180523

5장

03)분할, 적용, 재조합을 통한 데이터 분석

plyr 패키지

데이터를 분할하고, 분할된 데이터에 특정 함수를 적용한 뒤, 그 결과를 재조합하는 함수들
출력 형태로 데이터 프레임이 가능하다

{adl} {adl_} ply

a 입.출력 배열
d 입.출력 데이터 프레임
l 입.출력 리스트
_ 출력 아무런 출력도 없음

adply()

adply(data, margins, fun)

- 배열을 받아(행/열/행,열) 데이터 프레임으로 반환하는 함수
- 입력 데이터는 배열 뿐만 아니라 정수 색인이 가능한 데이터 셋이면 적용 가능
- apply()와 비슷하나, 행 방향 처리시 서로 다른 데이터 타입의 경우 결과값의 차이
  (apply()는 여러 행일 경우 행렬, 즉 한 가지 타입만 저장하는 결과를 반환)
- 데이터 프레임으로 출력
- 함수의 적용 결과가 마지막 컬럼으로 추가되어 나타남 적용된 값 함께 출력(개별적 내용 O)

```
> adply(iris,
+       1,
+       function(row){row$Sepal.Length >= 5.0 &
+           row$Species == "setosa"})
   Sepal.Length Sepal.Width Petal.Length Petal.Width   Species    V1
1           5.1         3.5          1.4         0.2    setosa    TRUE
2           4.9         3.0          1.4         0.2    setosa   FALSE
3           4.7         3.2          1.3         0.2    setosa   FALSE
4           4.6         3.1          1.5         0.2    setosa   FALSE
5           5.0         3.6          1.4         0.2    setosa    TRUE
6           5.4         3.9          1.7         0.4    setosa    TRUE
7           4.6         3.4          1.4         0.3    setosa   FALSE
```

...

-> 반환 값이 단순한 boolean 값, 그 결과가 임의의 컬럼명 V1에 저장됨.

=> 최종 반환 값이 데이터 프레임인 경우 함수의 반환 값을 **데이터 프레임으로 하는 것이 안전함**

```
> adply(iris,
+       1,
+       function(row){
+         data.frame(
+           sepal_5 = c(row$Sepal.Length >= 5.0 &
+                       row$Species == "setosa")
+         )
+
+       })
    Sepal.Length Sepal.Width Petal.Length Petal.Width   Species sepal_5
1           5.1         3.5          1.4         0.2    setosa    TRUE
2           4.9         3.0          1.4         0.2    setosa   FALSE
3           4.7         3.2          1.3         0.2    setosa   FALSE
4           4.6         3.1          1.5         0.2    setosa   FALSE
5           5.0         3.6          1.4         0.2    setosa    TRUE
6           5.4         3.9          1.7         0.4    setosa    TRUE
```

...

ddply()

ddply(data, .(variables), .fun)

- 데이터프레임(d)를 입력으로 받아 데이터프레임(d)을 내보내는 함수
- apply()는 행 또는 컬럼 단위로 함수를 적용하는 반면,
  ddply()는 variables에 나열한 컬럼에 따라 데이터를 나눈 뒤 함수를 적용 (tapply와 비슷)
- 그룹 연산에 **조건 사용** 가능
- 두 번째 인자에 그룹을 지을 변수를 **.()** 안에 기록, 여러 개 지정 시 콤마로 구분하여 나열
-> 여러개 시 .() 안에 벡터 X, 그냥 나열
- 그룹에 대한 내용만 나옴(개별적 내용 X)

```
> ddply(iris,
+       .(Species),
+       function(sub){
+         data.frame(sepal.width.mean=mean(sub$Sepal.Width))
+       })
     Species sepal.width.mean
1     setosa           3.428
2 versicolor           2.770
3  virginica           2.974
```

조건을 사용한 그룹 연산 가능 - 여러 조건 컴마로 나열 cf. 이전 그룹별 나눔과 다른 적용 (ddply의 장점)

(**tapply** - 벡터 전달 가능, **조건 가지고 있는 벡터 사용하면 가능**

but ddply는 **조건 자체로 삽입 가능)**

여러 변수로 그룹을 짓고자 한다면, .()안에 조건들 또는 컬럼명을 콤마로 구분해서 나열한다.

조건에 따른 그룹핑 가능

```
> ddply(iris,
+       .(Species, Sepal.Length > 5.0),
+       function(sub){
+         data.frame(mean(sub$Sepal.Width))
+       })
     Species Sepal.Length > 5 mean.sub.Sepal.Width.
1     setosa           FALSE            3.203571
2     setosa            TRUE            3.713636
3 versicolor           FALSE            2.233333
4 versicolor            TRUE            2.804255
5  virginica           FALSE            2.500000
6  virginica            TRUE            2.983673
```

- dataframe ~ 사용자함수 : 컬럼이름 쓰기 위해서, 뒤에 네 함수 이용하면 함수 안 써도 됨

baseball 데이터 in plyr 패키지

그룹마다 연산을 쉽게 수행하기 -in plyr 패키지 자주 쓰이는 함수

transform()

-객체를 변환한다.

mutate()

- 데이터프레임에 새로운 컬럼을 추가하거나 기존 컬럼을 수정한다.
- 컬럼명=**값 형태로 지정된 연산이 여러 개 일 때**! **앞의 연산 결과를 뒤의 연산에서 참조가능**
  (vs transform) -> mutate가 더 포괄적이어서 많이 사용
- **기존 데이터프레임이 기준, 매칭시켜서 새로운 컬럼에 할당하는 함수 (마치 join처럼)**

summarise()

- 데이터의 요약 정보를 만드는데 사용하는 함수
- transform(), mutate()와 달리 계산 결과만을 담은 데이터 프레임을 반환

★★subset()★★

subset(x, subset) / subset(x, subset, select)

subset - 행선택, select - 열선택

- 벡터, 행렬, 데이터 프레임의 일부를 반환한다.
- **subset -> 그룹별 비교할 때 사용**
- **ex. 최고 값을 가진 사람을 확인할 수 있도록 함(sub query)**
- **subset 조건 -> T/F 나올 수 있는 형태. 특히 == 사용!**
- **(cf. = 한 개는 컬럼 이름지정)**

---

cf. aggregate 한 후 원 데이터와 merge 한 것 = subset 사용! -- JOIN

4장 문제 중 부서별 최대값을 갖는 직원의 이름, 부서, 연봉, 최대연봉을 함께 출력하여라.

+ in plyr -> ddply(emp, .(DEPTNO), subset, SAL==max(SAL)) #5장

cf inline view in SQL

#merge로 조인해야 하므로(두 데이터 프레임 결합)
데이터프레임으로 만들어야 함 -> aggregate() 사용
#merge는 EQUI 조인만 가능 -> NON EQUI 조인은 merge 말고 사용자 정의 함수 만들거나
/ SQL 구문 통해서 먼저 그 조건에 맞는 데이터를 가져오는 것이 좋음
/ 혹은 library (SQLDF) 불러오기

```
f1<-aggregate(SAL ~ DEPTNO, emp, max)
colnames(f1)<-c("DEPTNO","MAX_SAL")
merge(emp, f1, by.x=c("DEPTNO","SAL"), by.y=c("DEPTNO", "MAX_SAL"))[, c(4,1,2,2)]
```

-*----------

transform()

```
> head(ddply(baseball, .(id), transform, cyear = year - min(year) + 1))
        id year stint team lg   g  ab   r   h X2b X3b hr rbi sb cs bb so ibb hbp sh sf gidp cyear
1 aaronha01 1954     1  ML1 NL 122 468  58 131  27   6 13  69  2  2 28 39  NA   3  6  4   13     1
2 aaronha01 1955     1  ML1 NL 153 602 105 189  37   9 27 106  3  1 49 61   5   3  7  4   20     2
3 aaronha01 1956     1  ML1 NL 153 609 106 200  34  14 26  92  2  4 37 54   6   2  5  7   21     3
4 aaronha01 1957     1  ML1 NL 151 615 118 198  27   6 44 132  1  1 57 58  15   0  0  3   13     4
5 aaronha01 1958     1  ML1 NL 153 601 109 196  34   4 30  95  4  1 59 49  16   1  0  3   21     5
6 aaronha01 1959     1  ML1 NL 154 629 116 223  46   7 39 123  8  0 51 54  17   4  0  9   19     6
```

mutate()

```
> head(ddply(baseball, .(id), mutate, cyear = year - min(year) + 1, log_cyear=log(cyear)))
        id year stint team lg   g  ab   r   h X2b X3b hr rbi sb cs bb so ibb hbp sh sf gidp cyear log_cyear
1 aaronha01 1954     1  ML1 NL 122 468  58 131  27   6 13  69  2  2 28 39  NA   3  6  4   13     1 0.0000000
2 aaronha01 1955     1  ML1 NL 153 602 105 189  37   9 27 106  3  1 49 61   5   3  7  4   20     2 0.6931472
3 aaronha01 1956     1  ML1 NL 153 609 106 200  34  14 26  92  2  4 37 54   6   2  5  7   21     3 1.0986123
4 aaronha01 1957     1  ML1 NL 151 615 118 198  27   6 44 132  1  1 57 58  15   0  0  3   13     4 1.3862944
5 aaronha01 1958     1  ML1 NL 153 601 109 196  34   4 30  95  4  1 59 49  16   1  0  3   21     5 1.6094379
6 aaronha01 1959     1  ML1 NL 154 629 116 223  46   7 39 123  8  0 51 54  17   4  0  9   19     6 1.7917595
```

in mutate 앞의 연산 결과를 뒤에 나오는 연산에서 재사용할 수 있다.

- cf. transform()에서 두 개 이상 컬럼 추가하면 에러 발생

summarise()

```
> head(ddply(baseball, .(id), summarise, minyear=min(year)))
        id minyear
1 aaronha01    1954
2 abernte02    1955
3 adairje01    1958
4 adamsba01    1906
5 adamsbo03    1946
6 adcocjo01    1950
> head(ddply(baseball, .(id), summarise, minyear=min(year), maxyear=max(year)))
        id minyear maxyear
1 aaronha01    1954    1976
2 abernte02    1955    1972
3 adairje01    1958    1970
4 adamsba01    1906    1926
5 adamsbo03    1946    1959
6 adcocjo01    1950    1966
```

- 여러 요약 값 계속 나열 가능

cf. 기존 요약 함수

-기술통계 ~ 평균/최대/최소/Quantile만

-summaryBy ~ 평균만

-Summarise -> 내가 보고 싶은 함수에 관한 것만 요약해서 보기 위해

★★★subset()★★★★

```
> head(ddply(baseball, .(id), subset, g==max(g)))
        id year stint team lg   g  ab   r   h X2b X3b hr rbi sb cs bb so ibb hbp sh sf gidp
1 aaronha01 1963     1  ML1 NL 161 631 121 201  29   4 44 130 31  5 78 94  18   0  0  5   11
2 abernte02 1965     1  CHN NL  84  18   1   3   0   0  0   2  0  0  0  7   0   1  3  0    0
3 adairje01 1965     1  BAL AL 157 582  51 151  26   3  7  66  6  4 35 65   7   2  4  2   26
4 adamsba01 1913     1  PIT NL  43 114  13  33   6   2  0  13  0 NA  1 16  NA   0  3 NA   NA
5 adamsbo03 1952     1  CIN NL 154 637  85 180  25   4  6  48 11  9 49 67  NA   0  8 NA   15
6 adcocjo01 1953     1  ML1 NL 157 590  71 168  33   6 18  80  3  2 42 82  NA   2  6 NA   22
```

- 조건에 맞는 데이터만 추출, 그룹별 비교연산 시 사용(sql은 복잡함) ~ ddply와 많이 활용,

조건은 T-F형태로 되도록

plyr 문제

1. 2013년_프로야구선수_성적.csv 파일을 불러와서 다음을 수행

pb<-read.csv("13probaseball.csv",stringsAsFactors = F)

str(pb) #구조 확인하는 습관 들이기

1) 경기수가 120 경기 이상인 선수만 출력하기

pb[pb$경기>=120,]

subset(pb,경기>=120)

2) 경기수가 120 경기 이상이면서 득점도 80 점 이상인 선수만 출력하기

pb[pb$경기>=120 & pb$득점>=80,]

subset(pb,경기>=120 & 득점>=80)

3) 팀별 출루율의 평균 출력

tapply(pb$출루율, pb$팀, mean)

aggregate(출루율 ~ 팀, pb, mean)

ddply(pb, .(팀), summarise, meanc=mean(출루율))

4) 3)의 결과에서 선수명,포지션,팀 컬럼 데이터만 조회하기

ddply(pb, .(팀), mutate, meanc=mean(출루율))[,c("선수명","포지션","팀")]

★summarise와 mutate의 차이점★

 # mutate 기존 데이터프레임이 기준, 매칭시켜서 새로운 컬럼에 할당하는 함수 (마치 join처럼)

#summarise는 결과값만 나타남

2. emp2.csv 파일을 불러와서 다음을 수행

emp2<-read.csv("emp2.csv", stringsAsFactors = F)

str(emp2)

1) 현재 직급이 없는 직원은 사원으로 치환

(내방법)

emp2[emp2$POSITION=="", "POSITION"] <- "사원"

----------강사님 방법----------.

emp2[str_length(emp2$POSITION) == 0, **"POSITION"**] <- **c("사원")**

과정

cf. is.na / is.null 검사

is.na는 벡터 원소별 T/F 출력 -> 벡터 연산 가능

is.null은 전체 벡터 검사

둘 다 아니면 length로 확인(cf. 오라클 -> length = 0 -> NULL, R은 아님)

length(emp2$POSITION)

벡터 원소 개수 출력

-> 원소별 길이 - stringr 패키지의 str_length() 사용

```
> library(stringr)
> str_length(emp2$POSITION)
[1] 4 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0
```

cf. 이것도 가능

ifelse(emp2$POSITION=="","사원",emp2$POSITION)

2) 직급별 평균연봉을 출력(tapply, ddply)

tapply(emp2$PAY, emp2$POSITION, mean)

aggregate(PAY ~ POSITION, emp2, mean)

ddply(emp2, .(POSITION), summarise, mp=mean(PAY))

3) 직급별 최대연봉자의 사원번호,이름,직급,연봉출력

ddply(emp2, .(POSITION), **subset**, PAY==max(PAY))[,c("EMPNO","NAME","POSITION","PAY")]

subset -> 그룹별 비교할 때 사용

cf merge로도 가능 -- 복잡

f1<-aggregate(PAY ~ POSITION, emp2, max)

colnames(f1)<-c("POSITION","MAX_PAY")

merge(emp2, f1, by.x=c("POSITION","PAY"), by.y=c("POSITION","MAX_PAY"))

cf. mutate, summarize

ddply(emp2, .(POSITION), mutate, mn=max(PAY))

원데이터 + 마지막 함수 적용된 새로운 컬럼 출력

ddply(emp2, .(POSITION), summarize, max(PAY))

그룹+함수값만 출력

#in oracle

```
select *
from emp2
where (position, pay) in (select position, max(pay)
                          from emp2
                          group by position)
```

3. emp.csv 파일을 불러와서 다음을 수행

```
emp<-read.csv("emp.csv",stringsAsFactors = F)
str(emp)
```

1) 부서별 각 사원의 이름을 다음과 같은 형태로 출력

	DEPTNO	ENAME
1	10	CLARK, KING, MILLER
2	20	SMITH, JONES, SCOTT, ADAMS, FORD
3	30	ALLEN, WARD, MARTIN, BLAKE, TURNER, JAMES

과정

#concate와 관련된 string_c(stringr패키지) / paste 함수

벡터형태로 전달 의미 알기

#str_c 함수의 사용법 : 벡터를 전달받아 각 벡터의 원소를 하나의 원소로 결합
(반복문자(sep), 구분자(collapse) 지정 가능)

```
> str_c(v1) # 반복문자, 구분자 없이 결합(자동 공백 구분자)
[1] "A" "B" "C"
> str_c(v1, collapse=",") # 반복문자 X, 구분자 "," 를 사용한 결합
[1] "A,B,C"
> str_c("원소 : ", v1, sep="") #  반복문자를 사용한 결합
[1] "원소 : A" "원소 : B" "원소 : C"
> str_c("원소", v1, sep=":") #  반복문자를 사용한 결합
[1] "원소:A" "원소:B" "원소:C"
```

aggregate () 함수가 제일 적합 - 옵션을 따로 쓰지 않아도 바로 나옴

1. tapply

원소 개수 다름->리스트로 반환

```
> tapply(emp$ENAME, emp$DEPTNO, str_c)
$`10`
[1] "CLARK"  "KING"   "MILLER"

$`20`
[1] "SMITH" "JONES" "SCOTT" "ADAMS" "FORD"

$`30`
[1] "ALLEN"  "WARD"   "MARTIN" "BLAKE"  "TURNER" "JAMES"
```

```
> tapply(emp$ENAME, emp$DEPTNO, str_c, collapse = ",")
                                10
               "CLARK,KING,MILLER"
                                20
      "SMITH,JONES,SCOTT,ADAMS,FORD"
                                30
"ALLEN,WARD,MARTIN,BLAKE,TURNER,JAMES"
```

```
> as.data.frame(ename=tapply(emp$ENAME, emp$DEPTNO, str_c, collapse = ","))
                            ename
10                 CLARK,KING,MILLER
20       SMITH,JONES,SCOTT,ADAMS,FORD
30 ALLEN,WARD,MARTIN,BLAKE,TURNER,JAMES
```

2. aggregate - 자동으로 ","가 붙는 형태로 출력

```
> aggregate(ENAME ~ DEPTNO, emp, str_c)
  DEPTNO                                ENAME
1     10                  CLARK, KING, MILLER
2     20        SMITH, JONES, SCOTT, ADAMS, FORD
3     30 ALLEN, WARD, MARTIN, BLAKE, TURNER, JAMES
```

cf. str_c -> c로 대체 가능 (in 결합함수)

```
> aggregate(ENAME ~ DEPTNO, emp, c)
  DEPTNO                                ENAME
1     10                   CLARK, KING, MILLER
2     20         SMITH, JONES, SCOTT, ADAMS, FORD
3     30 ALLEN, WARD, MARTIN, BLAKE, TURNER, JAMES
```

paste 함수도 가능

```
> aggregate(ENAME ~ DEPTNO, emp, paste)
  DEPTNO                                ENAME
1     10                   CLARK, KING, MILLER
2     20         SMITH, JONES, SCOTT, ADAMS, FORD
3     30 ALLEN, WARD, MARTIN, BLAKE, TURNER, JAMES
```

3. ddply - aggregate와 다르게 **옵션을 써줘야** 가능함

```
> ddply(emp, .(DEPTNO), summarise, c1=paste(ENAME, collapse=","))
  DEPTNO                                c1
1     10                 CLARK,KING,MILLER
2     20        SMITH,JONES,SCOTT,ADAMS,FORD
3     30 ALLEN,WARD,MARTIN,BLAKE,TURNER,JAMES

> ddply(emp, .(DEPTNO), summarise, c1=str_c(ENAME, collapse=","))
  DEPTNO                                c1
1     10                 CLARK,KING,MILLER
2     20        SMITH,JONES,SCOTT,ADAMS,FORD
3     30 ALLEN,WARD,MARTIN,BLAKE,TURNER,JAMES
```

#in oracle - listagg함수

그룹별로 한 행으로 묶어 결합하여 출력하기

```
select DEPTNO,
        listagg(ename, -- 결합할 대상값이 있는 컬럼
                ',') -- 각 값끼리의 구분자 지정 ,
            within group(order by ename) AS "각 부서별 인원 현황" -- 각 그룹내 정렬 순서 (필수)
    from emp
group by DEPTNO ;
```

sqldf 사용 -> 모든 문법이 다 적용되지 않음

180524

mdply()

mdply(.data, .fun)

- m{adl_}ply() / maply(), mdply(), mlply(), m_ply()
- 데이터프레임 또는 배열을 인자로 받아 각 컬럼을 주어진 함수에 적용하고 그 실행 결과를 조합한다.
- **mapply()**와 유사하나 차이점 : 입력 순서 - 데이터 -> 함수, 데이터프레임으로 반환

ex.

```
> (x<-data.frame(mean=1:5, sd=1:5))
  mean sd
1    1  1
2    2  2
3    3  3
4    4  4
5    5  5
> mdply(x, rnorm, n=2)
  mean sd        V1        V2
1    1  1 -1.411721 0.2498898
2    2  2 -1.216258 1.7266470
3    3  3  1.257360 3.2940622
4    4  4  7.061547 5.2922217
5    5  5  5.060172 5.7853972
> mdply(x, rnorm, n=8)
  mean sd          V1          V2        V3        V4         V5        V6         V7         V8
1    1  1  2.00167373  0.80351728 0.1226530  2.880503  0.8191335 0.2553776  1.0411511 0.71850675
2    2  2  2.04199275  1.02800854 0.2388321  1.336251 -0.9229510 2.1841748  1.5076883 4.51645468
3    3  3  0.08136932 -0.09518319 4.3607522  1.193678  0.1044883 0.2253267  1.6233690 2.11658458
4    4  4  7.45804443 -0.41207169 2.0154206  4.702974 -1.9117035 8.8725260 -0.2686646 0.03075555
5    5  5 12.74356968  0.45825090 3.8880042 -6.750397 -4.0625587 3.7124689  4.2248851 2.55048280
```

- rnorm ~ 원래 n이 먼저 순서인 함수. 현재 작성된 건 순서에 맞지 않으므로 인자의 이름을 사용해서 뒤에 명시
- rnorm : n을 2로 확정한 후 순차적으로 x에서 데이터를 가져와서 정보를 출력
- mdply -> 순서 뒤바꿈 ~ 주의

04) 데이터 구조의 변형과 요약

reshape2 패키지

- 데이터의 모양을 바꾸거나 그룹별 요약 값을 계산하는 함수들을 담고 있는 패키지
- 변환된 데이터는 측정치를 variable과 value라는 두 컬럼으로 표현
- stack, unstack은 제한적이어서 melt, cast 많이 씀

melt()

melt(data, id.vars, measure.vars, na.rm=FALSE, variable.name, value.name)

id.vars = 식별자 컬럼들

measure.vars = 측정치 컬럼들, 이 값이 생략되면 id.vars가 아닌 모든 컬럼이 측정치 컬럼

na.rm = FALSE NA인 행을 결과에 포함시킬지 여부. FALSE는 NA를 제거하지 않음

variable.name, value.name - 새로 생긴 컬럼 이름 지정

- 여러 컬럼으로 구성된 데이터를 데이터 식별자(id), 측정변수(variable), 측정값(value)로 표현
- Wide -> Long 형태로 변형하는 stack과 비슷 (차이점 : 식별자 지정 가능)
- stack보다 melt 많이 사용

ex

컬럼의 확장 형태로 되는 구조 지양.

2000년 2001년 .... 2018년이 컬럼인 이런 식의 wide 형태 데이터 X

- join, 그룹화 불가능 =>Tidy data화 하기(눕히기)

```
> library(googleVis)
> Fruits
    Fruit Year Location Sales Expenses Profit       Date
1  Apples 2008     West    98       78     20 2008-12-31
2  Apples 2009     West   111       79     32 2009-12-31
3  Apples 2010     West    89       76     13 2010-12-31
4 Oranges 2008     East    96       81     15 2008-12-31
5 Bananas 2008     East    85       76      9 2008-12-31
6 Oranges 2009     East    93       80     13 2009-12-31
7 Bananas 2009     East    94       78     16 2009-12-31
8 Oranges 2010     East    98       91      7 2010-12-31
9 Bananas 2010     East    81       71     10 2010-12-31
```

melt(fruits, id='year')

	Year	variable	value
1	2008	Fruit	Apples
2	2009	Fruit	Apples
3	2010	Fruit	Apples
4	2008	Fruit	Oranges
5	2008	Fruit	Bananas
6	2009	Fruit	Oranges
7	2009	Fruit	Bananas
8	2010	Fruit	Oranges
9	2010	Fruit	Bananas
10	2008	Location	West
11	2009	Location	West
12	2010	Location	West
13	2008	Location	East
14	2008	Location	East
15	2009	Location	East
16	2009	Location	East
17	2010	Location	East
18	2010	Location	East
19	2008	Sales	98
20	2009	Sales	111
21	2010	Sales	89
22	2008	Sales	96
23	2008	Sales	85
24	2009	Sales	93
25	2009	Sales	94
26	2010	Sales	98
27	2010	Sales	81
28	2008	Expenses	78
29	2009	Expenses	79

- id 고정, id의 변수들로 각각 구분되고 새로운 value 컬럼 생김
- 하나의 행이 고정된 컬럼인 id를 제외한 column들을 수직적으로 쌓음
- 컬럼별 특징을 가짐 ==========> 모델링 중요

책 예시

```
> m<-melt(french_fries, id.vars=1:4)
> head(m)
  time treatment subject rep variable value
1    1         1       3   1   potato   2.9
2    1         1       3   2   potato  14.0
3    1         1      10   1   potato  11.0
4    1         1      10   2   potato   9.9
5    1         1      15   1   potato   1.2
6    1         1      15   2   potato   8.8
> library(plyr)
> ddply(m, .(variable), summarise, mean=mean(value, na.rm=TRUE))
  variable      mean
1   potato 6.9525180
2  buttery 1.8236994
3   grassy 0.6641727
4   rancid 3.8522302
5   painty 2.5217579
> french_fries[!complete.cases(french_fries),]
    time treatment subject rep potato buttery grassy rancid painty
315    5         3      15   1     NA      NA     NA     NA     NA
455    7         2      79   1    7.3      NA    0.0    0.7      0
515    8         1      79   1   10.5      NA    0.0    0.5      0
520    8         2      16   1    4.5      NA    1.4    6.7      0
563    8         2      79   2    5.7       0    1.4    2.3     NA
> m<-melt(id=1:3, french_fries, na.rm=T)
> head(m)
  time treatment subject variable value
1    1         1       3      rep     1
2    1         1       3      rep     2
3    1         1      10      rep     1
4    1         1      10      rep     2
5    1         1      15      rep     1
6    1         1      15      rep     2
```

일반적으로 이런 경우 사용

```
> data1<-data.frame(class=1, "홍길동"=50, "박길동"=90, "최길동"=49, "김길동"=40)
> data1
  class 홍길동 박길동 최길동 김길동
1     1     50     90     49     40
> melt(data1, id="class")
  class variable value
1     1   홍길동    50
2     1   박길동    90
3     1   최길동    49
4     1   김길동    40
```

연습문제

실업률데이터 로딩 후 연도별 실업률의 평균, 월별 실업률의 평균을 구하세요

data1<-read.csv("2000-2013년_연령별실업율_40-49세.csv",stringsAsFactor=F)
data2<-melt(data1, id.vars="월")

cf . 처음이름설정

data2<-melt(data1, id.vars="월", **variable.name = "년도", value.name = "실업률"**)

cf. 2000년과 2001년만 가져오기

data2<-melt(data1, id.vars="월", **measure.vars = c("X2000년", "X2001년")**, variable.name = "년도", value.name = "실업률")

(1) 월별

tapply(data2$value, data2$월, mean) -- 리스트
aggregate(value ~ 월, data2, mean)
ddply(data2, .(월), summarise, 실업률평균=mean(value))

(2) 연도별

tapply(data2$value, data2$variable, mean) -- 리스트
aggregate(value ~ variable, data2, mean)
ddply(data2, .(variable), summarise, 실업률평균=mean(value))

(3) 각 해마다 실업률이 가장 높은 월을 출력

ddply(data2, .(variable), subset, value==max(value))
(각 해 마다 -> 구분 기준
가장 높은 실업률 -> max 함수
월 - 구분 기준X, 함수 X --> subset 사용)

cf.subset 함수는 select 인자 존재 / 원하는 정보만 선택

ddply(data2, .(variable), subset, value==max(value), **select=월**)

cf. merge() 사용

f1<-aggregate(value ~ variable, dd1, max)
merge(f1, dd1)

cast() ★★ 그래프 그릴 때 (막대그래프) 요약함수와 함께 사용★★

dcast(data, formula, fun.aggregate=NULL)

data - melt()된 데이터

formula **id변수(고정) ~ variable 변수(펼칠 변수)** 순서

아무 변수도 지정하지 않으려면 . 사용

모든 변수 표현하려면 ... 사용

- 얻고자 하는 데이터 타입에 따라 dcast(), acast()로 구분하여 사용
- long -> wide 형태로
- 데이터 요약

-> melt()에서 사용한 것보다 적은 개수의 식별자를 dcast()의 formula에 지정하면 된다.

그룹핑 -> 데이터프레임화 불가 -> dcast 사용 - 행방향 앞에, 물결 뒤 펴지게끔 (unstack도 가능)

cf. dcast의 subset 기능

dcast(mtest, name~var_name, sum, subset=.(name=='apple'))

grouping 한 값 중 저 조건을 만족하는 값만 출력 (in SQL - having 절)

앞 연습문제에 이어서

```
> dcast(data2, 월 ~ 년도)
Using 실업률 as value column: use value.var to override.
   월 X2000년 X2001년
1   1     4.0     4.3
2   2     4.5     4.5
3   3     3.9     4.1
4   4     3.6     3.3
5   5     3.4     2.7
6   6     3.3     2.4
7   7     3.1     2.6
8   8     3.2     2.9
9   9     3.2     2.5
10 10     3.1     2.3
11 11     3.2     2.2
12 12     3.5     2.2
```

```
> mtest
   Year    Fruit var_name val.name
1  2008   Apples Location     West
2  2009   Apples Location     West
3  2010   Apples Location     West
4  2008  Oranges Location     East
5  2008  Bananas Location     East
6  2009  Oranges Location     East
7  2009  Bananas Location     East
8  2010  Oranges Location     East
9  2010  Bananas Location     East
10 2008   Apples    Sales       98
```

```
> m2<- dcast(mtest, Year ~ Fruit + var_name )
Using val.name as value column: use value.var to override.
> m2
  Year Apples_Location Apples_Sales Apples_Expenses Apples_Profit Apples_Date Bananas_Location Bananas_Sales Bananas_Expenses
1 2008           West          98             78            20       14244            East            85              76
2 2009           West         111             79            32       14609            East            94              78
3 2010           West          89             76            13       14974            East            81              71
  Bananas_Profit Bananas_Date Oranges_Location Oranges_Sales Oranges_Expenses Oranges_Profit Oranges_Date
1              9        14244            East            96              81             15        14244
2             16        14609            East            93              80             13        14609
3             10        14974            East            98              91              7        14974
```

```
> m2<- dcast(mtest, Year + Fruit ~ var_name )
Using val.name as value column: use value.var to override.
> m2
   Year    Fruit Location Sales Expenses Profit  Date
1 2008   Apples     West    98       78     20 14244
2 2008  Bananas     East    85       76      9 14244
3 2008  Oranges     East    96       81     15 14244
4 2009   Apples     West   111       79     32 14609
5 2009  Bananas     East    94       78     16 14609
6 2009  Oranges     East    93       80     13 14609
7 2010   Apples     West    89       76     13 14974
8 2010  Bananas     East    81       71     10 14974
9 2010  Oranges     East    98       91      7 14974
```

요약

melt()에서 사용한 식별자를 dcast()에서 복구할 때,
포뮬러 '~'의 왼편에서 제외한다면
dcast()시 여러 행이 하나의 셀로 모이게 될 것이고,
이렇게 모인 값들에 요약치를 계산하는 것이 핵심.

- dcast()시 time만 포뮬러에서 '~' 왼쪽에 적고, 측정 변수를 오른쪽에 적은 예

```
> m<-melt(french_fries, id.vars=1:4)
> str(m)
'data.frame':   3480 obs. of  6 variables:
 $ time     : Factor w/ 10 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ treatment: Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
 $ subject  : Factor w/ 12 levels "3","10","15",..: 1 1 2 2 3 3 4 4 5 5 ...
 $ rep      : num  1 2 1 2 1 2 1 2 1 2 ...
 $ variable : Factor w/ 5 levels "potato","buttery",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ value    : num  2.9 14 11 9.9 1.2 8.8 9 8.2 7 13 ...
> dcast(m, time ~ variable)
Aggregation function missing: defaulting to length
   time potato buttery grassy rancid painty
1     1     72      72     72     72     72
2     2     72      72     72     72     72
3     3     72      72     72     72     72
4     4     72      72     72     72     72
5     5     72      72     72     72     72
6     6     72      72     72     72     72
7     7     72      72     72     72     72
8     8     72      72     72     72     72
9     9     60      60     60     60     60
10   10     60      60     60     60     60
```

time값에 해당하는 여러 개의 데이터가 결과 데이터 프레임의 한 셀에 모이게 되며,
dcast()는 자동으로 **length()**를 적용해 같은 셀에 모인 행의 개수를 센다. - 그룹의 행 개수 파악
cf. str_length - 각 원소의 길이

time에 따라 평균값이 어떻게 달라지는지 보여주는 예

```
> dcast(m, time ~ variable, mean, na.rm=T)
   time   potato  buttery    grassy   rancid   painty
1     1 8.562500 2.236111 0.9416667 2.358333 1.645833
2     2 8.059722 2.722222 1.1819444 2.845833 1.444444
3     3 7.797222 2.102778 0.7500000 3.715278 1.311111
4     4 7.713889 1.801389 0.7416667 3.602778 1.372222
5     5 7.328169 1.642254 0.6352113 3.529577 2.015493
6     6 6.670833 1.752778 0.6736111 4.075000 2.341667
7     7 6.168056 1.369014 0.4208333 3.886111 2.683333
8     8 5.431944 1.182857 0.3805556 4.272222 3.938028
9     9 5.673333 1.586667 0.2766667 4.670000 3.873333
10   10 5.703333 1.765000 0.5566667 6.068333 5.291667
```

-length 외에 sum, mean 또는 임의의 함수를 적용해 구할 수 있다.

cf ddply로 표현하기 - 괄호 안 두 개 다 !

ddply(m, **.(time, variable)**, summarise, mean(value, na.rm=T))

각 time마다 (treatment, variable) 순서쌍에 해당하는 value의 평균을 계산한 예

```
> dcast(m, time ~ treatment + variable, mean, na.rm=T)
   time 1_potato 1_buttery  1_grassy 1_rancid  1_painty 2_potato 2_buttery  2_grassy 2_rancid  2_painty 3_potato 3_buttery  3_grassy 3_rancid 3_painty
1     1 7.925000 1.7958333 0.9041667 2.758333 2.1500000 8.775000  2.491667 0.9958333 1.716667 0.8083333 8.987500  2.420833 0.9250000 2.600000 1.979167
2     2 7.591667 2.5250000 1.0041667 3.900000 1.9750000 8.537500  3.125000 0.9500000 2.141667 0.6625000 8.050000  2.516667 1.5916667 2.495833 1.695833
3     3 7.770833 2.2958333 0.8166667 4.650000 1.1166667 7.637500  2.079167 0.7250000 2.895833 1.5625000 7.983333  1.933333 0.7083333 3.600000 1.254167
4     4 8.404167 1.9791667 1.0250000 2.079167 0.4666667 8.204167  1.608333 0.6416667 3.512500 1.8583333 6.533333  1.816667 0.5583333 5.216667 1.791667
5     5 7.741667 1.3666667 0.7708333 4.279167 3.0083333 6.933333  1.858333 0.5833333 3.641667 0.7375000 7.308696  1.704348 0.5478261 2.630435 2.313043
6     6 6.079167 1.8250000 0.4666667 4.337500 2.5541667 7.016667  2.054167 0.9208333 3.841667 2.7500000 6.916667  1.379167 0.6333333 4.045833 1.720833
7     7 6.283333 1.2416667 0.1625000 3.204167 2.1958333 5.729167  1.356522 0.5500000 3.837500 3.0416667 6.491667  1.508333 0.5500000 4.616667 2.812500
8     8 5.175000 0.9869565 0.6333333 5.387500 4.5875000 5.641667  1.582609 0.3958333 3.825000 3.8652174 5.479167  0.987500 0.1125000 3.604167 3.358333
9     9 6.070000 1.8300000 0.1350000 3.950000 2.9050000 5.470000  1.665000 0.0800000 5.240000 4.0450000 5.480000  1.265000 0.6150000 4.820000 4.670000
10   10 5.465000 1.9600000 0.4550000 6.495000 5.4000000 5.580000  1.795000 0.6950000 6.310000 6.1700000 6.065000  1.540000 0.5200000 5.400000 4.305000
```

- dcast() 호출시 포뮬러의 '~' 우측은 측정 변수를 적는 곳으로, 이곳에 적은 변수는 결과에서 새로운 컬럼이 된다.

- **'~' 오른쪽에 +로 여러 컬럼 합침 --> 서로 결합된 상태로 컬럼이 펼쳐지게 됨(포뮬라의 확장)**

cf. 같은 연산을 ddply로 해보기

```
> ddply(m, .(time, treatment, variable), summarise,mean=mean(value, na.rm=T))
   time treatment variable      mean
1     1         1   potato 7.9250000
2     1         1  buttery 1.7958333
3     1         1   grassy 0.9041667
4     1         1   rancid 2.7583333
5     1         1   painty 2.1500000
6     1         2   potato 8.7750000
7     1         2  buttery 2.4916667
8     1         2   grassy 0.9958333
9     1         2   rancid 1.7166667
10    1         2   painty 0.8083333
11    1         3   potato 8.9875000
12    1         3  buttery 2.4208333
13    1         3   grassy 0.9250000
14    1         3   rancid 2.6000000
15    1         3   painty 1.9791667
16    2         1   potato 7.5916667
17    2         1  buttery 2.5250000
18    2         1   grassy 1.0041667
19    2         1   rancid 3.9000000
20    2         1   painty 1.9750000
```

연습문제 노선번호별 승 / 하차 각각 평균 구하기

```
> line1<-read.csv(file="1-4호선승하차승객수.csv",stringsAsFactors=F)
> line2<-melt(line1, id.vars = c("노선번호","시간"))
> ddply(line2, .(노선번호,variable), summarise,mean(value))
  노선번호 variable       ..1
1   line_1     승차  483346.7
2   line_1     하차  471542.3
3   line_2     승차 2369694.2
4   line_2     하차 2399158.0
5   line_3     승차  824213.4
6   line_3     하차  827766.8
7   line_4     승차  992552.9
8   line_4     하차 1001103.3
```

but, 이런 형태 말고 **노선번호, 승차, 하차**를 column 형식으로 출력하는 건 불가능 **->dcast**

2. **요약**

```
> line2
    노선번호 시간 variable   value
1     line_1  506     승차   88136
2     line_1  607     승차  114628
3     line_1  708     승차  259282
4     line_1  809     승차  384892
5     line_1  910     승차  315797
6     line_1 1011     승차  340972
7     line_1 1112     승차  411897
8     line_1 1213     승차  471989
9     line_1 1314     승차  558377
10    line_1 1415     승차  589343
```

dcast만 사용, 노선번호 ~ variable 사용 and **시간 생략** - 요약함수로 length() 자동 적용

```
> dcast(line2, 노선번호  ~ variable )
Aggregation function missing: defaulting to length
  노선번호 승차 하차
1   line_1   20   20
2   line_2   20   20
3   line_3   20   20
4   line_4   20   20
```

3. mean() 요약함수 사용

```
> dcast(line2, 노선번호  ~ variable , mean)
  노선번호       승차       하차
1   line_1  483346.7  471542.3
2   line_2 2369694.2 2399158.0
3   line_3  824213.4  827766.8
4   line_4  992552.9 1001103.3
```

-> 분리할 대상을 잘 보고 formula 잘 작성하기

groupby ~ 펼칠 column (생략하는 column 있어야 함) /// for 그래프 그리기용

cf. aggregate, ddply 이용해서 똑같은 형태 만들기(in 원데이터)

```
> ddply(data2, .(노선번호), summarise, 승차=mean(승차), 하차=mean(하차))
  노선번호       승차       하차
1   line_1  483346.7  471542.3
2   line_2 2369694.2 2399158.0
3   line_3  824213.4  827766.8
4   line_4  992552.9 1001103.3
> aggregate(cbind(승차, 하차) ~ 노선번호, data2, mean)
  노선번호       승차       하차
1   line_1  483346.7  471542.3
2   line_2 2369694.2 2399158.0
3   line_3  824213.4  827766.8
4   line_4  992552.9 1001103.3
```

연습문제

Fruit, Year 별로 sales / Expenses / Profit 평균 구하기

```
> Fruits
    Fruit Year Location Sales Expenses Profit       Date
1  Apples 2008     West    98       78     20 2008-12-31
2  Apples 2009     West   111       79     32 2009-12-31
3  Apples 2010     West    89       76     13 2010-12-31
4 Oranges 2008     East    96       81     15 2008-12-31
5 Bananas 2008     East    85       76      9 2008-12-31
6 Oranges 2009     East    93       80     13 2009-12-31
7 Bananas 2009     East    94       78     16 2009-12-31
8 Oranges 2010     East    98       91      7 2010-12-31
9 Bananas 2010     East    81       71     10 2010-12-31
```

```
> dd1 <- melt(Fruits, id.vars=c("Fruit", "Year"), measure.vars = 4:6)
> dd1
     Fruit Year variable value
1   Apples 2008    Sales    98
2   Apples 2009    Sales   111
3   Apples 2010    Sales    89
4  Oranges 2008    Sales    96
5  Bananas 2008    Sales    85
6  Oranges 2009    Sales    93
7  Bananas 2009    Sales    94
8  Oranges 2010    Sales    98
9  Bananas 2010    Sales    81
```

dcast / ddply

```
> dcast(dd1, Fruit ~ variable, mean)
    Fruit    Sales Expenses   Profit
1  Apples 99.33333 77.66667 21.66667
2 Bananas 86.66667 75.00000 11.66667
3 Oranges 95.66667 84.00000 11.66667
> ddply(dd1, .(Fruit, variable), summarise, mean(value))
    Fruit variable      ..1
1  Apples    Sales 99.33333
2  Apples Expenses 77.66667
3  Apples   Profit 21.66667
4 Bananas    Sales 86.66667
5 Bananas Expenses 75.00000
6 Bananas   Profit 11.66667
7 Oranges    Sales 95.66667
8 Oranges Expenses 84.00000
9 Oranges   Profit 11.66667
```

cf. 원데이터로 해보기

ddply는 가능

```
> ddply(Fruits, .(Fruit), summarise, smean=mean(Sales), emean=mean(Expenses), pm
ean=mean(Profit))
    Fruit    smean    emean    pmean
1  Apples 99.33333 77.66667 21.66667
2 Bananas 86.66667 75.00000 11.66667
3 Oranges 95.66667 84.00000 11.66667
```

dcast는 불가능

```
> dcast(Fruits, Fruit ~ Sales + Expenses + Profit, mean)
Using Date as value column: use value.var to override.
    Fruit 81_71_10 85_76_9 89_76_13 93_80_13 94_78_16 96_81_15 98_78_20
1  Apples       NaN     NaN    14974       NaN       NaN       NaN    14244
2 Bananas    14974   14244      NaN       NaN    14609       NaN      NaN
3 Oranges       NaN     NaN      NaN    14609       NaN    14244      NaN
  98_91_7 111_79_32
1     NaN     14609
2     NaN       NaN
3   14974       NaN
```

-> 왜냐면 Sales, Expenses, Profit이 한 Column 내에 있어야 그에 따른 분류를 dcast가 해주는데! 각각 column으로 데려와주지는 않는 것 같다. 그리고 value도 못 잡는 것 같고!

Sales + Expenses + Profit은 펼치는 형태가 아님

aggregate 편함

```
> aggregate(cbind(Sales,Expenses,Profit) ~ Fruit + Year, Fruits, mean)
    Fruit Year Sales Expenses Profit
1  Apples 2008    98       78     20
2 Bananas 2008    85       76      9
3 Oranges 2008    96       81     15
4  Apples 2009   111       79     32
5 Bananas 2009    94       78     16
6 Oranges 2009    93       80     13
7  Apples 2010    89       76     13
8 Bananas 2010    81       71     10
9 Oranges 2010    98       91      7
```

- 대신 여러 개 컬럼으로 표시하려면 cbind() 사용!

연습문제

sales 데이터 로딩 후 날짜별 / 품목별 판매량의 평균

```
s1<- read.csv("sales.csv",stringsAsFactors = F)
```

```
> ddply(s1, .(날짜, 품목), summarise, mean=mean(판매량))
         날짜   품목      mean
1  2018-01-01     TV 745.3333
2  2018-01-01 냉장고 862.6667
3  2018-01-01 세탁기 396.3333
4  2018-01-01 에어컨 704.6667
5  2018-01-02     TV 381.3333
6  2018-01-02 냉장고 547.3333
7  2018-01-02 세탁기 577.6667
8  2018-01-02 에어컨 518.6667
9  2018-01-03     TV 518.0000
10 2018-01-03 냉장고 512.0000
11 2018-01-03 세탁기 380.6667
12 2018-01-03 에어컨 454.0000
> dcast(s1, 날짜 ~ 품목, mean)
Using 판매량 as value column: use value.var to override.
        날짜       TV   냉장고   세탁기   에어컨
1 2018-01-01 745.3333 862.6667 396.3333 704.6667
2 2018-01-02 381.3333 547.3333 577.6667 518.6667
3 2018-01-03 518.0000 512.0000 380.6667 454.0000
>
```

dcast -> 지정 X인 경우 가장 오른쪽 걸로 계산됨

/ 만약 가격 컬럼이 옆에 있었다면 가격으로 계산되었을 것. in R, cast()에서 해당 value 지정 X

/ 펴고자 하는 데이터가 여러 개 있으면 각각 따로 테이블을 분리해야 함

(cf. 파이썬 - 여러개 선택 가능, 해당 value 지정 가능 - 멀티인덱스 / 상위칼럼)

cf. round 넣기 ~ 사용자함수 쓰기 round는 벡터별 연산 가능

```
> dcast(s1, 날짜 ~ 품목, function(row){round(mean(row))})
Using 판매량 as value column: use value.var to override.
       날짜  TV 냉장고 세탁기 에어컨
1 2018-01-01 745    863    396    705
2 2018-01-02 381    547    578    519
3 2018-01-03 518    512    381    454
```

cf. unstack은 날짜는 하기 어려붐... dcast 사용하는 게 좋음 첫번째 행이 무슨 데이턴지 모름

```
> dd<-ddply(s1, .(날짜, 품목), summarise, avg_sale=mean(판매량))
> unstack(dd,avg_sale ~ 품목)
        TV   냉장고   세탁기   에어컨
1 745.3333 862.6667 396.3333 704.6667
2 381.3333 547.3333 577.6667 518.6667
3 518.0000 512.0000 380.6667 454.0000
```

cf. ddply랑 aggregate는 힘을 못 쓴당!

```
> ddply(s1, .(날짜, 품목), summarise, mean(판매량))
         날짜   품목      ..1
1  2018-01-01     TV 745.3333
2  2018-01-01 냉장고 862.6667
3  2018-01-01 세탁기 396.3333
4  2018-01-01 에어컨 704.6667
5  2018-01-02     TV 381.3333
6  2018-01-02 냉장고 547.3333
7  2018-01-02 세탁기 577.6667
8  2018-01-02 에어컨 518.6667
9  2018-01-03     TV 518.0000
10 2018-01-03 냉장고 512.0000
11 2018-01-03 세탁기 380.6667
12 2018-01-03 에어컨 454.0000
> aggregate(판매량 ~ 날짜 + 품목, s1, mean)
         날짜   품목    판매량
1  2018-01-01     TV 745.3333
2  2018-01-02     TV 381.3333
3  2018-01-03     TV 518.0000
4  2018-01-01 냉장고 862.6667
5  2018-01-02 냉장고 547.3333
6  2018-01-03 냉장고 512.0000
7  2018-01-01 세탁기 396.3333
8  2018-01-02 세탁기 577.6667
9  2018-01-03 세탁기 380.6667
10 2018-01-01 에어컨 704.6667
11 2018-01-02 에어컨 518.6667
12 2018-01-03 에어컨 454.0000
>
```

지점 + 품목별로 나눠보기

-출력형태 비교

```
> dcast(s1, 날짜 ~ 지점 + 품목, mean)
Using 판매량 as value column: use value.var to override.
        날짜 c1_TV c1_냉장고 c1_세탁기 c1_에어컨 c2_TV c2_냉장고 c2_세탁기 c2_에어컨
1 2018-01-01   859       899       453       400 688.5     844.5     368.0     857.0
2 2018-01-02   345       565       234       645 399.5     538.5     749.5     455.5
3 2018-01-03   454       845       545       493 550.0     345.5     298.5     434.5
```

```
> dcast(s1, 날짜 +지점~ 품목, mean)
Using 판매량 as value column: use value.var to override.
        날짜 지점    TV 냉장고 세탁기 에어컨
1 2018-01-01   c1 859.0  899.0  453.0  400.0
2 2018-01-01   c2 688.5  844.5  368.0  857.0
3 2018-01-02   c1 345.0  565.0  234.0  645.0
4 2018-01-02   c2 399.5  538.5  749.5  455.5
5 2018-01-03   c1 454.0  845.0  545.0  493.0
6 2018-01-03   c2 550.0  345.5  298.5  434.5
```

지점을 먼저 쓰는 게 보기 좋다(지점별 비교시)

이 데이터 형식에서 ddply, aggregate는 행에 품목 이름을 가져오지 못함!

180525

05) 데이터 테이블 : 더 빠르고 편리한 데이터 프레임

데이터 테이블 - data.table 패키지

- 데이터 프레임을 대신하여 사용할 수 있는 더 빠르고 편리한 데이터 타입
- **key 값**을 통한 **색인**이 가능한 구조이므로 속도가 빠름 cf. in SQL - index ~ key 값
- 특히 조건 연산이 다른 데이터 구조보다 빠름
- 참조를 통한 데이터 갱신을 지원하여 데이터 복사에 대한 비용을 줄임
- 연산의 편의성 (특히 조건 연산의 편의성 제공)

```
install.packages("data.table")
library(data.table)
```

cf. in DB - row별 rowid가 존재해서 이를 가지고 인덱싱 / R은 없음. key값 지정필요

/ 데이터가 많을 수록 키를 설정하는 시간이 오래 걸림.
따라서 키를 먼저 설정하고 데이터를 만들면 조금 더 빠르게 할 수 있음

데이터 테이블 생성

data.table()
tag=value 형태로 컬럼과 값을 지정
stringsAsFactors
- as.data.frame() / as.data.table()을 통한 상호변환 가능
- 데이터 테이블을 생성한다.

as.data.table()
df 데이터프레임을 인자로 받음

tables()
- 모든 데이터 테이블 객체 목록을 저장한 데이터 테이블을 반환한다.

```
> (iris_table<-as.data.table(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
  4:          4.6         3.1          1.5         0.2    setosa
  5:          5.0         3.6          1.4         0.2    setosa
 ---
146:          6.7         3.0          5.2         2.3 virginica
147:          6.3         2.5          5.0         1.9 virginica
148:          6.5         3.0          5.2         2.0 virginica
149:          6.2         3.4          5.4         2.3 virginica
150:          5.9         3.0          5.1         1.8 virginica
```

```
> class(iris_table)
[1] "data.table" "data.frame"
> tables()
        NAME NROW NCOL MB                                                    COLS KEY
1: iris_table  150    5  0 Sepal.Length,Sepal.Width,Petal.Length,Petal.Width,Species
2:         x    3    2  0                                                     x,y
Total: 0MB
```

- class는 data.table + data.frame임

데이터 테이블의 활용

- 데이터 테이블의 클래스는 data.frame을 포함
- 클래스가 data.frame을 포함하므로 데이터 프레임을 처리하는 함수 적용 가능
- 아주 간혹 데이터 프레임에서 되던 함수가 데이터 테이블에서 안 될 수도 있음
  -> as.data.frame()을 사용해 변환

데이터 접근과 그룹 연산

색인

- 데이터 테이블의 색인은 [행, **표현식**, 옵션] 형태로 접근
- 행의 색인은 데이터 프레임과 동일(행 번호 / 행을 선택할지를 나타내는 진릿값)
- 열의 색인은 이름 색인 사용 권고, '컬럼명을 사용한 표현식' 가능
  -> **여러 컬럼을 지정할 경우 리스트**로 전달, 벡터로 전달시 벡터로 결과 출력
- 컬럼명을 사용한 표현식 가능(함수결과)

```
> DT<-as.data.table(iris)
> DT[1,]
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1:          5.1         3.5          1.4         0.2  setosa
> DT[DT$Species=="setosa",]
    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
 4:          4.6         3.1          1.5         0.2  setosa
```

```
> DT[1,Sepal.Length]
[1] 5.1
> DT[1,list(Sepal.Length, Species)]
   Sepal.Length Species
1:          5.1  setosa
> DT[,mean(Sepal.Length)]
[1] 5.843333
> DT[,mean(Sepal.Length-Sepal.Width)]
[1] 2.786
```

데이터 테이블의 그룹 연산

세 번째 인자에는 데이터를 그룹 지을 변수를 지정, 그룹 지을 변수가 여러 개일 경우 모두 나열

```
> DT[,mean(Sepal.Length), by="Species"]
      Species    V1
1:     setosa 5.006
2: versicolor 5.936
3:  virginica 6.588
```

```
> DT <- data.table(x=c(1,2,3,4,5),
+                  y=c("a","a","a","b","b"),
+                  z=c("c","c","d","d","d"))
> DT[,mean(x), by="y,z"]
   y z  V1
1: a c 1.5
2: a d 3.0
3: b d 4.5
```

key를 사용한 빠른 데이터 접근

setkey(x, 컬럼명 1, 컬럼명 2, ...)

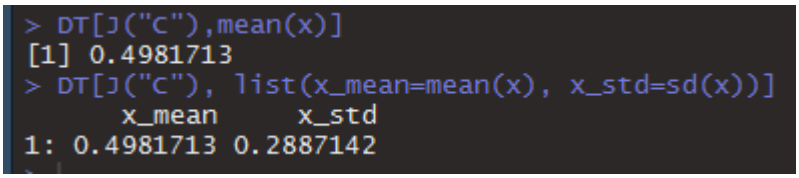
- 데이터 베이스의 테이블처럼 데이터를 찾기 위한 key 설정 가능
- key를 미리 생성하면 이진 탐색 트리가 생성되어 더 빠른 색인이 가능
- setkey 명령어로 조건 검색에 자주 사용되는 컬럼 / 연산에 자주 사용하는 컬럼에 key 생성
- key를 통한 색인은 J(컬럼명1, 컬럼명2, ...) 형태로 검색

```
> DF<-data.frame(x=runif(26000), y=rep(LETTERS,each=10000))
> str(DF)
'data.frame':   260000 obs. of  2 variables:
 $ x: num  0.1117 0.1802 0.8294 0.0813 0.1913 ...
 $ y: Factor w/ 26 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
> head(DF)
           x y
1 0.11170519 A
2 0.18018185 A
3 0.82939780 A
4 0.08128065 A
5 0.19131325 A
6 0.42161951 A
> system.time(x<-DF[DF$y=="C",])
 사용자  시스템 elapsed
      0       0       0
```

```
> DT <- as.data.table(DF)
> setkey(DT, y)
> system.time(x<-DT[J("C"),])
 사용자  시스템 elapsed
   0.01    0.00    0.02
```

두 번째 인자는 컬럼명을 사용한 표현식이 될 수 있으므로 검색과 동시에 계산을 수행할 수 있다. 여러 계산 결과를 원하면 list()안에 표현식을 나열한다.



연습문제

iris 데이터를 활용하여 종별 평균을 구하는 방식에 대해 속도 차 확인

1. 데이터 프레임
2. 데이터 테이블의 키 활용

1. system.time() 씌우기

```
ddply(iris, .(Species), summarise,
        m1=mean(Sepal.Length), m2=mean(Sepal.Width),
        m3=mean(Petal.Length), m4=mean(Petal.Width))
```

 # ddply 를 이용한 그룹연산을 하더라도 mean - 각 컬럼 나열해야 함

cf. tapply <- 첫 번째 인자는 반드시 vector 형식으로 전달해야함.
두 개 이상의 컬럼을 쓸 수 없음. 따라서 이 경우 맞지 않음
apply <- group by 불가능

2. system.time() 씌우기

```
dt <- as.data.table(iris)
setkey(dt, Species)
dt[ , list(m1=mean(Sepal.Length), m2=mean(Sepal.Width),
        m3=mean(Petal.Length), m4=mean(Petal.Width)), by="Species"]
```

 # 여러 컬럼의 평균 구할 시 리스트로 나열

속도 차 별로 안 남 - system.time()

내 흔적

```
library(data.table)

#1.
aggregate(cbind(iris$Sepal.Length, iris$Sepal.Width, iris$Petal.Length, iris$Petal.Width) ~ Species,
iris, mean)

#2.
ddply(iris, .(Species), summarise,
        mean(Sepal.Length), mean(Sepal.Width), mean(Petal.Length), mean(Petal.Width))
#3.
i1<-melt(iris,id.vars="Species")
dcast(i1,Species ~ variable, mean)

DT<-as.data.table(iris)
setkey(DT,Species)

dt[ , list(m1=mean(Sepal.Length), m2=mean(Sepal.Width),
        m3=mean(Petal.Length), m4=mean(Petal.Width)), by="Species"]
```

#이미 setkey 에서 Species 를 설정해서, by=에서 안 해도 되나봄!
색인 세 번째 순서에서 by=""로 컬럼이름 쓰기

#속도 차 별로 안 남 -> iris 데이터 확장시키기

```
> iris_big<-c()
> for (i in 1:1000){
+   iris_big <- rbind(iris_big, iris)
+   }
```

iris : 데이터 프레임 형태임을 기억하기 ! not vector -> c(), append()로 추가 안 함
rbind()로 행 추가함
iris_big 는 빈 벡터로 만들어주면 자동으로 rbind(, ___) 뒤에 있는 형태로 바뀌어짐
(빈 데이터 프레임 형태는 못 만듦)

```
> str(iris_big)
'data.frame':   150000 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

커진 데이터 속도 재확인

```
> #데이터 프레임 구조일 때 속도 확인
> system.time(ddply(iris_big, .(Species), summarise,
+                   m1=mean(Sepal.Length), m2=mean(Sepal.Width),
+                   m3=mean(Petal.Length), m4=mean(Petal.Width))
+ )
 사용자  시스템 elapsed
   0.02    0.00    0.01
```

(ddply 자체가 속도 빠른 함수)

```
> #데이터 테이블 구조일 때 속도 확인
> dt_big<-as.data.table(iris_big)
> setkey(dt_big, Species)
> system.time(dt_big[ , list(m1=mean(Sepal.Length), m2=mean(Sepal.Width),
+                      m3=mean(Petal.Length), m4=mean(Petal.Width)), by="Species"]
+ )
 사용자  시스템 elapsed
   0.01    0.00    0.02
```