Utility Packages

R 기본 함수 사용

- 테스트 데이터 준비

```
> install.packages("googleVis")
> 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
```

R 기본 함수 사용

- length() 함수 → 데이터 요소 또는 줄 수 반환

```
> a <-c(1, 2, 3, 4, 5)
> length(a)
[1] 5

> 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
> length(Fruits)
[1] 7
```

R 기본 함수 사용

- 데이터 정렬 함수 → sort()

```
> sort1 <- Fruits$Sales
> sort1
[1]  98 111  89  96  85  93  94  98  81
> sort(sort1)
[1]  81  85  89  93  94  96  98  98 111
> sort(sort1, decreasing = TRUE)
[1] 111  98  98  96  94  93  89  85  81
```

R 기본 함수 사용

- aggregate(계산될컬럼~기준컬럼, 데이터, 함수) 함수
  - 특정 값을 기준으로 그룹을 만들고 지정된 함수를 적용한 결과 생성

```
> aggregate(Sales~Year, Fruits, sum)
  Year Sales
1 2008   279
2 2009   298
3 2010   268

> aggregate(Sales~Fruit, Fruits, sum)
    Fruit Sales
1  Apples   298
2 Bananas   260
3 Oranges   287

> aggregate(Sales~Fruit, Fruits, max)
    Fruit Sales
1  Apples   111
2 Bananas    94
3 Oranges    98
```

R 기본 함수 사용

- aggregate() 함수 (계속)

```
> aggregate(Sales~Fruit+Year, Fruits, max)
    Fruit Year Sales
1  Apples 2008    98
2 Bananas 2008    85
3 Oranges 2008    96
4  Apples 2009   111
5 Bananas 2009    94
6 Oranges 2009    93
7  Apples 2010    89
8 Bananas 2010    81
9 Oranges 2010    98
```

R 기본 함수 사용

- apply(데이터, 행/열, 함수)
  - 행/열 전달인자에서 1은 행, 2는 열 기준
  - 주로 행렬(Matrix) 데이터에 유용하게 사용

```
> mat1 <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2, byrow = T)
> mat1
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6

> apply(mat1, 1, sum)
[1]  6 15

> apply(mat1, 2, sum)
[1] 5 7 9

> apply(mat1[,c(2, 3)], 2, max)
[1] 5 6
```

R 기본 함수 사용

- lapply(데이터, 함수) → 결과를 리스트 형식으로 반환

```
> list1 <- list(Fruits$Sales)

> list2 <- list(Fruits$Profit)

> lapply(c(list1, list2), max)
[[1]]
[1] 111

[[2]]
[1] 32

> lapply(subset(Fruits, select = c(Sales, Profit)), max)
$`Sales`
[1] 111

$Profit
[1] 32
```

R 기본 함수 사용

- sapply(데이터, 함수) → 결과를 벡터 또는 행렬 형식으로 반환

```
> sapply(c(list1, list2), max)
[1] 111  32
>
> lapply(Fruits[, c(4,5)], max)
$Sales
[1] 111

$Expenses
[1] 91

> sapply(Fruits[, c(4,5)], max)
   Sales Expenses
     111       91
```

R 기본 함수 사용

- tapply(출력컬럼, 기준컬럼, 함수) → 그룹별 집계 처리

```
> 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

> attach(Fruits)
> tapply(Sales, Fruit, sum)
 Apples Bananas Oranges
    298     260     287
> tapply(Sales, Year, sum)
2008 2009 2010
 279  298  268
```

R 기본 함수 사용

- mapply(함수, 벡터1, 벡터2, 벡터3, …)
  - 여러 벡터에서 같은 인덱스에 있는 값끼리 모아 함수 적용
  - 벡터의 요소의 개수는 같아야 한다

```
> vec1 <- c(1, 2, 3, 4, 5)
> vec2 <- c(10, 20, 30, 40, 50)
> vec3 <- c(100, 200, 300, 400, 500)
>
> mapply(sum, vec1, vec2, vec3)
```

- sweep(데이터1, 행/열, 데이터2)
  - 데이터1의 각 행/열 값에서 데이터2의 행 열 값을 빼는 함수
  - 데이터1의 행/열과 데이터2의 행/열의 크기는 같아야 한다

```
> mat1
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
> a <- c(1,1,1); a;
[1] 1 1 1
```

```
> sweep(mat1, 2, a)
     [,1] [,2] [,3]
[1,]    0    1    2
[2,]    3    4    5
```

stringr 패키지

- 효과적인 문자열 처리 기능 제공

- 설치

```
install.packages("stringr")

library(stringr)
```

stringr 패키지

- str_detect() 함수 : 문자열의 특정 문자 포함 여부 반환

```
> fruits <- c("apple", "Apple", "banana", "pineapple")

> str_detect(fruits, "A")
[1] FALSE  TRUE FALSE FALSE

> str_detect(fruits, "a")
[1]  TRUE FALSE  TRUE  TRUE

> str_detect(fruits, "^a")
[1]  TRUE FALSE FALSE FALSE

> str_detect(fruits, "e$")
[1]  TRUE  TRUE FALSE  TRUE
```

stringr 패키지

- str_detect() 함수 : 문자열의 특정 문자 포함 여부 반환 (계속)

```
> str_detect(fruits, "^[aA]")
[1]  TRUE  TRUE FALSE FALSE

> str_detect(fruits, "[aA]")
[1] TRUE TRUE TRUE TRUE

> str_detect(fruits, regex('a', ignore_case = TRUE)) # 대소문자 구분 X
[1] TRUE TRUE TRUE TRUE
```

- str_count() 함수 : 주어진 단어에서 지정된 문자가 발생하는 횟수 반환

```
> fruits
[1] "apple"     "Apple"     "banana"    "pineapple"

> str_count(fruits, regex('a', ignore_case = TRUE))
[1] 1 1 3 1

> str_count(fruits, 'a')
[1] 1 0 3 1
```

유용한 R 함수

- str_c() 함수 : 전달인자의 각 데이터를 결합해서 하나의 문자열 반환

```
> str_c("apple", "banana")
[1] "applebanana"
> str_c("Fruits : ", fruits)
[1] "Fruits : apple"     "Fruits : Apple"     "Fruits : banana"
"Fruits : pineapple"
> str_c(fruits, " name is ", fruits)
[1] "apple name is apple"          "Apple name is Apple"
"banana name is banana"       "pineapple name is pineapple"
> str_c(fruits, collapse = "")
[1] "appleApplebananapineapple“
> str_c(fruits, collapse = "-")
[1] "apple-Apple-banana-pineapple"
```

- str_dup() 함수 : 주어진 문자열을 주어진 횟수 만큼 반복해서 출력

```
> str_dup(fruits, 3)
[1] "appleappleapple"             "AppleAppleApple"
"bananabananabanana"          "pineapplepineapplepineapple"
```

유용한 R 함수

- str_length() 함수 : 주어진 문자열이 길이 반환

```
> str_length('apple')
[1] 5
> str_length(fruits)
[1] 5 5 6 9
```

- str_locate() 함수 : 주어진 문자열에서 특정 문자가 처음 나오는 위치와 마지막 나오는 위치 반환

```
> str_locate('apple', 'a')
     start end
[1,]     1   1
> str_locate(fruits, 'a')
     start end
[1,]     1   1
[2,]    NA  NA
[3,]     2   2
[4,]     5   5
```

정규 표현식

- 문자열의 형식을 검사하기 위해 사용하는 표기법
- 주요 정규표현식

표현식	설명	표현식	설명
\\d	Digit, 0, 1, 2, …, 9	.	New Line을 제외한 모든 문자
\\D	숫자가 아닌 것	[ab]	a 또는 b
\\s	공백	[^ab]	a, b를 제외한 모든 문자
\\S	공백이 아닌 것	[0-9]	모든 숫자
\\w	단어 문자	[A-Z]	영어 대문자
\\W	단어 문자가 아닌 것	[a-z]	영어 소문자
\\t	Tab	[A-z]	영어 대/소문자
\\n	New Line (개행)	i+	i가 1회 이상 발생
^	시작 표시	i*	i가 0회 이상 발생
$	끝 표시	i?	i가 0회 또는 1회 발생
\	탈출 문자	i{n}	i가 연속적으로 n회 발생
|	두 개 이상 조건 결합	i{n1,n2}	i가 n1회에서 n2회 발생
		i{n,}	i가 n회 이상 발생

정규 표현식

- 주요 정규표현식 (계속)

표현식	설명
[:alnum:]	문자와 숫자 [:alpha:] and [:digit:]
[:alpha:]	문자 [:lower:] and [:upper:]
[:blank:]	공백 : space, tab
[:cntrl:]	제어문자
[:digit:]	Digits : 0 ~ 9
[:graph:]	Graphical Character → [:alnum:] and [:punct:]
[:lower:]	소문자
[:print:]	숫자,문자,특수문자,공백 → [:alnum:],[:punct:],[:space]
[:punct:]	특수 문자 : !, “, #, $, %, &, ‘, …
[:space:]	공백문자 : tab, newline, vertical tab, space, …
[:upper:]	대문자
[:xdigit:]	16진수 : 0 ~ F

정규 표현식

- 사용 사례 – grep 함수를 사용해서 패턴 검출

```
> char <- c('apple', 'Apple', 'APPLE', 'banana', 'grape')
> grep('apple', char)
[1] 1

> grep('pp', char) # 데이터 위치 출력
[1] 1 2
> grep('pp', char, value = T) # 값으로 출력
[1] "apple" "Apple"
```

정규 표현식

- 사용 사례 – grep 함수를 사용해서 패턴 검출

```
> grep('^A', char, value = T) #A로 시작하는 데이터
[1] "Apple" "APPLE"

> grep('e$', char, value = T) #e로 끝나는 데이터
[1] "apple" "Apple" "grape"

> char2 <- c('grape1', 'apple1', 'apple', 'orange', 'Apple')
> grep('ap', char2, value = T) #ap가 포함된 데이터
[1] "grape1" "apple1" "apple"

> grep('[1-9]', char2, value = T) # 1 ~ 9 숫자 포함 데이터
[1] "grape1" "apple1"

> grep('[[:upper:]]', char2, value = T) #대문자가 포함된 데이터
[1] "Apple"
```

plyr 패키지

- 원본 데이터를 분석하기 쉬운 형태로 나누고 다시 새로운 형태로 변경

- apply() 함수 확장

- 함수명 규칙 : 원본자료형(d,l,y) + 반환자료형(d,l,y) + ply

	data frame	**list**	**array**
data frame	ddply()	ldply()	adply()
list	dlply()	llply()	alply()
array	daply()	laply()	aaply()

- 변환 방식
  - summarise → 기준 컬럼으로 집계
  - transform → 행 단위 변환 처리

plyr 패키지

- plyr::adply(데이터, 행/열 방향, 적용함수)
  - 배열을 받아 데이터프레임 반환 → 배열은 숫자 색인이 가능한 모든 형식

```
iris
adply(iris,
      1,
      function(row) {
        row$Sepal.Length >= 0.5 &
        row$Species == "setosa"
      })
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species	Species2	V1
1	5.1	3.5	1.4	0.2	setosa	1	TRUE
2	4.9	3.0	1.4	0.2	setosa	2	TRUE
3	4.7	3.2	1.3	0.2	setosa	1	TRUE
4	4.6	3.1	1.5	0.2	setosa	2	TRUE
5	5.0	3.6	1.4	0.2	setosa	1	TRUE

```
#직접 데이터 프레임 반환 (컬럼명 지정)
adply(iris,
      1,
      function(row) {
        data.frame(sepal_ge_5_setosa = c(row$Sepal.Length >= 0.5 &
                                         row$Species == "setosa"))
      })
```

plyr 패키지

- plyr::ddply(데이터, 기준컬럼, 적용함수 또는 결과 목록)

```
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
```

```
ddply(iris,
      .(Species, Sepal.Length > 5.0),
      function(sub) {
        data.frame(sepal.width.mean = mean(sub$Sepal.Width))
      })
```

```
     Species Sepal.Length > 5 sepal.width.mean
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
```

plyr 패키지

- plyr::ddply(데이터, 기준컬럼, 적용함수 또는 결과 목록)

```
> fruits <- read.csv("data-files/fruits-10.csv", header = T); fruits;
   year   name qty price
1  2000  apple   6  6000
2  2000 banana   2  1000
3  2000  peach   7  3500
4  2000  berry   9   900
5  2001  apple  10 10000
6  2001 banana   7  3500
7  2001  peach   3  1500
… (이하 생략)

> ddply(fruits, 'name', summarise,
        sum_qty = sum(qty), sum_price = sum(price))
    name sum_qty sum_price
1  apple      29     29000
2 banana      19      9500
3  berry      35      3500
4  peach      15      7500
```

plyr 패키지

- plyr::ddply(데이터, 기준컬럼, 적용함수 또는 결과 목록)

```
> ddply(fruits, 'name', summarise,
        max_qty = max(qty), min_price = min(price))
    name max_qty min_price
1  apple      13      6000
2 banana      10      1000
3  berry      15       900
4  peach       7      1500

> ddply(fruits, 'name', transform, sum_qty = sum(qty),
        pct_qty = (100 * qty) / sum(qty))
   year   name qty price sum_qty  pct_qty
1  2000  apple   6  6000      29 20.68966
2  2001  apple  10 10000      29 34.48276
3  2002  apple  13 13000      29 44.82759
4  2000 banana   2  1000      19 10.52632
5  2001 banana   7  3500      19 36.84211
6  2002 banana  10  5000      19 52.63158
7  2000  berry   9   900      35 25.71429
8  2001  berry  15  1500      35 42.85714
9  2002  berry  11  1100      35 31.42857
10 2000  peach   7  3500      15 46.66667
11 2001  peach   3  1500      15 20.00000
12 2002  peach   5  2500      15 33.33333
```

reshape2 패키지

- 설치

```
> install.packages("reshape2")
> library(reshape2)
```

- 데이터의 모양(구조)을 변경하는 데 사용하는 함수 제공

- 제공하는 변환은 크게 melt와 cast 이며 이 두가지 변환을 사용해 데이터의 모양을 변경하거나 요약할 수 있다.

- cast는 결과로 얻고자하는 데이터 타입에 따라 dcast(), acast()로 구분
  - dcast()는 결과로 데이터프레임 반환, acast()는 결과로 벡터, 행렬, 배열 반환

- cast는 데이터를 요약하는 기능도 제공
  - 데이터를 원래 차원으로 복원시키는 대신 더 작은 차원으로 변환하는 formula 지정

reshape2 패키지

- melt()
  - melt() 함수는 인자로 데이터를 구분하는 식별자, 측정 대상 변수, 측정치를 받아 데이터를 간략하게 표현
  - 측정 변수는 variable, 측정치는 value에 저장

```
library(reshape2)

str(french_fries)
head(french_fries, 10)
m <- melt(id = 1:4, french_fries); head(m, 100);

smiths
melt(id = 1:2, smiths)
melt(id = 1:2, smiths, na.rm = T) #NA행 제거

french_fries[!complete.cases(french_fries), ] #해당 행의 값 중 NA가 포함되지 않은 행 반환

m <- melt(id = 1:4, french_fries, na.rm = T); nrow(m);
m <- melt(id = 1:4, french_fries, na.rm = F); nrow(m);
```

reshape2 패키지

- dcast(데이터, formula, 집계/요약함수)
  - formula는 행 변수 ~ 열 변수 형태로 작성
  - formula에 나열되지 않은 모든 변수를 표현하려면 …를 작성

```
smiths
(m <- melt(id = 1:2, smiths))
(x <- dcast(m, subject + time ~ ...))

identical(x, smiths)

dcast(melt(id = 1:2, smiths, na.rm = T), subject + time ~ ...)

head(french_fries)
ffm <- melt(id = 1 : 4, french_fries)
head(ffm)

x <- dcast(ffm, time + treatment + subject + rep ~ variable)
head(x)

rownames(french_fries) <- NULL
rownames(x) <- NULL
identical(french_fries, x)
```

reshape2 패키지

- dcast(데이터, formula, 집계/요약함수)
  - 요약 기능 제공

```
ffm <- melt(id = 1:4, french_fries)
head(ffm, 100)

dcast(ffm, time ~ variable) # default 집계함수는 length
dcast(ffm, time ~ variable, length)

dcast(ffm, time ~ variable, mean)

dcast(melt(id = 1:4, french_fries, na.rm = T), time ~ variable, mean)

dcast(melt(id = 1:4, french_fries), time ~ variable, mean, na.rm = T)

dcast(ffm, treatment ~ rep + variable, mean, na.rm = T)
```

doBy 패키지

- 특정 변수를 기준으로 그룹화하고 집계와 같은 요약 정보를 생산하는 기능

- 설치

```
> install.packages("doBy")
> library(doBy)
```

- 주요 함수

함수명	설명
summaryBy	지정된 컬럼의 값에 따라 그룹화하고 요약 정보 생산
orderBy	지정된 컬럼 값에 따라 데이터프레임 정렬

doBy 패키지

- summaryBy(formula, 데이터프레임)
  - Formula에 지정된 컬럼을 기준으로 그룹화 및 집계 처리

```
> summaryBy(Sepal.Width + Sepal.Length ~ Species, iris)

     Species Sepal.Width.mean Sepal.Length.mean
1     setosa            3.428             5.006
2 versicolor            2.770             5.936
3  virginica            2.974             6.588

> summaryBy(Sepal.Width + Sepal.Length ~ Species, iris, FUN = sd)

     Species Sepal.Width.sd Sepal.Length.sd
1     setosa      0.3790644       0.3524897
2 versicolor      0.3137983       0.5161711
3  virginica      0.3224966       0.6358796
```

doBy 패키지

- orderBy(formula, 데이터프레임)
  - Formula에 지정된 컬럼을 기준으로 데이터프레임 정렬
  - Formula에 -를 지정하면 내림차순, +를 지정하면 오름차순

```
> orderBy(~ Species + Sepal.Width, iris)
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
42           4.5         2.3          1.3         0.3     setosa
9            4.4         2.9          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa

> orderBy(~ -Species + Sepal.Width, iris)
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
120          6.0         2.2          5.0         1.5  virginica
107          4.9         2.5          4.5         1.7  virginica
109          6.7         2.5          5.8         1.8  virginica

> orderBy(~ -Species - Sepal.Width, iris)
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
118          7.7         3.8          6.7         2.2  virginica
132          7.9         3.8          6.4         2.0  virginica
110          7.2         3.6          6.1         2.5  virginica
```

sqldf 패키지

- 데이터를 조회할 때 RDBMS에서 사용하는 SQL 문장을 사용하도록 지원
- 설치

```
> install.packages("sqldf")
> library(sqldf)
```

- 조회 대상 데이터 구성

```
>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
```

sqldf 패키지

- 단순 데이터 조회

```
> sqldf('select * from 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
```

- Where 절로 조건 사용

```
> sqldf('select * from Fruits where Fruit = \'Apples\' ')
   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
```

sqldf 패키지

- 출력되는 행 수 제어

```
> sqldf('select * from Fruits limit 3')
   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
```

- 정렬 적용

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

sqldf 패키지

- 집계 함수 사용

```
> sqldf('select sum(Sales) from Fruits')
  sum(Sales)
1        845
> sqldf('select max(Sales) from Fruits')
  max(Sales)
1        111
> sqldf('select min(Sales) from Fruits')
  min(Sales)
1         81
> sqldf('select avg(Sales) from Fruits')
  avg(Sales)
1   93.88889
> sqldf('select Fruit, avg(Sales) from Fruits group by Fruit')
    Fruit avg(Sales)
1  Apples   99.33333
2 Bananas   86.66667
3 Oranges   95.66667
> sqldf('select count(*) from Fruits')
  count(*)
1        9
```

sqldf 패키지

- 서브쿼리 사용

```
> sqldf('select * from Fruits where sales > (select sales from fruits
where expenses = 78)')
   Fruit Year Location Sales Expenses Profit       Date
1 Apples 2009     West   111       79     32 2009-12-31

> sqldf('select * from Fruits where sales in (select sales from Fruits
where sales > 95)')
    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 Oranges 2008     East    96       81     15 2008-12-31
4 Oranges 2010     East    98       91      7 2010-12-31
```

sqldf 패키지

- Join 사용

```
salaries = read.csv("data-files/salaries.csv")

teams = read.csv("data-files/teams.csv")

salaries.teams = sqldf("SELECT a.*, b.lgID, b.name FROM salaries a
INNER JOIN teams b ON(a.yearID = b.yearID AND a.teamID = b.teamID)")

salaries.teams

 yearID teamID lgID  playerID   salary lgID                    name
1       1985    BAL   AL murraed02  1472819   AL          Baltimore Orioles
2       1985    BAL   AL  lynnfr01  1090000   AL          Baltimore Orioles
3       1985    BAL   AL ripkeca01   800000   AL          Baltimore Orioles
4       1985    BAL   AL  lacyle01   725000   AL          Baltimore Orioles
5       1985    BAL   AL flanami01   641667   AL          Baltimore Orioles
6       1985    BAL   AL boddimi01   625000   AL          Baltimore Orioles
7       1985    BAL   AL stewasa01   581250   AL          Baltimore Orioles
```

sqldf 패키지

- Update 사용

```
> sqldf(c("update Fruits set profit = 25 where Fruit = 'Apples' and
year = 2008", 'select * from Fruits'))

    Fruit Year Location Sales Expenses Profit       Date
1  Apples 2008     West    98       78     25 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
```

sqldf 패키지

- Delete 사용

```
> sqldf(c('delete from Fruits where fruit = \'Apples\' and year =
2008', 'select * from Fruits'))

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