Post02-ruoming-dong.Rmd

R Data Processing|data.table package

Ruoming Dong

November.30.2017

Introduction

> There are two efficient data processing packages for R, which are dplyr and data.table. These two packages have their respective advantages. Dplyr package provides a natural language which is easier for people to understand, and we often use it in our homework. Data storage and data queries can be easily used with data.frame. It is convenient to use when the data is not too large. However, using the data.frame structure to process data is not very efficient, especially when the data size is slightly larger. However, the syntax of the data.table package is simple. Also it can accomplish a lot of things with a single code.

> **5 functions in dplyr package can effectively perform data processing. As follow:**
>
> - **select**
> - **filter**
> - **arrange**
> - **mutate**
> - **summarise**
>
> **data.table package provides concise generic format:**
>
> - **DT[i, j, by]**
> - **DT means dataset**
> - **i means select a subset**
> - **j is calculated by grouping "by"**

The difference between data.table and data.frame

> 1). In the data.table, the character types will not be converted to the factors. So, people do not have to add "stringAsFactors = False" in the statement.

> 2). When printing data, a colon is added to the line number: the area branch number and the first column data

> 3). When the number of data rows exceeds the options() parameter setting datatable.print.Nrows value (the default value is 100), it will only be the first 5 rows and the last 5 rows of the dataset are printed.

Mention the difference between the symbol square brackets in the data.table and in the basic funtion(data.frame)

> 1).Beause of the same symbol square brackets, the type of the dataset will be converted to data.table. Otherwise the symbol square brackets will be used as data.frame.

> 2).To take a value in dft[1,2], the result in dft is still in data.table dataset, while data.frame is automatically removed to a number. (will be mentioned in "extract")

> 3).In the data.table, dft[2] is a number and is going to be the second row.However, in the basic funtion, it takes the second column. It is convenience to sort and filter the rows instead of adding a comma. (will be mentioned in "extract")

> 4).Using column names can be used directly by name, without quotation marks. (will be mentioned in "extract")

> 5).Increased usage "!"" to reverse selection method. (will be mentioned in "delete the ranks")

> 6).The column names in this dataset are used in square brackets here, without the need to add $. (will be mentioned in "extract from logical values")

> please install data.table package and library it.

Content of the post

> In this post will explain the differences in data.table and other basic funtions. And use data.table package to explain systematically. The data.table package is basically based on square brackets, which not only covers the function of the basic function square brackets, but also has many other powerful functions. For example, they all use square brackets to extract, group, compute and so on.

Below is the list of this post:

- Create data
- Extraction function in data.table package
- Data.table package calculation and group calculation
- The merge of data.table
- Conclusion
- By reference

Part 1. Create Data

```
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 3.4.2
```

```
name1 <- c("Bob", "Mary", "Jane", "Kim")
name2 <- c("Bob", "Mary", "Kim", "Jane")
weight <- c(60, 65, 45, 55)
height <- c(170, 165, 140, 135)
birth <- c("1990-1", "1993-2", "1995-5", "1996-4")
accept <- c("no", "ok", "ok", "no")
dft <- data.table(name1, weight, height, accept)
dft
```

```
##    name1 weight height accept
## 1:   Bob     60    170     no
## 2:  Mary     65    165     ok
## 3:  Jane     45    140     ok
## 4:   Kim     55    135     no
```

Part 2. Extraction

The general extraction:

- Extract a point or multiple points according to the location coordinates
- Extract one or more points according to the name of the row
- Extract the entire line
- Extract the whole column
- Rows are extracted according to the logical value

> • Multiple points cannot be extracted from the matrix in data.table

1.The extraction of a coordinate or column name

1).Extract a point

```
dft[1,2] #This returns a data.table, while data.frame will return a value
```

```
##    weight
## 1:     60
```

```
dft[[1,2]] #this returns a value, and "drop" cannot be used in data.table
```

```
## [1] 60
```

```
dft[c(1,3),3]
```

```
##    height
## 1:    170
## 2:    140
```

```
dft[c(1,3), weight]
```

```
## [1] 60 45
```

2).Extract a row

```
dft[1]
```

```
##    name1 weight height accept
## 1:   Bob     60    170     no
```

```
dft[1:2, ]
```

```
##    name1 weight height accept
## 1:   Bob     60    170     no
## 2:  Mary     65    165     ok
```

3).Extract a column

```
dft[ ,2] #this retures to a data table
```

```
##    weight
## 1:     60
## 2:     65
## 3:     45
## 4:     55
```

```
dft[[3]]
```

```
## [1] 170 165 140 135
```

```
dft[["weight"]]
```

```
## [1] 60 65 45 55
```

```
dft[ ,weight]
```

```
## [1] 60 65 45 55
```

```
dft[ , weight:accept] # return to a dataset
```

```
##    weight height accept
## 1:     60    170     no
## 2:     65    165     ok
## 3:     45    140     ok
## 4:     55    135     no
```

2. Rows are extracted according to the logical value

1).Extract by logica

```
dft[weight > 40] #does NOT need to use $ in data.table
```

```
##    name1 weight height accept
## 1:   Bob     60    170     no
## 2:  Mary     65    165     ok
## 3:  Jane     45    140     ok
## 4:   Kim     55    135     no
```

```
dft[weight > 40 & height < 170]
```

```
##    name1 weight height accept
## 1:  Mary     65    165     ok
## 2:  Jane     45    140     ok
## 3:   Kim     55    135     no
```

```
dft[c(T,F,T,T)]
```

```
##    name1 weight height accept
## 1:   Bob     60    170     no
## 2:  Jane     45    140     ok
## 3:   Kim     55    135     no
```

2).Using the "on" parameter to extract a column is a row of a value (the equivalent of a logical value)

```
dft["Bob",on="name1"]
```

```
##    name1 weight height accept
## 1:   Bob     60    170     no
```

```
dft[name1=="Bob"]
```

```
##    name1 weight height accept
## 1:   Bob     60    170     no
```

```
dft[c("Bob","Mary"),on="name1"] #multiple choices in a column
```

```
##    name1 weight height accept
## 1:   Bob     60    170     no
## 2:  Mary     65    165     ok
```

> ### 3.Derived derivative
>
> - **Delete**
> - **Rank**

1).Delete the row/column

```
dft[-c(2,3)] # delete the second and third rows
```

```
##    name1 weight height accept
## 1:   Bob     60    170     no
## 2:   Kim     55    135     no
```

```
dft[,-c(2,3)] # delete the second and third columns
```

```
##    name1 accept
## 1:   Bob     no
## 2:  Mary     ok
## 3:  Jane     ok
## 4:   Kim     no
```

```
#the following is a function that not include in basic function(dplyr/data.frame)
dft[!2:3]
```

```
##    name1 weight height accept
## 1:   Bob     60    170     no
## 2:   Kim     55    135     no
```

```
dft[,!"weight"]
```

```
##    name1 height accept
## 1:   Bob    170     no
## 2:  Mary    165     ok
## 3:  Jane    140     ok
## 4:   Kim    135     no
```

```
dft[,-"weight"]
```

```
##    name1 height accept
## 1:   Bob    170     no
## 2:  Mary    165     ok
## 3:  Jane    140     ok
## 4:   Kim    135     no
```

```
dft[,-c("weight","height")]
```

```
##    name1 accept
## 1:   Bob     no
## 2:  Mary     ok
## 3:  Jane     ok
## 4:   Kim     no
```

```
dft[,!c("weight","height")]
```

```
##    name1 accept
## 1:   Bob     no
## 2:  Mary     ok
## 3:  Jane     ok
## 4:   Kim     no
```

2). Rank

```
dft[order(weight)] #No need to use $ in data.table
```

```
##    name1 weight height accept
## 1:  Jane     45    140     ok
## 2:   Kim     55    135     no
## 3:   Bob     60    170     no
## 4:  Mary     65    165     ok
```

```
dft[order(weight),] #Add a comma to indicate the sort for row, has same effect
```

```
##    name1 weight height accept
## 1:  Jane     45    140     ok
## 2:   Kim     55    135     no
## 3:   Bob     60    170     no
## 4:  Mary     65    165     ok
```

```
setcolorder(dft,rev(names(dft))) #Accept the rearranged column rank of column names
```

Part 3. Data.table package calculation and group calculation

> **The calculation of data.table process in "[]", and group is by adding a parameter. Therefore, it can be simple to realize the basic function or to realize the function of the package. Let's talk a little bit about the use of data.table in computing. Take three parameters in []:**
>
> - **The first specifies which rows to add to the calculation**
> - **The second specifies what kind of calculation to perform**
> - **The third specifies which variable to group the calculation**

1. Normal Calculation

1). In the second parameter position, what do you want to do with that column

```
dft[,sum(weight)]
```

```
## [1] 225
```

2). This extraction operation can also be viewed as the output of its own, without any other calculation

```
dft[,weight]
```

```
## [1] 60 65 45 55
```

3). The number of a list is calculated, and the result length is not equal

```
dft[,.(summary(weight),mean(weight))]
```

```
##       V1    V2
## 1: 45.00 56.25
## 2: 52.50 56.25
## 3: 57.50 56.25
## 4: 56.25 56.25
## 5: 61.25 56.25
## 6: 65.00 56.25
```

4). to show the results in vector notation

```
dft[,c(summary(weight),mean(weight))]
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   45.00   52.50   57.50   56.25   61.25   65.00   56.25
```

5). A number of the same column is evaluated and the calculated result column name is specified

```
dft[,.(wm=mean(weight),ws=sum(weight))]
```

```
##       wm  ws
## 1: 56.25 225
```

6). Descript the weight of the first two lines

```
dft[1:2,summary(weight)]
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   60.00   61.25   62.50   62.50   63.75   65.00
```

7). screening calculation

```
dft["Bob", weight-10, on="name1"]
```

```
## [1] 50
```

8). The second parameter provides a platform that can also be used by other data boxes

```
w <- dft[,c("weight","accept")]
dft[,w[,sum(weight)],by=accept]
```

```
##    accept  V1
## 1:     no 225
## 2:     ok 225
```

```
#The w  is calculated. Although it looks like the accept group is being displayed, the results are not actu
```

> **When the output is a matrix**
>
> - In the above calculation, we will find that the output results are automatically sorted into data.table datasets. If output are two matrices, how does this become a data box? If don't add anything else, this will automatically become a vector, and each of these will be shown as a column of data.table, so that will lose the value of the matrix. Below, will use two methods to solve this problem, which involves the special use of the location of the second parameter
> - Use {} to change the namespace
> - Use list

1. General method

```
dft[,weight%*%t(weight)] #Only return to one as matrix
```

```
##      [,1] [,2] [,3] [,4]
## [1,] 3600 3900 2700 3300
## [2,] 3900 4225 2925 3575
## [3,] 2700 2925 2025 2475
## [4,] 3300 3575 2475 3025
```

```
dft[,.(weight%*%t(weight),height%*%t(height))] #Use a list, but each will be automatically converted to a v
```

```
##       V1    V2
##  1: 3600 28900
##  2: 3900 28050
##  3: 2700 23800
##  4: 3300 22950
##  5: 3900 28050
##  6: 4225 27225
##  7: 2925 23100
##  8: 3575 22275
##  9: 2700 23800
## 10: 2925 23100
## 11: 2025 19600
## 12: 2475 18900
## 13: 3300 22950
## 14: 3575 22275
## 15: 2475 18900
## 16: 3025 18225
```

```
dft[,c(weight%*%t(weight),height%*%t(height))] #If use c, it's going to convert all of that to a vector
```

```
##  [1]  3600  3900  2700  3300  3900  4225  2925  3575  2700  2925  2025
## [12]  2475  3300  3575  2475  3025 28900 28050 23800 22950 28050 27225
## [23] 23100 22275 23800 23100 19600 18900 22950 22275 18900 18225
```

1). Use "{}"

```
dft[,{weight%*%t(weight)
  height%*%t(height)}]    #Use braces to run multiple commands, but only the last result can be returned. I
```

```
##       [,1]  [,2]  [,3]  [,4]
## [1,] 28900 28050 23800 22950
## [2,] 28050 27225 23100 22275
## [3,] 23800 23100 19600 18900
## [4,] 22950 22275 18900 18225
```

```
dft[,{print(weight%*%t(weight))
height%*%t(height)}]      #Can use "print" in the middle. But it only can be shown, not for call
```

```
##      [,1] [,2] [,3] [,4]
## [1,] 3600 3900 2700 3300
## [2,] 3900 4225 2925 3575
## [3,] 2700 2925 2025 2475
## [4,] 3300 3575 2475 3025

##       [,1]  [,2]  [,3]  [,4]
## [1,] 28900 28050 23800 22950
## [2,] 28050 27225 23100 22275
## [3,] 23800 23100 19600 18900
## [4,] 22950 22275 18900 18225
```

```
dft[,{u1 <- weight%*%t(weight)
  u2 <<- height%*%t(height)}]   #If use <- assignment, the variable cannot be found in the external enviror
```

```
##       [,1]  [,2]  [,3]  [,4]
## [1,] 28900 28050 23800 22950
## [2,] 28050 27225 23100 22275
## [3,] 23800 23100 19600 18900
## [4,] 22950 22275 18900 18225
```

2). Use list

```
(l <- dft[,.(.(weight%*%t(weight)),.(height%*%t(height)))])
```

```
##                                                                                 V1
## 1: 3600,3900,2700,3300,3900,4225,2925,3575,2700,2925,2025,2475,3300,3575,2475,3025,
##                                                                                 V2
## 1: 28900,28050,23800,22950,28050,27225,23100,22275,23800,23100,19600,18900,22950,22275,18900,18225,
```

```
l[[1]]
```

```
## [[1]]
##      [,1] [,2] [,3] [,4]
## [1,] 3600 3900 2700 3300
## [2,] 3900 4225 2925 3575
## [3,] 2700 2925 2025 2475
## [4,] 3300 3575 2475 3025
```

the matrix is folded into an element by the list, which can be called by index. It is hard for data.frame to do it.

Explain the difference between data.table and data.frame on list

```
ma <- cbind(1:3,1) #a matrix
ma
```

```
##      [,1] [,2]
## [1,]    1    1
## [2,]    2    1
## [3,]    3    1
```

```
data.frame(a=1,b=ma)
```

```
##   a b.1 b.2
## 1 1   1   1
## 2 1   2   1
## 3 1   3   1
```

```
data.table(a=1,b=ma) #same as data.frame
```

```
##    a b.V1 b.V2
## 1: 1    1    1
## 2: 1    2    1
## 3: 1    3    1
```

```
data.frame(a=1,b=list(ma)) #above three are returns to column 3 dataset
```

```
##   a b.1 b.2
## 1 1   1   1
## 2 1   2   1
## 3 1   3   1
```

```
(d <- data.table(a=1,b=list(ma))) #The element in the dataset that returns two columns, which is, dataset i
```

```
##    a           b
## 1: 1 1,2,3,1,1,1
```

```
d[[1,2]] #Although it's converted to a list, the output is still the matrix
```

```
##      [,1] [,2]
## [1,]    1    1
## [2,]    2    1
## [3,]    3    1
```

```
#list in the dataset
#The inside uses "I"" to make data.frame() see the list as a unit
df<-data.frame(x=1:3,y=I(list(1:2,1:3,1:4))) #The number of elements must be the same as the number of rows
df
```

```
##   x          y
## 1 1       1, 2
## 2 2    1, 2, 3
## 3 3 1, 2, 3, 4
```

```
df$m <- list(1:4,3:5,5:6) #if set up like this, do not have to use "I"
df$y #This whole column is a list
```

```
## [[1]]
## [1] 1 2
##
## [[2]]
## [1] 1 2 3
##
## [[3]]
## [1] 1 2 3 4
```

```
df$y[[1]][1]
```

```
## [1] 1
```

```
dft1 <- data.table(x=1:3,y=list(list(1:2),list(1:3),list(1:4))) #Three elements are placed on three lines
dft1[[1,2]]
```

```
## [[1]]
## [1] 1 2
```

```
dft2 <- data.table(x=1,y=list(list(list(1:2),list(1:3),list(1:4)))) #It's okay to be an element that is nes
dft2[[1,2]]
```

```
## [[1]]
## [[1]][[1]]
## [1] 1 2
##
##
## [[2]]
## [[2]][[1]]
## [1] 1 2 3
##
##
## [[3]]
## [[3]][[1]]
## [1] 1 2 3 4
```

```
df<-data.table(x=1:3,y=I(list(1:2,1:3,1:4))) # data.table can be used as data.frame
df
```

```
##    x       y
## 1: 1     1,2
## 2: 2   1,2,3
## 3: 3 1,2,3,4
```

```
#Add: use data.frame to fold the matrix into an element, which needs to be converted into a matrix before b
data.frame(cbind(1,list(ma), list(ma)))
```

```
##   X1              X2              X3
## 1  1 1, 2, 3, 1, 1, 1 1, 2, 3, 1, 1, 1
```

3. Group Calculation

```
dft[,sd(weight),by=accept]
```

```
##    accept        V1
## 1:     no  3.535534
## 2:     ok 14.142136
```

```
dft[,sd(weight),keyby=accept] #as "accept" order
```

```
##    accept        V1
## 1:     no  3.535534
## 2:     ok 14.142136
```

```
dft[,sd(weight),by=accept][order(accept)] #same as the one above
```

```
##    accept        V1
## 1:     no  3.535534
## 2:     ok 14.142136
```

```
dft[,mean(weight),by=height>150] #The variables are grouped after calculation
```

```
##    height   V1
## 1:   TRUE 62.5
## 2:  FALSE 50.0
```

1). Follow the multi-column grouping

```
DT = data.table(x=rep(c("b","a","c"),each=6), y=c(1,3,6), v=1:18)
DT[,sum(v),by=x]
```

```
##    x V1
## 1: b 21
## 2: a 57
## 3: c 93
```

```
DT[,sum(v),by=y]
```

```
##    y V1
## 1: 1 51
## 2: 3 57
## 3: 6 63
```

```
DT[,sum(v),by=.(x,y)]
```

```
##    x y V1
## 1: b 1  5
## 2: b 3  7
## 3: b 6  9
## 4: a 1 17
## 5: a 3 19
## 6: a 6 21
## 7: c 1 29
## 8: c 3 31
## 9: c 6 33
```

```
DT[,sum(v),by=c("x","y")]
```

```
##    x y V1
## 1: b 1  5
## 2: b 3  7
## 3: b 6  9
## 4: a 1 17
## 5: a 3 19
## 6: a 6 21
## 7: c 1 29
## 8: c 3 31
## 9: c 6 33
```

```
DT[,sum(v),by=.(x,y)][,sum(V1),by=x]
```

```
##    x V1
## 1: b 21
## 2: a 57
## 3: c 93
```

Part 3. The merge of data.table

```
dt1 <- dft[1:3]
dt1
```

```
##    accept height weight name1
## 1:     no    170     60   Bob
## 2:     ok    165     65  Mary
## 3:     ok    140     45  Jane
```

```
dt2 <- data.table(name1=name2[1:3],birth[1:3],friend=name1[c(2,4,3)])
dt2
```

```
##    name1     V2 friend
## 1:   Bob 1990-1   Mary
## 2:  Mary 1993-2    Kim
## 3:   Kim 1995-5   Jane
```

```
dt1[dt2,on="name1"] #Use the same column name to merge the two data boxes, retaining all values of name1 ir
```

```
##    accept height weight name1     V2 friend
## 1:     no    170     60   Bob 1990-1   Mary
## 2:     ok    165     65  Mary 1993-2    Kim
## 3:     NA     NA     NA   Kim 1995-5   Jane
```

```
dt2[dt1,on="name1"] #Keep dt1 and Na replace missing value in dt2
```

```
##    name1     V2 friend accept height weight
## 1:   Bob 1990-1   Mary     no    170     60
## 2:  Mary 1993-2    Kim     ok    165     65
## 3:  Jane     NA     NA     ok    140     45
```

```
dt2[dt1,on="name1",nomatch=0] #intersecting
```

```
##    name1     V2 friend accept height weight
## 1:   Bob 1990-1   Mary     no    170     60
## 2:  Mary 1993-2    Kim     ok    165     65
```

```
dt1[!dt2,on="name1"] #intersect Na in dt2
```

```
##    accept height weight name1
## 1:     ok    140     45  Jane
```

```
dt1[dt2,on=.(name1==friend)] # to merge the content column name not the same, use == to match
```

```
##    accept height weight name1 i.name1     V2
## 1:     ok    165     65  Mary     Bob 1990-1
## 2:     NA     NA     NA   Kim    Mary 1993-2
## 3:     ok    140     45  Jane     Kim 1995-5
```

```
dt1[dt2,on="name1==friend"] #same as above
```

```
##    accept height weight name1 i.name1     V2
## 1:     ok    165     65  Mary     Bob 1990-1
## 2:     NA     NA     NA   Kim    Mary 1993-2
## 3:     ok    140     45  Jane     Kim 1995-5
```

If the fusion basis is a number, it can also be used to match the inequality with <= >=

```
dt1[dt2,.(name1,w=weight),on="name1==friend"] #Select which columns to return in the position of the second
```

```
##    name1  w
## 1:   Bob 65
## 2:  Mary NA
## 3:   Kim 45
```

```
dt1[dt2,on="name1",mult="first"] # Select the first of each group
```

```
##    accept height weight name1     V2 friend
## 1:     no    170     60   Bob 1990-1   Mary
## 2:     ok    165     65  Mary 1993-2    Kim
## 3:     NA     NA     NA   Kim 1995-5   Jane
```

```
dt1[dt2,on="name1",mult="last"] # Select the last of each group
```

```
##    accept height weight name1     V2 friend
## 1:     no    170     60   Bob 1990-1   Mary
## 2:     ok    165     65  Mary 1993-2    Kim
## 3:     NA     NA     NA   Kim 1995-5   Jane
```

Part 4. Conclusion

> Data.table contains less and more efficient code, especically when dealing with large data, which will be faster than dplyr and python's pandas processing data. On the other hand, data.table can read aggregation sorting, grouping operation and flexible nature of grammar.

Part 5.Reference

- ####https://stackoverflow.com/questions/21435339/data-table-vs-dplyr-can-one-do-something-well-the-other-cant-or-does-poorly
- ####https://www.quora.com/Which-is-better-to-use-for-data-manipulation-dplyr-package-or-data-table-library
- ####https://datascienceplus.com/best-packages-for-data-manipulation-in-r/
- ####https://cran.r-project.org/web/packages/data.table/vignettes/datatable-intro.html
- ####https://www.r-bloggers.com/intro-to-the-data-table-package/
- ####https://github.com/Rdatatable/data.table/wiki
- ####https://www.rdocumentation.org/packages/data.table/versions/1.10.4-2