data.table

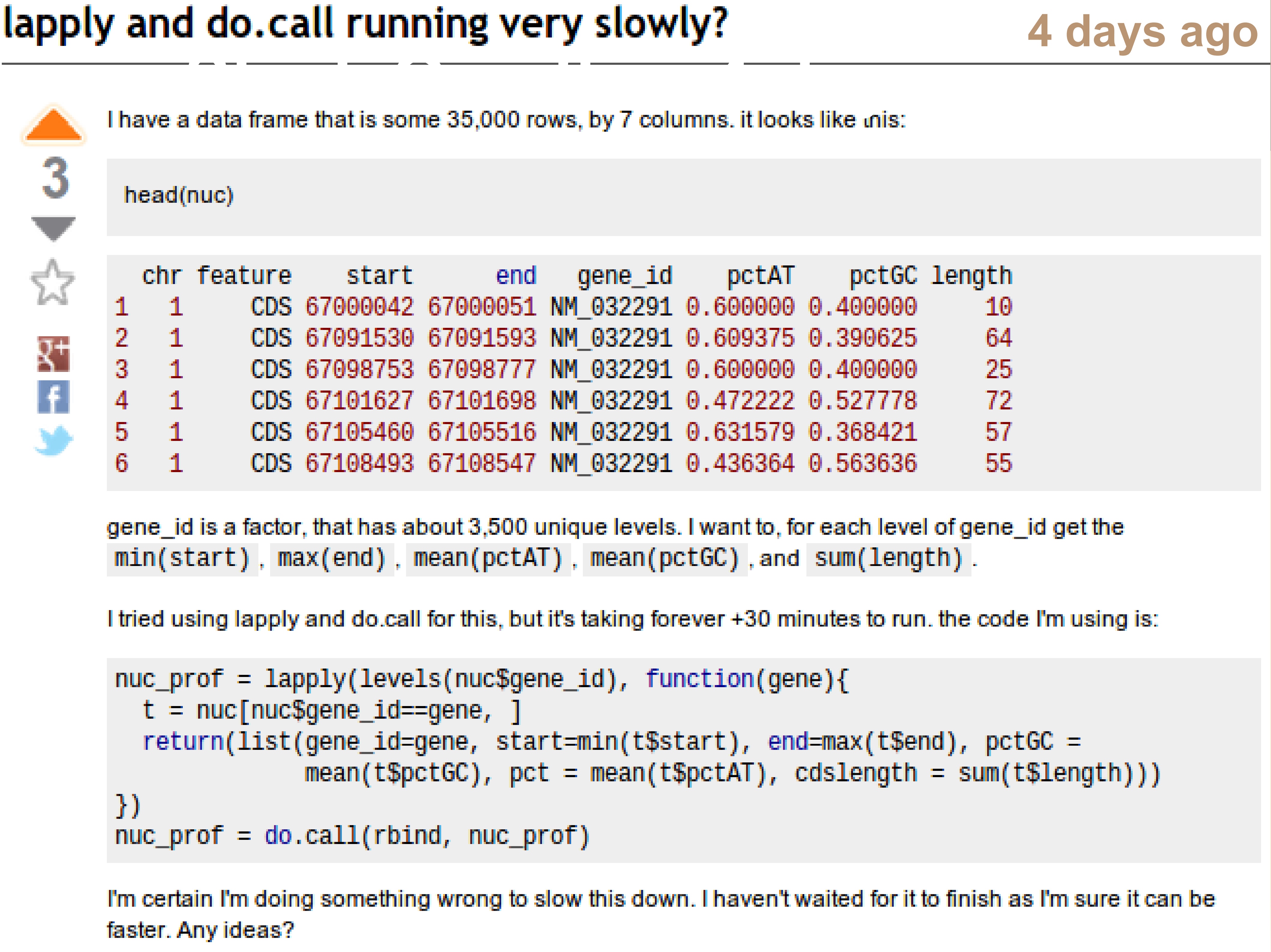
News from 1.6, 1.7 & 1.8

Matthew Dowle

LondonR, June 2012

Overview

- Real example
- Review of last presentation 2 years ago
- Package statistics
- New features
- Q&A



lapply and do.call running very slowly?

4 days ago

3

I have a data frame that is some 35,000 rows, by 7 columns. it looks like this:

```
head(nuc)
```

```
  chr feature     start       end   gene_id    pctAT    pctGC length
1   1     CDS 67000042 67000051 NM_032291 0.600000 0.400000     10
2   1     CDS 67091530 67091593 NM_032291 0.609375 0.390625     64
3   1     CDS 67098753 67098777 NM_032291 0.600000 0.400000     25
4   1     CDS 67101627 67101698 NM_032291 0.472222 0.527778     72
5   1     CDS 67105460 67105516 NM_032291 0.631579 0.368421     57
6   1     CDS 67108493 67108547 NM_032291 0.436364 0.563636     55
```

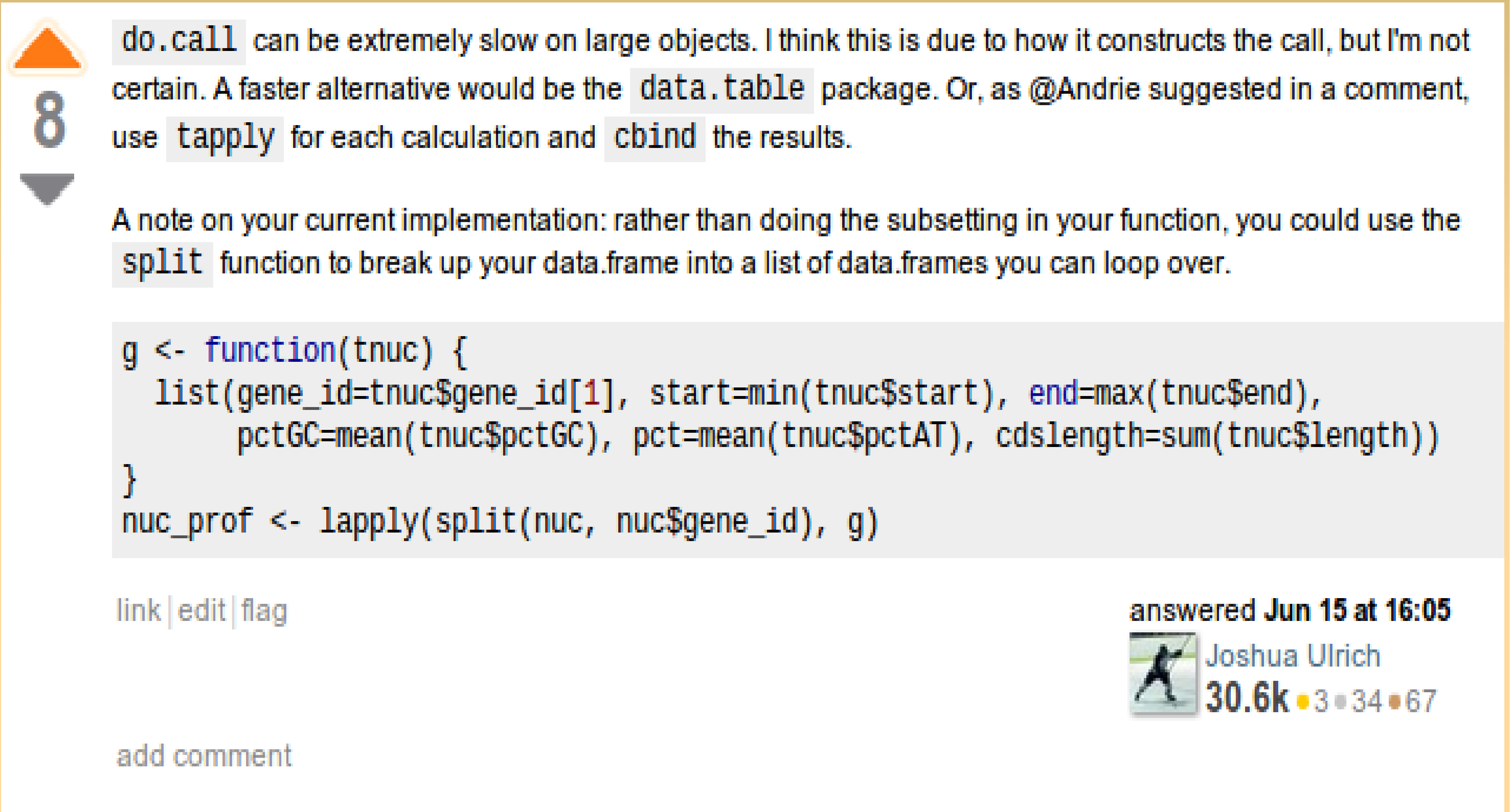
gene_id is a factor, that has about 3,500 unique levels. I want to, for each level of gene_id get the `min(start)`, `max(end)`, `mean(pctAT)`, `mean(pctGC)`, and `sum(length)`.

I tried using lapply and do.call for this, but it's taking forever +30 minutes to run. the code I'm using is:

```
nuc_prof = lapply(levels(nuc$gene_id), function(gene){
  t = nuc[nuc$gene_id==gene, ]
  return(list(gene_id=gene, start=min(t$start), end=max(t$end), pctGC =
                  mean(t$pctGC), pct = mean(t$pctAT), cdslength = sum(t$length)))
})
nuc_prof = do.call(rbind, nuc_prof)
```

I'm certain I'm doing something wrong to slow this down. I haven't waited for it to finish as I'm sure it can be faster. Any ideas?

First answer



8

do.call can be extremely slow on large objects. I think this is due to how it constructs the call, but I'm not certain. A faster alternative would be the data.table package. Or, as @Andrie suggested in a comment, use tapply for each calculation and cbind the results.

A note on your current implementation: rather than doing the subsetting in your function, you could use the split function to break up your data.frame into a list of data.frames you can loop over.

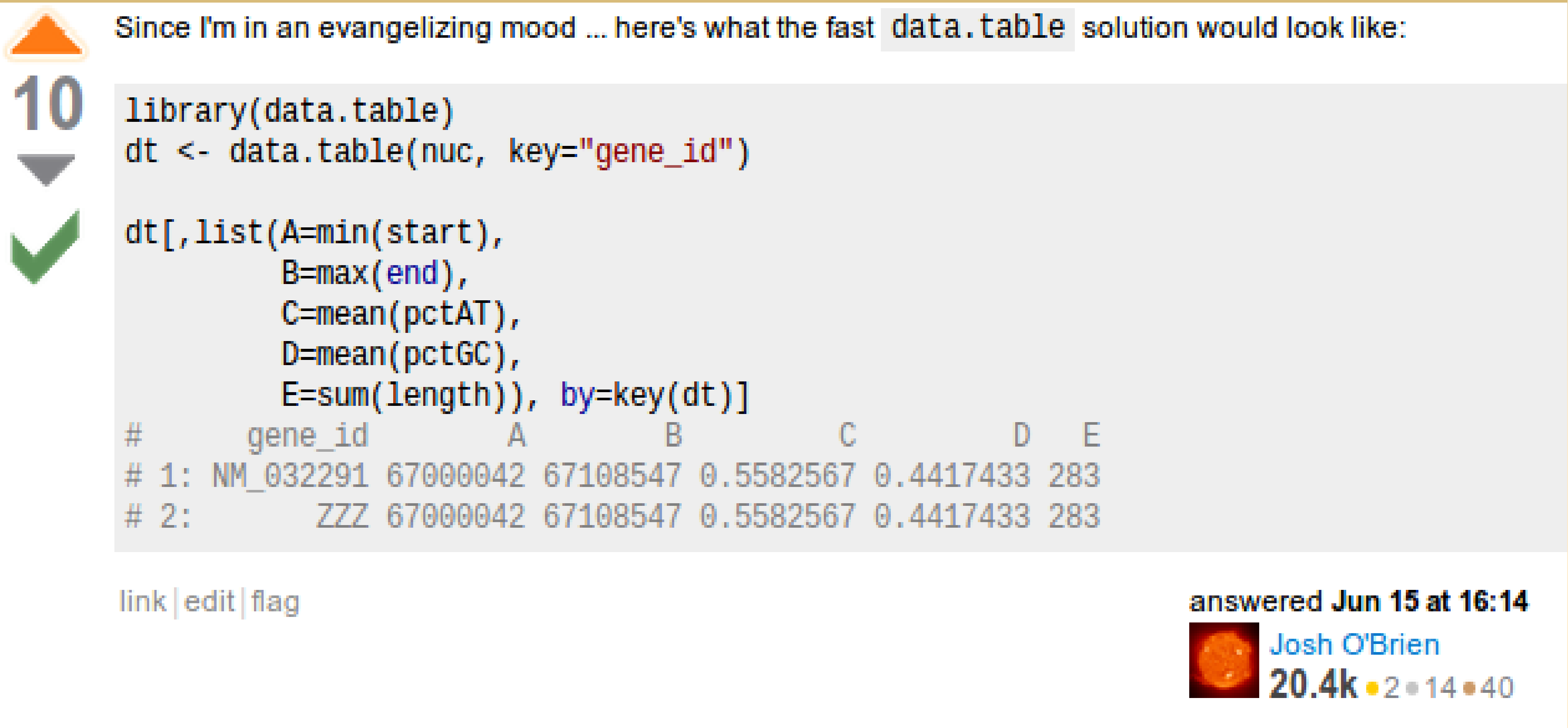
```
g <- function(tnuc) {
  list(gene_id=tnuc$gene_id[1], start=min(tnuc$start), end=max(tnuc$end),
       pctGC=mean(tnuc$pctGC), pct=mean(tnuc$pctAT), cdslength=sum(tnuc$length))
}
nuc_prof <- lapply(split(nuc, nuc$gene_id), g)
```

answered **Jun 15 at 16:05**

Joshua Ulrich

30.6k ●3 ●34 ●67

Follow up answer



Since I'm in an evangelizing mood ... here's what the fast `data.table` solution would look like:

```
library(data.table)
dt <- data.table(nuc, key="gene_id")

dt[,list(A=min(start),
         B=max(end),
         C=mean(pctAT),
         D=mean(pctGC),
         E=sum(length)), by=key(dt)]
#      gene_id        A        B         C         D   E
# 1: NM_032291 67000042 67108547 0.5582567 0.4417433 283
# 2:       ZZZ 67000042 67108547 0.5582567 0.4417433 283
```

answered **Jun 15 at 16:14**

Josh O'Brien

20.4k ● 2 ● 14 ● 40

NB: It isn't just the speed, but the simplicitly. It's easy to write and easy to read.

User's reaction

Holy fudge buckets!!! data.table is awesome! That took about 3 seconds for the whole thing!!!

I think that congratulations are well in order for the frankly amazingly well written quick start guide and FAQ. Seriously. Where is the button to make all R and bioconductor packages like this one?

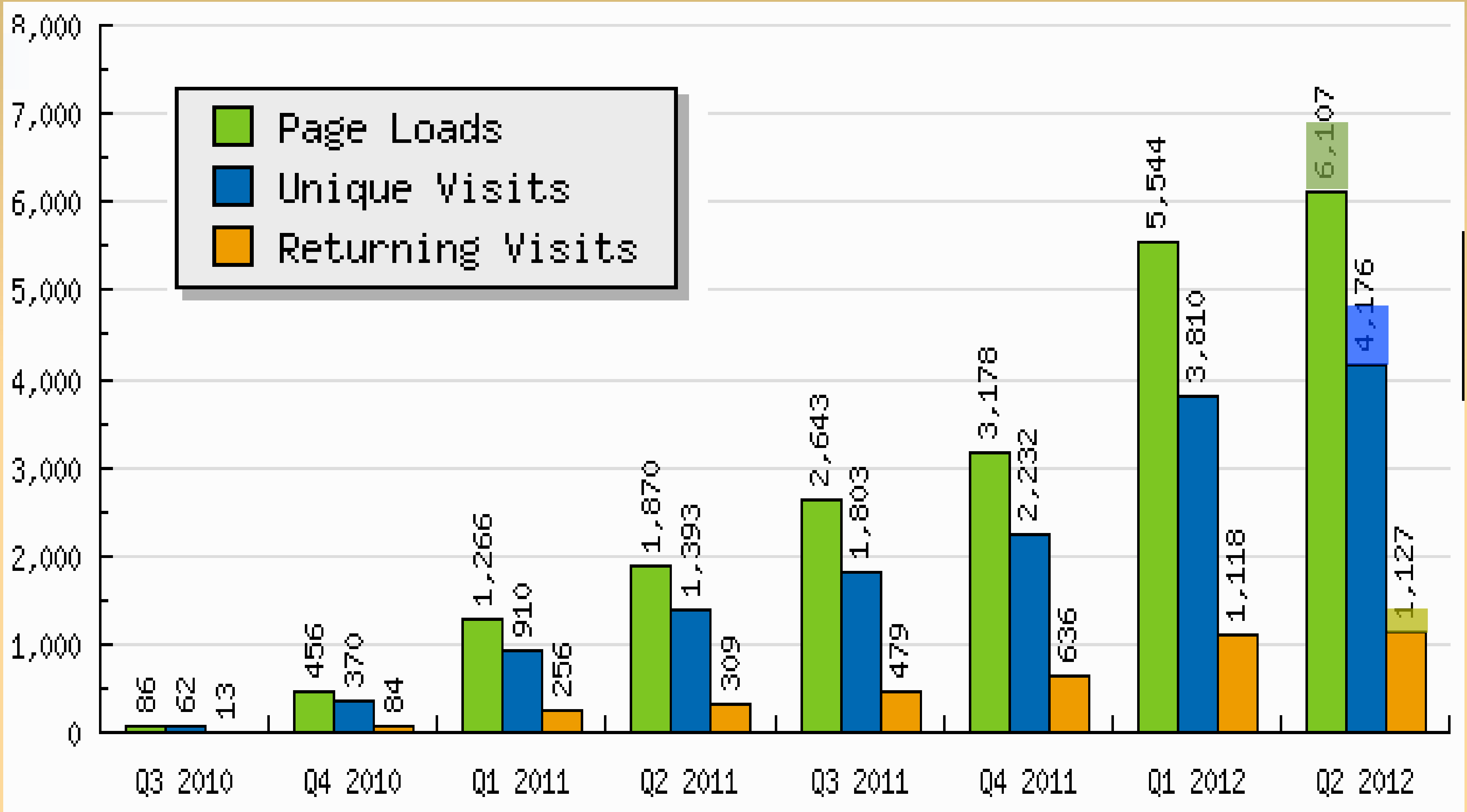
Davy Kavanagh, 15 Jun 2012

Review of presentation 2 years ago

(Including why grouping is fast)

Link to pdf

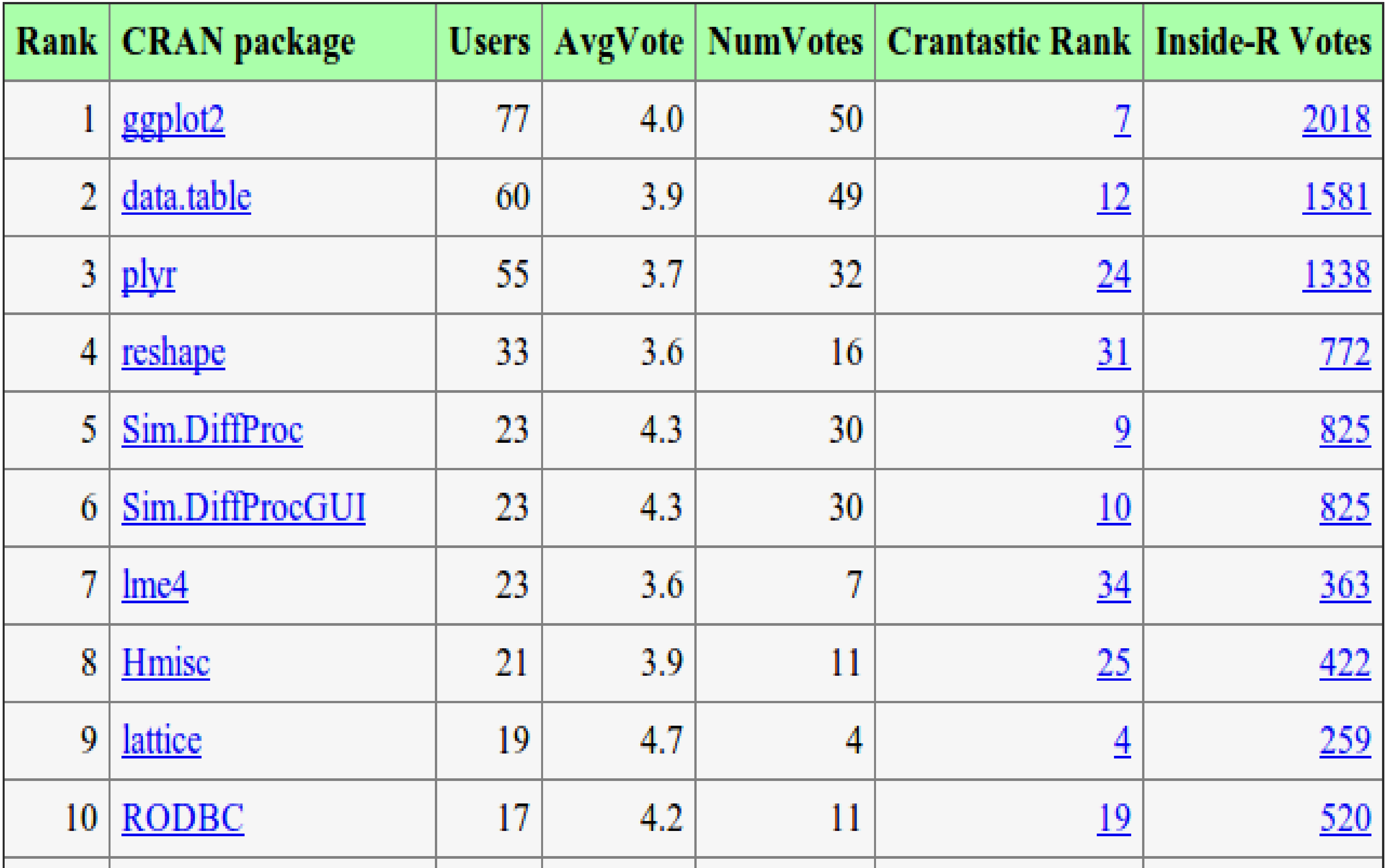
Since then



8,000
7,000
6,000
5,000
4,000
3,000
2,000
1,000
0
🟩 Page Loads
🟦 Unique Visits
🟧 Returning Visits
86
62
13
456
370
84
1,266
910
256
1,870
1,393
309
2,643
1,803
479
3,178
2,232
636
5,544
3,810
1,118
6,107
4,176
1,127
Q3 2010
Q4 2010
Q1 2011
Q2 2011
Q3 2011
Q4 2011
Q1 2012
Q2 2012

Rank all packages by users

Link



Rank	CRAN package	Users	AvgVote	NumVotes	Crantastic Rank	Inside-R Votes
1	ggplot2	77	4.0	50	7	2018
2	data.table	60	3.9	49	12	1581
3	plyr	55	3.7	32	24	1338
4	reshape	33	3.6	16	31	772
5	Sim.DiffProc	23	4.3	30	9	825
6	Sim.DiffProcGUI	23	4.3	30	10	825
7	lme4	23	3.6	7	34	363
8	Hmisc	21	3.9	11	25	422
9	lattice	19	4.7	4	4	259
10	RODBC	17	4.2	11	19	520

15 reviews

Link

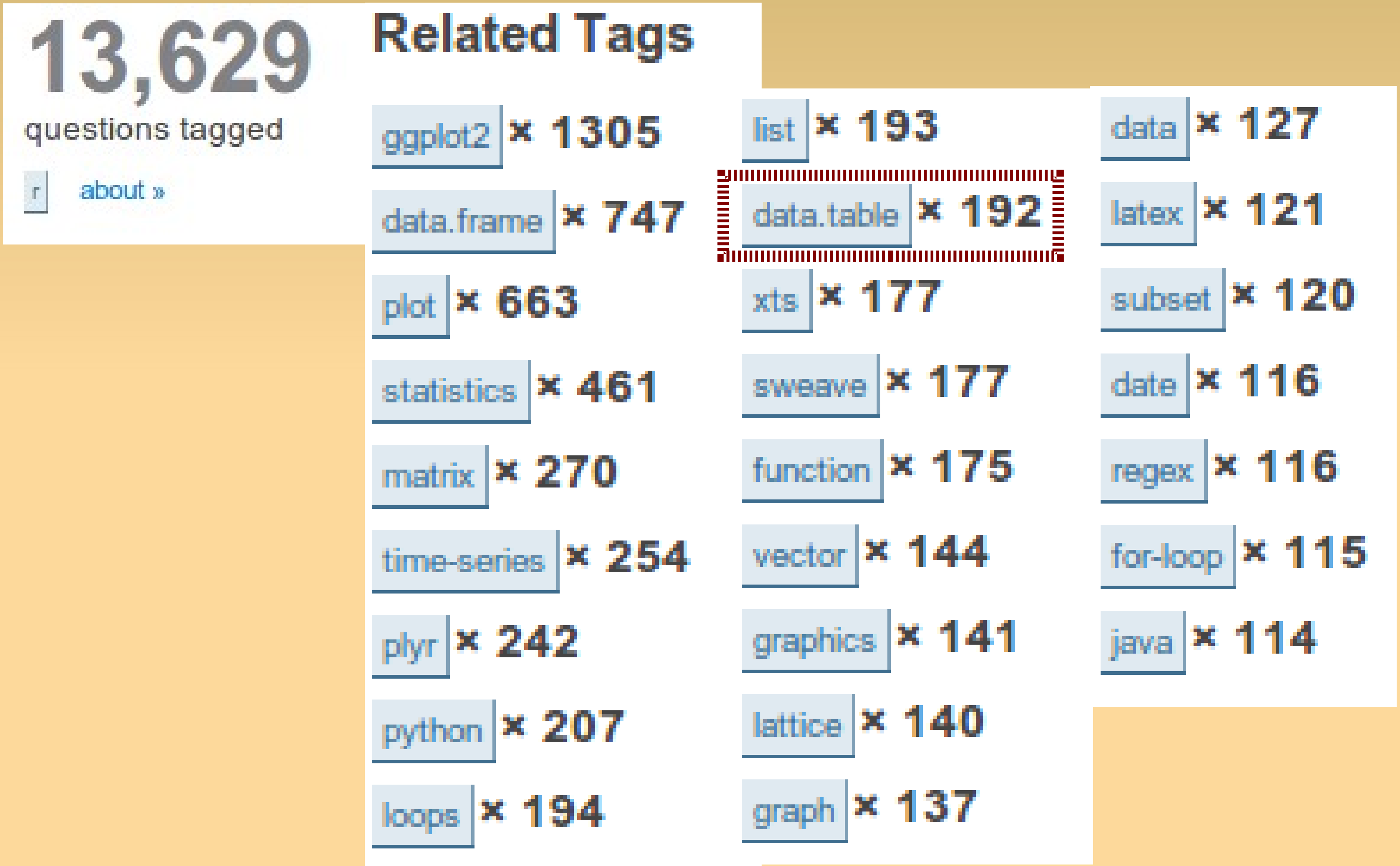
- ”It is much easier to subset, summarize, and investigate data.tables”
- ”Improves programming and computing speed”
- ”Great library for data.mining”
- ”data.table is a perfect combination of useability and speed”
- ”The more I use it, the better it gets”
- ”A very useful package!”
- ”The fast way to do SQL like operations in R”

Reviews continued

- ”data.table is fast compared to ddply and ave”
- ”data.table rocks!”
- ”Efficient and simple”
- ”I use it on a regular basis and I managed to cut computing time dramatically.”
- ”Amazing package!”
- ”Fast”
- ”I don't know where I would be w/o data.table”
- ”Fast splitting/sorting operations in frames”

Stack Overflow tag

Link



13,629

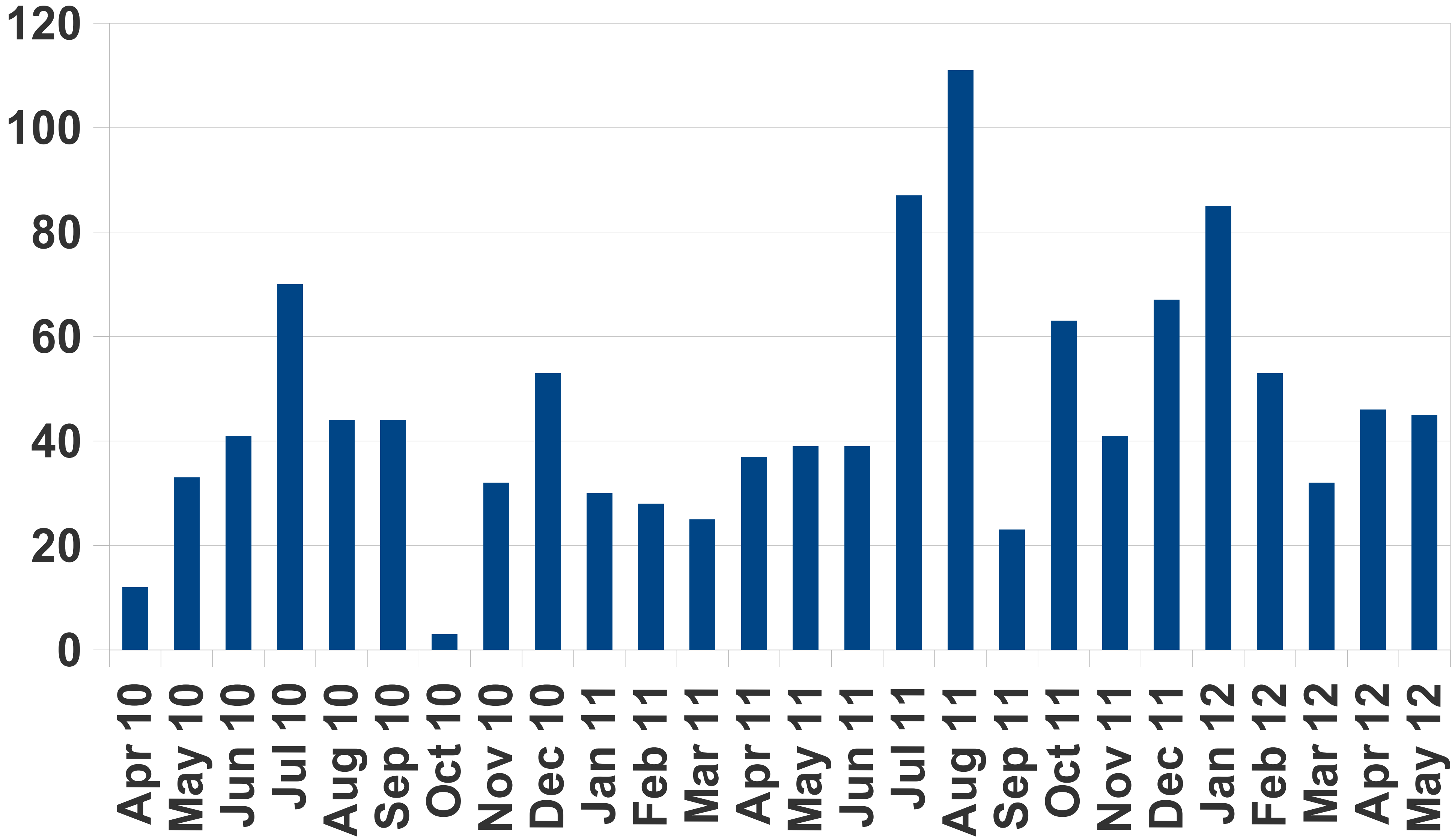
questions tagged

r about »

Related Tags

ggplot2	× 1305
data.frame	× 747
plot	× 663
statistics	× 461
matrix	× 270
time-series	× 254
plyr	× 242
python	× 207
loops	× 194
list	× 193
data.table	× 192
xts	× 177
sweave	× 177
function	× 175
vector	× 144
graphics	× 141
lattice	× 140
graph	× 137
data	× 127
latex	× 121
subset	× 120
date	× 116
regex	× 116
for-loop	× 115
java	× 114

datatable-help: posts per month



Other stats

- 24 articles ”data.table” on R-bloggers
- 108 bugs fixed, 5 outstanding
- 66 feature requests implemented, 64 left
- 191 items in NEWS 1.6.0-1.8.1
- 2,600 lines of R
- 2,000 lines of C
- 653 unit tests

First released Aug 2008

Thanked in NEWS

Chris Neff
Yike Lu
user1393348
Leon Baum
Michael Weylandt
Christoph Jaeckel
Muhammad Waliji
Josh O'Brien
Malcolm Cook
Joseph Voelkel
user1165199
Karl Ove Hufthammer
Ina
Prasad Chalasani
Helge Liebert
Dieter Menne
Yang Zhang
Steven Bagley
Ivan Zhang
Joshua Ulrich
Eric
DM
Damian Betebenner
Jim Holtman
Timothée Carayol
Johann Hibschman
Stavros Macrakis
Branson Owen
RYogi
Prof Brian Ripley
Ivo Welch
Simon Urbanek
Luke Tierney
Eugene Tyurin
Nicolas Servant
Jean-Francois Rami
Jelmer Ypma
Thell Fowler
Andreas Borg
gkaupas
Juliet Hannah
Iterator
Allan Engelhardt
Sean Creighton
ilprincipe
Dennis Murphy
Vanja
Alexander Peterhansl

In appearance order in NEWS. Special thanks to Chris Neff for weeks of help to solve difficult crash bug Jan 2012.

Hierarchical indexes

- 4 years old (released Aug 2008)
- setkey(DT, id, date)
- setkey(DT, category, id, date)
- DT[X] or merge(DT,X)
- But, it was integer columns only
- NEW : character and double now ok
fractional seconds in POSIXct ok

Assign to a subset

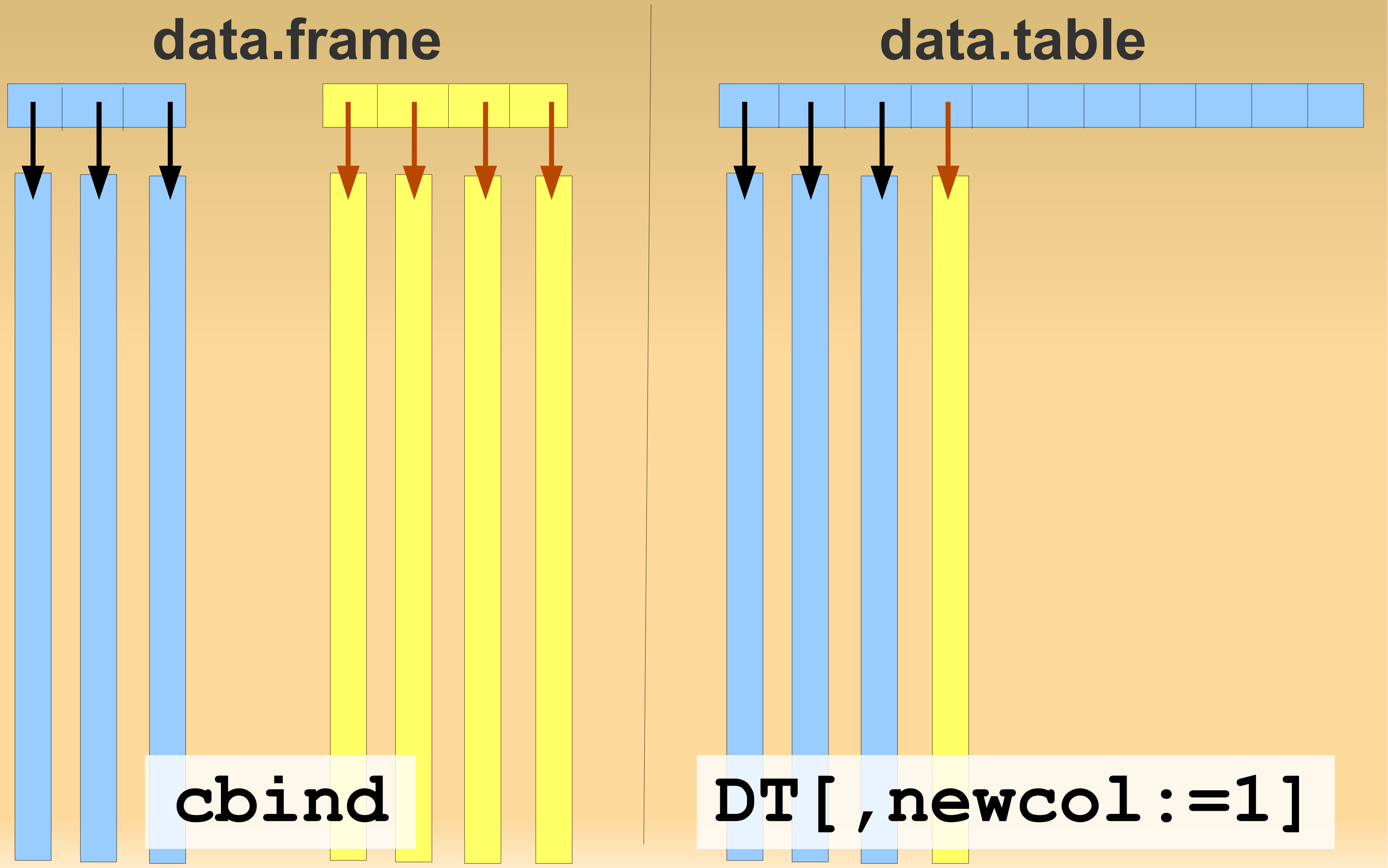
Link to question on S.O.

” In R I find myself doing something like this a lot:”

```
adataframe[adataframe$col==something]<-
adataframe[adataframe$col==something)]+1
```

- DT[col1==something, col2:=col3+1]
- Easy to write, easy to read

Over allocation (add by reference)



Delete in place

- DT[,colname:=NULL]
- Instant, regardless of size
- By reference, memmove internally
- Don't have to copy all but that column

copy()

- data.table IS copied-on-change by `<-` as usual
- No copy by `set*` functions (`setkey`, `setnames`, `setattr`)
- No copy by `:=`
- When you need a copy, call `copy(DT)`
- Why copy a 20GB data.table, even once.

Other

- Print method now prints head and tail
- Automatic optimization (sum -vs- mean)
- rbenchmark() replications default of 100 times overhead. Set to 1 and increase size of data, instead.
- Notice variable name repetition
- := by group now in v1.8.1

Analogous to SQL

- Link to data.table FAQ

DT[where,

select | update,

group by]

[having]

[order by]

[ ]...[ ]

Compound `[]` is key reason it's all inside `[.data.table`

Not (that) much to learn

- One manual page: `?data.table`
- Run `example(data.table)` at the prompt
- No methods, no functions, just use what you're used to in R

list columns

- Each cell can be a different type
- Each cell can be vector
- Each cell can itself be a data.table

- Combining list columns with i and by

list column example

```
data.table(x=letters[1:3],
 y=list(1:10,
        letters[1:4],
        data.table(a=1:3,b=4:6) ))
   x            y
1: a 1,2,3,4,5,6,
2: b      a,b,c,d
3: c <data.table>
```

Questions?

Suggestions?

Feedback?

Thank you!

Homepage