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
Multiple users (clients) - Windows
One database (server) – Unix / Windows
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
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!
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.
Matt: \texttt{DF[2:3, sum(B)]} \# 3+5 == 8

Pat: Ah, no.

Matt: Why not?

Pat: It's \texttt{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.
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.
*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.
Matt: Pat, remember how I first thought [.data.frame should work?

DF[2:3, sum(B)]

Pat: Good luck with that.
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) ]
I realise I need group by:

\[
\text{DF[ region=="U.S.", sum(population), by=State ]}
\]
I realise **chaining** comes for free:

```r
DF[region=="U.S.", sum(population), by=State ][ order(-population), ]
```
I release data.table as GPL:

```
DT[ where, select, group by ][ ... ][ ... ]
```
I define := in \( j \) to do assignment by reference, combined with subset and grouping.

\[
\text{DT[ where, select | update, group by ][ ... ][ ... ]}
\]

From v1.6.3 NEWS:

\[
\begin{align*}
\text{for (i in 1:1000) DF[i,1] <- i} & \quad \# 591s \\
\text{for (i in 1:1000) DT[i,V1:=i]} & \quad \# 1s
\end{align*}
\]
I have a data frame that is some 35,000 rows, by 7 columns. It looks like this:

```r
head(nuc)
```

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

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:

```r
cur 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),
              cds_length = sum(t$length)))
})
cur prof = do.call(rbind, cur 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

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

```r
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)]
```

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

```r
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

```r
read.csv("big.csv", ...)  # hours
fread("big.csv")  # 8m
```
Update by reference using :=

Add new column "sectorMCAP" by group:
\[DT[,sectorMCAP:=\text{sum(MCAP)},by=\text{Sector}]\]

Delete a column (0.00s even on a 20GB table):
\[DT[,\text{colToDelete:=NULL}]\]

Be explicit to really copy entire 20GB:
\[DT2 = \text{copy(DT)}\]
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”**

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>1.1</td>
</tr>
<tr>
<td>A</td>
<td>9</td>
<td>1.2</td>
</tr>
<tr>
<td>A</td>
<td>11</td>
<td>1.3</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>1.4</td>
</tr>
</tbody>
</table>

```plaintext
setkey(DT, x, y)
```

```plaintext
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 :

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

Learn by example :
> example(data.table)