

data.table

14 July 2014

R in Insurance, London

Session: R in a production environment

Matt Dowle

Some history

1996

I graduate in Maths and Computing

Start work at Lehman Brothers, London

Technology :

VB/Excel and Sybase SQL

Multiple users (clients) - Windows

One database (server) – Unix / Windows

1999

I move to Salomon Brothers, 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.

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.

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 to CRAN:

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
```



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?

data.table answer

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

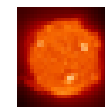
```
library(data.table)
dt <- data.table(nuc, key="gene_id")

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 |

[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 [was 30 mins] 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=,  
        rows=, 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)
```

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"]
```

+ forwards, backwards, limited and ends

Reducing programming time

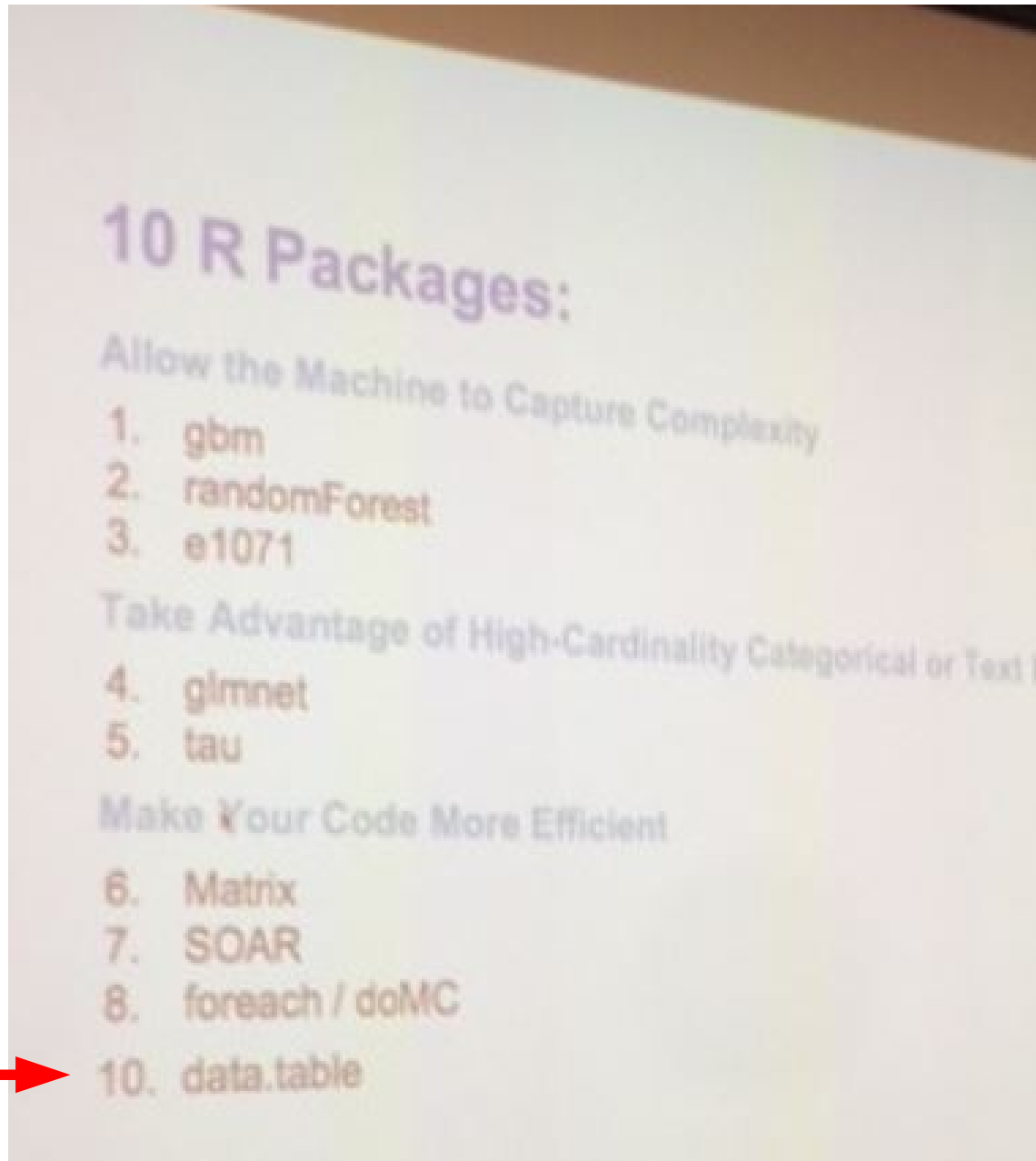
```
trades [  
    filledShares < orderedShares,  
    sum( (orderedShares-filledShares)  
        * orderPrice / fx ),  
    by = "date,region,algo"  
]
```

R : i j by

SQL : WHERE SELECT GROUP BY

"10 R packages to win Kaggle competitions", useR! 2014

By Xavier Conort
of DataRobot.com



data.table →

Why R?

- 1) R's lazy evaluation enables the syntax :
 - `DT[filledShares < orderedShares]`
 - query optimization before evaluation
- 2) Pass DT to any package taking DF. It works.
`is.data.frame(DT) == TRUE`
- 3) CRAN (cross platform release, quality control)
- 4) Thousands of statistical packages to use with `data.table`

data.table support

20

Last 7 Days

20% unanswered

80

Last 30 Days

16.3% unanswered

1,577

All Time

8.8% unanswered

As of 13 July 2014

Highest voted "unanswered"

Emacs tab auto-complete for R data.table?

▲ Anyone know how to get auto-complete working in Emacs for R data.tables (ess-mode)?

11 For example when I type tab below I'd like autocomplete to add "alpha"



```
DT <- data.table(alpha = 1:5)
DT[<type tab here>
```

thanks, jason

r emacs data.table

share | edit | close | flag | protect

edited Jan 28 at 6:51

asked Jan 28 at 1:32



JasonB

76 ● 1 ● 5

Comments usually "answer"



3 ↑ I've had the same request for Sublime Text. I would love to know if there is a way to make this happen –
Ricardo Saporta Jan 28 at 2:00

2 This seems hard to do because it's context sensitive. And to recognise the context (DT) you have parse code that isn't complete yet. Not impossible, but hard. – hadley Jan 28 at 17:27

1 Tab completion works for column names specified using the \$ syntax: DT\$<tab> will work. I don't think completion is available in ESS for DT[<tab>,] – Tyler Feb 10 at 15:38

Popular question 1

When should I use the := operator in data.table?



37



21



39



`data.table` objects now have a `:=` operator. What makes this operator different from all other assignment operators? Also, what are its uses, how much faster is it, and when should it be avoided?

r `data.table`

share | edit | close | flag | protect

asked Aug 11 '11 at 17:01

asked by [Ari D. Friedman](#)
24.4k ● 5 ● 74 ● 128

Here is an example showing 10 minutes reduced to 1 second (from NEWS on homepage). It's like subassigning to a `data.frame` but doesn't copy the entire table each time.

```
m = matrix(1, nrow=100000, ncol=100)
DF = as.data.frame(m)
DT = as.data.table(m)

system.time(for (i in 1:1000) DF[i,1] <- i)
  user system elapsed
287.062 302.627 591.984

system.time(for (i in 1:1000) DT[i,V1:=i])
  user system elapsed
 1.148   0.000   1.158   ( 511 times faster )
```

R 3.1 has largely solved this. Needs updating.

This is the main answer now. + loopable `set()` + combining `:=` with `i` and `by`

Putting the `:=` in `j` like that allows more idioms:

```
DT["a", done:=TRUE] # binary search for group 'a' and set a flag
DT[, newcol:=42]    # add a new column by reference (no copy of existing data)
DT[, col:=NULL]     # remove a column by reference
```

Popular question 2

How to delete a row by reference in R data.table?



40



15

My question is related to assignment by reference versus copying in data.table. I want to know if one can delete rows by reference, similar to

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

I want to know about

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

asked 2 years ago

viewed 6536 times

active 5 months ago



38



Good question. `data.table` can't delete rows by reference yet.

`data.table` can add and delete *columns* by reference since it over-allocates the vector of column pointers, as you know. The plan is to do something similar for rows and allow fast `insert` and `delete`. A row delete would use `memmove` in C to budge up the items (in each and every column) after the deleted rows. Deleting a row in the middle of the table would still be quite inefficient compared to a row store database such as SQL, which is more suited for fast insert and delete of rows wherever those rows are in the table. But still, it would be a lot faster than copying a new large object without the deleted rows.

Still true as it happens.
Always check dates.

answered May 29 '12 at 0:20



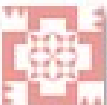

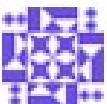

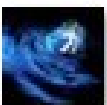
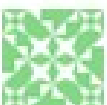




Matt Dowle





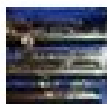




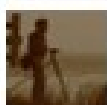

20.2k ● 4 ● 44 ● 87

data.table answerers

Last 30 Days

| | | | |
|----|----|---|---|
| 38 | 10 |  | Matt Dowle  20.2k ● 4 ● 44 ● 87 |
| 36 | 13 |  | Arun  38.2k ● 6 ● 41 ● 90 |
| 27 | 11 |  | eddi  17k ● 2 ● 15 ● 44 |
| 11 | 3 |  | Simon O'Hanlon 26.6k ● 3 ● 24 ● 59 |
| 10 | 2 |  | G. Grothendieck 39.9k ● 2 ● 31 ● 65 |
| 10 | 2 |  | Roland 32.2k ● 3 ● 24 ● 54 |
| 8 | 4 |  | Ananda Mahto 61.1k ● 7 ● 51 ● 108 |

All Time

| | | | |
|------|-----|---|--|
| 1.3k | 215 |  | Matt Dowle  20.2k ● 4 ● 44 ● 87 |
| 859 | 201 |  | Arun  38.2k ● 6 ● 41 ● 90 |
| 732 | 154 |  | mnel 46.4k ● 6 ● 72 ● 104 |
| 541 | 72 |  | Josh O'Brien 64.1k ● 3 ● 85 ● 161 |
| 474 | 172 |  | eddi  17k ● 2 ● 15 ● 44 |
| 258 | 79 |  | BondedDust 102k ● 3 ● 60 ● 142 |
| 248 | 68 |  | Ricardo Saporta  21.2k ● 2 ● 26 ● 56 |

Number of answers provided

 code contributors

Number of +1 votes for those answers

Testing

data.table has :

3,700 lines of R code

7,300 lines of C code

+ 3,400 tests

4,900 lines of test code

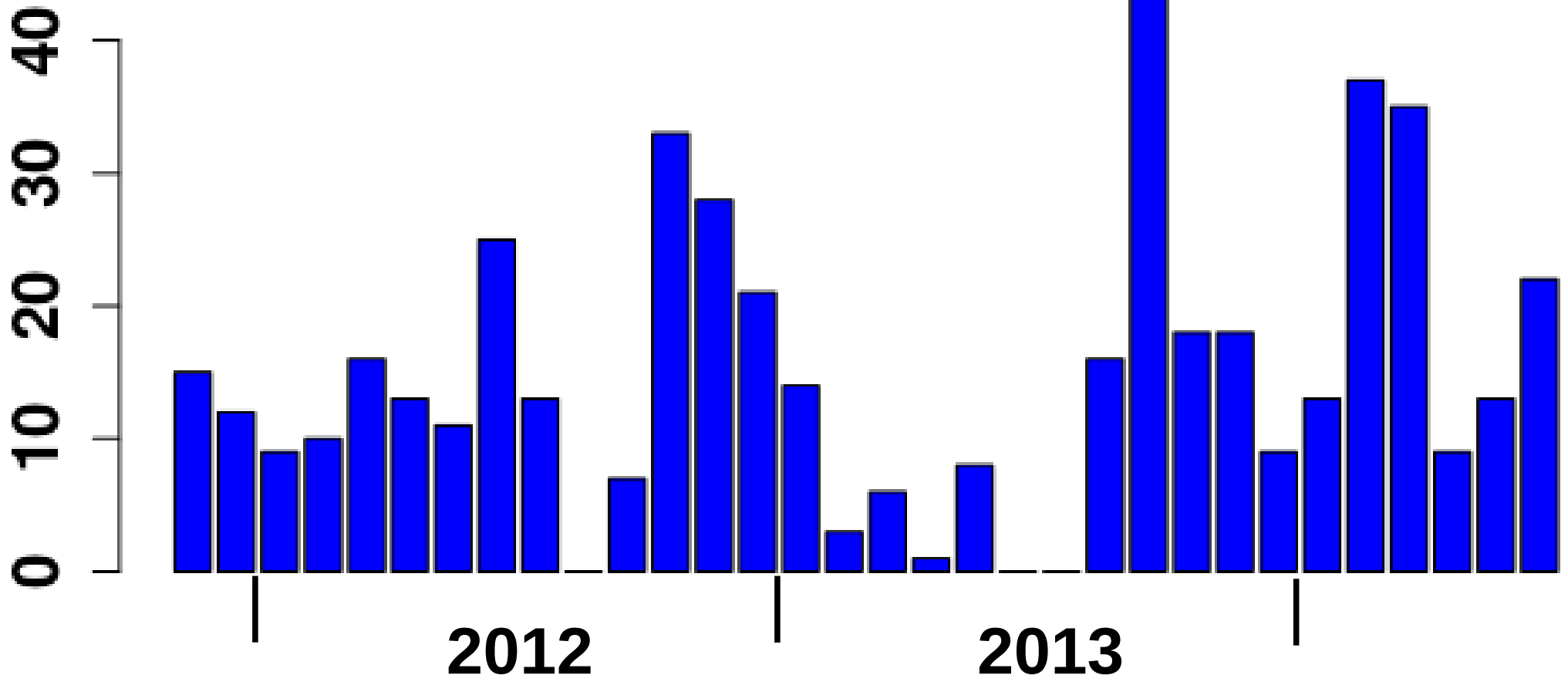
Run by CRAN every day

Includes tests with other packages

e.g. ggplot2, reshape

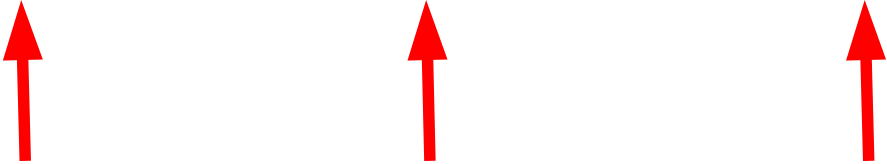
Tests are added *at the time*

Number of commits to tests, per month



Example tests

```
DT = data.table(a=1:6, b=7:12)
test(1348, DT[.N], DT[6])
test(1349, DT[.N-1:3], DT[5:3])
test(1350, DT[.N+1], DT[NA])
```


Test ID **this == that**

Whichever test framework you use, needs to be easy to read and easy to add new tests

Plus tests in dependent packages

- 44 CRAN packages

| | | | | |
|------------------|----------|-------------|-----------------|-----------|
| ALFQ | Causata | DataCombine | dplyr | ecoengine |
| edmr | eeptools | FAOSTAT | freqweights | gems |
| greport | IAT | installr | Kmisc | Lahman |
| lar | llcrc | LogisticDx | optiRum | psidR |
| RAPIDR | rbison | Rbitcoin | rfisheries | rgauges |
| rgbif | rnoaa | rplos | SciencesPo | sdcmicro |
| sdcmicro | SGP | simPH | spocc | survMisc |
| sweSCB | taxize | treebase | treemap | ttwa |
| benford.analysis | | randomNames | RecordLinkage | |
| ProjectTemplate | | CAGExploreR | splitstackshape | |

- 14 Bioconductor packages

| | | | |
|-----------|---------|---------------|----------|
| CAGER | COMPASS | flowWorkspace | GGtools |
| GOTHIC | MIMOSA | openCyto | phyloseq |
| QUALIFIER | R3Cseq | rBiopaxParser | rfPred |
| rTANDEM | RTN | | |

All tests run daily on Linux, Mac and Windows. Thanks to CRAN.

Last updated on 2014-07-13 16:46:54.

| | Flavor | Version | T_{install} | T_{check} | T_{total} | Status | Flags |
|--------------------------------------|---|---------|----------------------|--------------------|--------------------|--------|-------|
| R-devel latest daily commit | r-devel-linux-x86_64-debian-clang | 1.9.2 | 7.46 | 107.67 | 115.14 | OK | |
| | r-devel-linux-x86_64-debian-gcc | 1.9.2 | 10.45 | 108.56 | 119.01 | OK | |
| | r-devel-linux-x86_64-fedora-clang | 1.9.2 | | | 250.97 | OK | |
| | r-devel-linux-x86_64-fedora-gcc | 1.9.2 | | | 234.76 | OK | |
| | r-devel-osx-x86_64-clang | 1.9.2 | | | 198.47 | OK | |
| | r-devel-windows-ix86+x86_64 | 1.9.2 | 36.00 | 198.00 | 234.00 | OK | |
| | r-patched-linux-x86_64 | 1.9.2 | 10.29 | 108.39 | 118.68 | OK | |
| | r-patched-solaris-sparc | 1.9.2 | | | 587.80 | ERROR | |
| R-release now 3.1.1 | r-patched-solaris-x86 | 1.9.2 | | | 290.80 | OK | |
| | r-release-linux-ix86 | 1.9.2 | 12.56 | 130.71 | 143.26 | OK | |
| | r-release-linux-x86_64 | 1.9.2 | 10.26 | 106.58 | 116.84 | OK | |
| | r-release-osx-x86_64-mavericks | 1.9.2 | | | | OK | |
| | r-release-osx-x86_64-snowleopard | 1.9.2 | | | | OK | |
| | r-release-windows-ix86+x86_64 | 1.9.2 | 35.00 | 204.00 | 239.00 | OK | |
| Old=R3.0.3 | r-oldrel-windows-ix86+x86_64 | 1.9.2 | 36.00 | 217.00 | 253.00 | OK | |

Backwards compatibility

- *We sometimes* make backwards incompatible changes, *where it warrants*. **However ...**
- Long warning/deprecation period e.g. `rolltolast` was deprecated 6 March 2013. It still works now and will be just a warning in next release, over a year later. Use `rollends` instead. Read `README.md` on GitHub.
- `by-without-by` now `by=.EACHI` in dev. We'll provide an option to return the old behaviour.
- Require only R 2.14.0 (nearly 3 years old). We still don't use `paste0` internally since it was added later to R 2.15.0. Current R is 3.1.1.

Run tests yourself

```
require(data.table)
```

```
test.data.table()
```

```
...
```

```
All 3429 tests (last id = 1351.1)  
in inst/tests/tests.Rraw completed  
ok in 00:02:19
```

Maybe something in *your environment* causes some to fail.

Call `test.data.table()` at the start of your production code. **Create tests of *your* code and run them routinely, too.**

Test frameworks

- `data.table` has its own `test()` function for flexibility, written 10 years ago. Also uses `testthat` for S4 tests.
- **`testthat`** by Hadley Wickham used by over 250 packages
- **`RUnit`** by Burger, Juenemann and Koenig used by over 100 packages
- **`svUnit`** by Philippe Grosjean not as widely used but definitely worth reviewing
- R's built-in method: `tests/*.R > *.Rout` compared using `diff` to corresponding `*.Rout.save`

testCoverage

Which lines of source code *don't* the tests test?
(of course if no tests, then 0% coverage)

New package from Mango Solutions.

R code coverage, not C code yet.

Important statistic to publish. This might encourage packages to add tests.

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

Further reading

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

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

3 hour data.table tutorial at useR! 2014, Los Angeles:

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)
```

By the way: H2O

Flagged by John Chambers at useR! 2014

Machine learning on very large in-memory clusters e.g. 1TB+ RAM

e.g. GLM, GBM, PCA, SVM, Random forest, K-means, MCMC, and more.

Applications in insurance? :

- Internet-of-things; e.g. vehicle telematics, health monitoring
- weather forecasting and impact
- fraudulent transactions / claims

Open-source! <http://0xdata.com/>