FAQs about the **data.table** package in R

Matthew Dowle

April 15, 2010

The first section, Beginner FAQs, is intended to be read in order from start to finish. It may be read before reading the Introduction vignette.

Contents

1 Beginner FAQs **2**
- 1.1 Why does DT[,5] return 5 ? . . . 2
- 1.2 Why does DT[,"region"] return "region"? . . . 2
- 1.3 Why does DT[,region] return a vector? I want a 1-column data.table. There is no drop argument like I'm used to in data.frame. . . . 2
- 1.4 Why does DT[,x,y,z] not work? I wanted the 3 columns x,y and z. . . . 2
- 1.5 I assigned a variable mycol="x" but then DT[,mycol] returns "x". How do I get it to look up the column name contained in the mycol variable? . . . 2
- 1.6 This is really hard. Whats the point ? . . . 2
- 1.7 Ok, I'm starting to see what data.table is about, but why didn't you enhance data.frame? Why does it have to be a new package? . . . 3
- 1.8 Why are the defaults the way they are? Why does it work the way it does? . . . 3
- 1.9 Isn't this already done by with() and subset() in base? . . . 3
- 1.10 Why does x[y] just return the columns from x? Shouldn't it return the y columns too? . . . 3

2 General syntax **3**
- 2.1 How can I avoid writing a really long j expression? You've said I should use the column names, but I've got loads of them. . . . 3
- 2.2 Why is the default for mult "first" ? If there are duplicates in the key, shouldn't it return them all by default? . . . 4
- 2.3 Comment from test 81 . . . 4
- 2.4 I have build up a complex table with many columns. I want to use it as a template for a new table i.e. create a new table with no rows, but with the columns types copied from my table. Can I do that easily? . . . 4
- 2.5 Is a NULL data.table the same as DT[0]? . . . 4
- 2.6 Why has the DT alias been removed? . . . 4
- 2.7 What are the scoping rules for j expression? . . . 4
- 2.8 Can I trace my code as it runs through the groups? . . . 5

3 Questions relating to compute time **5**
- 3.1 I have 20 columns in data.table x. Why is x[,sum(V1),by=grp] so quick? . . . 5
- 3.2 I don't have a key on a large table, but grouping is still really quick. Why is that? 6

4 General questions about the package **6**
- 4.1 v1.3 appears to be missing from the CRAN archive ? . . . 6
- 4.2 Is data.table compatible with S-plus ? . . . 6
- 4.3 I think its great. How do I suggest it to other people ? . . . 6
- 4.4 I think its not great. How do I warn others about my experience ? . . . 6

4.5 I have a question. I know the posting guide tells me to contact the maintainer, but thats just one person. Isn't there a group of people I can ask ? 6
4.6 Where are the datatable-help archives? 6
4.7 I'd prefer not to contact datatable-help, can I mail just one or two people privately? 6
4.8 Why is this FAQ a pdf? Can we have the FAQ on a website? 6

1 Beginner FAQs

1.1 Why does DT[,5] return 5 ?

Because, by default, unlike with a data.frame the 2nd argument is an <italic>expression</italic> which is evaluated within the scope of DT. 5 evaluates to 5. It is generally bad practice to refer to columns by number rather than name. If someone else comes along and reads your code later, they may have to hunt around to find out which column is number 5. Furthermore, if you, or someone else changes the column ordering of DT higher up in your R program, you might get bugs if you forget to change all the places in your code which refer to column number 5.

Say column 5 is called "region", just do DT[,region] instead. Notice there are no quotes around the column name. This is what we mean by j being evaluated within the scope of the data.table. That scope consists of an environment where the column names are variables.

Having said this, there are some circumstances where refering to a column by number is ok, particular a sequence of columns. In these situations just do DT[,5:10,with=FALSE] or DT[,c(1,4,10),with=FALSE]. See ?"[.data.table" for an explanation of the 'with' argument.

Note that with() has been a base function for a long time. Thats why we say that data.table builds upon base functionality. There is nothing new here really, data.table is just making use of with() and building it into the syntax.

1.2 Why does DT["region"] return "region"?

See answer to 1.1 above. Try DT[,region] instead. Or if you must then DT["region",with=FALSE].

1.3 Why does DT[,region] return a vector? I want a 1-column data.table. There is no drop argument like I'm used to in data.frame.

Try DT[,list(region)] instead.

1.4 Why does DT[,x,y,z] not work? I wanted the 3 columns x,y and z.

The j expression is the 2nd argument. The correct way to do this is DT[,list(x,y,z)].

1.5 I assigned a variable mycol="x" but then DT[,mycol] returns "x". How do I get it to look up the column name contained in the mycol variable?

This is what we mean when we say the j expression 'sees' objects in the calling scope. Because 'mycol' does not exist as a column name of DT, R then looked in the calling scope and found mycol there, and returned its value. This is correct behaviour. Had "mycol" been a column name, then that column's data would have been returned. What you probably meant was DT[,mycol,with=FALSE] which will return the region column's data as you wanted. Alternatively you could do DT[,eval(mycol)].

1.6 This is really hard. Whats the point ?

j doesn't have to be just column names. You can put any R expression of column names, directly as the j e.g. DT[,mean(x*y/z)]. The same applies to i. You have been used to i being row numbers or row names only. Its nice to just write DT[x>1000,sum(y*z)]. What does that mean? Well it just runs the j expression on the set of rows where the i is true. The i can be any expression

of column names that evaluates to logical. You don't even need to return data e.g. DT[x>1000, plot(y,z)]. When we get to compound table joins we will see how i (and j) can themselves be other data.table queries. We are going to stretch i and j much further than this. But to get there we need you on board first with FAQs 1.1-1.5.

1.7 Ok, I'm starting to see what data.table is about, but why didn't you enhance data.frame? Why does it have to be a new package?

As FAQ 1.1 highlights, j is fundamentally different from j in data.frame. Even something as simple as DF[,1] would break existing code in many packages and user code. This is by design and we want it to work this way, for more complicated syntax to work. There are other differences too.
To convert a data.frame DF to a data.table, just write data.table(DF).

1.8 Why are the defaults the way they are? Why does it work the way it does?

The simple answer is because the author designed it for his own use, and he wanted it that way. He finds it a more natural, faster way to write code, which also executes more quickly.

1.9 Isn't this already done by with() and subset() in base?

Some of the features discussed so far are, yes. The package builds upon base functionality. It does the same sorts of things but with less code required, and executes many times faster, if used correctly.

1.10 Why does x[y] just return the columns from x? Shouldn't it return the y columns too?

Good question. The thinking was that, more often than not, you don't actually want the columns from y which aren't in the key. By default, we try to keep things efficient. In general we don't want to create memory for the union of things, only to select out a few columns from it in the end. So if you want the computer to do more work, then you need to tell it do more work. You can either do cbind(y,x[y]) or merge(x,y). There are many different ways to do the same thing in data.table. Its your choice to understand the differences and write good code. The other thinking is that x[y] is after all a subset of x. The "[" operator does mean subset. By default we thought it was more consistent with base R for x[y] to just return the columns from x. However it is now apparent that x[y] returning all columns from both tables would be useful, so an argument inci will be added. We mentioned the merge method for data.table too, but does merge(x,y) mean x[y] or y[x] ? Those are different things. Again, this is your choice which syntax you prefer and find clearer.
Finally, although it appears as though x[y] does not return the columns in y, you can actually use the columns from y in the j expression. This is what we mean by âĂIJjoin inherited scopeâĂİ. Why not just return the union of all the columns from x and y and then run expressions on that? Its down to efficiency of code and quicker to program. When you write x[y,foo*boo], data.table automatically inspects the j expression to see which columns it uses. It will only subset, or group, those columns only. Memory is only created for the columns the j uses. Lets say foo is in x, and boo is in y (along with 20 other columns in y). Isn't x[y,foo*boo] quick to program and quick to run, than a merge step followed by another subset step ?

2 General syntax

2.1 How can I avoid writing a really long j expression? You've said I should use the column names, but I've got loads of them.

There is a special .SD object, which stands for Sub Data. The j expression can use column names as variables, as you know, but it can also use .SD which refers to the sub data.table as a whole.

So to sum up all your columns its just DT[,lapply(.SD,sum),by=grp]. It might seem tricky, but its fast to write and fast to run. Notice you don't have to create an anonymous function(). See the timing vignette and comparison to other methods. The .SD object is efficently implemented internally, its more efficient than passing an argument to a function. Please don't do this though : DT[,.SD[,"sales",with=FALSE],by=grp]. That works but its very inefficient and inelegant. This is what was intended : DT[,sum(x),by=grp] and could be 100's of times faster if DT contains many columns. No data.table may contain a column called .SD, thats why it has a "." at the start as you are unlikely to really want a column called ".SD".

2.2 Why is the default for mult "first" ? If there are duplicates in the key, shouldn't it return them all by default?

If there are no duplicates then "all" does no harm. No, "all" is slower. Internally "all" is implemented by joining using "first" then again using "last" and then does a diff between them to work out the spans of groups. Almost always we work with unique keys, and we prefer maximum performance for the majority of the situations. If you are working with a non-unique key then you need to specify "all".

A future version may allow a distinction between a key and a unique key. The mult argument could then we defaulted more wisely. data.table would then need to add checks on insert and update to make sure a unique key is maintained. That would be one of the advantages to the user of specifying a unique key i.e. data.table would make sure a duplicate could not be inserted.

Currently, there is no distinction. Almost always the key will be unique and for efficiency either "first" or "last" are faster than "all", so we chose "first".

Note that when i (or i's key if it has one) has less columns that x's key, mult is automatically set to "all". This is why grouping by i works i.e. DT[J(grp),mean(v)] where key(DT) has 2 or more columns.

2.3 Comment from test 81

2.4 I have build up a complex table with many columns. I want to use it as a template for a new table i.e. create a new table with no rows, but with the columns types copied from my table. Can I do that easily?

Yes. If your complex table is called DT, try DT[0].

2.5 Is a NULL data.table the same as DT[0]?

No. Despite the print method indicating otherwise. Strictly speaking its not possible to have is.null(data.table(NULL)) returns FALSE. Perhaps look at this?

2.6 Why has the DT alias been removed?

It hasn't been removed, yet. It is no longer encouraged or mentioned in the docs or examples. We now use and encourage list() to be passed as the j. list() is a primitive and is much faster than passing DT() or data.table() as the j, especially when there are many groups. Internally this was a non trivial change, but the speedups are very large.

2.7 What are the scoping rules for j expression?

Think of the subset as an environment where all the column names are variables. When a variable is used in the j expression, it is looked for in the following order :

1. The scope of the subset i.e. the column names
2. The scope of the calling frame e.g. the line that appears before the data.table query
3. TO CHECK. Does it ripple up or go straight to .GlobalEnv?

4. The global environment

This is "lexical scoping" explained by R FAQ 3.3.1. Note that the environment that the function was created is not relevant though, because no anonymous function is passed to the j. Note that an anonymous body can be passed to the j, for example:

```
> DT = data.table(x=rep(c("a","b"),c(2,3)),y=1:5)
> DT

     x y
[1,] a 1
[2,] a 2
[3,] b 3
[4,] b 4
[5,] b 5

> DT[,{print(x);list(sum(y))},by="x"]

[1] a a
Levels: a b
[1] b b b
Levels: a b
     x V1
[1,] a  3
[2,] b 12
```

2.8 Can I trace my code as it runs through the groups?

Try something this :

```
> DT[,{
+   cat("Trace:",as.character(x[1]),length(y),"\n");
+   print(objects());
+   sum(y)
+ },by=x]

Trace: a 2
[1] "x" "y"
Trace: b 3
[1] "x" "y"
     x V1
[1,] a  3
[2,] b 12
```

3 Questions relating to compute time

3.1 I have 20 columns in data.table x. Why is x[,sum(V1),by=grp] so quick?

Several reasons.

- Only the V1 column is grouped, the other 19 are ignored because `data.table` detects that j expression doesn't use them.
- One memory allocation is made for the largest group of V1 only, then that memory is re-used for the other groups, there is very little garbage to collect.
- R is an in memory column store i.e. V1 is contiguous in RAM. Page fetches from RAM into L2 cache are minimised.

3.2 I don't have a key on a large table, but grouping is still really quick. Why is that?

data.table uses radix sorting for integer vectors. This is significantly faster than other sort algorithims. Unfortunately radix is restricted to integer only. See ?sort.list(method="radix"). This is also one reason why `setkey()` is quick.

4 General questions about the package

4.1 v1.3 appears to be missing from the CRAN archive ?

That is correct. v1.2 was released to CRAN in Aug 2008. Development of v1.3 then continued on r-forge, from which users installed the latest nightly build, and were very helpful giving feedback and suggestions. As many versions of v1.3 are in use, we released v1.4 to CRAN as a clean line in the sand. From now on our hope is to have a more regular release cycle to CRAN.

4.2 Is data.table compatible with S-plus ?

Not currently.

- A few core parts of the package are written in C and use internal functions and structures.
- The package uses lexical scoping which is one of the differences between R and S-plus explained by R FAQ 3.3.1.

4.3 I think its great. How do I suggest it to other people ?

Thank you. Please put your comments on `http://crantastic.org/`

4.4 I think its not great. How do I warn others about my experience ?

Please put your comments on `http://crantastic.org/`. Try to make it constructive so we have a chance to improve.

4.5 I have a question. I know the posting guide tells me to contact the maintainer, but thats just one person. Isn't there a group of people I can ask ?

Yes. You can post to datatable-help@lists.r-forge.r-project.org. Its like r-help, but just for this package. Feel free to answer questions there too.

4.6 Where are the datatable-help archives?

`http://lists.r-forge.r-project.org/pipermail/datatable-help/`

4.7 I'd prefer not to contact datatable-help, can I mail just one or two people privately?

Sure.

4.8 Why is this FAQ a pdf? Can we have the FAQ on a website?

This FAQ is a vignette and as such is written using Sweave. The benefits include:

- We include R code in the answers. This is actually run when the file is created, its not copy and pasted.
- This document is reproducible. Grab the .Rnw and you can run it yourself.

- This file is built every night on r-forge so its another way we check the package.
- This file is bound into each version of the package. The package is not accepted on CRAN unless this file passes checks. Each version of the package will have its own FAQ file which will be relevant for that version. Contrast to a single website, which can be ambiguous if the answer depends on the version.
- You can open it offline, from your R prompt using vignette().
- It prints out easily.
- Its quicker and easier for us to write and maintain the FAQ in .Rnw form.