7장. 데이터 처리성능 향상

plyr 패키지

https://cran.r-project.org/web/packages/plyr/plyr.pdf

plyr(플라이어) 패키지는 R의 split-apply-combine 패턴[27]을 구현하는 깨끗하고 일관된 도구 세트입니다. 이 패키지는 데이터의 분할, 함수 적용, 재조합 등에 사용하는 함수를 포함합니다. plyr 패키지의 함수는 apply(), lapply(), sapply() 함수를 대체할 수 있습니다. plyr 패키지의 함수는 분할 된 데이터 구조의 종류와 반환하는 데이터 구조의 종류에 따라 이름이 지정됩니다.

함수의 이름은 입력 데이터의 타입에 따라 x, 출력 데이터의 타입에 따라 y가 달라집니다.

```
xyply(.data, ...)
```

- *x* : 입력 데이터의 타입을 지정합니다.(a: array, l: list, d: data.frame, m: multiple inputs, r: repeat multiple times)
- *y* : 출력 데이터의 타입을 지정합니다.(a: array, l: list, d: data.frame, m: multiple inputs, r: repeat multiple times, _: nothing)

ddply, adply

다음 구문은 ddply() 함수의 구문입니다.

```
ddply(.data, .variables, .fun=NULL, ...,
      .progress="none", .inform=FALSE, .drop=TRUE,
      .parallel=FALSE, .paropts=NULL)
```

다음은 adply() 함수의 구문입니다.

```
adply(.data, .margins, .fun=NULL, ..., .expand=TRUE,
  .progress="none", .inform=FALSE, .parallel=FALSE,
  .paropts=NULL, .id=NA)
```

adply, ddply 예

다음 코드는 adply() 함수를 이용하여 데이터셋의 열 별로 함수를 적용하는 예입니다. .margins 속성의 값은 열 별로 적용을 의미하는 2입니다. 아래 구문은 Sepal.Length, Sepal.Width, Sepal.Length, Sepal.Width의 합을 출력합니다.

```
> adply(iris[, 1:4], 2, function(col) { sum(col) })
            X1    V1
1 Sepal.Length 876.5
2  Sepal.Width 458.6
3 Petal.Length 563.7
4  Petal.Width 179.9
```

데이터를 group by하여 함수를 적용할 수 있습니다. 다음 코드는 ddply() 함수를 사용하는 예입니다.

```
> ddply(iris, .(Species),          # Species로 group by
+       function(group) { data.frame(mean=mean(group$Sepal.Length)) })
     Species  mean
1     setosa 5.006
2 versicolor 5.936
3  virginica 6.588
```

reshape2 패키지

reshape2 패키지는 데이터의 구조를 변경하기 위한 함수를 제공합니다. reshape2 패키지는 reshape 패키지가 다시 작성된 것입니다. reshape2 패키지는 훨씬 더 집중적이고 훨씬 더 빠릅니다. reshape2 버전은 기능성으로 인해 속도가 향상되므로 기존 사용자에게 문제가 발생되지 않도록 reshape2로 이름을 변경했습니다. 초기 벤치마킹을 통해 melt()의 속도는 최대 10배, cast의 속도는 최대 100배, 집계하는 cast의 용도는 최대 10배 더 빨라졌습니다.

reshape2 패키지

- 칼럼 이름과 값을 variable, value 칼럼에 저장된 형태로 변환하는 함수 제공
- install.packages("reshape2")

- 5, 6번째 칼럼을 고정하고, 나머지 칼럼을 variable과 value로 변환
  - library(reshape2)
  - data <- melt(airquality, id=c(5, 6), na.rm=TRUE)

- 행과 열의 형태로 데이터 표시 (수식=행 ~ 열), data.frame 반환
  - dcast(data, Month ~ variable, sum)
  - dcast(data, Month + Day ~ variable, NULL)

- 행과 열의 형태로 데이터 표시 (수식=행 ~ 열), matrix 반환
  - acast(data, Month ~ variable, sum)
  - acast(data, Month + Day ~ variable, sum)
  - acast(data, Day ~ variable ~ Month, sum)

melt

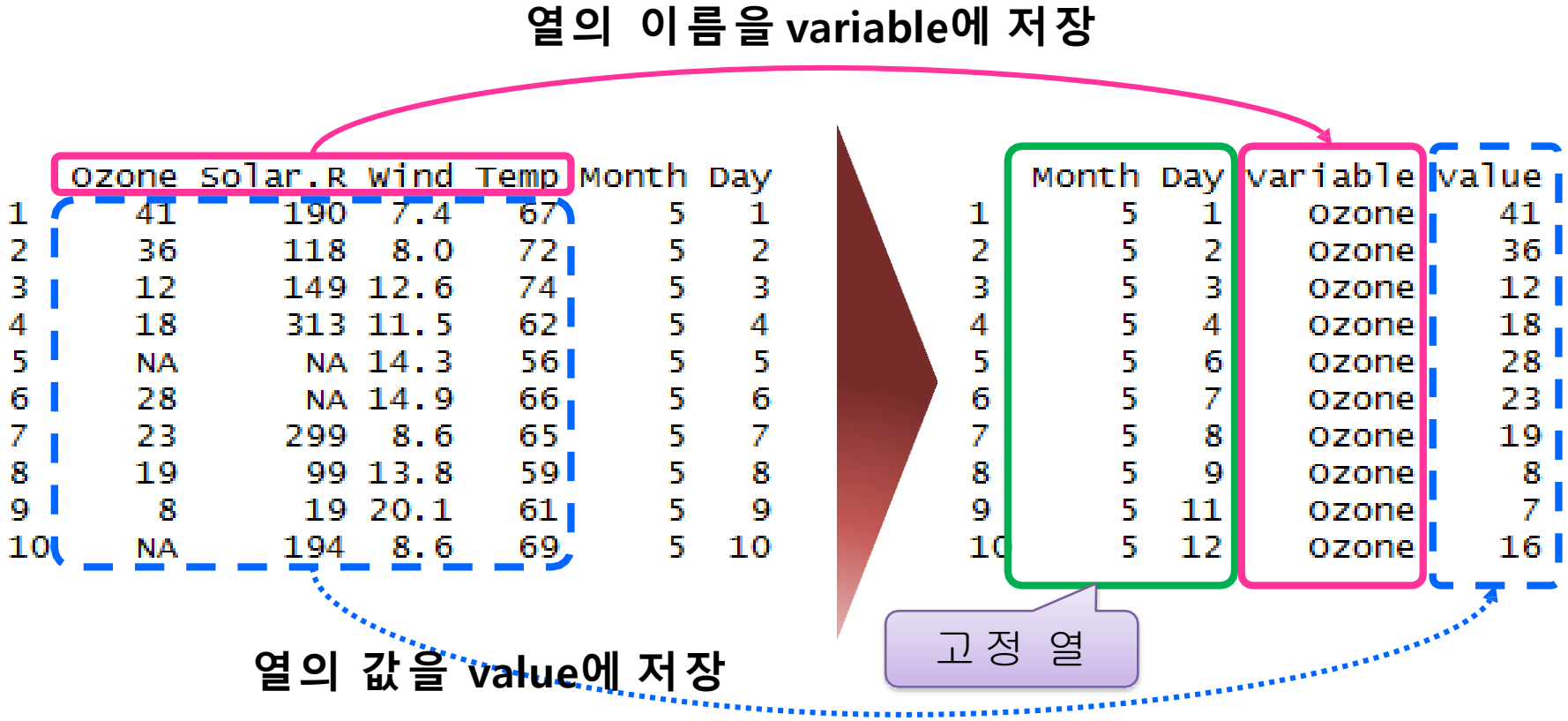
melt() 함수는 열 이름과 값을 variable, value 열에 저장된 형태로 변환하는 함수를 제공합니다. 이는 열 단위 데이터 구조를 행 단위 데이터 구조로 바꿉니다.

```
melt(data, ..., na.rm=FALSE, value.name="value")
```

구문에서...

- data : melt 할 데이터셋입니다.
- ... : 함수에 전달할 인수입니다.
- na.rm : NA 값을 데이터셋에서 삭제할지 여부를 지정합니다. FALSE(기본값)이면 NA 값도 재구조화에 사용합니다.
- value.name : 값을 저장하기 위해 사용할 변수의 이름입니다.

melt


열의 이름을 variable에 저장
Ozone Solar.R Wind Temp Month Day
1 41 190 7.4 67 5 1
2 36 118 8.0 72 5 2
3 12 149 12.6 74 5 3
4 18 313 11.5 62 5 4
5 NA NA 14.3 56 5 5
6 28 NA 14.9 66 5 6
7 23 299 8.6 65 5 7
8 19 99 13.8 59 5 8
9 8 19 20.1 61 5 9
10 NA 194 8.6 69 5 10
Month Day variable value
1 5 1 Ozone 41
2 5 2 Ozone 36
3 5 3 Ozone 12
4 5 4 Ozone 18
5 5 6 Ozone 28
6 5 7 Ozone 23
7 5 8 Ozone 19
8 5 9 Ozone 8
9 5 11 Ozone 7
10 5 12 Ozone 16
고정 열
열의 값을 value에 저장

melt 예


```
> head(airquality)
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
> airquality.melt <- melt(airquality, id=c("Month", "Day"), na.rm=TRUE)
> head(airquality.melt)
  Month Day variable value
1     5   1    Ozone    41
2     5   2    Ozone    36
3     5   3    Ozone    12
4     5   4    Ozone    18
6     5   6    Ozone    28
7     5   7    Ozone    23
```

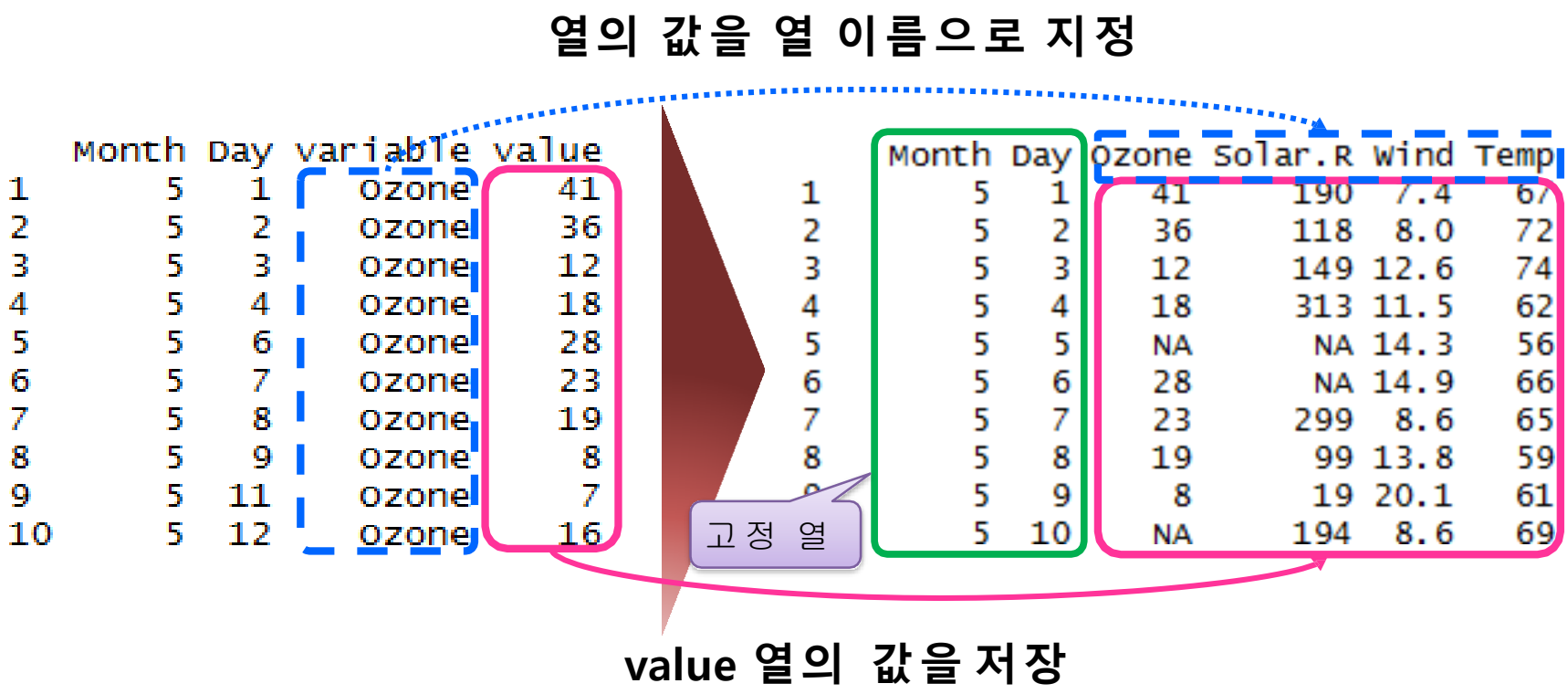
cast

cast() 함수는 molten(melt의 형용사) 데이터 프레임을 배열 또는 데이터 프레임으로 캐스팅합니다. cast() 함수는 reshape 패키지의 함수 이름입니다. reshape2 패키지에서는 dcast()와 acast() 함수로 제공합니다. dcast()는 캐스팅 한 결과가 데이터 프레임(data.frame) 타입이고, acast()는 캐스팅 한 결과의 타입이 벡터(vector)/행렬(matrix)/배열(array) 입니다.

```
dcast(data, formula, fun.aggregate=NULL, ..., margins=NULL,
      subset=NULL, fill=NULL, drop=TRUE,
      value.var=guess_value(data))
```

```
acast(data, formula, fun.aggregate=NULL, ..., margins=NULL,
      subset=NULL, fill=NULL, drop=TRUE,
      value.var=guess_value(data))
```

cast


열의 값을 열 이름으로 지정
Month Day variable value
1 5 1 Ozone 41
2 5 2 Ozone 36
3 5 3 Ozone 12
4 5 4 Ozone 18
5 5 6 Ozone 28
6 5 7 Ozone 23
7 5 8 Ozone 19
8 5 9 Ozone 8
9 5 11 Ozone 7
10 5 12 Ozone 16
고정 열
Month Day Ozone Solar.R Wind Temp
1 5 1 41 190 7.4 67
2 5 2 36 118 8.0 72
3 5 3 12 149 12.6 74
4 5 4 18 313 11.5 62
5 5 5 NA NA 14.3 56
6 5 6 28 NA 14.9 66
7 5 7 23 299 8.6 65
8 5 8 19 99 13.8 59
5 5 9 8 19 20.1 61
5 10 NA 194 8.6 69
value 열의 값을 저장

dcast 예

다음 코드는 앞에서 melt 한 데이터를 cast() 함수로 캐스팅하는 예입니다.

```
> reshape.cast <- dcast(airquality.melt,
+                       Month + Day ~ variable,
> head(reshape.cast)
  Month Day Ozone Solar.R Wind Temp
1     5   1    41     190  7.4   67
2     5   2    36     118  8.0   72
3     5   3    12     149 12.6   74
4     5   4    18     313 11.5   62
5     5   5    NA      NA 14.3   56
6     5   6    28      NA 14.9   66
```

데이터 테이블

- 데이터 테이블(data.table)은 데이터 프레임(data.frame)에서 상속
- 빠른 개발을 위해 짧고 유연한 구문을 사용하여 파일 읽기와 쓰기, 집계, 업데이트, 동등 비교, 범위 및 내부 조인 등을 빠르고 메모리 효율적으로 제공
- 데이터 테이블은 R의 기본 타입인 데이터 프레임을 대신하여 사용할 수 있는 더 빠르고 편리한 데이터 타입

데이터 테이블을 이용한 분석에서 데이터 분할, 부분 집합, 그룹, 수정, 조인 등과 같은 데이터 조작 작업은 모두 데이터 프레임의 기능을 상속 받습니다. 데이터 테이블을 이용하면 다음 장점을 가질 수 있습니다.

- 간결함과 일관성 : 최종 목표를 달성하기 위해 수행하고자하는 작업 세트와 상관없이 간결하고 일관성 있는 구문을 제공합니다.
- 유동적 : 분석을 수행하기 전에 사용할 수 있는 함수 집합에서 특정 함수로 각 작업을 매핑해야 하는 인지적 부담 없이 데이터를 유동적으로 분석을 수행합니다.
- 자동화 : 각 작업에 필요한 데이터를 정확하게 파악함으로써 매우 빠르고 효율적으로 내부적으로 작업을 자동으로 최적화합니다.

data.table

데이터 테이블은 향상된 버전의 데이터 프레임을 제공하는 R 패키지입니다. 앞의 데이터 섹션에서 fread()를 사용하여 데이터 테이블을 이미 만들었습니다. data.table()함수를 사용하여 생성 할 수도 있습니다.

```
data.table(..., keep.rownames=FALSE, check.names=FALSE,
           key=NULL, stringsAsFactors=FALSE)
```

데이터 부분집합 만들기

데이터 프레임과는 달리 데이터 테이블의 [...]에서 행을 부분 집합하고 열을 선택하는 것보다 훨씬 많은 작업을 수행 할 수 있습니다. 이를 이해하기 위해 아래와 같이 일반적인 형식의 데이터 테이블 구문을 살펴보아야합니다.

먼저 i와 j를 살펴보고 행의 하위 집합과 열에서 작업을 시작하겠습니다.

```
dt[i, j, by, keyby, WITH=TRUE,
   nomatch = getOption("datatable.nomatch"), mult="all",
   .SDcols ]
```

구문에서...

- R에서 i, j, by 는...
- SQL에서 where, select | update, group by 와 같습니다.
- 위 구문은 'dt를 사용하고, i를 사용하여 부분 집합을 만들고 j를 계산하며, by를 이용하여 그룹화합니다.'라고 할 수 있습니다.
- by, keyby, mult, .SDcols 등은 이 섹션을 통해 후반부에 설명됩니다.

i, 조건으로 부분집합 만들기

다음 코드는 flights 데이터 테이블에서 origin이 JFK이고 month가 6L인 모든 행을 얻습니다.

```
> result <- flights[origin=="JFK" & month==5L]
> head(result)
   year month day dep_time dep_delay arr_time arr_delay cancelled carrier
1: 2014     5   1     1743        43     1955         5         0      AA
2: 2014     5   1      759        -1     1057       -38         0      AA
3: 2014     5   1     1540         0     1854        14         0      AA
4: 2014     5   1     1823        78     2104        54         0      AA
5: 2014     5   1      756        -4      912         2         0      AA
6: 2014     5   1     1527        -3     1845       -15         0      AA
   tailnum flight origin dest air_time distance hour min
1:  N3ELAA     45    JFK  LAS      288     2248   17  43
2:  N789AA     59    JFK  SFO      330     2586    7  59
3:  N3JEAA     65    JFK  DFW      219     1391   15  40
4:  N3KEAA     67    JFK  SAN      308     2446   18  23
5:  N3AEAA     84    JFK  BOS       42      187    7  56
6:  N351AA     85    JFK  SFO      339     2586   15  27
```

i, 색인으로 부분집합 만들기

다음 코드는 처음 두 행의 부분집합을 생성합니다.

```
> result <- flights[1:2]
> result
   year month day dep_time dep_delay arr_time arr_delay cancelled carrier
1: 2014     1   1      914        14     1238        13         0      AA
2: 2014     1   1     1157        -3     1523        13         0      AA
   tailnum flight origin dest air_time distance hour min
1:  N338AA      1    JFK  LAX      359     2475    9  14
2:  N335AA      3    JFK  LAX      363     2475   11  57
```

i, 정렬하기

다음 코드는 flights 데이터를 origin 으로 오름차순, desc로 내림차순 정렬합니다. 이를 위해 R의 함수인 order()를 사용할 수 있습니다. 컬럼 이름에 "-"를 사용하여 내림차순으로 정렬 할 수 있습니다.

```
> result <- flights[order(origin, -dest)]
> head(result)
   year month day dep_time dep_delay arr_time arr_delay cancelled carrier
1: 2014     1   5      836         6     1151        49         0      EV
2: 2014     1   6      833         7     1111        13         0      EV
3: 2014     1   7      811        -6     1035       -13         0      EV
4: 2014     1   8      810        -7     1036       -12         0      EV
5: 2014     1   9      833        16     1055         7         0      EV
6: 2014     1  13      923        66     1154        66         0      EV
   tailnum flight origin dest air_time distance hour min
1:  N12175   4419    EWR  XNA      195     1131    8  36
2:  N24128   4419    EWR  XNA      190     1131    8  33
3:  N12142   4419    EWR  XNA      179     1131    8  11
4:  N11193   4419    EWR  XNA      184     1131    8  10
5:  N14198   4419    EWR  XNA      181     1131    8  33
6:  N12157   4419    EWR  XNA      188     1131    9  23
```

flights_df 데이터프레임에 대해 작성 제출(yisy0703@naver.com)

1. origin이 JFK이고 month가 5월인 모든 행을 resul에 얻는다
2. 처음 두 행을 resul에 얻는다
3. origin으로 오름차순, desc로 내림차순으로 정렬하여 출력
4. arr_delay열만 출력
5. year열부터 dep_time열까지 출력
6. year열과 dep_time열 출력
7. arr_delay열과 dep_delay열을 출력하되 열이름을 delay_arr과 delay_dep로 변경
8. 지연시간(arr_delay, dep_delay모두 0미만인 비행이 몇 번인지 출력
9. 6월에 출발 공항이 JFK인 모든 항공편의 도착지연 및 출발지연 시간의 평균을 계산
10. 9번의 결과에 title에 mean_arr, mean_dep로 출력
11. JFK 공항의 6월 운항 횟수
12. JFK 공항의 6월 운항 데이터 중 arr_delay열과 dep_delay열을 출력
13. JFK 공항의 6월 운항 데이터 중 arr_delay열과 dep_delay열을 제외한 모든 열 출력
14. 출발 공항(origin)별 비행 수 출력 (JFK 81483 LGA 84433 EWR 87400)
15. 항공사코드(carrier)가 AA에 대해 출발공항별 비행횟수 계산
16. origin, dest별로 비행횟수 출력
17. 항공사코드(carrier)가 AA에 대해 origin, dest별로 비행횟수 출력
18. 항공사 코드가 AA에 대해, origin, dest, 월별 평균arr_delay, 평균 dep_delay 출력
19. dep_delay>0가 참이거나 거짓, arr_delay>0가 참이거나 거짓인 각각의 비행횟수
20. Origin=="JFK"에 대해 월별 최대 출발 지연 시간 출력(month로 정렬)

https://raw.githubusercontent.com/wiki/arunsrinivasan/flights/NYCflights14/flights14.csv

j로 열 조회

다음 코드는 arr_delay 열을 조회합니다. 단일 열을 조회하는 경우 리턴하는 데이터타입은 벡터입니다.

```
> result <- flights[, arr_delay]
> head(result)
[1]  13  13   9 -26   1   0
> is.vector(result)
[1] TRUE
```

열은 데이터 테이블에서 변수 인 것처럼 참조 될 수 있으므로 우리는 하위 집합으로 만들 변수를 직접 참조합니다. 위 구문은 열에 대한 모든 행을 반환합니다.

list()와 .()

데이터 테이블은 .()을 사용하여 열을 감싸는 것을 허용합니다. .()은 list()의 별명입니다. 둘 다 같은 기능이므로 선호하는 것을 사용하세요. 이 책에서 이후부터는 list() 대신 .()을 사용할 것입니다.

```
> result <- flights[, .(arr_delay, dep_delay)]
> head(result)
   arr_delay dep_delay
1:        13        14
2:        13        -3
3:         9         2
4:       -26        -8
5:         1         2
6:         0         4
```

j, 열 이름 변경 조회

다음 코드에서처럼 목록을 만들 때 열 이름을 지정할 수 있습니다.

```
> result <- flights[, .(delay_arr = arr_delay, delay_dep = dep_delay)]
> head(result)
   delay_arr delay_dep
1:        13        14
2:        13        -3
3:         9         2
4:       -26        -8
5:         1         2
6:         0         4
```

j에서 표현식

다음 코드는 delay 〈 0 인 전체 여행이 몇 번인지 출력합니다.

```
> result <- flights[, sum((arr_delay + dep_delay) < 0)]
> result
[1] 141814
```

i와 j로 부분집합 만들기

다음 코드는 6월에 출발 공항이 "JFK"인 모든 항공편의 평균 도착 및 출발 지연을 계산합니다.

```
> result <- flights[origin == "JFK" & month == 6L,
+                   .(m_arr = mean(arr_delay), m_dep = mean(dep_delay))]
> result
      m_arr    m_dep
1: 5.839349 9.807884
```

열 선택 취소

마이너스 기호(-) 또는 Not 기호(!)를 사용하여 열의 선택을 취소 할 수도 있습니다. 다음 두 코드는 같습니다. 아래 코드를 굳이 실행 시킬 필요까지는 없을 것입니다.

```
> result <- flights[, !c("arr_delay", "dep_delay"), with=FALSE]
> result <- flights[, -c("arr_delay", "dep_delay"), with=FALSE]
```

:을 이용한 열 선택

v1.9.5+ 부터 시작 및 끝 열 이름(예 : year:day)을 지정하여 열을 선택할 수도 있습니다.

```
> result <- flights[, year:day, with=FALSE]
> head(result)
```

```
> result <- flights[, day:cancelled, with=FALSE]
> head(result)
   day dep_time dep_delay arr_time arr_delay cancelled
1:   1      914        14     1238        13         0
2:   1     1157        -3     1523        13         0
3:   1     1902         2     2224         9         0
4:   1      722        -8     1014       -26         0
5:   1     1347         2     1706         1         0
6:   1     1824         4     2145         0         0
```

by에 의한 그룹화

다음 코드는 출발 공항별 여행의 수를 출력합니다.

```
> result <- flights[, .(.N), by=.(origin)]
> result
   origin     N
1:    JFK 81483
2:    LGA 84433
3:    EWR 87400
```

위 구문은 다음 구문과 같습니다.

```
> result <- flights[, .(.N), by="origin"]
```

- .N은 현재 그룹의 행의 수를 저장하는 특수 변수입니다. 출발 공항별로 그룹화하면 각 그룹에 대해 행 수(.N)가 확보됩니다.

by 속성에 여러 열 지정

다음 코드는 항공사 코드 "AA"에 대한 각 월 별로 그리고 origin 및 dest 별로 출발 지연과 도착 지연의 평균을 계산합니다.

```
> result <- flights[carrier=="AA",
+                   .(mean(arr_delay), mean(dep_delay)),
+                   by=.(origin, dest, month)]
> result
     origin dest month         V1         V2
  1:    JFK  LAX     1   6.590361 14.2289157
  2:    LGA  PBI     1  -7.758621  0.3103448
  3:    EWR  LAX     1   1.366667  7.5000000
  4:    JFK  MIA     1  15.720670 18.7430168
  5:    JFK  SEA     1  14.357143 30.7500000
 ---
196:    LGA  MIA    10  -6.251799 -1.4208633
197:    JFK  MIA    10  -1.880184  6.6774194
198:    EWR  PHX    10  -3.032258 -4.2903226
199:    JFK  MCO    10 -10.048387 -1.6129032
200:    JFK  DCA    10  16.483871 15.5161290
```

- j에서 표현식에 대한 열 이름을 제공하지 않았고 자동으로 (V1, V2)가 생성되었습니다.

keyby 속성


```
> result <- flights[carrier=="AA",
+                   .(mean(arr_delay), mean(dep_delay)),
+                   keyby=.(origin, dest, month)]
> result
     origin dest month         V1         V2
  1:    EWR  DFW     1   6.427673 10.0125786
  2:    EWR  DFW     2  10.536765 11.3455882
  3:    EWR  DFW     3  12.865031  8.0797546
  4:    EWR  DFW     4  17.792683 12.9207317
  5:    EWR  DFW     5  18.487805 18.6829268
 ---
196:    LGA  PBI     1  -7.758621  0.3103448
197:    LGA  PBI     2  -7.865385  2.4038462
198:    LGA  PBI     3  -5.754098  3.0327869
199:    LGA  PBI     4 -13.966667 -4.7333333
200:    LGA  PBI     5 -10.357143 -6.8571429
```

- 이 코드에서 우리가 한 것은 keyby로 바꾸는 것 이었습니다. 이렇게 하면 자동으로 결과가 그룹화 변수에 따라 오름차순으로 정렬됩니다. keyby()는 연산을 수행 한 후, 즉 계산 된 결과에 적용됩니다.
- Keys : 실제로 keyby는 정렬 이상의 것을 합니다. 또한 sorted라는 속성을 설정으로 정렬하는 것 뒤에 keys를 설정합니다. 다음 절에서 keys에 대해 더 많이 배웁니다.

Chaining

항공사 코드 "AA"에 대한 각 출발지, 목적지 쌍에 대한 총 여행수를 계산하는 작업을 다시 생각해 봅시다.

```
> result <- flights[carrier=="AA", .N, by=.(origin, dest)]
```

이 결과를 origin의 오름차순으로, 그리고 dest의 내림차순으로 정렬할 수 있을까요? 우리는 결과를 변수에 저장할 수 있고, 그리고 그 변수에 order(origin, -dest)를 이용할 것입니다. 이것은 상당히 간단합니다.

```
> result <- result[order(origin, -dest)]
> head(result)
   origin dest    N
1:    EWR  PHX  121
2:    EWR  MIA  848
3:    EWR  LAX   62
4:    EWR  DFW 1618
5:    JFK  STT  229
6:    JFK  SJU  690
```

- 데이터 테이블에서 order()의 문자열에 "-"를 사용할 수 있다는 것을 기억하세요. 이것은 데이터 테이블의 내부 쿼리 최적화로 인해 가능합니다.

Chaining

표현식을 연결함으로써 변수에 이 중간 할당을 피할 수 있습니다.

```
> result <- flights[carrier=="AA", .N, by=.(origin, dest)][order(origin, -dest)]
> head(result, 10)
    origin dest    N
 1:    EWR  PHX  121
 2:    EWR  MIA  848
 3:    EWR  LAX   62
 4:    EWR  DFW 1618
 5:    JFK  STT  229
 6:    JFK  SJU  690
 7:    JFK  SFO 1312
 8:    JFK  SEA  298
 9:    JFK  SAN  299
10:    JFK  ORD  432
```

표현을 하나씩 차례로 붙이면 일련의 작업을 형성 할 수 있습니다.

```
DT [...] [...] [...]
```

또는 수직으로 연결할 수도 있습니다.

```
DT [ ...
 ] [ ...
 ] [ ...
 ]
```

by에서 표현식

by가 수식을 가질 수 있습니다.

```
> result <- flights[, .N, .(dep_delay>0, arr_delay>0)]
> result
   dep_delay arr_delay      N
1:      TRUE      TRUE  72836
2:     FALSE      TRUE  34583
3:     FALSE     FALSE 119304
4:      TRUE     FALSE  26593
```

키 사용 예

다음 코드는 첫 번째 키 열 origin이 "JFK"와 일치하고 두 번째 키 열 dest가 "MIA"와 일치하는 모든 행을 부분 집합합니다.

```
> flights[.("JFK", "MIA")]
      year month day dep_time dep_delay arr_time arr_delay cancelled
   1: 2014     1   1     1509        -1     1828       -17         0
```

다음 코드는 첫 번째 키 열 origin 만 "JFK"와 일치하는 모든 행을 부분 집합합니다.

```
> key(flights)
[1] "origin" "dest"
> flights[.("JFK")]   # flights["JFK"]
       year month day dep_time dep_delay arr_time arr_delay cancelled
    1: 2014     1   1     2011        10     2308         4         0
```

다음 코드는 두 번째 키 열 dest가 "MIA"와 일치하는 모든 행을 부분 집합합니다.

```
> flights[.(unique(origin), "MIA")]
      year month day dep_time dep_delay arr_time arr_delay cancelled
   1: 2014     1   1     1655        -5     2003       -17         0
```

by를 사용한 집계

예제를 위해 origin과 dest 열을 키로 설정합니다.

```
> setkey(flights, origin, dest)
> key(flights)
[1] "o
```

다음 코드는 origin="JFK"에 해당하는 매월 최대 출발 지연 시간을 출력합니다. 결과를 month로 정렬합니다.

```
> result <- flights["JFK", max(dep_delay), keyby=month]
> head(result)
   month   V1
1:     1  881
2:     2 1014
3:     3  920
4:     4 1241
5:     5  853
6:     6  798
> key(result)
[1] "month"
```