Big data pipelines

Hadley Wickham
@hadleywickham
Chief Scientist, RStudio

February 2015
What is big data?
<table>
<thead>
<tr>
<th>Size</th>
<th>Can’t fit in memory on one computer:</th>
<th>Fits in memory on a server:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Big</td>
<td>$&gt;1$ TB</td>
<td>$10$ GB-$1$ TB</td>
</tr>
<tr>
<td>Medium</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small</td>
<td></td>
<td>$&lt;10$ GB</td>
</tr>
</tbody>
</table>
The big data mirage
1. Can be reduced to a small/medium data problem with subsetting/sampling/summarising (90%)

2. Can be reduced to a very large number of small data problems (9%)

3. Is irreducibly big (1%)
Small data is still big!

<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doubles</td>
<td>$2 \times 10^9$</td>
<td>200 vars, 10 million obs</td>
</tr>
<tr>
<td>Integers</td>
<td>$4 \times 10^9$</td>
<td>40 vars, 100 million obs</td>
</tr>
<tr>
<td>Characters</td>
<td>$16 \times 10^9$</td>
<td>5,000 copies of war and peace</td>
</tr>
</tbody>
</table>
Pipelines
Think it

Describe it (precisely)

Do it

Cognitive

Computational
Cognition time » Computation time
Tidy

Transform

Visualise
Surprises, but doesn't scale

Model
Scales, but doesn't (fundamentally) surprise
x %>% f(y)
# f(x, y)

x %>% f(z, .)
# f(z, x)

x %>% f(y) %>% g(z)
# g(f(x, y), z)

# Turns function composition (hard to read)
# into sequence (easy to read)
foo_foo <- little_bunny()

bop_on(
  scoop_up(
    hop_through(foo_foo, forest),
    field_mouse
  ),
  head
)

# vs
foo_foo %>%
  hop_through(forest) %>%
  scoop_up(field_mouse) %>%
  bop_on(head)
Any function can use it. Only needs a simple property: the type of the first argument needs to be the same as the type of the result.

tidyr: pipelines for messy -> tidy data
dplyr: pipelines for manipulation of tidy data
ggvis: pipelines for visualisations

rvest: pipelines for html/xml DOMs
purrr: pipelines for lists
tidyr
**Tidy data** = data that makes data analysis easy

<table>
<thead>
<tr>
<th>Storage</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table / File</td>
<td>Data set</td>
</tr>
<tr>
<td>Rows</td>
<td>Observations</td>
</tr>
<tr>
<td>Columns</td>
<td>Variables</td>
</tr>
</tbody>
</table>
```r
library(tidyr)
library(dplyr, warn = FALSE)

# Load the data

tb <- tbl_df(read.csv("tb.csv", stringsAsFactors = FALSE))

# Display the data

tb
```

What are the variables in this dataset? (Hint: f = female, u = unknown, 1524 = 15-24)
# To convert this messy data into tidy data
# we need two verbs. First we need to **gather**
# together all the columns that aren't variables

tb2 <- tb %>%
gather(demo, n, -iso2, -year, na.rm = TRUE)
tb2
# Then **separate** the demographic variable into
# sex and age

tb3 <- tb2 %>%
  separate(demo, c("sex", "age"), 1)
tb3

# tidyr provides a few other useful verbs:
# **spread** (opposite of gather)
# **extract** (like separate, but uses regexp groups)
# **unite** (opposite of extract/gather)
# **nest** & **unnest**, ...
Google for “"tidyr"” & “"tidy data""
dplyr
Think it

Describe it
(precisely)

Do it

Cognitive

Computational
One table verbs

- **select**: subset variables by name
- **filter**: subset observations by value
- **mutate**: add new variables
- **summarise**: reduce to a single row
- **arrange**: re-order the rows

+ group by
Demo
<table>
<thead>
<tr>
<th>Mutating</th>
<th>Filtering</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>inner_join()</td>
<td>semi_join()</td>
<td>intersect()</td>
</tr>
<tr>
<td>left_join()</td>
<td>anti_join()</td>
<td>setdiff()</td>
</tr>
<tr>
<td>right_join()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>full_join()</td>
<td></td>
<td>union()</td>
</tr>
</tbody>
</table>
Move computation; not the data
dplyr sources

• Local data frame
• Local data table
• Local data cube (experimental)
• RDMS: Postgres, MySQL, SQLite, Oracle, MS SQL, JDBC, Impala
• MonetDB, BigQuery
Google for “dplyr”
What is ggvis?

• A grammar of graphics (like ggplot2)
• Reactive (interactive & dynamic) (like shiny)
• A pipeline (a la dplyr)
• Of the web (drawn with vega)
Google for “ggvis”
Challenges /
Future work
broom

- [https://github.com/dgretwo/broom](https://github.com/dgretwo/broom)
- By David Robinson
- Provides verbs to convert model objects into tidy data frames
Modelling

R provides a huge variety of modelling tools, and the formula interface is common.

But... otherwise there's not a lot of consistency, and I think a lot of room for making easier to use (Zelig and caret notwithstanding)
End game

Provide a fluent interface where you spent your mental energy on the specific data problem, not general data analysis process.

The best tools become invisible with time!

(Currently focusing on data ingest: on disk, from databases, web apis & scraping)