cloudera
Ask Bigger Questions

# Infrastructures and Interfaces for Data Science

Jeff Hammerbacher

# What Will Be Covered

- For a large organization with dozens of data scientists
  - What are these data scientists making?
  - What infrastructure will help them do data science faster?
  - What interfaces to this infrastructure are most useful?

# Assertions

- Data scientists make data and probabilistic models
- Primary infrastructure: warehouse-scale computer
- Productivity gains last ten years: infrastructure
- Productivity gains next ten years: interface

# Data Science

# Work Products of Data Science

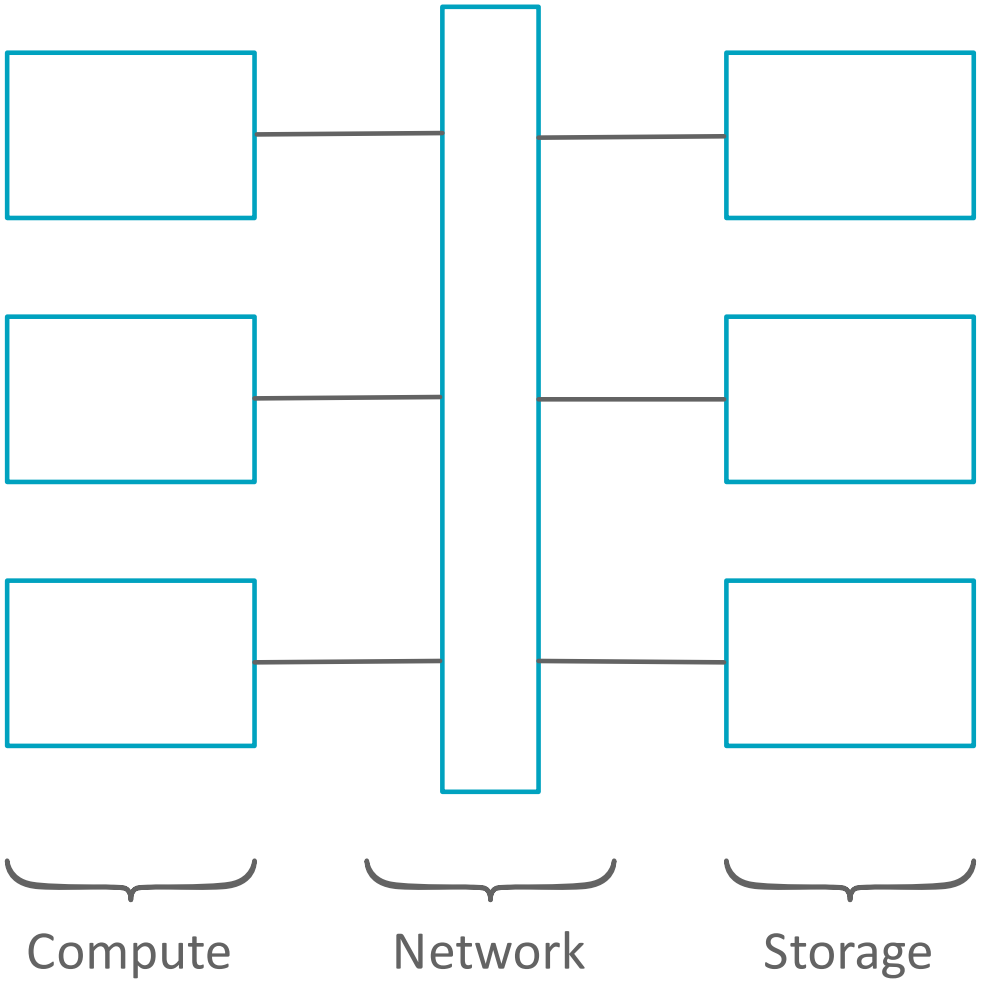
- Models (data)
  - Cleansed, integrated data set
  - Interfaces to explore and manipulate the data set
- Models (probability)
  - Understanding
  - Prediction

# Infrastructure

# 3 Kinds of Infrastructures

- HPC
- MPP
- WSC

# HPC



# HPC: Software

- Distributed file system
  - OneFS, GPFS, Lustre, GlusterFS, Ceph, PanFS, SiliconFS
- Resource management and job scheduling
  - LSF, SGE, Torque, Condor, SLURM
- Data management
  - NetCDF, HDF5
- Library and application development
  - MPI

# HPC: Pros

- Fast inter-node communication
- Arbitrary communication topologies
- Scale compute separate from storage
- Sophisticated resource management and scheduling
- High performance, POSIX-compliant file system
- Large collection of libraries and applications

# HPC: Cons

- Specialized hardware for network
- Throughput bottleneck on network
- Resource management optimized for compute
- Data management with files
- Library and application development with MPI
- Libraries and applications developed by scientists

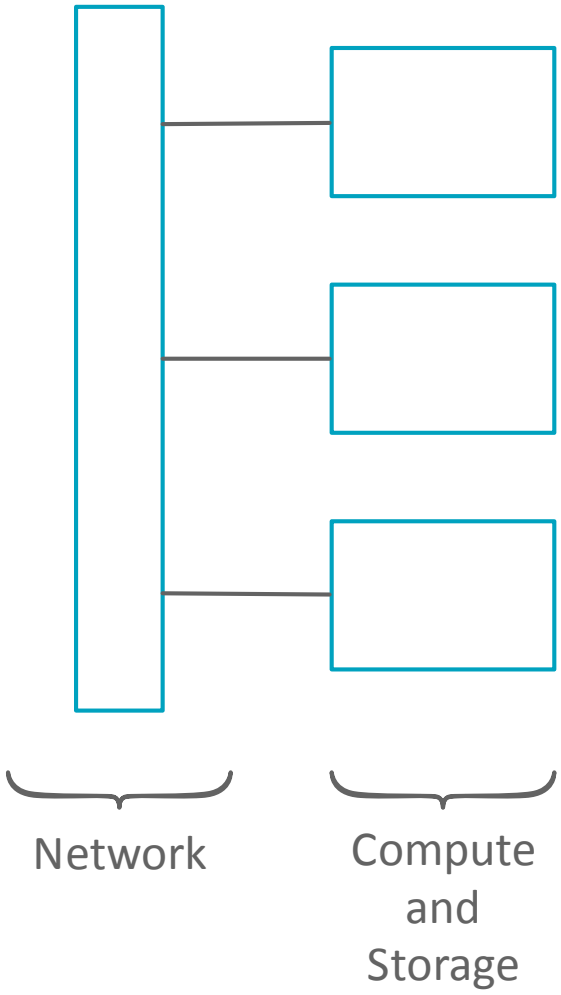
# HPC: Workloads

- Linear algebra
- Differential equations
- Distance matrices
- Monte Carlo
- FFT

# HPC: Industries

- Science ($4B)
  - Government
  - University
- Life sciences ($1B)
- CAE ($1B)
- Defense
- EDA
- Earth sciences

# MPP



# MPP: Software

- IBM DB2 and Netezza
- Teradata and Teradata Aster
- Microsoft PDW
- Pivotal Greenplum
- HP Vertica
- Actian ParAccel
- Calpont InfiniDB
- StormDB Stado
- Percona Shard-Query

# MPP: Pros

- Commodity hardware (some of them)
- Compute co-located with storage
- Data management with tables
- Library and application development with SQL
- Large collection of applications

# MPP: Cons

- No high-quality open source implementations
- Adding/removing nodes is expensive (usually)
- Schema-on-write/early binding of schemas
- Difficult to handle non-tabular data
- Difficult to handle non-SQL development
- Difficult to mix in non-SQL workloads

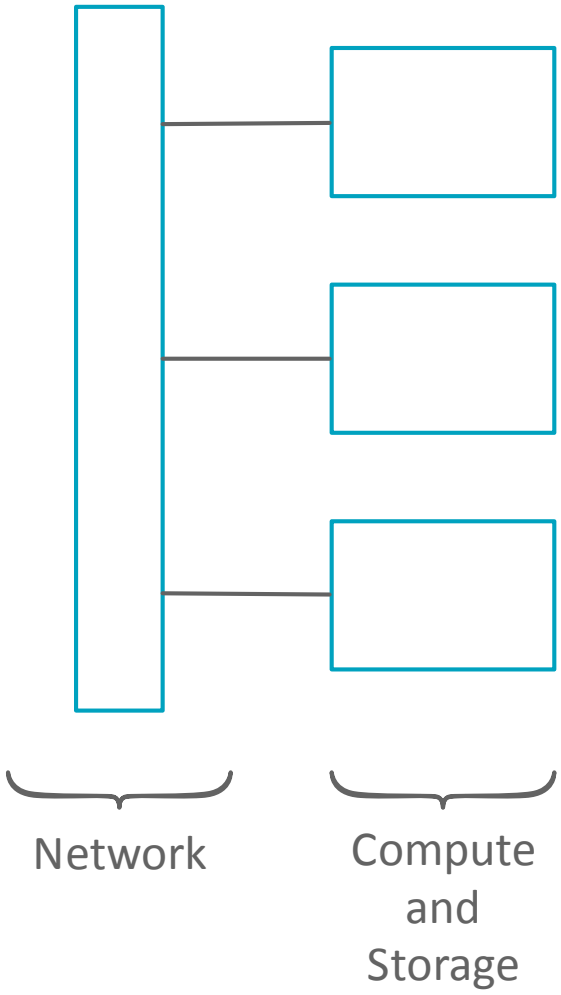
# MPP: Workloads

- Data preparation
- Index creation and maintenance
- Interactive SQL query
  - Filtering
  - Aggregation
  - Join
- Model fitting, barely

# MPP: Industries

- Financial services
- Communications
- Retail
- Manufacturing

# WSC



# WSC: Software

- Distributed file storage: HDFS
- Data management
  - Avro
  - Parquet
  - Metastore
  - HBase
  - Sentry
  - SolrCloud

# WSC: Software

- Resource management: YARN
- Parallel Frameworks
  - MapReduce
  - Impala
  - Spark
  - SolrCloud
- Interfaces
  - Crunch, Scrunch
  - Pig
  - SQL
  - Keyword search

# WSC: Pros

- Commodity hardware
- Open source software
- Compute co-located with storage
- General purpose resource management
- Data management with files, tables, indexes, …
- Application development with MapReduce, SQL, …

# WSC: Cons

- Commodity interconnect
- Limited communication topologies
- Non-POSIX file system
- Limited number of libraries and applications

# WSC: Workloads

- Data storage
- Data transformation
- Interactive SQL queries
- Interactive keyword and faceted search queries
- Machine learning: model fitting and evaluation
- Graph computations
- Stream processing

# WSC: Industries

- Communications
  - Web
  - Media
  - Telecommunications
- Government
- Retail
- Financial services

# Alternatives

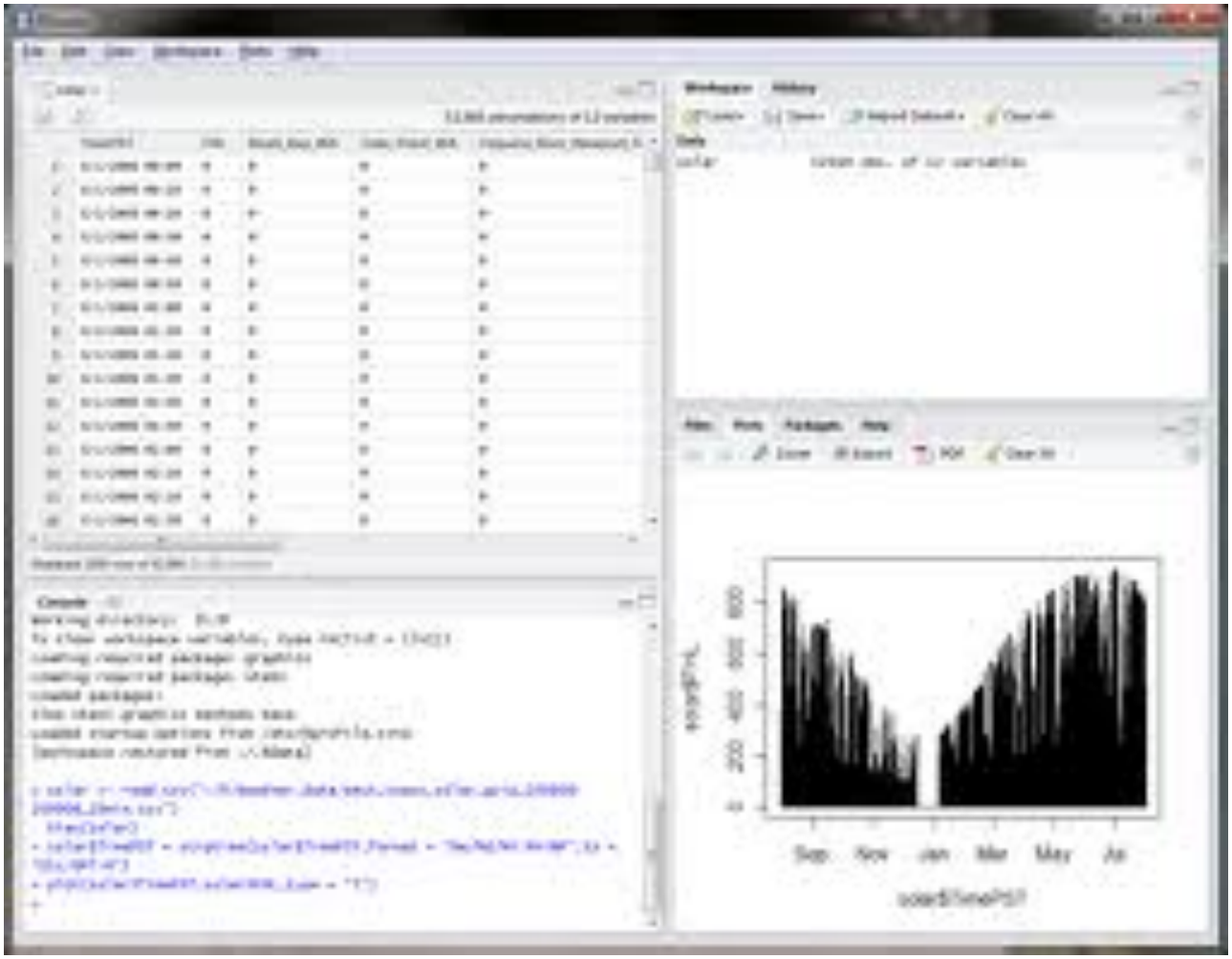
- Scale up
- Cloud
- SaaS federation

# Interface

# Wish List: Part One

- Graphics and code
- First class language
- Expressive types
- Domain syntax
- Mixed initiative

# Graphics and code: RStudio



# Graphics and code: iPython Notebook

IPython Notebook

Spectrogram Save Idle

Notebook

Actions: New Open

Download ipynb

Print

Cell

Actions: Delete

Format: Code Markdown

Output: Toggle ClearAll

Insert: Above Below

Move: Up Down

Run: Selected All

Autoindent:

Kernel

Actions: Interrupt Restart

Kill kernel upon exit:

Help

Links: Python IPython NumPy SciPy MPL SymPy

Shift-Enter : run selected cell

Ctrl-Enter : run in terminal mode

Ctrl-m h : show keyboard shortcuts

## Simple spectral analysis

An illustration of the Discrete Fourier Transform

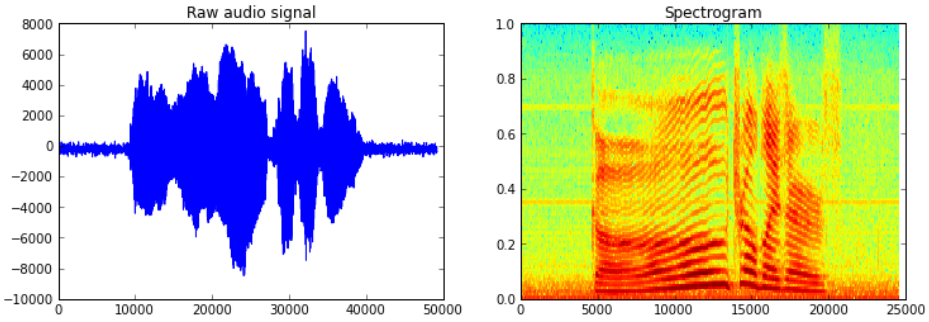
$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0, \ldots, N-1$$

using windowing, to reveal the frequency content of a sound signal.
We begin by loading a datafile using SciPy's audio file support:

```
In [1]: from scipy.io import wavfile
        rate, x = wavfile.read('/home/fperez/teach/py4science/book/examples/test_mono.wav')
```

And we can easily view its spectral structure using matplotlib's builtin `specgram` routine:

```
In [3]: fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
        ax1.plot(x); ax1.set_title('Raw audio signal')
        ax2.specgram(x); ax2.set_title('Spectrogram');
```



# First class language: LINQ

```
//c#
public void Linq2()
{
    List<Product> products = GetProductList();

    var soldOutProducts =
        from p in products
        where p.UnitsInStock == 0
        select p;

    Console.WriteLine("Sold out products:");
    foreach (var product in soldOutProducts)
    {
        Console.WriteLine("{0} is sold out!", product.ProductName);
    }
}
```

# Expressive types: Biocaml

```
type raw_to_item = [ `header_line_not_first of int
        | `header_line_without_version of (string * string) list
        | `header_line_wrong_sorting of string
        | `invalid_header_tag of int * string
        | `invalid_tag_value_list of int * string list
        | `reference_sequence_not_found of Biocaml_bam.raw_alignment
        | `wrong_auxiliary_data of
          [ `array_size of int
          | `null_terminated_hexarray
          | `null_terminated_string
          | `out_of_bounds
          | `unknown_type of char
          | `wrong_int32 of string ] * string
        | `wrong_cigar of string
        | `wrong_cigar_length of int
        | `wrong_flag of Biocaml_bam.raw_alignment
        | `wrong_mapq of Biocaml_bam.raw_alignment
        | `wrong_pnext of Biocaml_bam.raw_alignment
        | `wrong_pos of Biocaml_bam.raw_alignment
        | `wrong_qname of Biocaml_bam.raw_alignment
        | `wrong_tlen of Biocaml_bam.raw_alignment ]
```

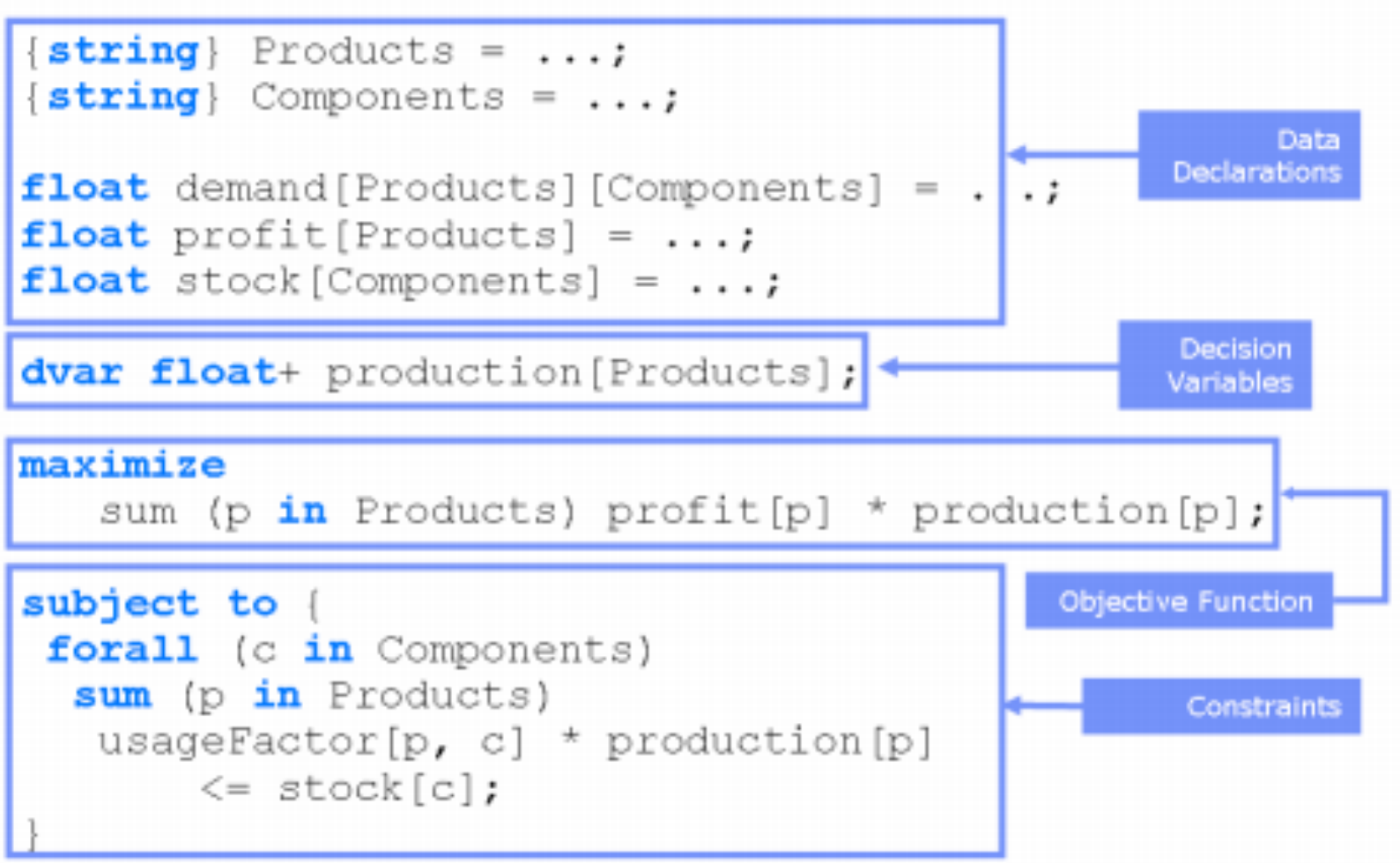
# Domain syntax: R formulas

```
kid_score ~ mom_hs + mom_iq + mom_hs:mom_iq
```

# Domain syntax: ggplot2

```
ggplot(mpg, aes(displ, hwy))+
  geom_point(aes(color = factor(cyl)))+
  geom_line()
```

# Domain syntax: IBM ILOG OPL



# Domain syntax: Fortress

- Programming language notation can become closer to mathematical notation (Unicode helps a lot)
  - `v_norm = v / ‖v‖`
  - `Σ[k=1:n] a[k] x^k`
  - `C = A ∪ B`

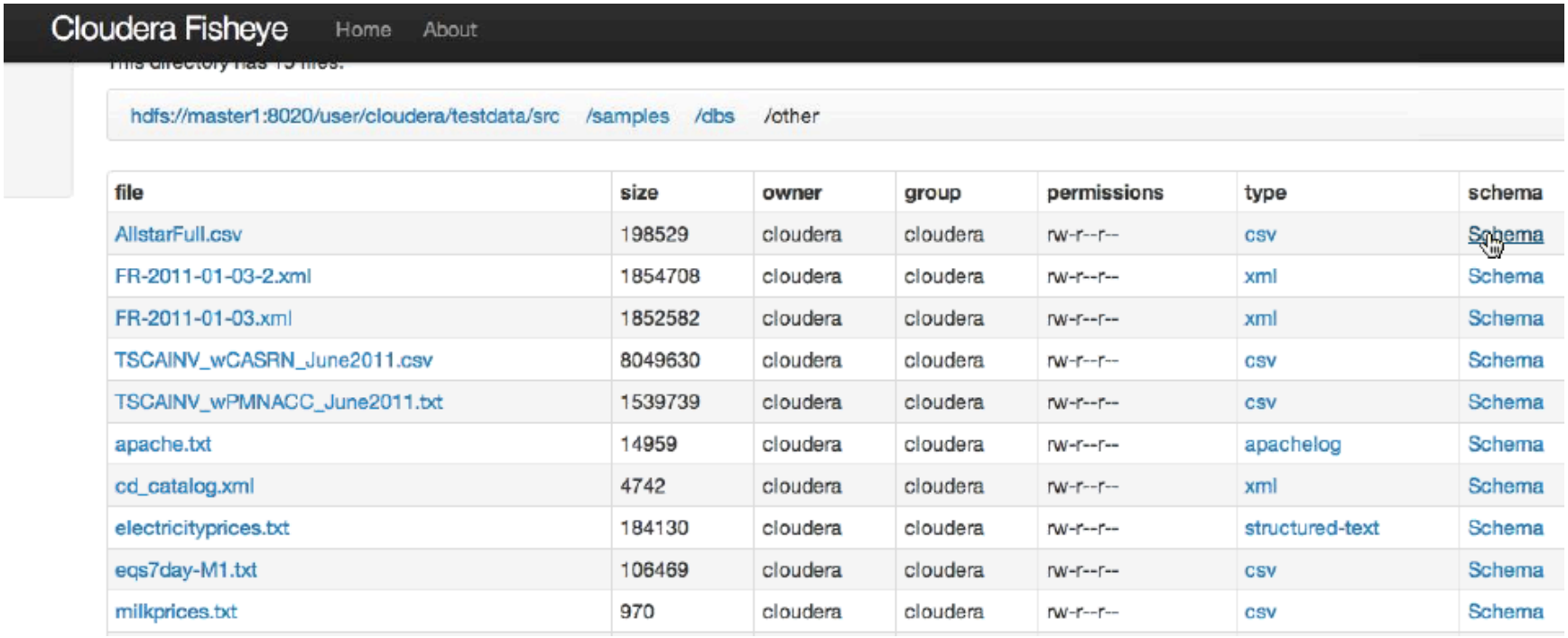
# Domain syntax: BUGS

```
model {
theta ~ dbeta(a, b)
y ~ dbin(theta, n)
Y.pred ~ dbin(theta, n.pred)
P.crit <- step(Y.pred - n.crit + 0.5)
}
```

# Domain syntax: Figaro

```
class Firm {
  val efficient = Flip(0.3)
  val bid = If(efficient, Uniform(0.0,10.0), Uniform(5.0,20.0))
}
val firms = Array.fromFunction(i => new Firm)(20)
def clause(firm: Firm) = (Constant(1.0), Constant(firm))
val winner = Dist(firms map clause)
val winningBid = Chain(winner, (f: Firm) => f.bid)
winningBid.constrain((bid: Double) => 20.0 - bid)
```

# Mixed initiative: FishEye




| file | size | owner | group | permissions | type | schema |
|---|---|---|---|---|---|---|
| AllstarFull.csv | 198529 | cloudera | cloudera | rw-r--r-- | csv | Schema |
| FR-2011-01-03-2.xml | 1854708 | cloudera | cloudera | rw-r--r-- | xml | Schema |
| FR-2011-01-03.xml | 1852582 | cloudera | cloudera | rw-r--r-- | xml | Schema |
| TSCAINV_wCASRN_June2011.csv | 8049630 | cloudera | cloudera | rw-r--r-- | csv | Schema |
| TSCAINV_wPMNACC_June2011.txt | 1539739 | cloudera | cloudera | rw-r--r-- | csv | Schema |
| apache.txt | 14959 | cloudera | cloudera | rw-r--r-- | apachelog | Schema |
| cd_catalog.xml | 4742 | cloudera | cloudera | rw-r--r-- | xml | Schema |
| electricityprices.txt | 184130 | cloudera | cloudera | rw-r--r-- | structured-text | Schema |
| eqs7day-M1.txt | 106469 | cloudera | cloudera | rw-r--r-- | csv | Schema |
| milkprices.txt | 970 | cloudera | cloudera | rw-r--r-- | csv | Schema |

# Mixed initiative: Wrangler

**Transform Script** Import Export

- Split **data repeatedly** on **newline** into **rows**
- Split **split repeatedly** on **','**
- Promote **row 0** to header

Text | Columns | Rows | Table | Clear

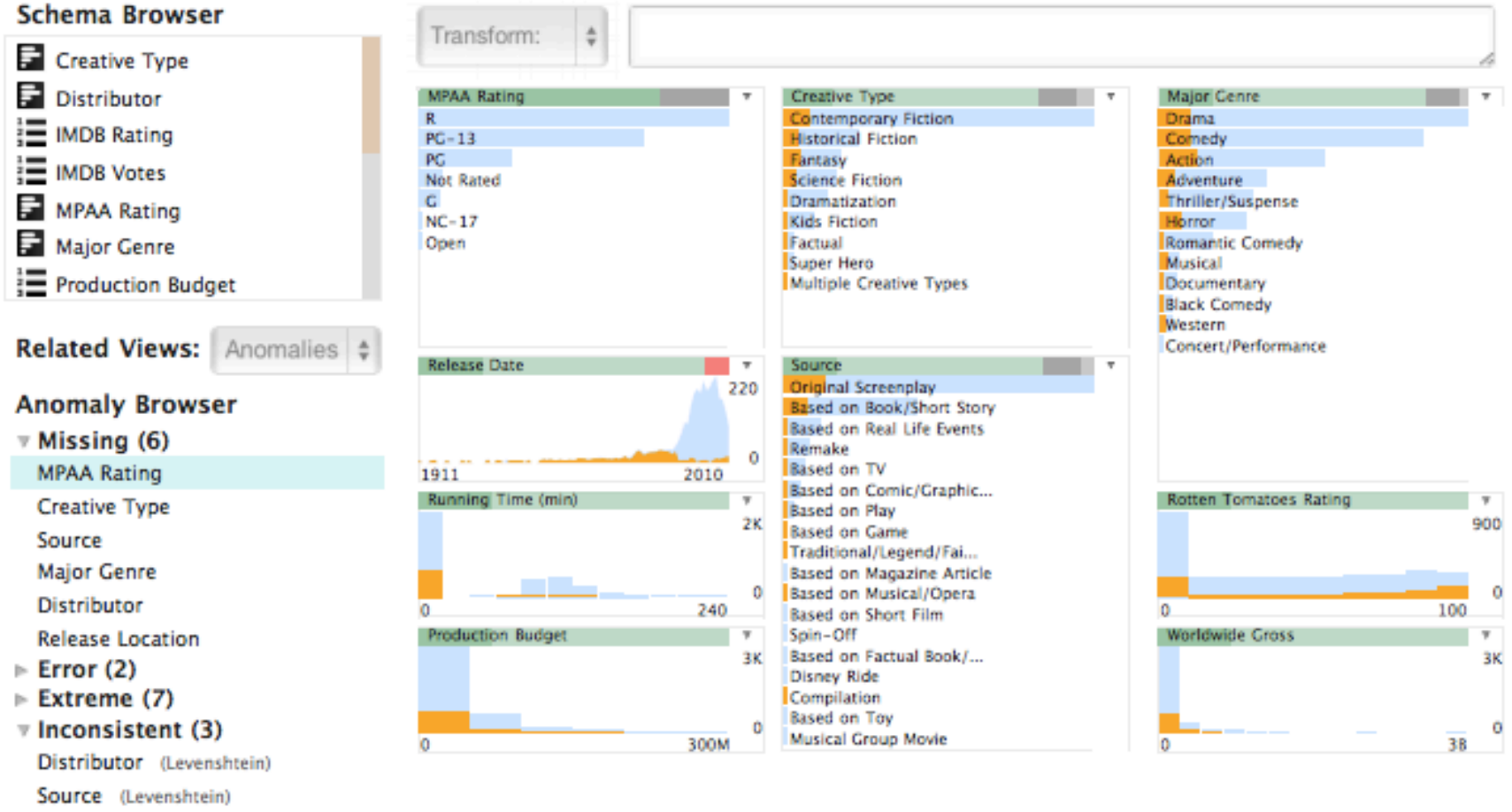
Delete **row 7**

Delete **empty rows**

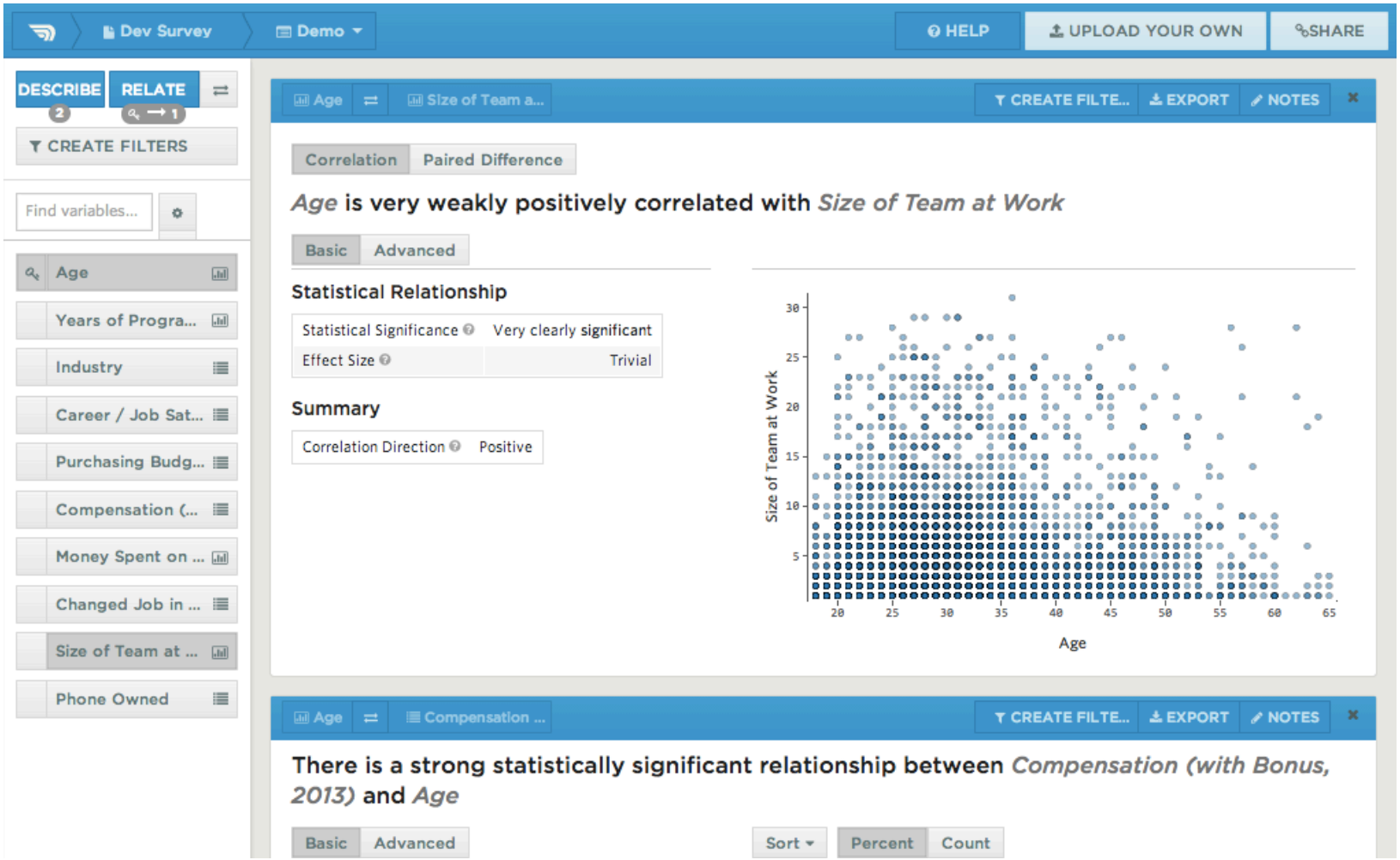
Fill **row 7** by **copying** values from **above**

| | Year | Property_crime_rate |
|---|---|---|
| 0 | Reported crime in Alabama | |
| 1 | | |
| 2 | 2004 | 4029.3 |
| 3 | 2005 | 3900 |
| 4 | 2006 | 3937 |
| 5 | 2007 | 3974.9 |
| 6 | 2008 | 4081.9 |
| 7 | | |
| 8 | Reported crime in Alaska | |
| 9 | | |
| 10 | 2004 | 3370.9 |
| 11 | 2005 | 3615 |
| 12 | 2006 | 3582 |

# Mixed initiative: Profiler



# Mixed initiative: Statwing



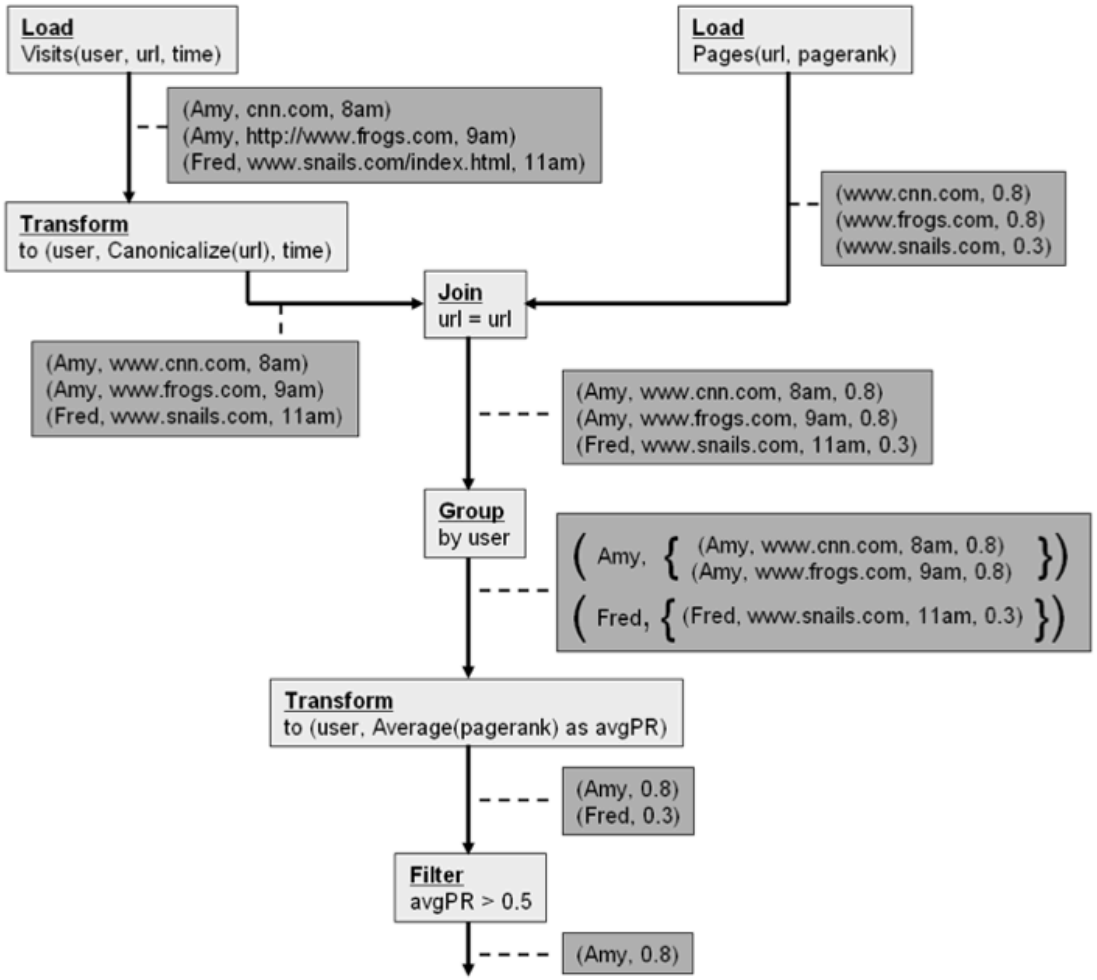
# Wish List: Part One

- Graphics and code
- First class language
- Expressive types
- Domain syntax
- Mixed initiative

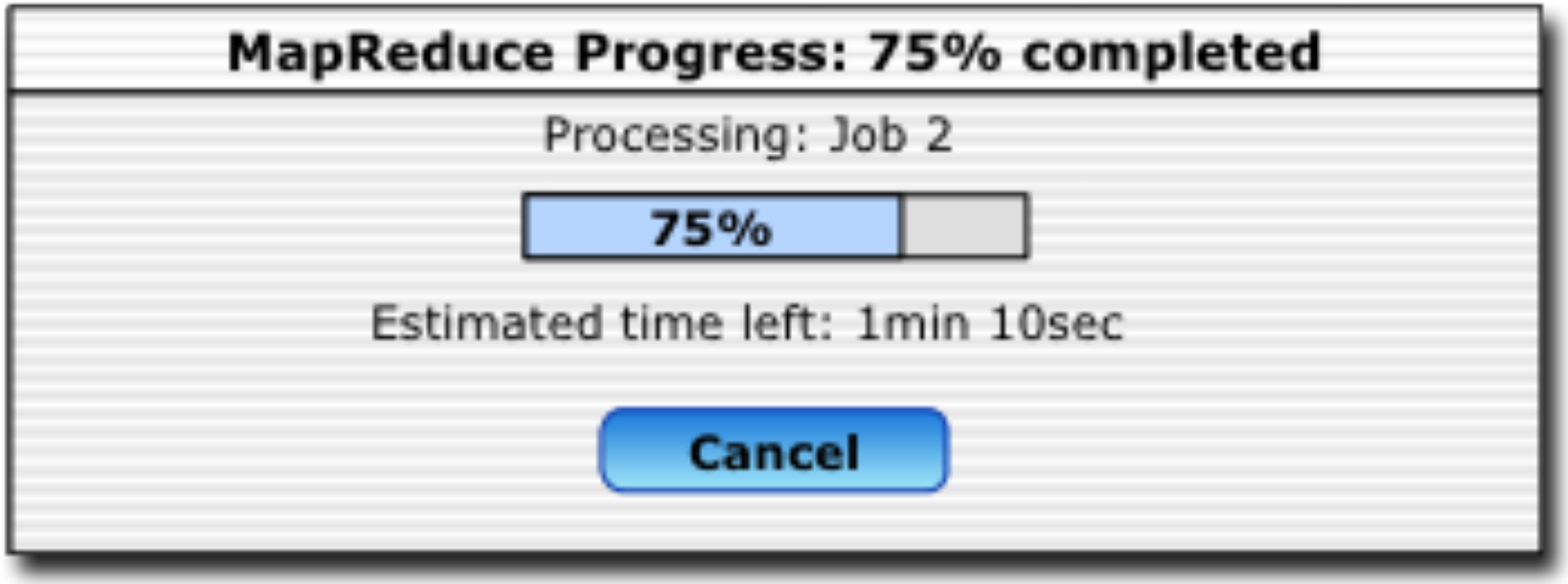
# Wish List: Part Two

- Continuous feedback
- Improve over time
- Rapid problem resolution
- Unify data at rest and in flight
- Libraries across languages

# Continuous feedback: ILLUSTRATE



# Continuous feedback: Parallax



# Continuous feedback: APPROXIMATE

```
SELECT DATE_TRUNC('day',event_time),
        APPROXIMATE COUNT(DISTINCT user_id),
        APPROXIMATE COUNT(DISTINCT url)
FROM weblog
GROUP BY 1
```

# Improve over time: Wrangler

**Transform Script** Import Export

- Split **data repeatedly** on **newline** into **rows**
- Split **split repeatedly** on **','**
- Promote **row 0** to header

Text | Columns | Rows | Table | Clear

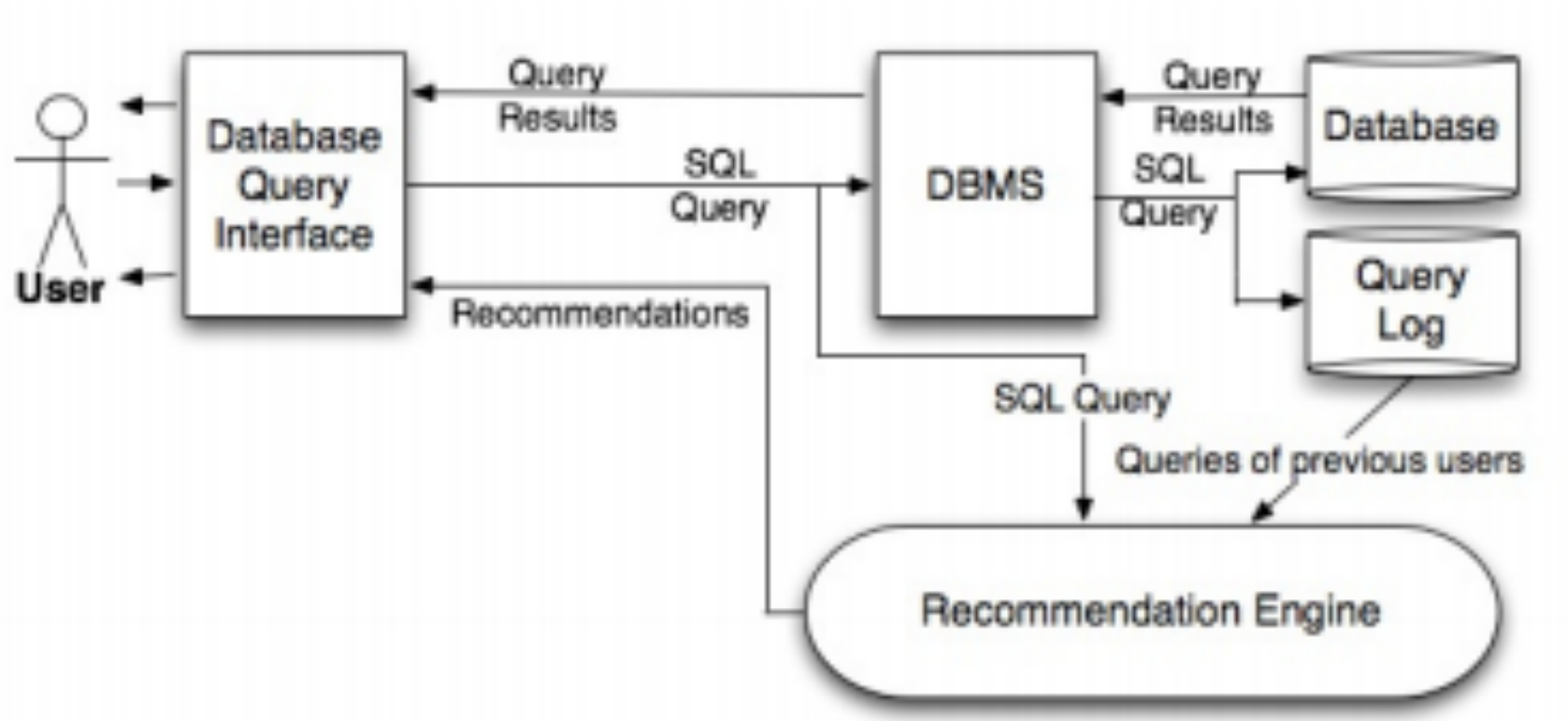
Delete **row 7**

Delete **empty rows**

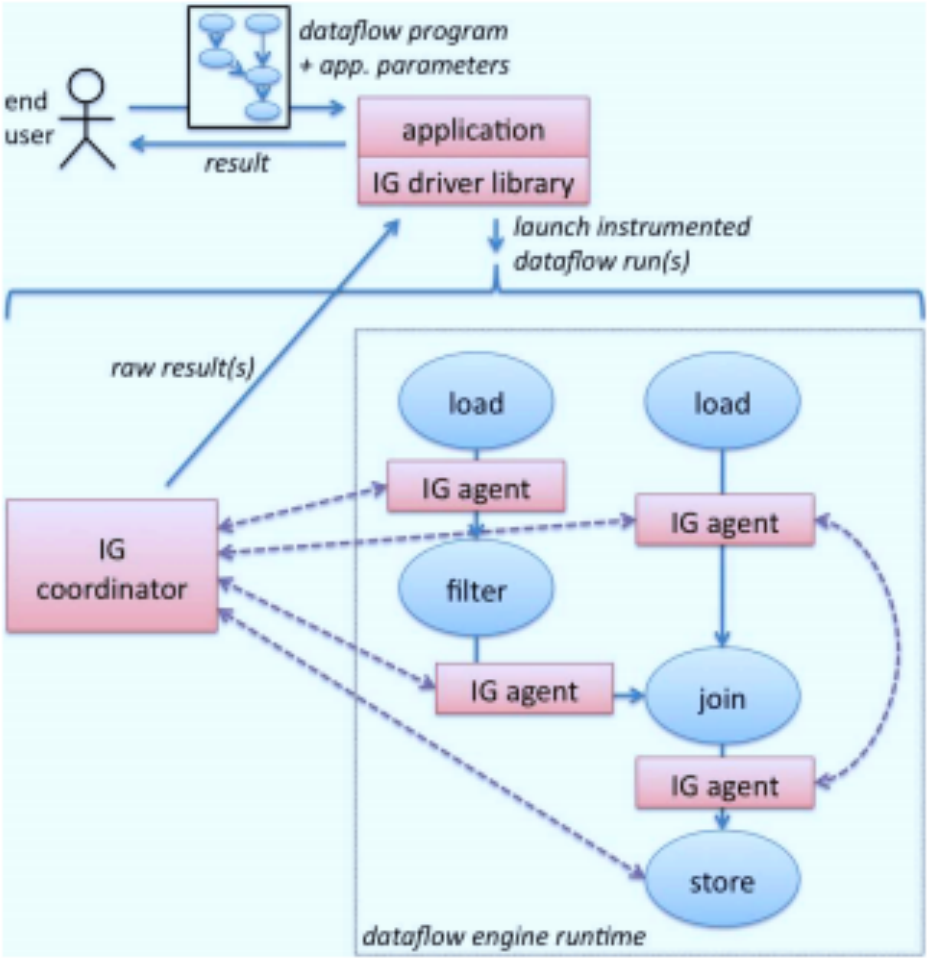
Fill **row 7** by **copying** values from **above**

| | Year | Property_crime_rate |
|---|---|---|
| 0 | Reported crime in Alabama | |
| 1 | | |
| 2 | 2004 | 4029.3 |
| 3 | 2005 | 3900 |
| 4 | 2006 | 3937 |
| 5 | 2007 | 3974.9 |
| 6 | 2008 | 4081.9 |
| 7 | | |
| 8 | Reported crime in Alaska | |
| 9 | | |
| 10 | 2004 | 3370.9 |
| 11 | 2005 | 3615 |
| 12 | 2006 | 3582 |

# Improve over time: QueRIE



# Rapid problem resolution: Penny



# Unify data at rest and in flight: TruCQ

```
create view impsn_count as
  (select campaign, sum(pc) c, cq_close(*) t
   from   i_c_p <slices '1 minute'>
   group by campaign);

create view impsn_count_ar as
  (select campaign, sum(pc) as c, t
   from   i_c_p_ar
   group by campaign, t);
```

# Unify data at rest and in flight: Summingbird

```
def wordCount[P <: Platform[P]]
  (source: Producer[P, String], store: P#Store[String, Long]) =
    source.flatMap { sentence =>
      toWords(sentence).map(_ -> 1L)
    }.sumByKey(store)
```

# Libraries across languages

- Arrays
- Data frames
- Serialization
- Workflow
- Statistics
- Machine learning
- Simulation
- Visualization

# Wish List: Part Two

- Continuous feedback
- Improve over time
- Rapid problem resolution
- Unify data at rest and in flight
- Libraries across languages


cloudera®
Ask Bigger Questions

# Language Improvements

- Compiler directives
  - Parakeet
- Syntax extensions
  - R formulas
  - Matlab and APL array operations
  - Mathematica symbolic manipulation
  - Fortress mathematical operators
  - Fortress units and dimensions

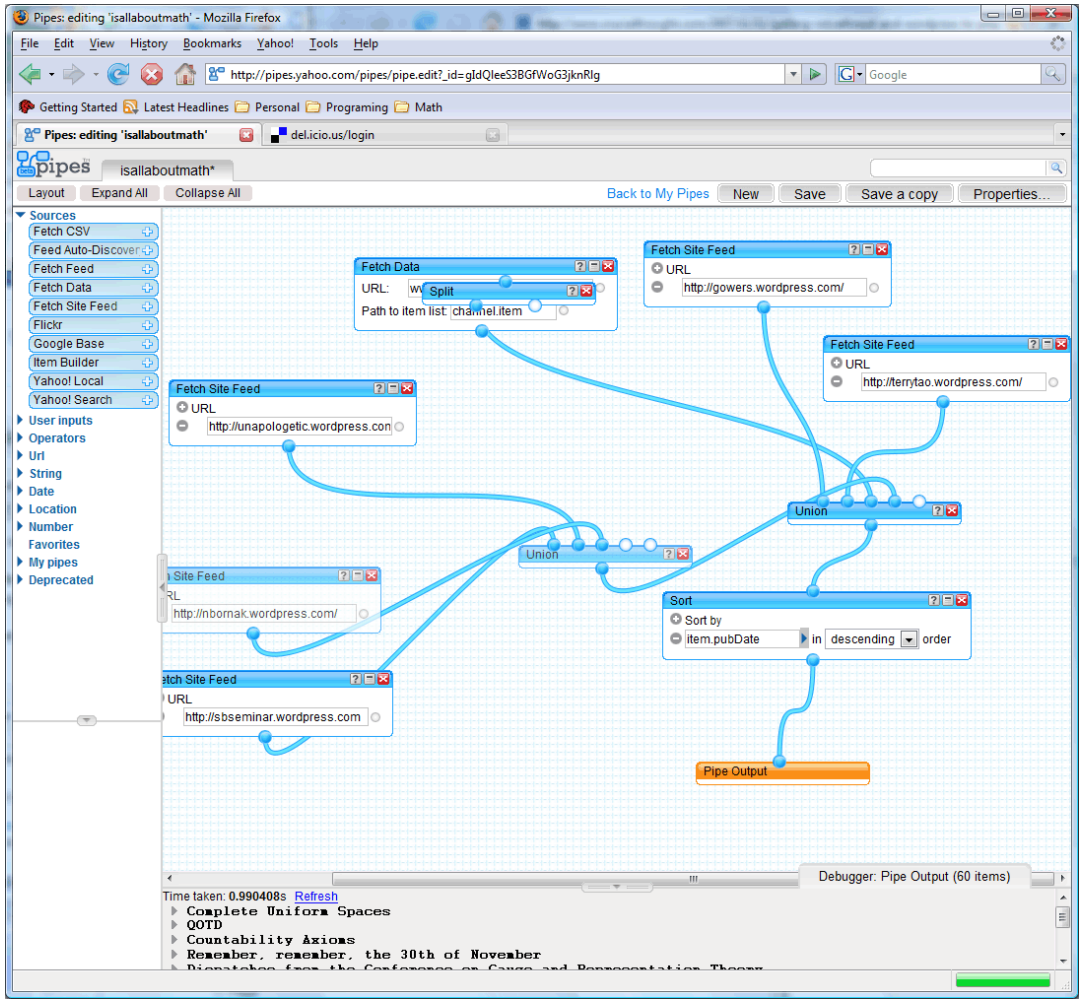
# Language Improvements

- Embedded DSLs
  - LINQ
  - plyr
  - ggplot2
  - FACTORIE
  - Figaro
  - Infer.NET
  - Weaver

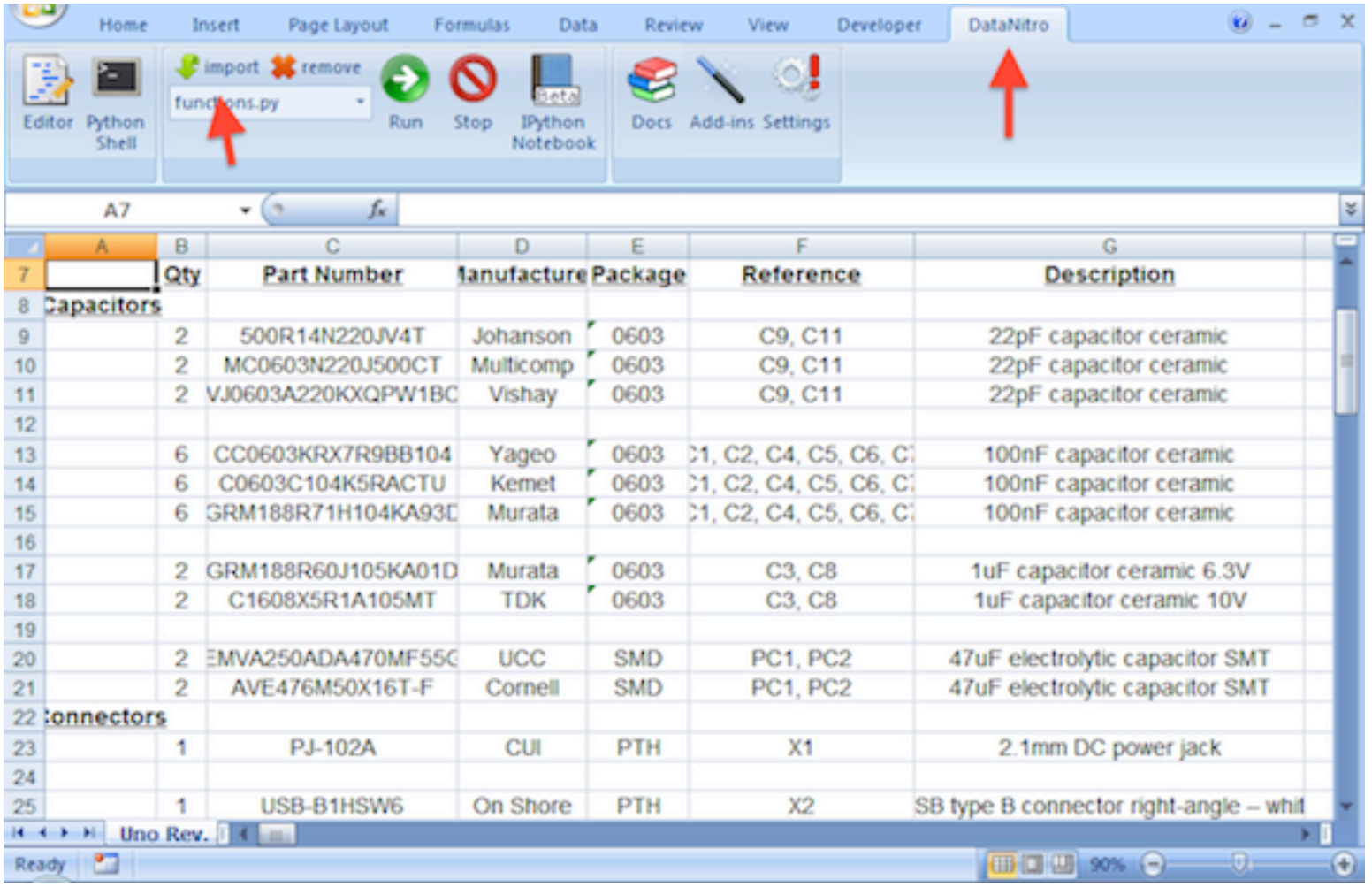
# Language Improvements

- External DSLs
  - IBM ILOG OPL
  - Palantir Hedgehog
  - Splunk SPL
  - Palantir VizQL
  - BUGS
  - MIT Church
  - Cray Chapel
  - Google Sawzall
  - Apache Pig

# Yahoo! Pipes



# DataNitro




| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 7 | | Qty | Part Number | Manufacture | Package | Reference | Description |
| 8 | Capacitors | | | | | | |
| 9 | | 2 | 500R14N220JV4T | Johanson | 0603 | C9, C11 | 22pF capacitor ceramic |
| 10 | | 2 | MC0603N220J500CT | Multicomp | 0603 | C9, C11 | 22pF capacitor ceramic |
| 11 | | 2 | VJ0603A220KXQPW1BC | Vishay | 0603 | C9, C11 | 22pF capacitor ceramic |
| 12 | | | | | | | |
| 13 | | 6 | CC0603KRX7R9BB104 | Yageo | 0603 | C1, C2, C4, C5, C6, C | 100nF capacitor ceramic |
| 14 | | 6 | C0603C104K5RACTU | Kemet | 0603 | C1, C2, C4, C5, C6, C | 100nF capacitor ceramic |
| 15 | | 6 | GRM188R71H104KA93D | Murata | 0603 | C1, C2, C4, C5, C6, C | 100nF capacitor ceramic |
| 16 | | | | | | | |
| 17 | | 2 | GRM188R60J105KA01D | Murata | 0603 | C3, C8 | 1uF capacitor ceramic 6.3V |
| 18 | | 2 | C1608X5R1A105MT | TDK | 0603 | C3, C8 | 1uF capacitor ceramic 10V |
| 19 | | | | | | | |
| 20 | | 2 | EMVA250ADA470MF55G | UCC | SMD | PC1, PC2 | 47uF electrolytic capacitor SMT |
| 21 | | 2 | AVE476M50X16T-F | Cornell | SMD | PC1, PC2 | 47uF electrolytic capacitor SMT |
| 22 | Connectors | | | | | | |
| 23 | | 1 | PJ-102A | CUI | PTH | X1 | 2.1mm DC power jack |
| 24 | | | | | | | |
| 25 | | 1 | USB-B1HSW6 | On Shore | PTH | X2 | SB type B connector right-angle – whit |