How do you look at a billion data points?

Carlos Scheidegger, Assistant Professor
Dept. of Computer Science - University of Arizona

NOAO Meeting, tools for Big Data
As computational methods get better, so must our understanding
Why look at data at all?

```
summary(anscombe)
```

<table>
<thead>
<tr>
<th></th>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min.</td>
<td>4.0</td>
<td>4.0</td>
<td>4.0</td>
<td>4.0</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>6.5</td>
<td>6.5</td>
<td>6.5</td>
<td>8.0</td>
</tr>
<tr>
<td>Median</td>
<td>9.0</td>
<td>9.0</td>
<td>9.0</td>
<td>9.0</td>
</tr>
<tr>
<td>Mean</td>
<td>9.0</td>
<td>9.0</td>
<td>9.0</td>
<td>9.0</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>11.5</td>
<td>11.5</td>
<td>11.5</td>
<td>8.0</td>
</tr>
<tr>
<td>Max.</td>
<td>14.0</td>
<td>14.0</td>
<td>14.0</td>
<td>19.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>y1</th>
<th>y2</th>
<th>y3</th>
<th>y4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min.</td>
<td>4.260</td>
<td>3.100</td>
<td>5.390</td>
<td>5.2500</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>6.315</td>
<td>6.695</td>
<td>6.250</td>
<td>6.1700</td>
</tr>
<tr>
<td>Median</td>
<td>7.580</td>
<td>8.140</td>
<td>7.110</td>
<td>7.0400</td>
</tr>
<tr>
<td>Mean</td>
<td>7.501</td>
<td>7.501</td>
<td>7.500</td>
<td>7.5010</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>8.570</td>
<td>8.950</td>
<td>7.980</td>
<td>8.1900</td>
</tr>
<tr>
<td>Max.</td>
<td>10.840</td>
<td>9.260</td>
<td>12.740</td>
<td>12.5000</td>
</tr>
</tbody>
</table>
Why look at data at all?

\begin{verbatim}
lm(y1 ~ x1, data=anscombe)
##
## Call:
## lm(formula = y1 ~ x1, data = anscombe)
##
## Coefficients:
## (Intercept) x1
## 3.0001 0.5001

lm(y2 ~ x2, data=anscombe)
##
## Call:
## lm(formula = y2 ~ x2, data = anscombe)
##
## Coefficients:
## (Intercept) x2
## 3.001 0.500

lm(y3 ~ x3, data=anscombe)
##
## Call:
## lm(formula = y3 ~ x3, data = anscombe)
##
## Coefficients:
## (Intercept) x3
## 3.0025 0.4997

lm(y4 ~ x4, data=anscombe)
##
## Call:
## lm(formula = y4 ~ x4, data = anscombe)
##
## Coefficients:
## (Intercept) x4
## 3.0017 0.4999
\end{verbatim}
Why look at data at all?

anscombe

```r
##   x1 x2 x3 x4  y1  y2  y3  y4
## 1 10 10 10  8 8.04 9.14 7.46 6.58
## 2  8  8  8  8 6.95 8.14 6.77 5.76
## 3 13 13 13  8 7.58 8.74 12.74 7.71
## 4  9  9  9  8 8.81 8.77 7.11 8.84
## 5 11 11 11  8 8.33 9.26 7.81 8.47
## 6 14 14 14  8 9.96 8.10 8.84 7.04
## 7  6  6  6  8 7.24 6.13 6.08 5.25
## 8  4  4  4 19 4.26 3.10 5.39 12.50
## 9 12 12 12  8 10.84 9.13 8.15 5.56
##10  7  7  7  8 4.82 7.26 6.42 7.91
##11  5  5  5  8 5.68 4.74 5.73 6.89
```
Why look at data at all?
If it’s bad with 11 points, imagine 1 billion
Nanocubes

Lins, Scheidegger, Klosowski, IEEE TVCG 2013

NOAO Meeting, tools for Big Data, March 16 2015
Let’s explore the space of solutions

- Prerequisites
  - Support many different queries
  - with small memory usage
  - and fast query times
What must it do?

- Query: produce a heatmap of the world
What must it do?

- Produce a heatmap of the world in 2005
What must it do?

• Query: produce a time series of tweet counts
What must it do?

- Query: produce a time series of tweet counts in Central Texas
Nanocubes are..

- ... multiscale
- ... spatiotemporal
- ... sparse
- ... in-memory
- data cubes

(It seems that “Data cubes” means something different to you!)
Demos
How does it work?

• We avoid exponential memory blowup by carefully reusing results of different queries

• eg. Don’t store results twice if query for year=2005 is equal to query for year=2005 and month=January

• Many more ugly, uninteresting data structures tricks

NOAO Meeting, tools for Big Data, March 16 2015
### Performance numbers

#### Build time:

<table>
<thead>
<tr>
<th>dataset</th>
<th>n</th>
<th>memory</th>
<th>time</th>
<th>keys</th>
<th>cardinality</th>
</tr>
</thead>
<tbody>
<tr>
<td>brightkite</td>
<td>4.5M</td>
<td>1.6GB</td>
<td>3.5m</td>
<td>3.5M</td>
<td>$2^{74}$</td>
</tr>
<tr>
<td>cust. tix</td>
<td>7.8M</td>
<td>2.5GB</td>
<td>8.47m</td>
<td>7.8M</td>
<td>$2^{69}$</td>
</tr>
<tr>
<td>flights</td>
<td>121M</td>
<td>2.3GB</td>
<td>31.13m</td>
<td>43.3M</td>
<td>$2^{75}$</td>
</tr>
<tr>
<td>twitter-small</td>
<td>210M</td>
<td>10.2GB</td>
<td>1.23h</td>
<td>116M</td>
<td>$2^{53}$</td>
</tr>
<tr>
<td>twitter</td>
<td>210M</td>
<td>46.4GB</td>
<td>5.87h</td>
<td>136M</td>
<td>$2^{60}$</td>
</tr>
<tr>
<td>cdrs</td>
<td>1B</td>
<td>3.6GB</td>
<td>3.08h</td>
<td>96.3M</td>
<td>$2^{69}$</td>
</tr>
</tbody>
</table>

*Query time is dominated by network latency and bandwidth (<0.1s)*

*Preprocessing time is ~100k events/s*
Implementation

• C++ backend, HTML5 front-end
  • Program reads data sequentially, then opens a web server

• Open source: https://github.com/laurolins/nanocube

• Runs on cell phones and tablets (!)
Astronomy demos

• (I’m not an astronomer, so apologies in advance!)

• But imagine an interactive version of the Hertzsprung-Russell diagram

• it would not be hard to create an interactive tool to select/visualize subsets of stars based on

  • temperature x magnitude x other attributes (sky location, etc).

• Today: two small star catalogs I could find and parse myself
Limitations

• Relatively small number of dimensions (4-8 ideal)

• in-memory for now, so it won’t work for arbitrarily-large dataset

  • External memory implementation is coming

  • still, very large ones, 1B daily events with d=5 in production use

• work-in-progress, usability-wise
Where do we go from here?

• Store more than counts
  • Anything that behaves like a **monoid**: lots of statistics are monoids

• Rebuild the infrastructure of EDA assuming this is the available backend
  • Clustering, data fitting, modelling
  • Push interactive exploration into the computation infrastructure

• How to reconcile interaction with the multiple-comparisons problem?
Thank you!

- [http://nanocubes.net](http://nanocubes.net) for links to paper, source code, documentation
- [http://github.com/laurolins/nanocube](http://github.com/laurolins/nanocube) is the github page