data.table: data.frame 2.0
A better kind of data.frame

Mick Cooney

18 Feb 2015

Introduction

`data.table` (DT) is very similar to a `data.frame` (DF)

Introduction

`data.table` (DT) is very similar to a `data.frame` (DF)

```
> DF = data.frame(x = 1:3, y = c('a', 'b', 'c'));
> print(DF)
  x y
1 1 a
2 2 b
3 3 c
```

```
> library(data.table)
data.table 1.8.2  For help type: help(``data.table'')
> DT = data.table(x = 1:3, y = c('a', 'b', 'c'));
> print(DT)
   x y
1: 1 a
2: 2 b
3: 3 c
```

Introduction

`data.table` (DT) is very similar to a `data.frame` (DF)

```
> DF = data.frame(x = 1:3, y = c('a', 'b', 'c'));
> print(DF)
  x y
1 1 a
2 2 b
3 3 c
```

```
> library(data.table)
data.table 1.8.2  For help type: help(``data.table'')
> DT = data.table(x = 1:3, y = c('a', 'b', 'c'));
> print(DT)
   x y
1: 1 a
2: 2 b
3: 3 c
```

Note the colon (':') after row number

DT inherits from DF, so DF $\rightarrow$ DT easy.

DT inherits from DF, so DF → DT easy.

```
> mtcars.dt <- data.table(mtcars)
> mtcars.dt
     mpg cyl  disp  hp drat    wt  qsec vs am gear carb
 1: 21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
 2: 21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
 3: 22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
 4: 21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
 5: 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
 6: 18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
 7: 14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
 8: 24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
 9: 22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
10: 19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
11: 17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
12: 16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
13: 17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
14: 15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
15: 10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
16: 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
17: 14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
18: 32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
19: 30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
20: 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
21: 21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
22: 15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
23: 15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
24: 13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
25: 19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
26: 27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
27: 26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
28: 30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
29: 15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
30: 19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
31: 15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
```

`data.table` truncates long output

data.table truncates long output

```
> library(ggplot2)
> diamonds.dt <- data.table(diamonds)
> diamonds.dt
       carat       cut color clarity depth table price    x    y    z
    1:  0.23     Ideal     E     SI2  61.5    55   326 3.95 3.98 2.43
    2:  0.21   Premium     E     SI1  59.8    61   326 3.89 3.84 2.31
    3:  0.23      Good     E     VS1  56.9    65   327 4.05 4.07 2.31
    4:  0.29   Premium     I     VS2  62.4    58   334 4.20 4.23 2.63
    5:  0.31      Good     J     SI2  63.3    58   335 4.34 4.35 2.75
   ---
53936:  0.72     Ideal     D     SI1  60.8    57  2757 5.75 5.76 3.50
53937:  0.72      Good     D     SI1  63.1    55  2757 5.69 5.75 3.61
53938:  0.70 Very Good     D     SI1  62.8    60  2757 5.66 5.68 3.56
53939:  0.86   Premium     H     SI2  61.0    58  2757 6.15 6.12 3.74
53940:  0.75     Ideal     D     SI2  62.2    55  2757 5.83 5.87 3.64
```

Row/Column referencing slightly different:

Row/Column referencing slightly different:

```
> diamonds.dt[1:5]
   carat     cut color clarity depth table price    x    y    z
1:  0.23   Ideal     E     SI2  61.5    55   326 3.95 3.98 2.43
2:  0.21 Premium     E     SI1  59.8    61   326 3.89 3.84 2.31
3:  0.23    Good     E     VS1  56.9    65   327 4.05 4.07 2.31
4:  0.29 Premium     I     VS2  62.4    58   334 4.20 4.23 2.63
5:  0.31    Good     J     SI2  63.3    58   335 4.34 4.35 2.75
```

```
> diamonds.dt[, list(carat, cut, table)]
       carat       cut table
    1:  0.23     Ideal    55
    2:  0.21   Premium    61
    3:  0.23      Good    65
    4:  0.29   Premium    58
    5:  0.31      Good    58
   ---
53936:  0.72     Ideal    57
53937:  0.72      Good    55
53938:  0.70 Very Good    60
53939:  0.86   Premium    58
53940:  0.75     Ideal    55
```

Row/Column referencing slightly different:

```
> diamonds.dt[1:5]
   carat     cut color clarity depth table price    x    y    z
1:  0.23   Ideal     E     SI2  61.5    55   326 3.95 3.98 2.43
2:  0.21 Premium     E     SI1  59.8    61   326 3.89 3.84 2.31
3:  0.23    Good     E     VS1  56.9    65   327 4.05 4.07 2.31
4:  0.29 Premium     I     VS2  62.4    58   334 4.20 4.23 2.63
5:  0.31    Good     J     SI2  63.3    58   335 4.34 4.35 2.75
```

```
> diamonds.dt[, list(carat, cut, table)]
       carat       cut table
    1:  0.23     Ideal    55
    2:  0.21   Premium    61
    3:  0.23      Good    65
    4:  0.29   Premium    58
    5:  0.31      Good    58
   ---
53936:  0.72     Ideal    57
53937:  0.72      Good    55
53938:  0.70 Very Good    60
53939:  0.86   Premium    58
53940:  0.75     Ideal    55
```

Note syntax of columns — colnames not quoted (more later)

Why use data.table()?

Why bother?

Why use data.table()?

Why bother?

Three main reasons:

Why use data.table()?

Why bother?

Three main reasons:

- More efficient memory use

Why use data.table()?

Why bother?

Three main reasons:

- More efficient memory use
- Faster merging

Why use data.table()?

Why bother?

Three main reasons:

- More efficient memory use
- Faster merging
- $[i, j]$ syntax incredibly powerful

Keys

DT can have key

Keys

DT can have key

- Index in SQL DBs

Keys

DT can have key

- Index in SQL DBs
- Sorted by index

Keys

DT can have key

- Index in SQL DBs
- Sorted by index
- Only one per DT

Keys

DT can have key

- Index in SQL DBs
- Sorted by index
- Only one per DT
- Binary search

Keys

DT can have key

- Index in SQL DBs
- Sorted by index
- Only one per DT
- Binary search

Huge speedup over vector scan

Merging Tables

Inspired by `A[B]` in base R (`A` matrix, `B` a 2-col matrix)

```
> X  ### 'a' is the key
     a  b
 1:  1  2
 2:  2  4
 3:  3  6
 4:  4  8
 5:  5 10
 6:  6 12
 7:  7 14
 8:  8 16
 9:  9 18
10: 10 20
```

```
> Y  ### 'a' is the key
     a    c
 1:  3    2
 2:  6    4
 3:  9    8
 4: 12   16
 5: 15   32
 6: 18   64
 7: 21  128
 8: 24  256
 9: 27  512
10: 30 1024
```

```
> X[Y]
     a  b    c
 1:  3  6    2
 2:  6 12    4
 3:  9 18    8
 4: 12 NA   16
 5: 15 NA   32
 6: 18 NA   64
 7: 21 NA  128
 8: 24 NA  256
 9: 27 NA  512
10: 30 NA 1024
```

```
> Y[X]
     a  c  b
 1:  1 NA  2
 2:  2 NA  4
 3:  3  2  6
 4:  4 NA  8
 5:  5 NA 10
 6:  6  4 12
 7:  7 NA 14
 8:  8 NA 16
 9:  9  8 18
10: 10 NA 20
```

```
> merge(X, Y)
   a  b c
1: 3  6 2
2: 6 12 4
3: 9 18 8
```

DT[i, j, by = x]

```
DT[where, select | update, group-by] [having]
[order by]
```

DT[i, j, by = x]

```
DT[where, select | update, group-by] [having]
[order by]
```

- SQL-like syntax

DT[i, j, by = x]

```
DT[where, select | update, group-by] [having]
[order by]
```

- SQL-like syntax
- Probably most powerful feature

DT[i, j, by = x]

```
DT[where, select | update, group-by] [having]
[order by]
```

- SQL-like syntax
- Probably most powerful feature
- *i* is row-select, *j* is row-output

DT[i, j, by = x]

```
DT[where, select | update, group-by] [having]
[order by]
```

- SQL-like syntax
- Probably most powerful feature
- *i* is row-select, *j* is row-output
- 'by =' allows for grouping

```
> X <- data.table(grp = c("a","a","b","b","b","c","c"), foo = 1:7, key = 'grp');
> X
   grp foo
1:   a   1
2:   a   2
3:   b   3
4:   b   4
5:   b   5
6:   c   6
7:   c   7

> Y <- data.table(c("b","c"), bar = c(4,2))
> Y
   V1 bar
1:  b   4
2:  c   2
```

```
> X[Y]
   grp foo bar
1:   b   3   4
2:   b   4   4
3:   b   5   4
4:   c   6   2
5:   c   7   2
```

```
> X[Y, sum(foo * bar)]
   grp V1
1:   b 48
2:   c 26
```

```
> X[Y, list(val = sum(foo * bar))]
   grp val
1:   b  48
2:   c  26
```

In-place Assignment

Easy to efficiently add and remove columns

In-place Assignment

Easy to efficiently add and remove columns

```
> test.dt <- data.table(x = 1:3, y = (1:3)^2)
> test.dt
   x y
1: 1 1
2: 2 4
3: 3 9
```

In-place Assignment

Easy to efficiently add and remove columns

```
> test.dt <- data.table(x = 1:3, y = (1:3)^2)
> test.dt
   x y
1: 1 1
2: 2 4
3: 3 9


> test.dt[, z := x * 2]
> test.dt
   x y z
1: 1 1 2
2: 2 4 4
3: 3 9 6
```

In-place Assignment

Easy to efficiently add and remove columns

```
> test.dt <- data.table(x = 1:3, y = (1:3)^2)
> test.dt
   x y
1: 1 1
2: 2 4
3: 3 9
```

```
> test.dt[, z := x * 2]
> test.dt
   x y z
1: 1 1 2
2: 2 4 4
3: 3 9 6
```

```
> test.dt[, z := NULL]
> test.dt
   x y
1: 1 1
2: 2 4
3: 3 9
```

Gotchas with data.table()

DTs inherit from DF

Gotchas with data.table()

DTs inherit from DF

- Not completely interchangeable for DF

Gotchas with data.table()

DTs inherit from DF

- Not completely interchangeable for DF
- Compatibility usually fixed by DF casting

Gotchas with data.table()

DTs inherit from DF

- Not completely interchangeable for DF
- Compatibility usually fixed by DF casting
- Binary search relies on keys

Gotchas with data.table()

DTs inherit from DF

- Not completely interchangeable for DF
- Compatibility usually fixed by DF casting
- Binary search relies on keys
- Proper use needs to be coded (e.g. `setkey()` function)

Gotchas with data.table()

DTs inherit from DF

- Not completely interchangeable for DF
- Compatibility usually fixed by DF casting
- Binary search relies on keys
- Proper use needs to be coded (e.g. `setkey()` function)
- Code breaks (1.8.x particularly notorious)

Gotchas with data.table()

DTs inherit from DF

- Not completely interchangeable for DF
- Compatibility usually fixed by DF casting
- Binary search relies on keys
- Proper use needs to be coded (e.g. `setkey()` function)
- Code breaks (1.8.x particularly notorious)
- New functionality detailed in NEWS file

How to Get Started

data.table page on R-forge
(`datatable.r-forge.r-project.org`)

How to Get Started

data.table page on R-forge
(`datatable.r-forge.r-project.org`)

- Documentation biggest issue

How to Get Started

data.table page on R-forge
(`datatable.r-forge.r-project.org`)

- Documentation biggest issue
- Read quick introduction vignette (bit lightweight)

How to Get Started

data.table page on R-forge
(`datatable.r-forge.r-project.org`)

- Documentation biggest issue
- Read quick introduction vignette (bit lightweight)
- FAQ is main manual

How to Get Started

data.table page on R-forge
(`datatable.r-forge.r-project.org`)

- Documentation biggest issue
- Read quick introduction vignette (bit lightweight)
- FAQ is main manual
- Good questions on StackOverflow (`stackoverflow.com`)
  – tag `data.table`

How to Get Started

data.table page on R-forge (`datatable.r-forge.r-project.org`)

- Documentation biggest issue
- Read quick introduction vignette (bit lightweight)
- FAQ is main manual
- Good questions on StackOverflow (`stackoverflow.com`) – tag `data.table`
- As of Feb 2015, proper vignettes being written (and look good)

Summary

Summary

`data.table` is the schnizzle