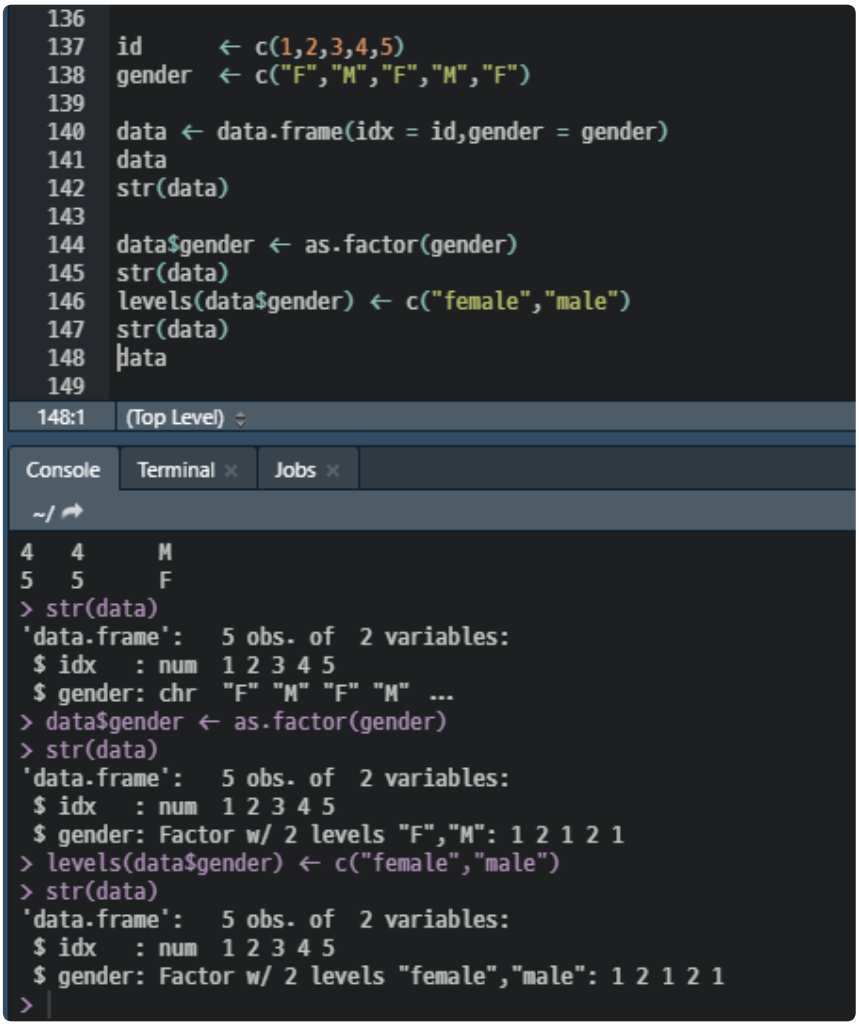
```
140          6.9          5.4  virginica
141          6.7          5.6  virginica
142          6.9          5.1  virginica
143          5.8          5.1  virginica
144          6.8          5.9  virginica
145          6.7          5.7  virginica
146          6.7          5.2  virginica
147          6.3          5.0  virginica
148          6.5          5.2  virginica
149          6.2          5.4  virginica
150          5.9          5.1  virginica
> str(iris_df)
'data.frame':   93 obs. of  3 variables:
 $ Sepal.Length: num  7 6.4 6.9 5.5 6.5 5.7 6.3 6.6 5.2 5.9 ...
 $ Petal.Length: num  4.7 4.5 4.9 4 4.6 4.5 4.7 4.6 3.9 4.2 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 2 2 2 2 2 2 2 2 2 2 ...
```

#Factor - 범주형 변수를 나타낸다.

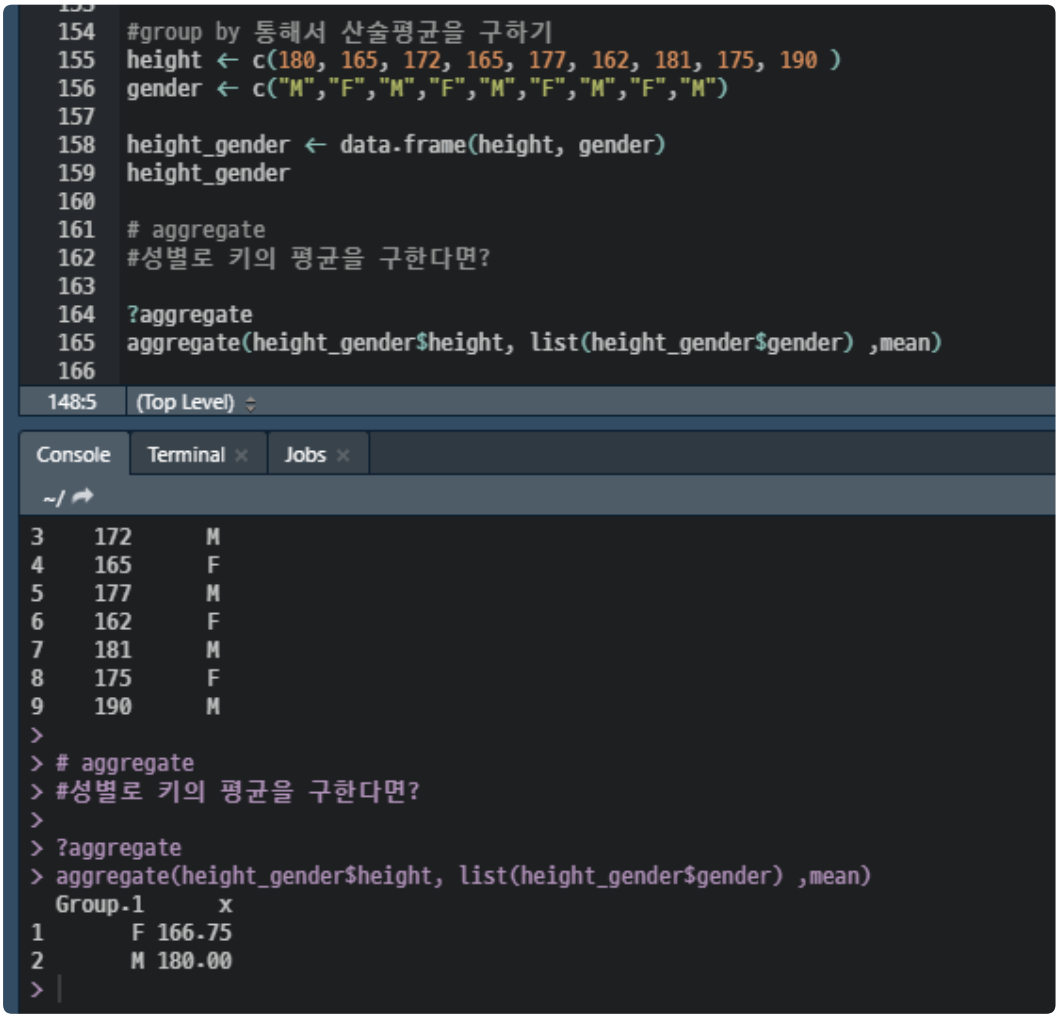
as.factor / factor 로 생성할 수 있다.

```
107  ## Factor?
108  # 범주형 변수
109  ?factor
110  gender ← factor("m",c("m","f"))
111  gender
112
113  #레벨의 갯수를 확인할 수 있다.
114  nlevels(gender)
115  #구성되어 있는 레벨을 출력해준다.
116  levels(gender)
117  levels(gender)[1]
118
119  blood.type ← factor(c("A","A","AB","O","B"))
120  blood.type[6] ← "D" #범주에 없는 값을 넣지 못한다.
121
122  is.factor(blood.type) #factor형인지 확인하는 함수
123
124  lettersVec ← c("a","b","b","c","a","c","a","a","a")
125  lettersVec
126  class(lettersVec)
127  lettersVec.fac ← as.factor(lettersVec) # factor 형으로 변환
128  lettersVec.fac
129
130  lettersVec.fac ← factor(lettersVec,  # factor를 생성하는데 이때는 라벨이랑 레벨이 가능하다
131                          levels = c("a","b","c"),
132                          labels = c("best","middle","low"))
133
134  lettersVec.fac
135
101:84  (Top Level)

Console  Terminal  Jobs
~/
> lettersVec.fac ← as.factor(lettersVec) # factor 형으로 변환
> lettersVec.fac
[1] a b b c a c a a a
Levels: a b c
> lettersVec.fac ← factor(lettersVec,  # factor를 생성하는데 이때는 라벨이랑 레벨이 가능하다
+                         levels = c("a","b","c"),
+                         labels = c("best","middle","low"))
> lettersVec.fac
[1] best   middle middle low    best   low    best   best   best
Levels: best middle low
>
```

factor로 형변환


136
137 id <- c(1,2,3,4,5)
138 gender <- c("F","M","F","M","F")
139
140 data <- data.frame(idx = id,gender = gender)
141 data
142 str(data)
143
144 data$gender <- as.factor(gender)
145 str(data)
146 levels(data$gender) <- c("female","male")
147 str(data)
148 data
149
148:1 (Top Level)
Console Terminal Jobs
~/
4 4 M
5 5 F
> str(data)
'data.frame': 5 obs. of 2 variables:
$ idx : num 1 2 3 4 5
$ gender: chr "F" "M" "F" "M" ...
> data$gender <- as.factor(gender)
> str(data)
'data.frame': 5 obs. of 2 variables:
$ idx : num 1 2 3 4 5
$ gender: Factor w/ 2 levels "F","M": 1 2 1 2 1
> levels(data$gender) <- c("female","male")
> str(data)
'data.frame': 5 obs. of 2 variables:
$ idx : num 1 2 3 4 5
$ gender: Factor w/ 2 levels "female","male": 1 2 1 2 1
>

#산술평균 구하기



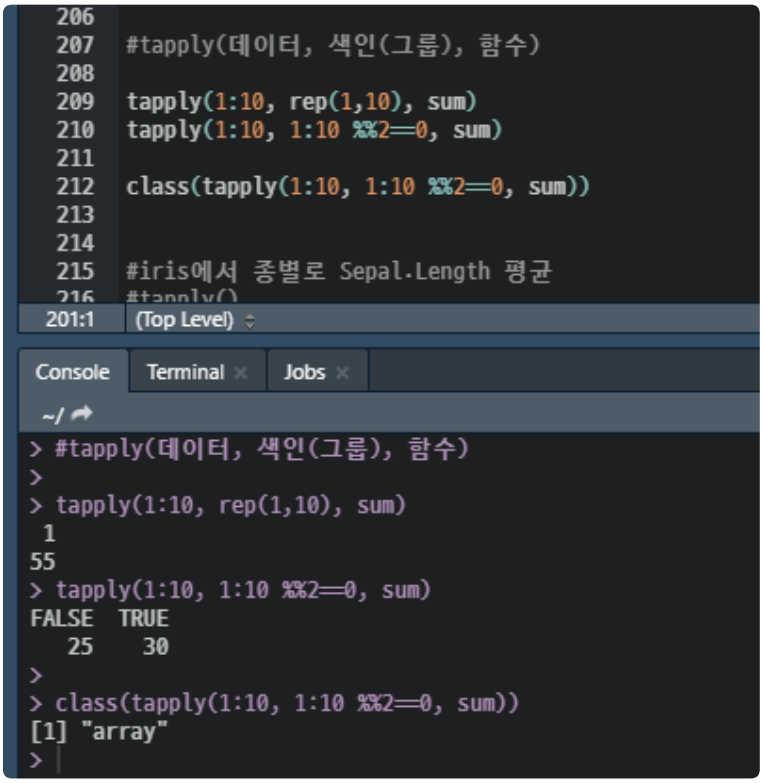
```
154  #group by 통해서 산술평균을 구하기
155  height ← c(180, 165, 172, 165, 177, 162, 181, 175, 190 )
156  gender ← c("M","F","M","F","M","F","M","F","M")
157
158  height_gender ← data.frame(height, gender)
159  height_gender
160
161  # aggregate
162  #성별로 키의 평균을 구한다면?
163
164  ?aggregate
165  aggregate(height_gender$height, list(height_gender$gender) ,mean)
166
148:5  (Top Level)

Console  Terminal  Jobs
~/

3    172      M
4    165      F
5    177      M
6    162      F
7    181      M
8    175      F
9    190      M
>
> # aggregate
> #성별로 키의 평균을 구한다면?
>
> ?aggregate
> aggregate(height_gender$height, list(height_gender$gender) ,mean)
  Group.1      x
1       F 166.75
2       M 180.00
>
```

 # aggregate(x, by, fun) - by를 기준으로 fun 함수를 사용해 x를 구한다.

```r
# aggregate(x,by,fun) - by를 기준으로 fun 함수를 사용해 x 를 구한한다.

  #성별로 키의 평균을 구한다면?
?aggregate
aggregate(height_gender$height, list(height_gender$gender) ,mean)


#예제 데이터 mtcars
mtcars
str(mtcars)
head(mtcars)
mtcars[,6]


# cyl 컬럼을 기준으로 나머지 컬럼의 평균 값 구하기
aggregate(mtcars, list(cylStandard = mtcars$cyl) ,mean)


# disp 컬럼이 120 이상인 조건 추가
aggregate(mtcars, list(cylStandard = mtcars$cyl , dispHigh = mtcars['disp'] > 120) ,mean)

# cyl 컬럼을 기준으로 wt 컬럼의 평균만 구하기

aggregate(mtcars$wt, list(cyl = mtcars$cyl) ,mean)
aggregate(mtcars[,6], list(cyl = mtcars$cyl) ,mean)
aggregate(wt~cyl, data = mtcars ,mean) #formula


# carb , gear 컬럼 두가지를 기준으로 wt 구하기
aggregate(wt ~ carb + gear, data = mtcars, mean)


# gear 기준으로 disp, wt 평균 구하기(컬럼을 cbind로 합친다.)
aggregate( cbind(disp,wt) ~ gear, data = mtcars, mean)

```

```
> aggregate(height_gender$height, list(height_gender$gender) ,mean)
  Group.1      x
1       F 166.75
2       M 180.00
> # cyl 컬럼을 기준으로 나머지 컬럼의 평균 값 구하기
> aggregate(mtcars, list(cylStandard = mtcars$cyl) ,mean)
  cylStandard      mpg cyl     disp        hp     drat       wt     qsec        vs        am     gear     carb
1           4 26.66364   4 105.1364  82.63636 4.070909 2.285727 19.13727 0.9090909 0.7272727 4.090909 1.545455
2           6 19.74286   6 183.3143 122.28571 3.585714 3.117143 17.97714 0.5714286 0.4285714 3.857143 3.428571
3           8 15.10000   8 353.1000 209.21429 3.229286 3.999214 16.77214 0.0000000 0.1428571 3.285714 3.500000
> # disp 컬럼이 120 이상인 조건 추가
> aggregate(mtcars, list(cylStandard = mtcars$cyl , dispHigh = mtcars[,'disp'] > 120) ,mean)
  cylStandard dispHigh      mpg cyl     disp        hp     drat       wt     qsec        vs        am     gear     carb
1           4    FALSE 29.53333   4  84.6000  75.83333 4.155000 1.903000 18.71667 1.0000000 1.0000000 4.166667 1.333333
2           4     TRUE 23.22000   4 129.7800  90.80000 3.970000 2.745000 19.64200 0.8000000 0.4000000 4.000000 1.800000
3           6     TRUE 19.74286   6 183.3143 122.28571 3.585714 3.117143 17.97714 0.5714286 0.4285714 3.857143 3.428571
4           8     TRUE 15.10000   8 353.1000 209.21429 3.229286 3.999214 16.77214 0.0000000 0.1428571 3.285714 3.500000
> aggregate(wt~cyl, data = mtcars ,mean) #formula
  cyl       wt
1   4 2.285727
2   6 3.117143
3   8 3.999214
> # carb , gear 컬럼 두가지를 기준으로 wt 구하기
> aggregate(wt ~ carb + gear, data = mtcars, mean)
   carb gear       wt
1     1    3 3.046667
2     2    3 3.560000
3     3    3 3.860000
4     4    3 4.685800
5     1    4 2.072500
6     2    4 2.683750
7     4    4 3.093750
8     2    5 1.826500
9     4    5 3.170000
10    6    5 2.770000
11    8    5 3.570000
> # carb, gear 컬럼 기준으로 disp, wt 평균
> aggregate( cbind(disp,wt) ~carb + gear, data = mtcars, mean)
   carb gear     disp       wt
1     1    3 201.0333 3.046667
2     2    3 345.5000 3.560000
3     3    3 275.8000 3.860000
4     4    3 416.4000 4.685800
5     1    4  84.2000 2.072500
6     2    4 121.0500 2.683750
7     4    4 163.8000 3.093750
8     2    5 107.7000 1.826500
9     4    5 351.0000 3.170000
10    6    5 145.0000 2.770000
11    8    5 301.0000 3.570000
```

#tapply - 데이터를 색인에 따라 그룹을 한 후 함수에 따른 결괏값을 내는 함수



'**R**' 카테고리의 다른 글

[R] R 사용자 정의 함수(FUNCTION)와 데이터 전처리를 위한 기본적인 함수

[R] R로 만드는 제어문 (if, else if, for)과 예제

[R] R에서 사용되는 Data.frame 과 Factor 에 사용되는 다양한 함수

[R] R에 사용되는 배열(array)과 리스트(list)의 개념 및 사용되는 함수

[R] R에 사용되는 행렬(matrix)의 개념 및 사용되는 함수

[R] R에서 사용되는 정규표현식(Regex) 표현 방법과 함수를 통한 사용 예제

aggregate cbind colnames data.frame rbind rownames tapply with Within 데이터프레임

꾸까꾸

혼자 끄적끄적하는 블로그 입니다.