data.table

1 July 2014

useR! - Los Angeles

Matt Dowle

Some history

1996

I graduate in Maths and Computing

Start work at Lehman Brothers (investment bank), London

Technology :

VB/Excel and Sybase SQL

Mutiple users (clients) - Windows

One database (server) – Unix / Windows

1999

I move to Salomon Brothers (another investment bank), London

Day 1 and I meet Patrick Burns (author of S Poetry)

Pat: We use S-PLUS here.

Matt: What's S-PLUS?

Pat shows me S-PLUS

```
> DF <- data.frame(
          A = letters[1:3],
          B = c(1,3,5)       )
> DF
  A B
1 a 1
2 b 3
3 c 5
```

Already easier than SQL

Pat: It's a set of columns. All columns have the same length but can be different types.

Matt: So data.frame is like a database table?

Pat: Yes

Matt: Great. I get it. You didn't have to do CREATE TABLE first and then INSERT data?

Pat: Correct. It's one step.

Matt: Show me more!

Cool

Pat:

```
> DF[2:3,]
  A B
2 b 3
3 c 5
```

Matt: WOW! I don't need to create a column containing row numbers like I do in SQL?

Pat: Nope. The row order is how it's stored in memory. That's why it's good for time series.

My first thought

Matt: `DF[2:3, sum(B)]  # 3+5 == 8`

Pat: Ah, no.

Matt: Why not?

Pat: It's `sum(DF[2:3,"B"])`

Matt: Ok, but why not what I tried?

Pat: It doesn't work like that.

Matt: Why not?

Pat: Because it doesn't.

Matt: What does it do then?

Pat: Nothing, don't do it.

Matt: I tried it anyway. It's an error.

```
object 'B' not found
```

Pat: Yeah I told you not to do that.

Matt: Can we ask S-PLUS to change it?

Pat: Good luck with that.

Matt: Ok ok. I'll move on.

3 years pass, 2002

One day S-PLUS crashes

It's not my code, but a corruption in S-PLUS

Support: Are you sure it's not *your* code.

Matt: Yes. See, here's how you reproduce it.

Support: Yes, you're right. We'll fix it, thanks!

Matt: Great, when?

When

Support: Immediately for the next release.

Matt: Great, when's that?

Support: 6 months

Matt: Can you do a patch quicker?

Support: No because it's just you with the problem.

Matt: But I'm at Salomon/Citigroup, the biggest financial corporation in the world!

Support: True but it's still just you, Matt.

When continued

Matt: I understand. Can you send me the code and I'll fix it? I don't mind - I'll do it for free. I just want to fix it to get my job done.

Support: Sorry, can't do that. Lawyer says no.

Matt: Pat, any ideas?

Pat: Have you tried R?

Matt: What's R?

R in 2002

I took the code I had in S-PLUS and ran it in R.

Not only didn't it crash, but it took 1 minute instead of 1 hour.

R had improved the speed of `for` loops (*) and was in-memory rather than on-disk.

(*) The code generated random portfolios and couldn't be vectorized, due to its nature.

Even better

If R does error or crash, I can fix it. We have the source code! Or I can hire someone to fix it for me.

I can get my work done and not wait 6 months for a fix.

And it has **packages**.

I start to use R.

My first thought, again

Matt: Pat, remember how I first thought `[.data.frame` should work?

```
DF[2:3, sum(B)]
```

Pat: Good luck with that.

2004, day 1

I join a new firm and leave S-PLUS behind. Now use R only.

I create my own `[.data.frame` and make `sum(B)` work.

`DF[2:3, sum(B)]` is born.

Only possible because R (uniquely) has lazy evaluation.

2004, day 2

I do the same for `i`

DF[region=="U.S.", sum(population)]

2004, day 3

I realise I need group by :

DF[region=="U.S.", sum(population), by=State]

2004, day 4

I realise **chaining** comes for free:

```
DF[region=="U.S.", sum(population), by=State
   ][ order(-population), ]
```

2008

I release data.table as GPL:

DT[where, select, group by][...][...]

2011

I define `:=` in `j` to do assignment by reference, combined with subset and grouping

DT[where, select | update, group by][…][…]

From v1.6.3 NEWS :

```
for (i in 1:1000) DF[i,1] <- i       # 591s
for (i in 1:1000) DT[i,V1:=i]        #     1s
```

lapply and do.call running very slowly?

June 2012

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?

data.table answer

10

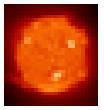
Since I'm in an evangelizing mood ... here's what the fast `data.table` solution would look like:

```
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
```

link | edit | flag

answered **Jun 15 at 16:14**



Josh O'Brien

20.4k ● 2 ● 14 ● 40

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

User's reaction

”data.table is awesome! That took about 3 seconds for the whole thing!!!”

Davy Kavanagh, 15 Jun 2012

Present day ...

Fast and friendly file reading

e.g. 50MB .csv, 1 million rows x 6 columns

```
read.csv("test.csv")              # 30-60s
read.csv("test.csv", colClasses=,
         nrows=, etc...)          #    10s
fread("test.csv")                 #     3s
```

e.g. 20GB .csv, 200 million rows x 16 columns

```
read.csv("big.csv", ...)          #  hours
fread("big.csv")                  #     8m
```

Update by reference using :=

Add new column ”sectorMCAP” by group :

```
DT[,sectorMCAP:=sum(MCAP),by=Sector]
```

Delete a column (0.00s even on a 20GB table) :

```
DT[,colToDelete:=NULL]
```

Be explicit to really copy entire 20GB :

```
DT2 = copy(DT)
```

data.table support

21	Last 7 Days	19% unanswered
85	Last 30 Days	15.3% unanswered
1,542	All Time	8.6% unanswered

roll = ”nearest”

x	y	value
A	2	1.1
A	9	1.2
A	11	1.3
B	3	1.4

```
setkey(DT, x, y)

DT[.(”A”,7), roll=”nearest”]
```

Not (that) much to learn

- Main manual page: `?data.table`
- Run `example(data.table)` at the prompt (53 examples)
- No methods, no functions, just use what you're used to in R

Thank you

https://github.com/Rdatatable/datatable/

http://stackoverflow.com/questions/tagged/data.table

3 hour data.table tutorial yesterday :

http://user2014.stat.ucla.edu/files/tutorial_Matt.pdf

```
> install.packages("data.table")
> require(data.table)
> ?data.table
> ?fread
```

Learn by example :

```
> example(data.table)
```