In [1]:
```
library(data.table) # version 1.13.0

DT <- data.table(Fruit = rep(c("banana", "apple", "orange"), 3:1),
                 Year  = c(2008, 2009, 2010, 2009, 2010, 2010),
                 Count = 1:6)
DT
```

A data.table: 6 × 3

Fruit	Year	Count
<chr>	<dbl>	<int>
banana	2008	1
banana	2009	2
banana	2010	3
apple	2009	4
apple	2010	5
orange	2010	6

DT[i, j]

In [2]:
```
# operations on rows
DT[Fruit == "banana", ]
```

A data.table: 3 × 3

Fruit	Year	Count
<chr>	<dbl>	<int>
banana	2008	1
banana	2009	2
banana	2010	3

In [3]:
```
DT[Fruit == "banana" & Year > 2008] # 不加逗号，默认为行
```

A data.table: 2 × 3

Fruit	Year	Count
<chr>	<dbl>	<int>
banana	2009	2
banana	2010	3

In [4]:
```
DT[order(Fruit), ] #按指定列排序a a
```

A data.table: 6 × 3

Fruit <chr>	Year <dbl>	Count <int>
apple	2009	4
apple	2010	5
banana	2008	1
banana	2009	2
banana	2010	3
orange	2010	6

In [5]:
```
DT[order(Fruit, -Year)]
```

A data.table: 6 × 3

Fruit <chr>	Year <dbl>	Count <int>
apple	2010	5
apple	2009	4
banana	2010	3
banana	2009	2
banana	2008	1
orange	2010	6

In [6]:
```
DT[sample(.N, 3), ]
```

A data.table: 3 × 3

Fruit <chr>	Year <dbl>	Count <int>
apple	2010	5
banana	2010	3
banana	2009	2

In [7]:
```
# operations on columns
DT[, Count]
```

1 · 2 · 3 · 4 · 5 · 6

In [8]:
```
DT[, list(Count)]
```

A data.table: 6 × 1

Count <int>
1
2
3
4
5
6

In [9]:
```
DT[, .(Count)]
```

A data.table: 6 × 1

Count <int>
1
2
3
4
5
6

In [10]:
```
DT[, .(Fruit, Count)]
```

A data.table: 6 × 2

Fruit <chr>	Count <int>
banana	1
banana	2
banana	3
apple	4
apple	5
orange	6

In [11]:
```
cols <- c("Fruit", "Year")
DT[, ..cols]
```

A data.table: 6 × 2

Fruit	Year
<chr>	<dbl>
banana	2008
banana	2009
banana	2010
apple	2009
apple	2010
orange	2010

In [12]:
```
DT[, cumsum(Count)]
```

1 · 3 · 6 · 10 · 15 · 21

In [13]:
```
DT[, .(cumsum(Count))]
```

A data.table: 6 × 1

V1
<int>
1
3
6
10
15
21

In [14]:
```
DT[, .(CumsumCount = cumsum(Count))]
```

A data.table: 6 × 1

CumsumCount
<int>
1
3
6
10
15
21

In [15]:
```
DT[, .(sum(Count), max(Year))]
```

A data.table: 1 × 2

V1	V2
<int>	<dbl>
21	2010

In [16]:
```
DT[, .(SUM = sum(Count),
       MAX = max(Year))]
```

A data.table: 1 × 2

SUM	MAX
<int>	<dbl>
21	2010

modifying

In [20]:
```
# operations on columns (modifying the data.table)
DT[, Cumsum_Count := cumsum(Count)]
DT
```

A data.table: 6 × 6

Fruit	Year	Count	Cumsum_Count	CountX3	CountX4
<chr>	<dbl>	<int>	<int>	<dbl>	<dbl>
banana	2008	1	1	3	4
banana	2009	2	3	6	8
banana	2010	3	6	9	12
apple	2009	4	10	12	16
apple	2010	5	15	15	20
orange	2010	6	21	18	24

In [21]:
```
DT[, ':='(CountX3 = Count * 3,
          CountX4 = Count * 4)]
DT
```

A data.table: 6 × 6

Fruit	Year	Count	Cumsum_Count	CountX3	CountX4
<chr>	<dbl>	<int>	<int>	<dbl>	<dbl>
banana	2008	1	1	3	4
banana	2009	2	3	6	8
banana	2010	3	6	9	12
apple	2009	4	10	12	16
apple	2010	5	15	15	20
orange	2010	6	21	18	24

In [23]:
```
cols <- c("CountX3", "CountX4")

DT[, (cols) := .(Count * 3, Count * 4)]
DT
```

A data.table: 6 × 6

Fruit	Year	Count	Cumsum_Count	CountX3	CountX4
<chr>	<dbl>	<int>	<int>	<dbl>	<dbl>
banana	2008	1	1	3	4
banana	2009	2	3	6	8
banana	2010	3	6	9	12
apple	2009	4	10	12	16
apple	2010	5	15	15	20
orange	2010	6	21	18	24

In [26]:
```
DT[, Cumsum_Count := NULL]
DT
```

A data.table: 6 × 5

Fruit	Year	Count	CountX3	CountX4
<chr>	<dbl>	<int>	<dbl>	<dbl>
banana	2008	1	3	4
banana	2009	2	6	8
banana	2010	3	9	12
apple	2009	4	12	16
apple	2010	5	15	20
orange	2010	6	18	24

In [27]:
```
# operations on both rows and columns
DT[Fruit != "apple", sum(Count)]
```

12

In [28]:
```
DT[Fruit == "banana" & Year < 2011, .(sum(Count))]
```

A data.table: 1 × 1

V1
<int>
6

In [29]:
```
DT[Fruit == "banana" & Year < 2010, Count := Count + 1]
```

In [31]:
```
DT[Fruit == "orange", Orange := "orange"]
DT
```

A data.table: 6 × 6

Fruit	Year	Count	CountX3	CountX4	Orange
<chr>	<dbl>	<int>	<dbl>	<dbl>	<chr>
banana	2008	2	3	4	NA
banana	2009	3	6	8	NA
banana	2010	3	9	12	NA
apple	2009	4	12	16	NA
apple	2010	5	15	20	NA
orange	2010	6	18	24	orange

DT[i, j, by]

In [32]:
```
# aggregation by group
DT[, sum(Count), by = Fruit]
```

A data.table: 3 × 2

Fruit	V1
<chr>	<int>
banana	8
apple	9
orange	6

In [33]:
```
DT[, sum(Count), by = (IsApple = Fruit == "apple")]
```

A data.table: 2 × 2

IsApple	V1
<lgl>	<int>
FALSE	14
TRUE	9

In [34]:
```
DT[, sum(Count), by = c("Fruit", "Year")]
```

A data.table: 6 × 3

Fruit	Year	V1
<chr>	<dbl>	<int>
banana	2008	2
banana	2009	3
banana	2010	3
apple	2009	4
apple	2010	5
orange	2010	6

In [35]:
```
DT[, .(SumCount = sum(Count)), by = .(Fruit, Before2011 = Year < 2011)]
```

A data.table: 3 × 3

Fruit <chr>	Before2011 <lgl>	SumCount <int>
banana	TRUE	8
apple	TRUE	9
orange	TRUE	6

In [36]:
```
DT[Fruit != "orange",
   max(Count),
   by = Fruit]
```

A data.table: 2 × 2

Fruit <chr>	V1 <int>
banana	3
apple	5

In [38]:
```
DT[, N := .N, by = Fruit]
DT
```

A data.table: 6 × 7

Fruit <chr>	Year <dbl>	Count <int>	CountX3 <dbl>	CountX4 <dbl>	Orange <chr>	N <int>
banana	2008	2	3	4	NA	3
banana	2009	3	6	8	NA	3
banana	2010	3	9	12	NA	3
apple	2009	4	12	16	NA	2
apple	2010	5	15	20	NA	2
orange	2010	6	18	24	orange	1

In [40]:
```
DT[, MeanCountByFruit := round(mean(Count), 2), by = Fruit]
DT
```

A data.table: 6 × 8

Fruit	Year	Count	CountX3	CountX4	Orange	N	MeanCountByFruit
<chr>	<dbl>	<int>	<dbl>	<dbl>	<chr>	<int>	<dbl>
banana	2008	2	3	4	NA	3	2.67
banana	2009	3	6	8	NA	3	2.67
banana	2010	3	9	12	NA	3	2.67
apple	2009	4	12	16	NA	2	4.50
apple	2010	5	15	20	NA	2	4.50
orange	2010	6	18	24	orange	1	6.00

chaining

In [41]:
```
# chaining
DT[, MeanCountByFruit := round(mean(Count), 2), by = Fruit][MeanCountByFruit > 2
```

A data.table: 6 × 8

Fruit	Year	Count	CountX3	CountX4	Orange	N	MeanCountByFruit
<chr>	<dbl>	<int>	<dbl>	<dbl>	<chr>	<int>	<dbl>
banana	2008	2	3	4	NA	3	2.67
banana	2009	3	6	8	NA	3	2.67
banana	2010	3	9	12	NA	3	2.67
apple	2009	4	12	16	NA	2	4.50
apple	2010	5	15	20	NA	2	4.50
orange	2010	6	18	24	orange	1	6.00

In [42]:
```
DT[, c("Orange", "N", "MeanCountByFruit") := NULL][]
```

A data.table: 6 × 5

Fruit	Year	Count	CountX3	CountX4
<chr>	<dbl>	<int>	<dbl>	<dbl>
banana	2008	2	3	4
banana	2009	3	6	8
banana	2010	3	9	12
apple	2009	4	12	16
apple	2010	5	15	20
orange	2010	6	18	24

More details about DT[, j]

❗ the j element can be any arbitrary expression, or set of expressions written within curly braces. For example:
DT[, 1 + 1] (pdf)

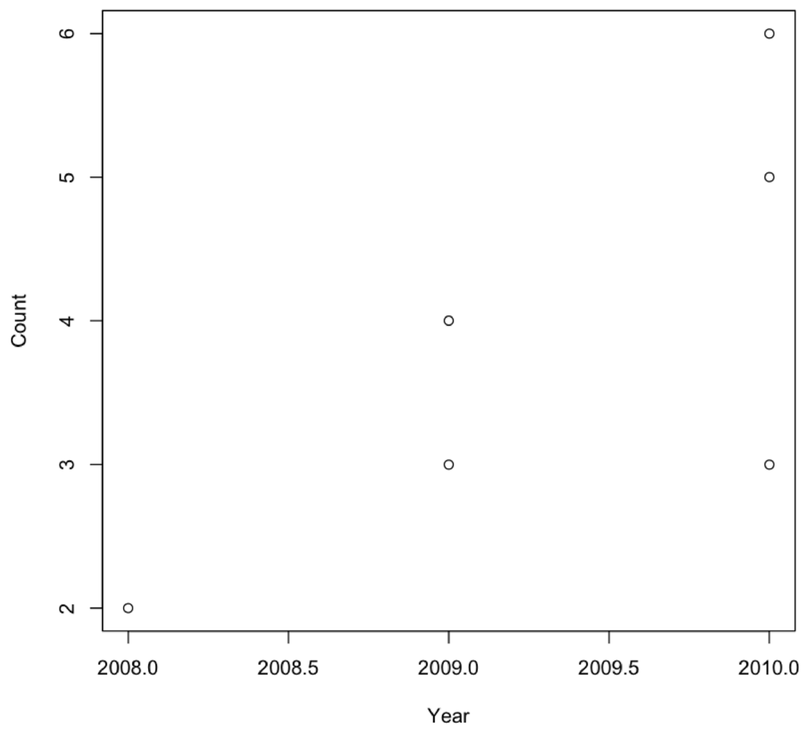
j元素可以是任意表达式，也可以是用花括号写的一组表达式。例如：

In [44]:
```
DT[,1+1]
```

2

In [45]:
```
DT[, plot(Year, Count)]
```

```
NULL
```



In [46]:
```
DT[, {sum_count <- sum(Count)
      print("The sum of the Count column is:")
      sum_count}]
```

```
[1] "The sum of the Count column is:"
```

23

请注意，在大括号中传递几个表达式是有效的基本R代码，用于计算几个命令，但只返回最后一个结果:

In [47]:
```
{sum123 <- sum(1:3);1 + 2;sum123}
```

6

只要j表达式返回长度相等的元素列表(或长度为1的元素)，列表中的每个元素将被转换为结果data.table中的一列。这很重要!记住这一点，我们将在下一节看到其含义。但请注意，这也解释了为什么我们在前面使用list()别名.()来对列进行操作。

In [48]:
```
DT[, list(1:3, 4:6, 7)]
```

A data.table: 3 × 3

V1	V2	V3
<int>	<int>	<dbl>
1	4	7
2	5	7
3	6	7

In [49]:
```
DT[, {2 + 3             # this command is evaluated but not returned
      list(Col1 = 1:3,
           Col2 = 4:6,
           Col3 = 7)}]
```

A data.table: 3 × 3

Col1	Col2	Col3
<int>	<int>	<dbl>
1	4	7
2	5	7
3	6	7

In [50]:
```
DT[, print(.SD), by = Fruit] #. sd对应于"当前组的当前数据(不包括分组变量)"
```

```
   Year Count CountX3 CountX4
1: 2008     2       3       4
2: 2009     3       6       8
3: 2010     3       9      12
   Year Count CountX3 CountX4
1: 2009     4      12      16
2: 2010     5      15      20
   Year Count CountX3 CountX4
1: 2010     6      18      24
```

A data.table: 0 × 1

Fruit
<chr>

In [51]:
```
# If there is no by, then .SD is DT itself.
DT[, .SD]
```

A data.table: 6 × 5

Fruit <chr>	Year <dbl>	Count <int>	CountX3 <dbl>	CountX4 <dbl>
banana	2008	2	3	4
banana	2009	3	6	8
banana	2010	3	9	12
apple	2009	4	12	16
apple	2010	5	15	20
orange	2010	6	18	24

In [52]:
```
DT[, lapply(.SD, min), by = Fruit]   # lapply(.SD, min) is used as the j express
```

A data.table: 3 × 5

Fruit <chr>	Year <dbl>	Count <int>	CountX3 <dbl>	CountX4 <dbl>
banana	2008	2	3	4
apple	2009	4	12	16
orange	2010	6	18	24

In [53]:
```
DT[Fruit != "apple", lapply(.SD, min), by = Fruit]
```

A data.table: 2 × 5

Fruit <chr>	Year <dbl>	Count <int>	CountX3 <dbl>	CountX4 <dbl>
banana	2008	2	3	4
orange	2010	6	18	24

In [54]:
```
DT[, c("MeanYear", "MeanCount") := lapply(.SD, mean),
     by = Fruit]
DT
```

A data.table: 6 × 7

Fruit	Year	Count	CountX3	CountX4	MeanYear	MeanCount
<chr>	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<dbl>
banana	2008	2	3	4	2009.0	2.666667
banana	2009	3	6	8	2009.0	2.666667
banana	2010	3	9	12	2009.0	2.666667
apple	2009	4	12	16	2009.5	4.500000
apple	2010	5	15	20	2009.5	4.500000
orange	2010	6	18	24	2010.0	6.000000

In [55]:
```
# .SDcols to pass a vector of colnames
DT[, lapply(.SD, min), by = Fruit, .SDcols = c("Count", "MeanCount")]
```

A data.table: 3 × 3

Fruit	Count	MeanCount
<chr>	<int>	<dbl>
banana	2	2.666667
apple	4	4.500000
orange	6	6.000000

In [56]:
```
selected_cols <- "Year"
# indenting the code
DT[, by = Fruit,                  # for each fruit
     lapply(.SD, min),            # retrieve the min value
     .SDcols = selected_cols]     # for each column provided in the selected_cols v
```

A data.table: 3 × 2

Fruit	Year
<chr>	<dbl>
banana	2008
apple	2009
orange	2010

In [57]:
```
# regular expression can also be passed using patterns():
DT[, lapply(.SD, min),
     by = Fruit,
     .SDcols = patterns("^Co")]
```

A data.table: 3 × 4

Fruit	Count	CountX3	CountX4
<chr>	<int>	<dbl>	<dbl>
banana	2	3	4
apple	4	12	16
orange	6	18	24

In [58]:
```
DT[, lapply(.SD, min),
     by = Fruit,
     .SDcols = is.integer] # !is.integer can also be used
```

A data.table: 3 × 2

Fruit	Count
<chr>	<int>
banana	2
apple	4
orange	6

In [59]:
```
foo <- function(x) {is.numeric(x) && mean(x) > 2000}
DT[, lapply(.SD, min),
     by = Fruit,
     .SDcols = foo]
```

A data.table: 3 × 3

Fruit	Year	MeanYear
<chr>	<dbl>	<dbl>
banana	2008	2009.0
apple	2009	2009.5
orange	2010	2010.0

In [60]:
```
sessionInfo()
```

```
R version 4.3.2 (2023-10-31)
Platform: x86_64-apple-darwin20 (64-bit)
Running under: macOS Sonoma 14.0

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/lib/libRbla
s.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/lib/libRlap
ack.dylib;  LAPACK version 3.11.0

locale:
[1] zh_CN.UTF-8/zh_CN.UTF-8/zh_CN.UTF-8/C/zh_CN.UTF-8/zh_CN.UTF-8

time zone: Asia/Shanghai
tzcode source: internal

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base     

other attached packages:
[1] data.table_1.14.8 jsonlite_1.8.7   

loaded via a namespace (and not attached):
 [1] digest_0.6.33       IRdisplay_1.1       utf8_1.2.4         
 [4] base64enc_0.1-3     fastmap_1.1.1       glue_1.6.2         
 [7] htmltools_0.5.7     repr_1.1.6          lifecycle_1.0.4    
[10] cli_3.6.1           fansi_1.0.5         vctrs_0.6.4        
[13] pbdZMQ_0.3-10       compiler_4.3.2      tools_4.3.2        
[16] evaluate_0.23       pillar_1.9.0        crayon_1.5.2       
[19] rlang_1.1.2         IRkernel_1.3.2.9000 uuid_1.1-1         
```

```
In [ ]:
```