Welcome to the course!

DATA MANIPULATION WITH DATA.TABLE IN R

Matt Dowle and Arun Srinivasan
Instructors, DataCamp

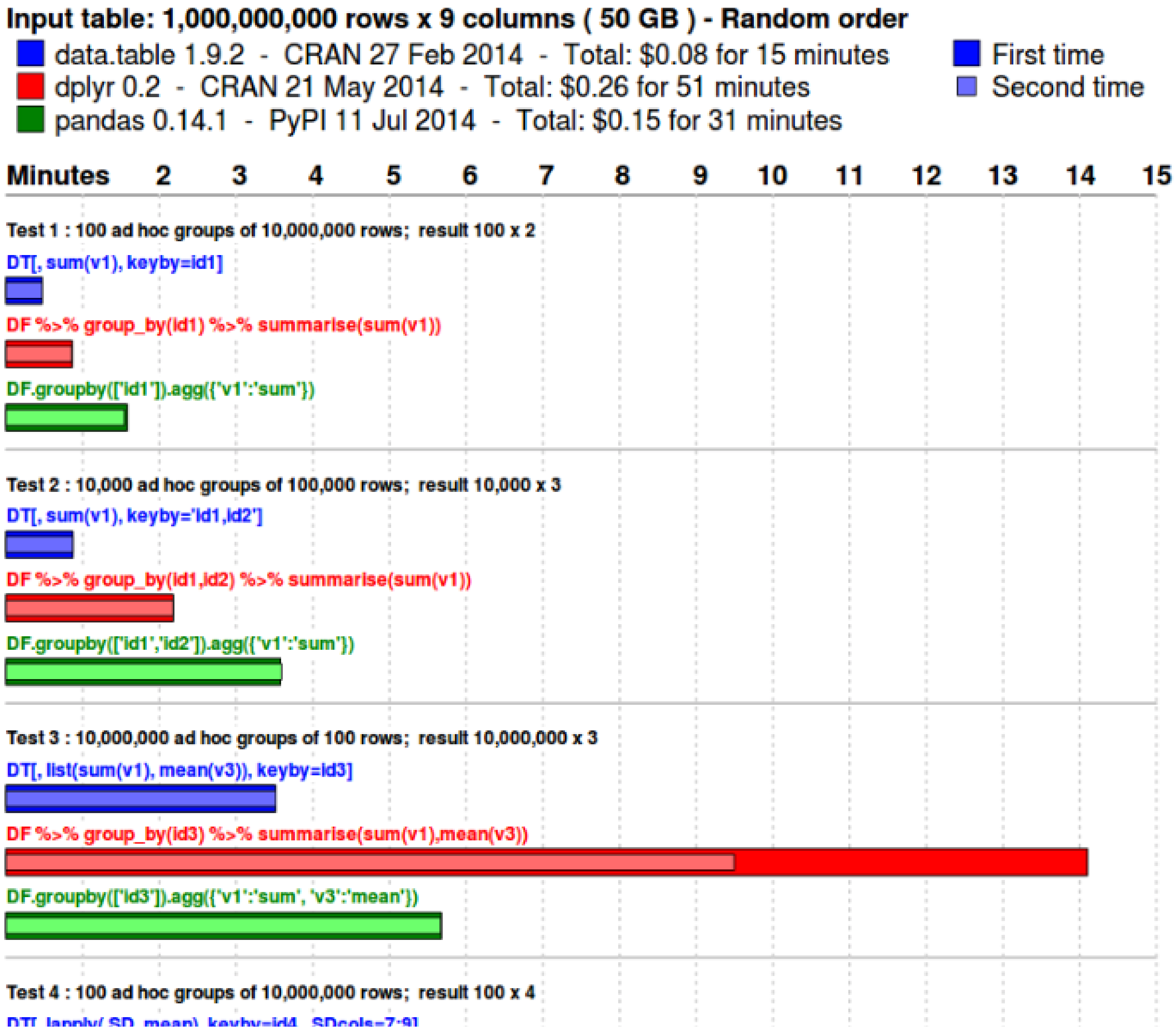
What is a data.table?

- Enhanced `data.frame`
    - Inherits from and extends `data.frame`
- Columnar data structure
- Every column must be of same length but can be of different type

Why use data.table?

- Concise and consistent syntax
  - Think in terms of `rows`, `columns` and `groups`
  - Provides a *placeholder* for each

```
# General form of data.table syntax
DT[i, j, by]
   |  |  |
   |  |  --> grouped by what?
   |  -----> what to do?
   --------> on which rows?
```


Input table: 1,000,000,000 rows x 9 columns (50 GB) - Random order
data.table 1.9.2 - CRAN 27 Feb 2014 - Total: $0.08 for 15 minutes
dplyr 0.2 - CRAN 21 May 2014 - Total: $0.26 for 51 minutes
pandas 0.14.1 - PyPI 11 Jul 2014 - Total: $0.15 for 31 minutes
First time
Second time
Minutes 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Test 1 : 100 ad hoc groups of 10,000,000 rows; result 100 x 2
DT[, sum(v1), keyby=id1]
DF %>% group_by(id1) %>% summarise(sum(v1))
DF.groupby(['id1']).agg({'v1':'sum'})
Test 2 : 10,000 ad hoc groups of 100,000 rows; result 10,000 x 3
DT[, sum(v1), keyby='id1,id2']
DF %>% group_by(id1,id2) %>% summarise(sum(v1))
DF.groupby(['id1','id2']).agg({'v1':'sum'})
Test 3 : 10,000,000 ad hoc groups of 100 rows; result 10,000,000 x 3
DT[, list(sum(v1), mean(v3)), keyby=id3]
DF %>% group_by(id3) %>% summarise(sum(v1),mean(v3))
DF.groupby(['id3']).agg({'v1':'sum', 'v3':'mean'})
Test 4 : 100 ad hoc groups of 10,000,000 rows; result 100 x 4
DT[, lapply(.SD, mean), keyby=id4, .SDcols=7:9]

Why use data.table?

- Feature-rich
  - Parallelization
  - Fast updates *by reference*
  - Powerful joins (**Joining Data with data.table in R**)

Creating a data.table

Three ways of creating data tables:

- `data.table()`
- `as.data.table()`
- `fread()`

Creating a data.table

```
library(data.table)
x_df <- data.frame(id = 1:2, name = c("a", "b"))
x_df
```

```
id name
 1    a
 2    b
```

```
x_dt <- data.table(id = 1:2, name = c("a", "b"))
x_dt
```

```
id name
 1    a
 2    b
```

Creating a data.table

```r
y <- list(id = 1:2, name = c("a", "b"))
y
```

```
$id
1 2
$name
"a" "b"
```

```r
x <- as.data.table(y)
x
```

```
id name
 1    a
 2    b
```

data.tables and data.frames (I)

Since a data.table *is* a data.frame ...

```
x <- data.table(id = 1:2,
                name = c("a", "b"))
x
```

```
id name
 1    a
 2    b
```

```
class(x)
```

```
"data.table" "data.frame"
```

data.tables and data.frames (II)

Functions used to query data.frames also work on data.tables

```
nrow(x)
```

```
2
```

```
ncol(x)
```

```
2
```

```
dim(x)
```

```
2 2
```

data.tables and data.frames (III)

A data table never automatically converts character columns to factors

```
x_df <- data.frame(id = 1:2, name = c("a", "b"))
class(x_df$name)
```

```
"factor"
```

```
x_dt <- data.table(id = 1:2, name = c("a", "b"))
class(x_dt$name)
```

```
"character"
```

data.tables and data.frames (IV)

Never sets, needs or uses *row names*

```
rownames(x_dt) <- c("R1", "R2")
x_dt
```

```
   id name
1:  1    a
2:  2    b
```

Let's practice!

DATA MANIPULATION WITH DATA.TABLE IN R

Filtering rows in a data.table

DATA MANIPULATION WITH DATA.TABLE IN R

Matt Dowle and Arun Srinivasan

Instructors, DataCamp

General form of data.table syntax

First argument `i` is used to *subset* or *filter* rows

```
# General form of data.table syntax
DT[i, j, by]
   |  |  |
   |  |  --> grouped by what?
   |  -----> what to do?
   --------> on which rows?
```

Row numbers

```
# Subset 3rd and 4th rows from batrips
batrips[3:4]

# Same as
batrips[3:4, ]
```

```
# Subset everything except first five rows
batrips[-(1:5)]

# Same as
batrips[!(1:5)]
```

Special symbol .N

- `.N` is an integer value that contains the number of rows in the data.table
- Useful alternative to `nrow(x)` in `i`

```
nrow(batrips)
```

```
326339
```

```
batrips[326339]
```

```
trip_id duration
588914       364
```

```
# Returns the last row
batrips[.N]
```

```
trip_id duration
588914       364
```

```
# Return all but the last 10 rows
ans <- batrips[1:(.N-10)]
nrow(ans)
```

```
326329
```

Logical expressions (I)

```
# Subset rows where subscription_type is "Subscriber"
batrips[subscription_type == "Subscriber"]

# If batrips was only a data frame
batrips[batrips$subscription_type == "Subscriber", ]
```

Logical expressions (II)

```
# Subset rows where start_terminal = 58 and end_terminal is not 65
batrips[start_terminal == 58 & end_terminal != 65]

# If batrips was only a data frame
batrips[batrips$start_terminal == 58 & batrips$end_terminal != 65]
```

Logical expressions (III)

Optimized using secondary indices for speed automatically

```
set.seed(1)
dt <- data.table(x = sample(10000, 10e6, TRUE),
                 y = sample(letters, 1e6, TRUE))
indices(dt)
```

```
NULL
```

```
# 0.207s on first run
#(time to create index + subset)
system.time(dt[x == 900])
```

```
user   system elapsed
0.207    0.015    0.226
```

```
indices(dt)
```

```
"x"
```

```
# 0.002s on subsequent runs
#(instant subset using index)
system.time(dt[x == 900])
```

```
user   system elapsed
0.002    0.000    0.002
```

Let's practice!

DATA MANIPULATION WITH DATA.TABLE IN R

Helpers for filtering

DATA MANIPULATION WITH DATA.TABLE IN R

Matt Dowle and Arun Srinivasan

Instructors, DataCamp

%like%

- `%like%` allows you to search for a *pattern* in a *character* or a *factor* vector
  - Usage: `col %like% pattern`

```
# Subset all rows where start_station starts with San Francisco
batrips[start_station %like% "^San Francisco"]


# Instead of
batrips[grepl("^San Francisco", start_station)]
```

%between%

- `%between%` allows you to search for values in the closed interval `[val1, val2]`
  - Usage: `numeric_col %between% c(val1, val2)`

```
# Subset all rows where duration is between 2000 and 3000
batrips[duration %between% c(2000, 3000)]


# Instead of
batrips[duration >= 2000 & duration <= 3000]
```

%chin%

- `%chin%` is similar to `%in%`, but it is *much* faster and only for character vectors
    - Usage: `character_col %chin% c("val1", "val2", "val3")`

```
# Subset all rows where start_station is
# "Japantown", "Mezes Park" or "MLK Library"
batrips[start_station %chin% c("Japantown", "Mezes Park", "MLK Library")]

# Much faster than
batrips[start_station %in% c("Japantown", "Mezes Park", "MLK Library")]
```

Let's practice!