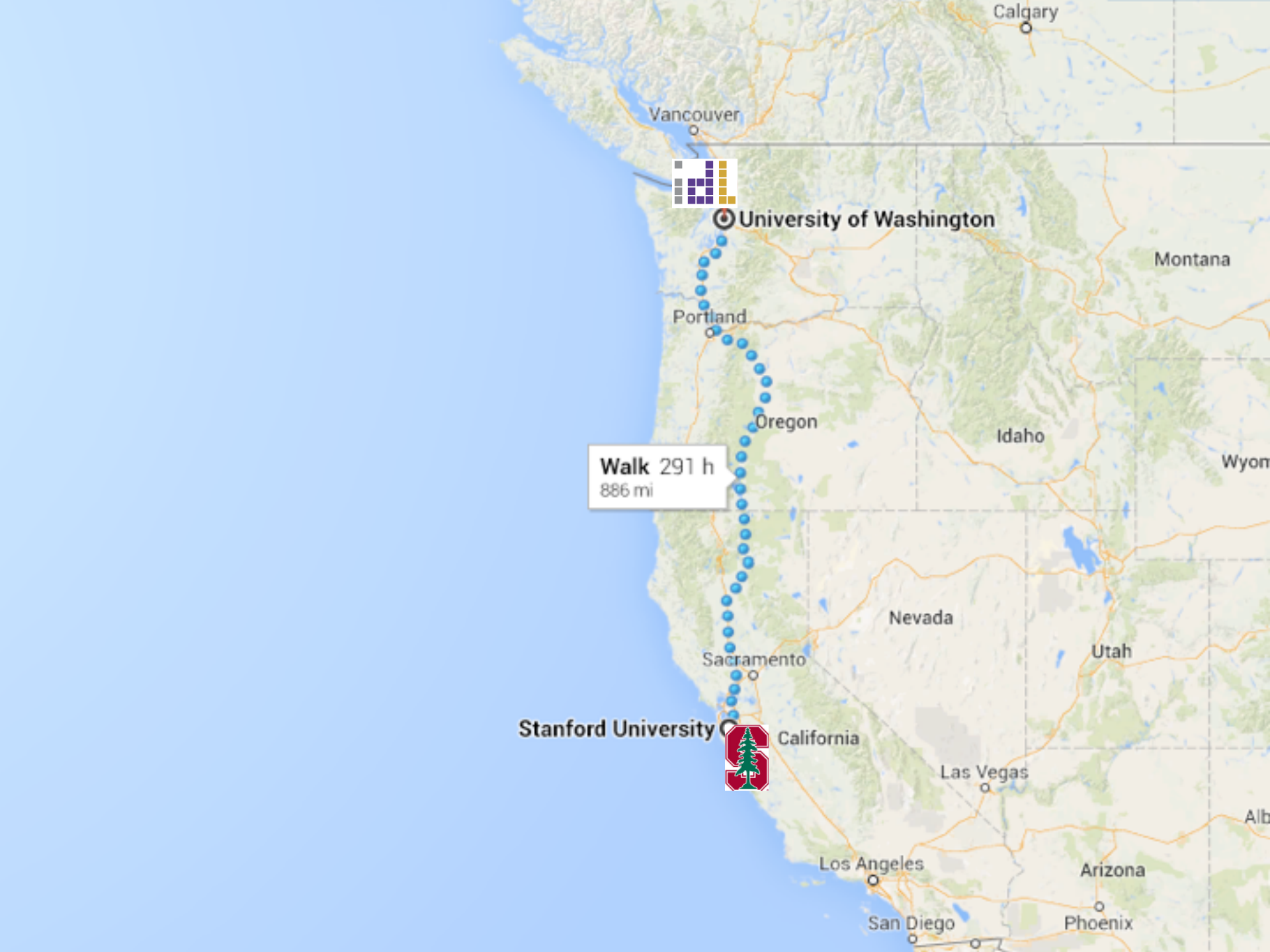
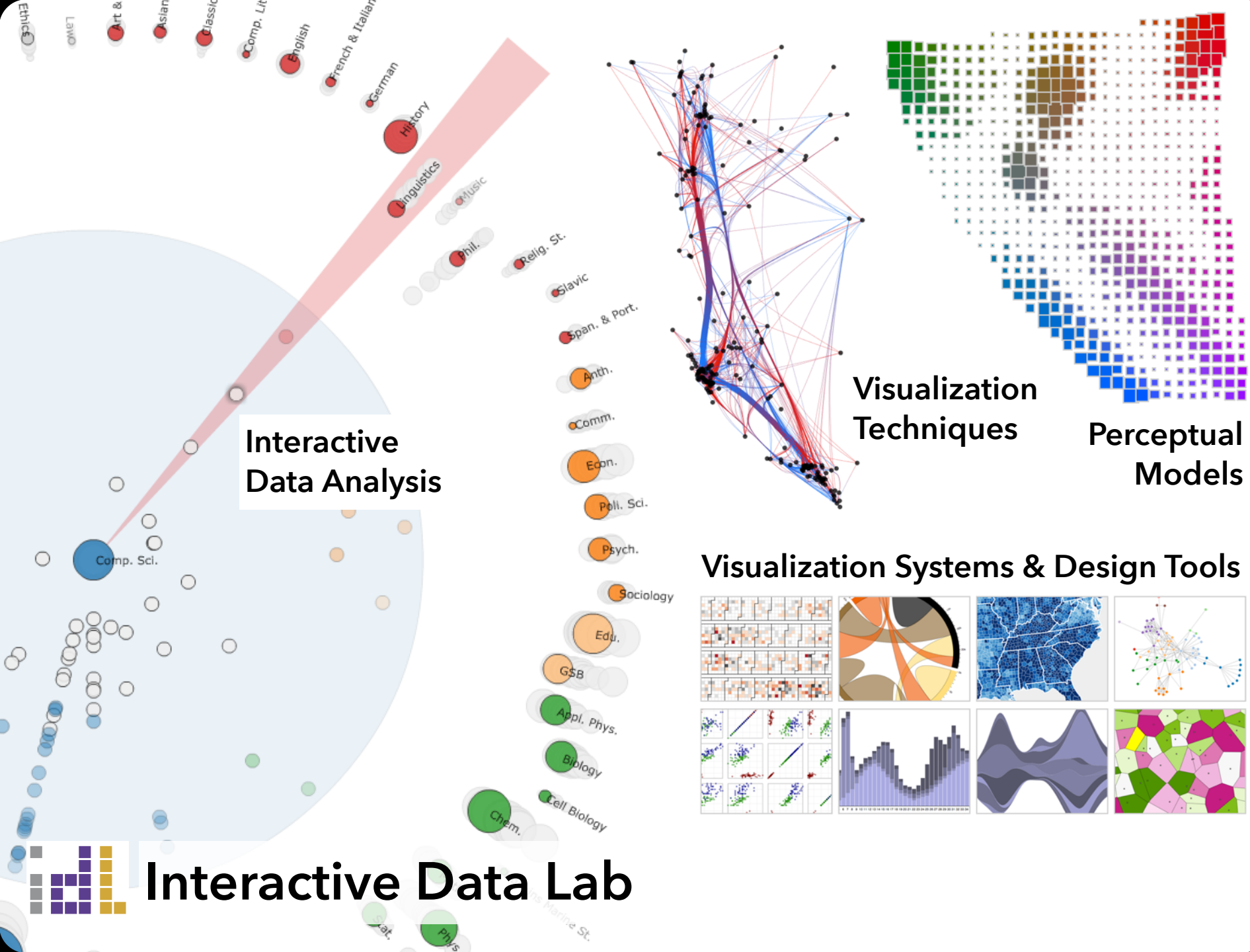
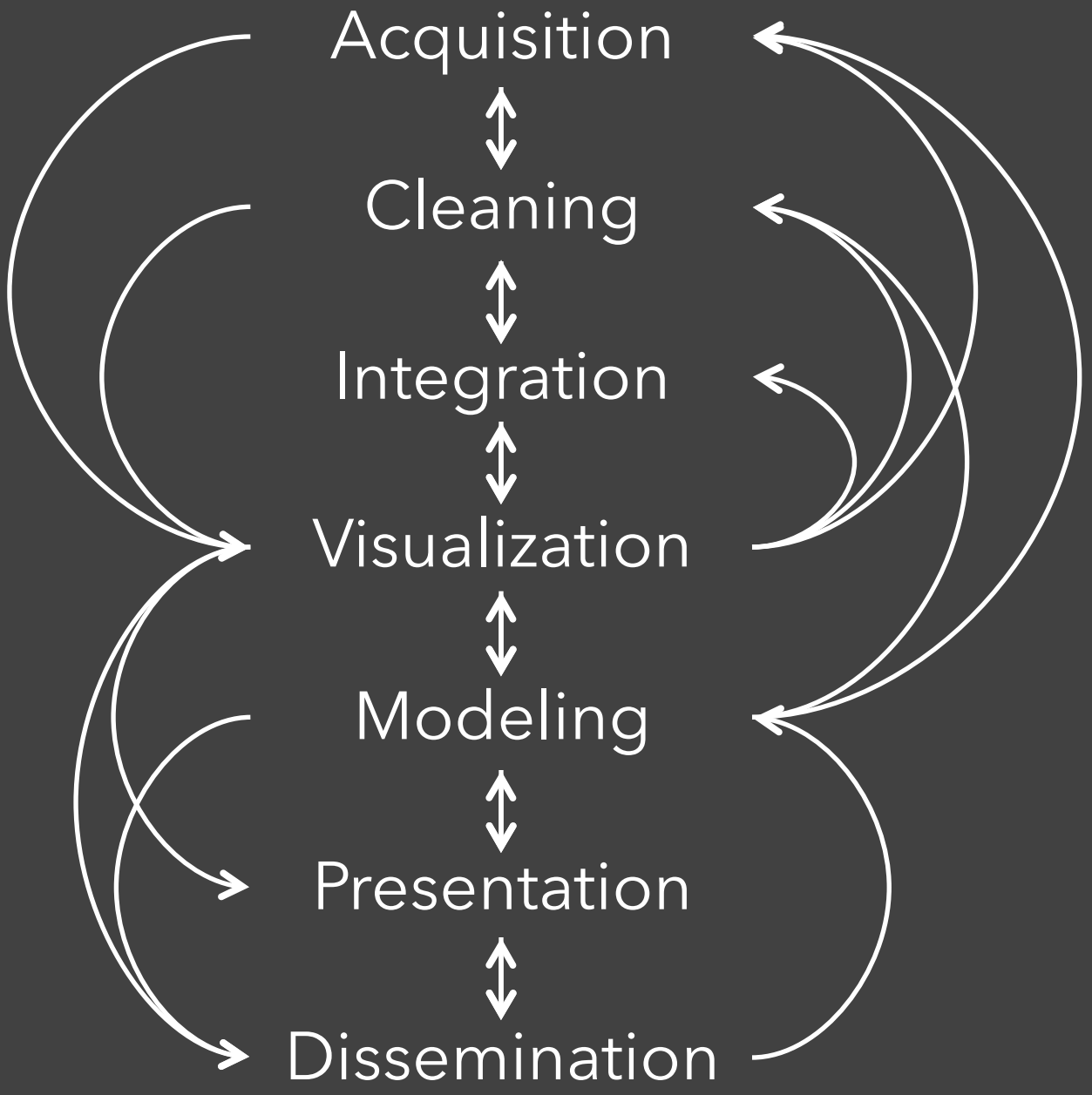
# UW Interactive Data Lab

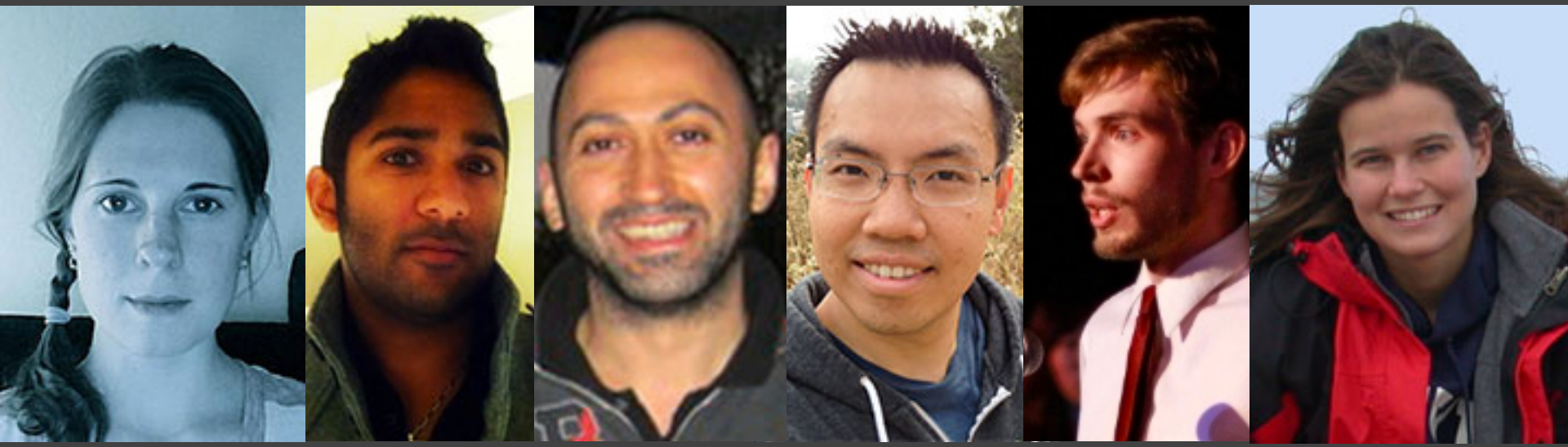
idl.cs.washington.edu




Calgary
Vancouver
University of Washington
Montana
Portland
Oregon
Idaho
Wyom
Walk 291 h
886 mi
Nevada
Utah
Sacramento
Stanford University
California
Las Vegas
Alb
Los Angeles
Arizona
San Diego
Phoenix


Interactive
Data Analysis
Visualization
Techniques
Perceptual
Models
Visualization Systems & Design Tools
Interactive Data Lab
Comp. Sci.
History
Linguistics
Music
Phil.
Relig. St.
Slavic
Span. & Port.
Anth.
Comm.
Econ.
Poli. Sci.
Psych.
Sociology
Edu.
GSB
Appl. Phys.
Biology
Cell Biology
Chem.
German
French & Italian
English


Acquisition
Cleaning
Integration
Visualization
Modeling
Presentation
Dissemination



Jane Hoffswell

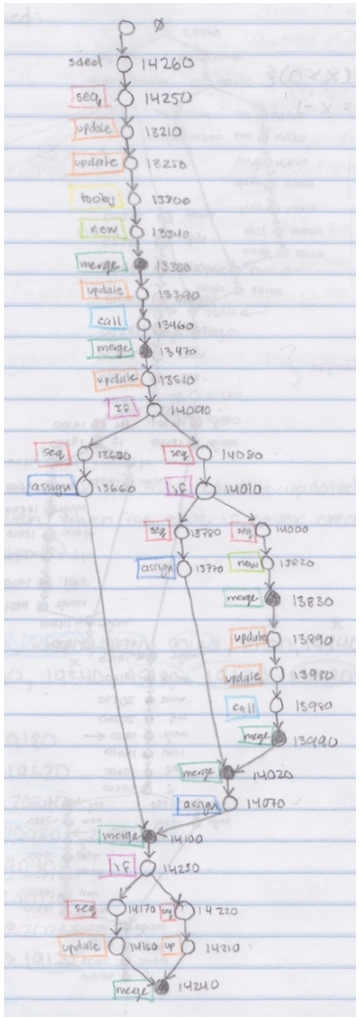
# About Me


HARVEY
MUDD
COLLEGE


University of Washington
Computer Science & Engineering

```
x = Math.random();
if (x < 0.5)
     y = true;
else
     y = false;
```

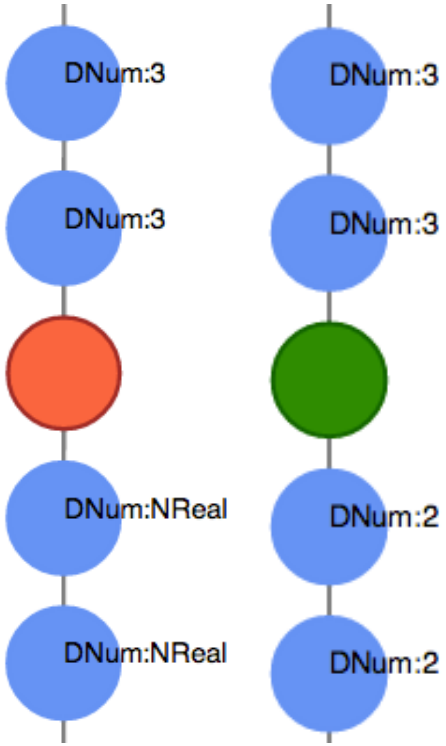
```
SDecl                                  ID: 14180
   Seq                                 ID: 14170
     Update                            ID: 13130
       Var: `window`0 (0)              ID: 13100
       String: dummyAddress            ID: 13110
       Undef                           ID: 13120
     Update                            ID: 13170
       Var: `window`0 (0)              ID: 13140
       String: Arguments               ID: 13150
       Undef                           ID: 13160
     ToObj                             ID: 13220
       Scratch: 0                      ID: 13180
       Binop: acc                      ID: 13210
         Var: `window`0 (0)            ID: 13190
         String: Math                  ID: 13200
     New                               ID: 13260
       Scratch: 1                      ID: 13230
       Var: `argumentsVar`11 (11)      ID: 13240
       Var: `dummyAddressVar`13 (13)   ID: 13250
```



Reset Untwist Help Choose File No file chosen Choose File No file chosen Choose a store value

```
 1 decl t`0 = undef, `dummy`2 = undef, `arrayVar`3 = `window`0 r·· "Array", `fu
 2   scratch (8) in                                                        ID: 76
 3       (`window`0).("dummyAddress") = undef                              ID: 69
 4       (`window`0).("Arguments") = undef                                 ID: 69
 5       (`window`0).("CObject") = undef                                   ID: 69
 6       (`window`0).("btop") = undef                                      ID: 69
 7       (`window`0).("foo") = undef                                       ID: 69
 8       (`window`0).("fact") = undef                                      ID: 69
 9       (`window`0).("fail") = undef                                      ID: 69
10       scratch_0 = newfun (1.0)                                          ID: 71
11         (`self`14, arguments) =>                                        ID: 71
12           decl n = arguments r·· "0" in                                 ID: 71
13             scratch (11) in                                             ID: 71
14               :RETURN::                                                 ID: 71
15                   if typeof n r== "string" r&&· false                   ID: 70
16                       scratch_0 = n r<· 0.0                             ID: 70
17                   else                                                  ID: 70
18                       if isprim n                                       ID: 70
19                           scratch_1 = tonum n                           ID: 70
20                       else                                              ID: 70
21                           scratch_2 = new `argumentsVar`11(`dummyAddressVar`
22                           merge                                         ID: 70
23                           (scratch_2).("0") = n                         ID: 70
24                           (scratch_2).("length") = 1.0                  ID: 70
25                           scratch_1 = `numberVar`8(`window`0, scratch_2)ID:
26                           merge                                         ID: 70
27                       merge                                             ID: 70
28                       scratch_0 = scratch_1 r<· 0.0                     ID: 70
29                   merge                                                 ID: 70
30                   if tobool scratch_0                                   ID: 71
31                       jmp :RETURN: 1.0                                  ID: 70
```

| Variable | Address | Value |
|---|---|---|
| `self`14 | 69860 | DAddr:Set(Address(-3)) |
| arguments | 69870 | DAddr:Set(Address(71921)) |
| `window`0 | -2 | DAddr:Set(Address(-3)) |
| `dummyAddressVar`13 | 69190 | DAddr:Set(Address(-177)) |
| `argumentsVar`11 | 69170 | DAddr:Set(Address(-29)) |
| `numberVar`8 | 69140 | DAddr:Set(Address(-26)) |




Imprecise (1 CFA)

Precise (5 CFA)

```
 8         (`window`0).("fact") = undef
 9         (`window`0).("fail") = undef
10         scratch_0 = newfun (1.0)
11           (`self`14, arguments) =>
12             decl n = arguments r?? "0" in
13               scratch (11) in
14                 :RETURN::
15                     if typeof n r== "string" r&&? false
16                         scratch_0 = n r*? 0.0
17                     else
18                         if isprim n
19                             scratch_1 = tonum n
20                         else
21                             scratch_2 = new `argumentsVar`11(`dummyAddr
22                             merge
23                             (scratch_2).("0") = n
24                             (scratch_2).("length") = 1.0
25                             scratch_I = `numberVar`8(`window`0, scratch
26                             merge
27                         merge
28                         scratch_0 = scratch_1 r<? 0.0
29                     merge
30                     if tobool scratch_0
31                         jmp :RETURN: I.0
32                     else
33                         if isprim n
34                             scratch_3 = tonum n
35                         else
36                             scratch_4 = new `argumentsVar`11(`dummyAddr
37                             merge
38                             (scratch_4).("0") = n
```

# Future Work

Visual Context of Program

Improve/Augment User's Mental Model

Develop Creative/Robust Code

Dominik Moritz

# Big Data

Volume

Velocity

**Variety**

**Big Data Borat**
@BigDataBorat

Follow

#BigData2014Predictions #1: Falling #bigdata hype force 3 Vs model to be reduced to 2 Vs model. "Variety" temporary loan to 3-D Printing

12:07 PM - 1 Jan 2014

**12** RETWEETS **4** FAVORITES

***"High variety is what keeps me up at night"***

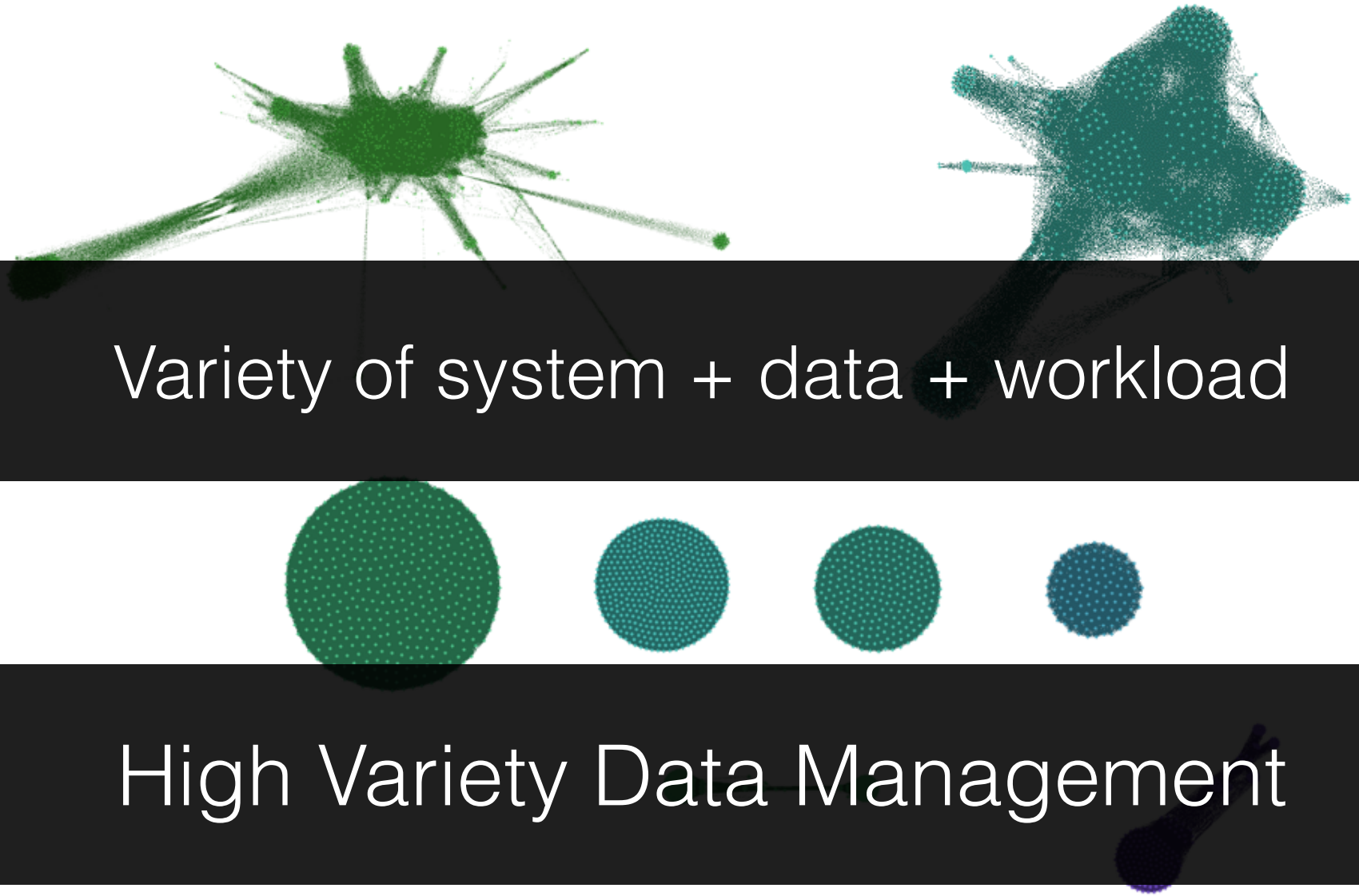
- Some data scientist

"*Variety, not volume, is the greatest big data challenge*"

- Ben Plummer, www.datawatch.com

"*Variety is the hardest V in telecom*"

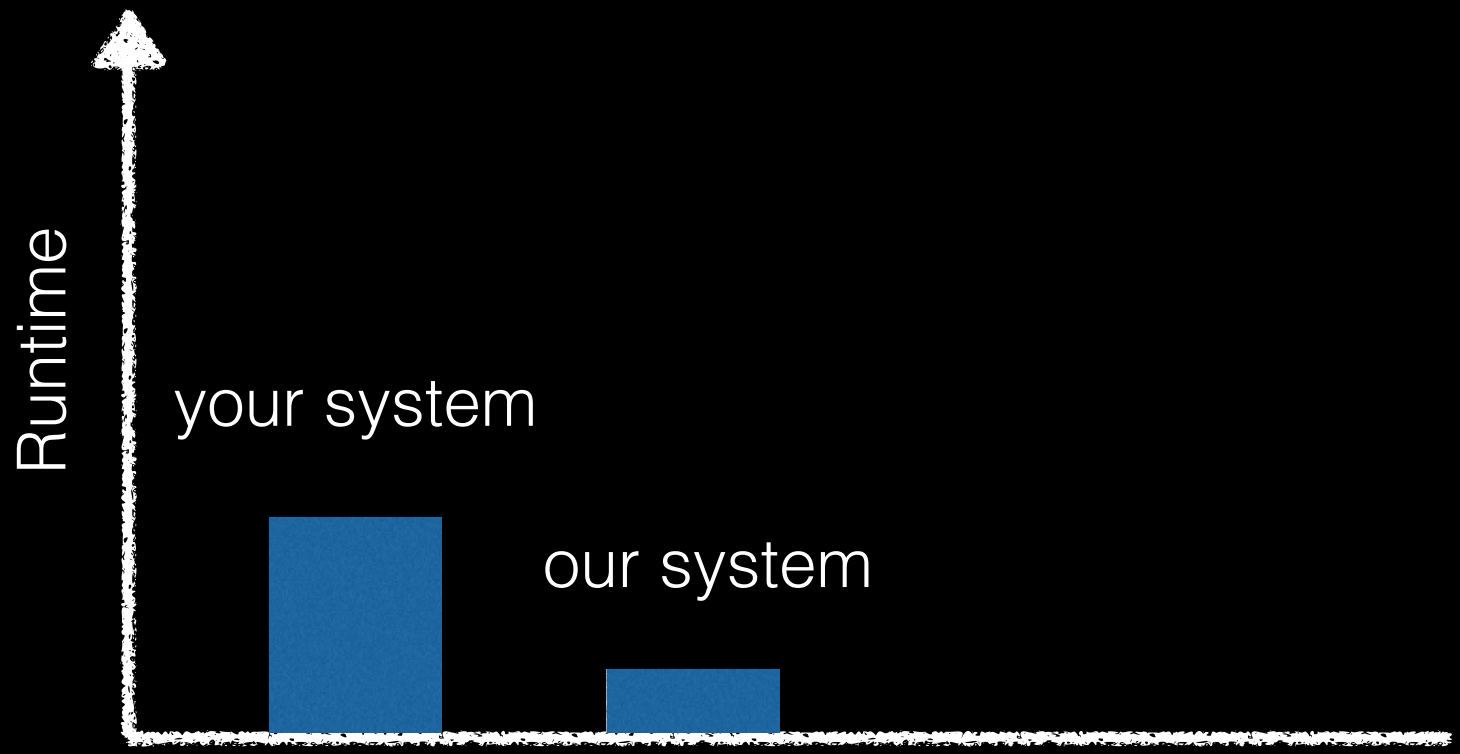
- www.ontology.com



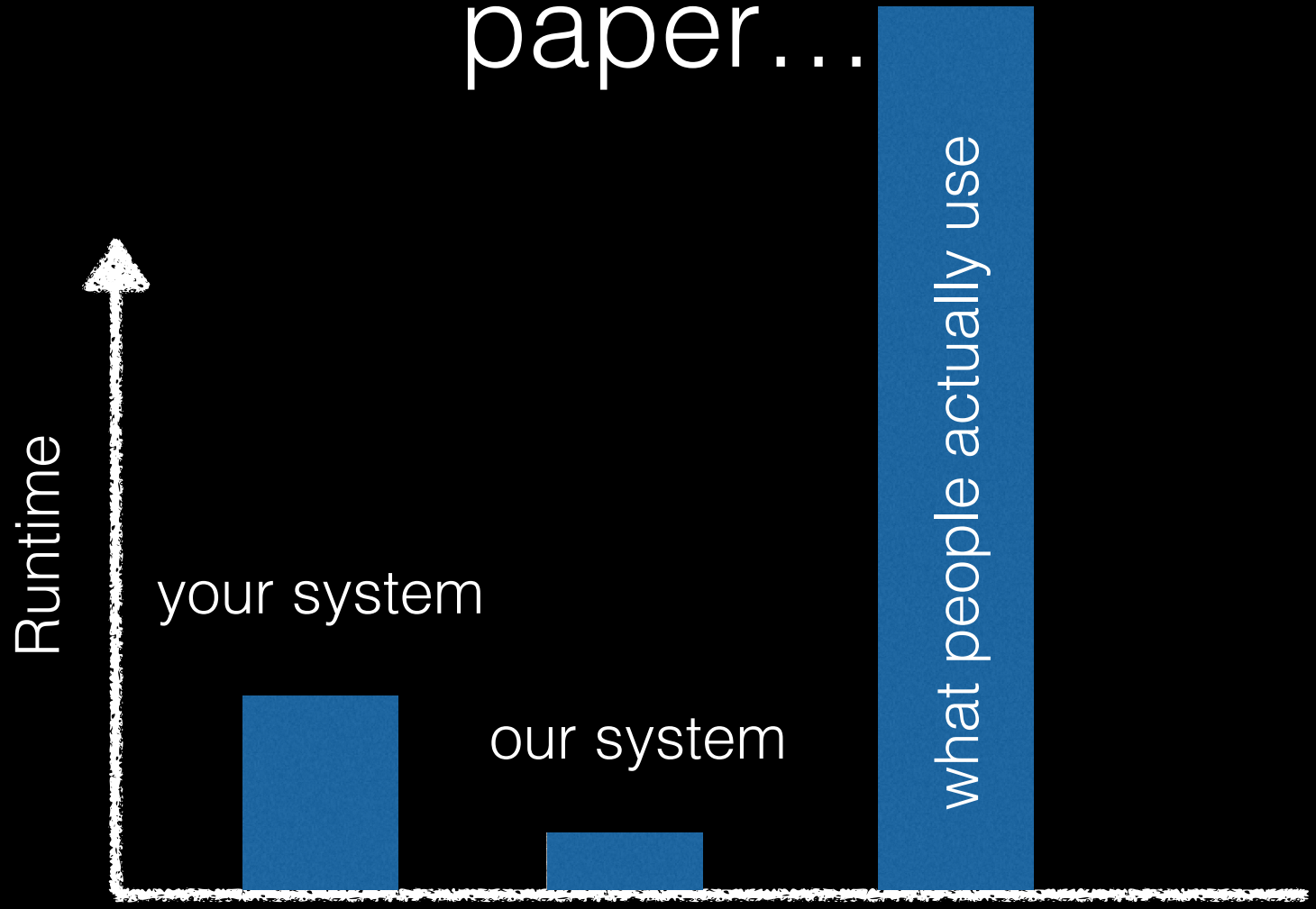
Variety of system + data + workload

High Variety Data Management

# Every database system paper…



# Every database system paper…



Myria Editor Queries Datasets Report an issue rest.myria.cs.washington.edu:1776 [72/72]

Write your code here, perhaps starting from one of the examples at the right.

```
1  OppData = scan(all_opp_v3);
2  VctData = scan(all_vct);
3
4  OppWithPop = select opp.*, vct.pop
5               from OppData as opp,
6                    VctData as vct
7               where opp.Cruise = vct.Cruise
8                 and opp.Day = vct.Day
9                 and opp.File_Id = vct.File_Id
10                and opp.Cell_Id = vct.Cell_Id;
11
12 PlanktonCount = select Cruise, count(*) as Phytoplankton
13                 from OppWithPop
14                 where pop != "beads" and pop != "noise"
15                   and fsc_small > 10000;
16
17 store(PlanktonCount, public:demo:PlanktonCount);
```

▶ Execute the Query | Parse | Myria JSON

Query Language: MyriaL

Developer Options

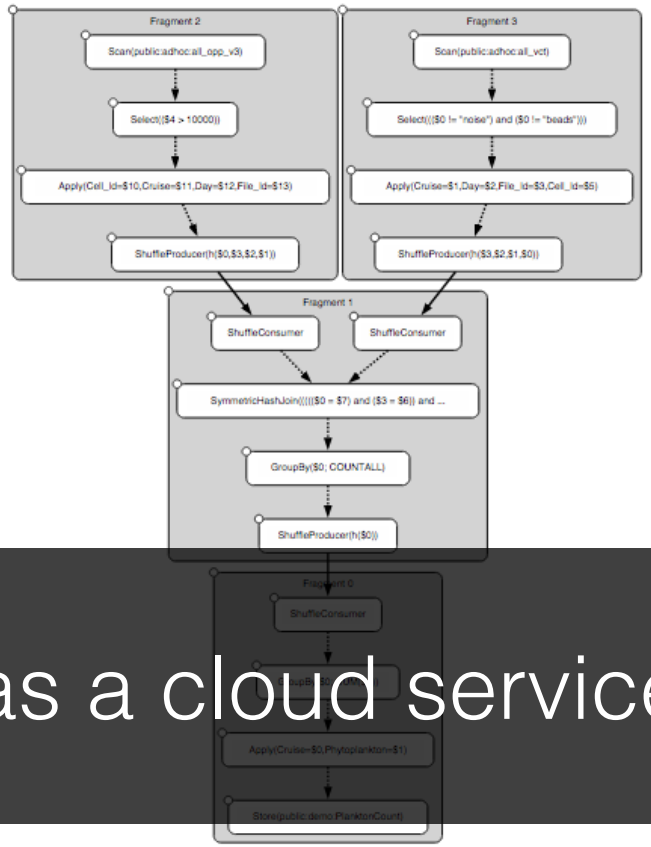
- Profile Query
  Profiling will make the query run a little bit slower but allows you to examine exactly how the query was executed.
- Push initial computation into database
  Generate SQL code to push work into the backend database.
- Compile to Multiway Join

Examples | Datasets | Query Plan | Results

Visualization of the logical and optimized physical query plan.

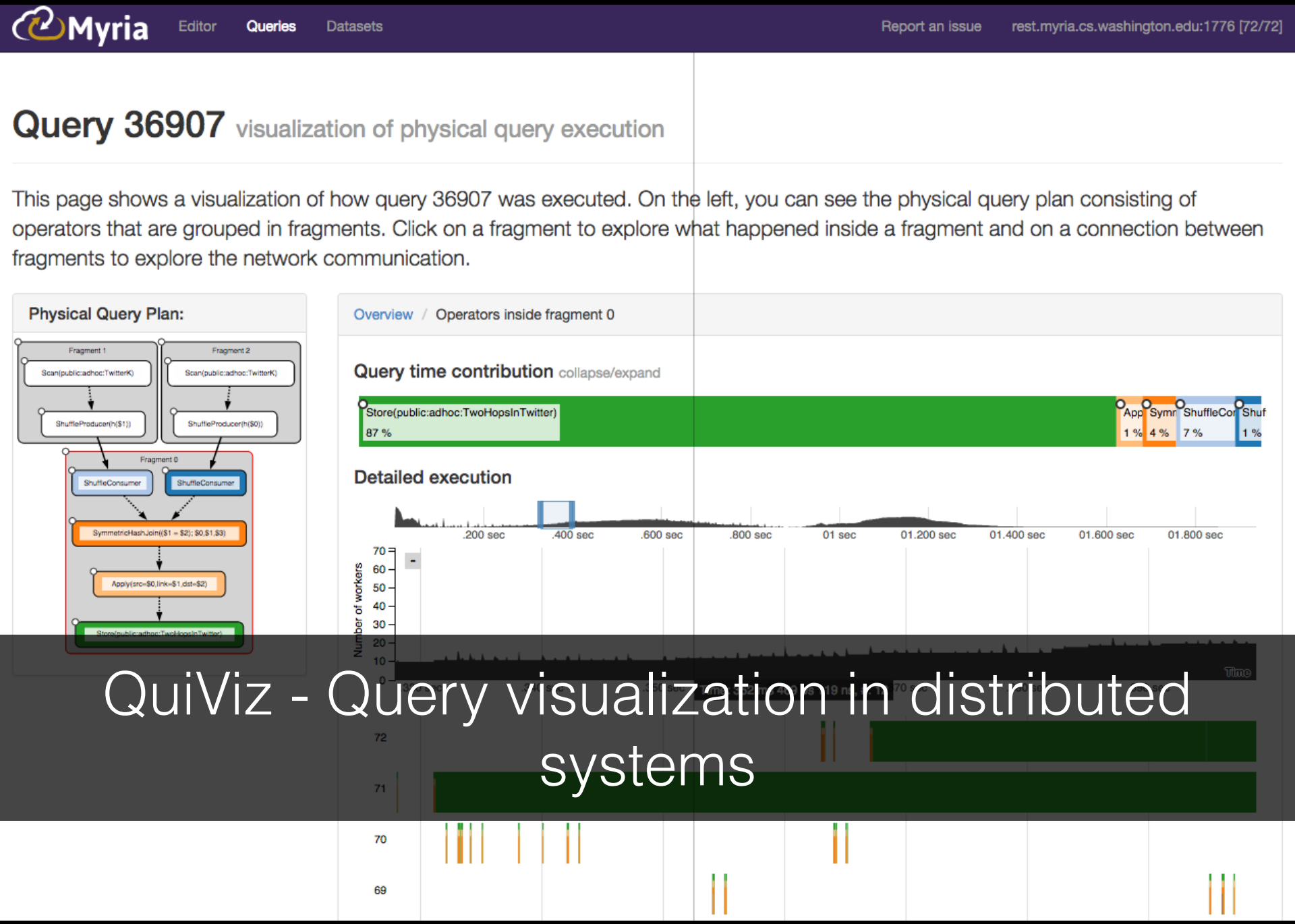
Code parsed as Relational Algebra

Relational algebra converted and optimized into a Myria Physical Plan

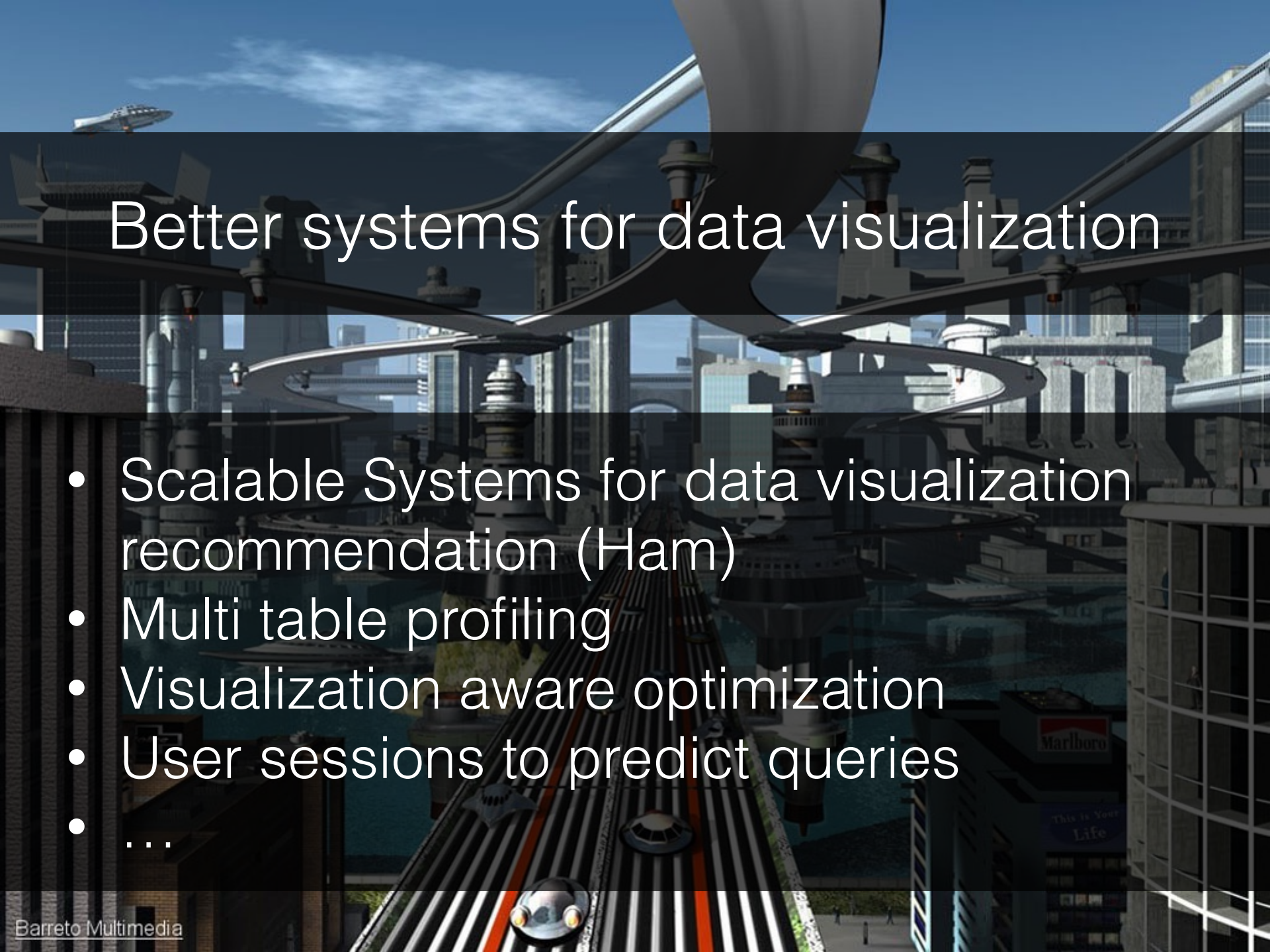



```
MyriaStore(public:demo:PlanktonCount)[MyriaApply(Cr
uise=$0,Phytoplankton=$1)[MyriaGroupBy($0; SUM($1))
[MyriaShuffleConsumer[MyriaShuffleProducer(h($0))[M
yriaGroupBy($0; COUNTALL)[MyriaSymmetricHashJoin(((
```

# Myria - Big Data analytics as a cloud service




# QuiViz - Query visualization in distributed systems




# Better systems for data visualization

- Scalable Systems for data visualization recommendation (Ham)
- Multi table profiling
- Visualization aware optimization
- User sessions to predict queries
- …

Arvind Satyanarayan

# Declarative Interaction Design for Data Visualization

Arvind Satyanarayan
Stanford University
@arvindsatya1

Kanit Wongsuphasawat
University of Washington
@kanitw

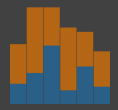
Jeffrey Heer
University of Washington
@jeffrey_heer

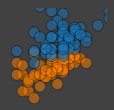


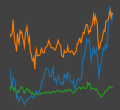
# Grammars of Graphics

| | |
|---|---|
| **Data** | Input data to visualize |
| **Transforms** | Filtering, grouping, stats, projection, layout |
| **Scales** | Map data values to visual values |
| **Guides** | Axes & legends to visualize scales |
| **Marks** | Data-representative graphics |


Area


Rect


Symbol


Line


Arc

Text

- Grammar of Graphics. Wilkinson, 2005.
- Protovis. Bostock and Heer, *InfoVis '09*.
- D3. Bostock et al. *InfoVis '11*.
- ggplot2
- Tableau (VizQL)

# Imperative Event Handling

```
var dragging = false;
d3.selectAll("rect")
  .on("mousedown", function() {
     dragging = true;
  })
  .on("mouseup", function() {
     dragging = false;
     d3.event.stopPropagation();
  })
  .on("mousemove", function() {
     var e = d3.event;
     if(!dragging) return;
     d3.select(this)
         .attr("x", e.pageX)
         .attr("y", e.pageY);
  });
```

1. Inconsistent syntactic forms for similar semantics.
   Blackwell et al. *Cognitive Dimensions of Notations*. 2001.

2. Manually maintain state and dependencies.
   Myers. *Eliminating the Spaghetti of Callbacks*. UIST 2001.
   Cooper et al. *Programming Languages and Systems*. 2006.

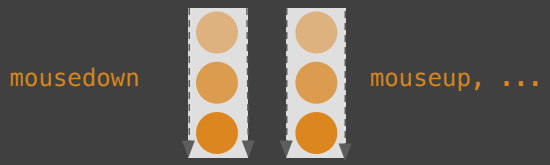
3. "Side-effects" break encapsulation.
   Cooper. *Integrating dataflow evaluation into a practical higher-order call-by-value language*. 2008.

4. "Callback hell": execution order can be unpredictable and interleaved.
   Edwards. *Coherent Reaction*. SIGPLAN 2009.

# An Interaction Grammar

# An Interaction Grammar

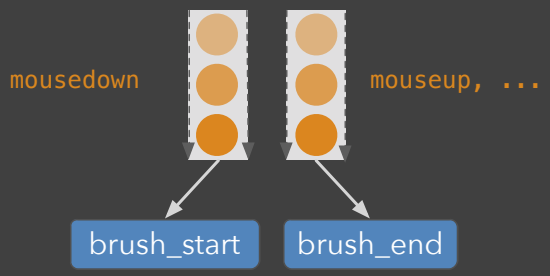
**Event Streams** Capture and compose input events



# An Interaction Grammar

**Event Streams** Capture and compose input events

**Signals** To build reactive expressions

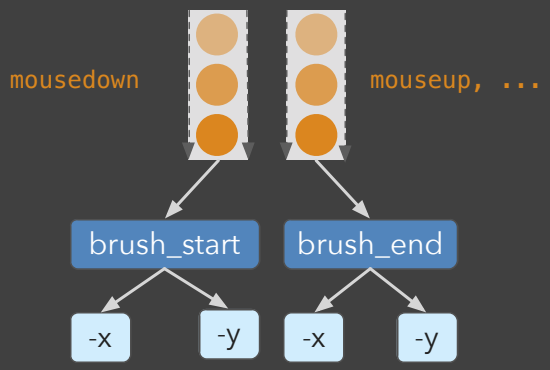


# An Interaction Grammar

**Event Streams** Capture and compose input events

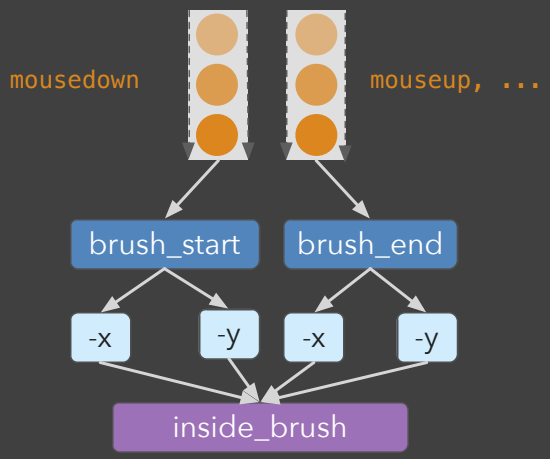
**Signals** To build reactive expressions

**Scale Inversions** Lift interactions to the data domain



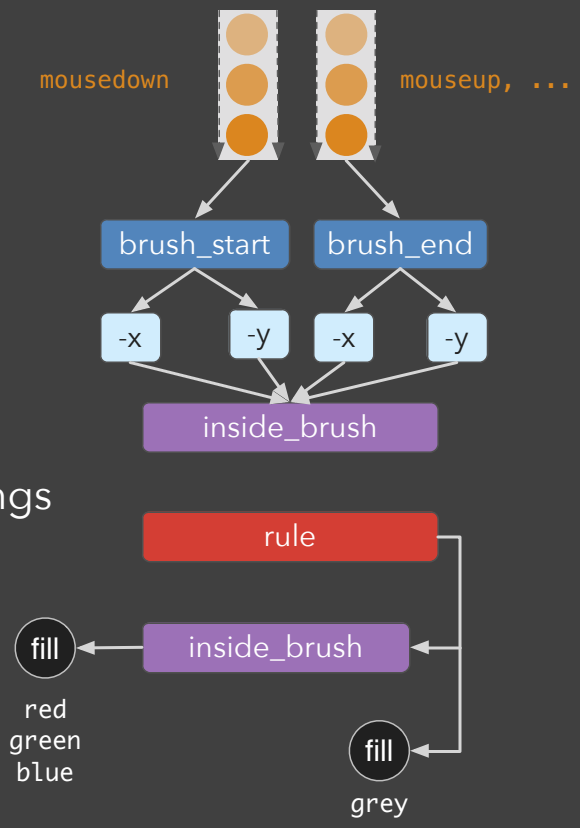
# An Interaction Grammar

| | |
|---|---|
| **Event Streams** | Capture and compose input events |
| **Signals** | To build reactive expressions |
| **Scale Inversions** | Lift interactions to the data domain |
| **Predicates** | Define interactive selections |



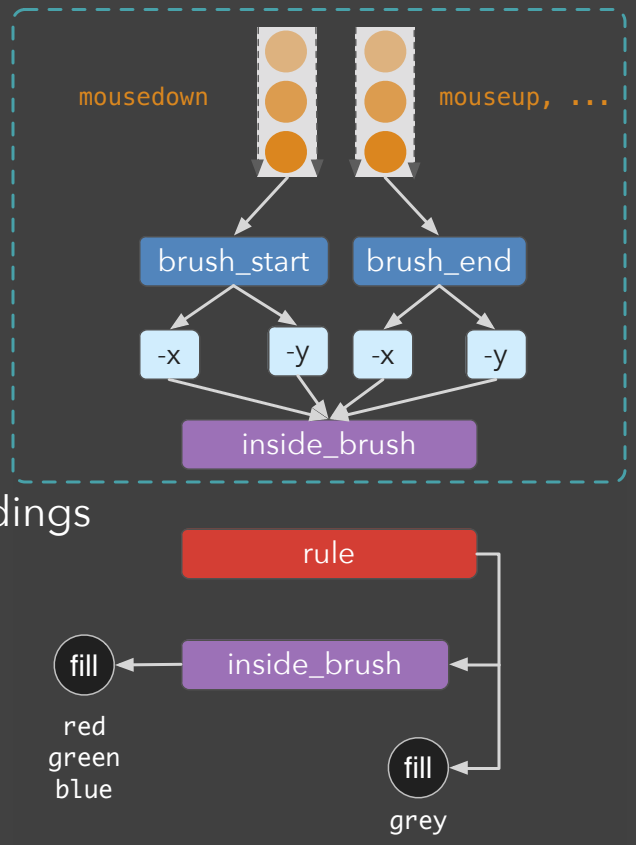
# An Interaction Grammar

| | |
|---|---|
| **Event Streams** | Capture and compose input events |
| **Signals** | To build reactive expressions |
| **Scale Inversions** | Lift interactions to the data domain |
| **Predicates** | Define interactive selections |
| **Production Rules** | Predicate chain to manipulate visual encodings |



# An Interaction Grammar

| | |
|---|---|
| **Event Streams** | Capture and compose input events |
| **Signals** | To build reactive expressions |
| **Scale Inversions** | Lift interactions to the data domain |
| **Predicates** | Define interactive selections |
| **Production Rules** | Predicate chain to manipulate visual encodings |
| **Interactors** | “File-format” for reusable interactions |

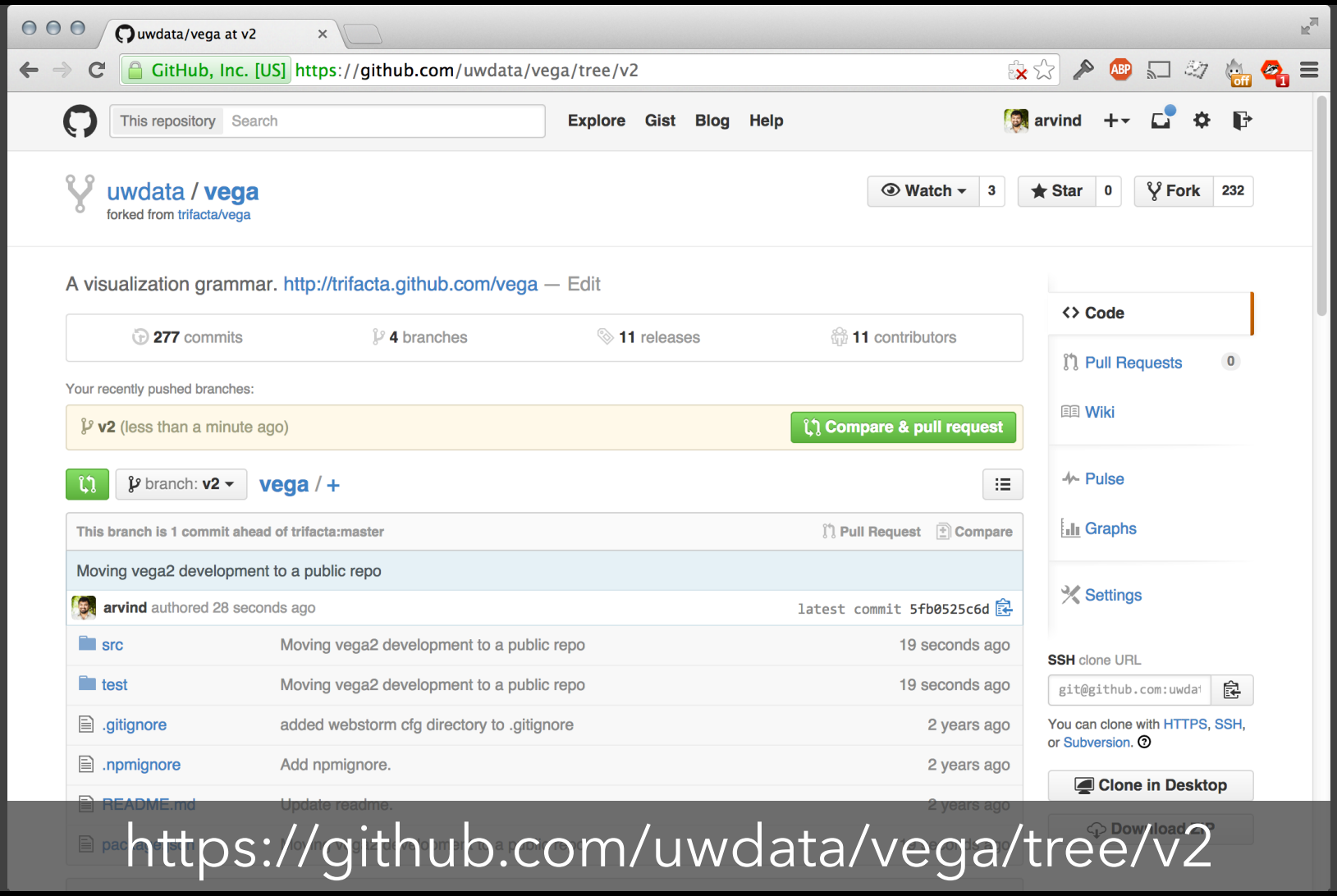


# Current Work

1. What is the optimal implementation?

   Vega. http://trifacta.github.io/vega/

   Arasu et al. *Stream: The Stanford Data Stream Management System*. 2004.




https://github.com/uwdata/vega/tree/v2

# Current Work

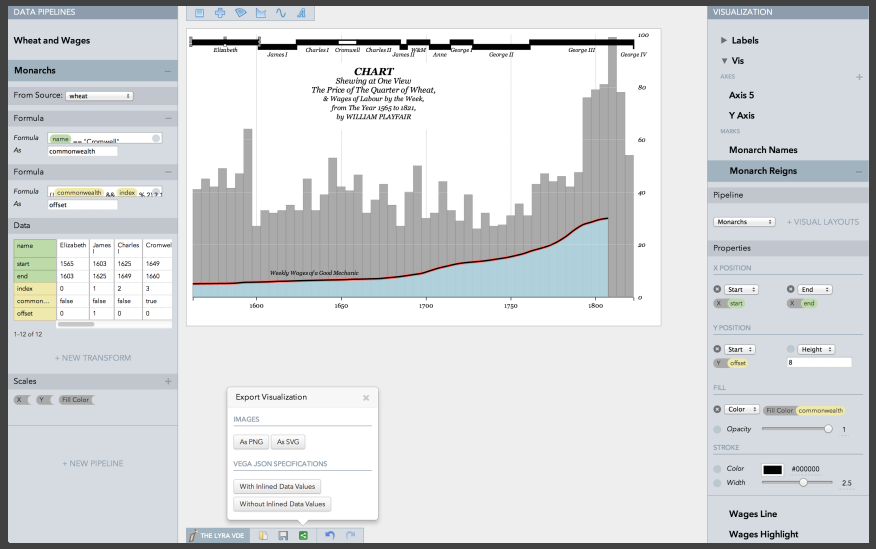
1. What is the optimal implementation?

   Vega. http://trifacta.github.io/vega/

   Arasu et al. *Stream: The Stanford Data Stream Management System*. 2004.

2. Interaction by Demonstration?

   Satyanarayan & Heer. *Lyra: A Interactive Visualization Design Environment*. EuroVis 2014.



# Declarative Interaction Design for Data Visualization

https://github.com/uwdata/vega/tree/v2

Arvind Satyanarayan
Stanford University
@arvindsatya1

Kanit Wongsuphasawat
University of Washington
@kanitw

Jeffrey Heer
University of Washington
@jeffrey_heer



Allison Smith

# Allison Smith (post-doc, eScience Institute, UW)

**Scientific disciplines:** Ecophysiology and Oceanography

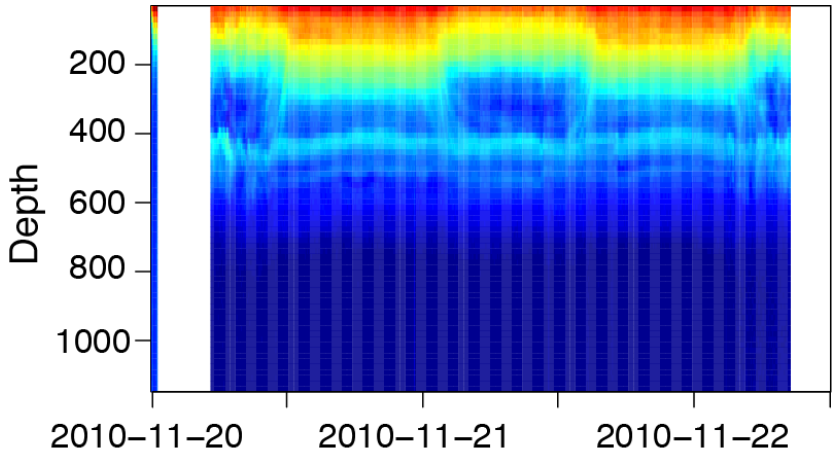
**Data sources:** Multi-dimensional environmental data from ships, autonomous sensors, satellites, etc.

**Applications:** Forecasting the effects of climate change and verifying ecosystem models

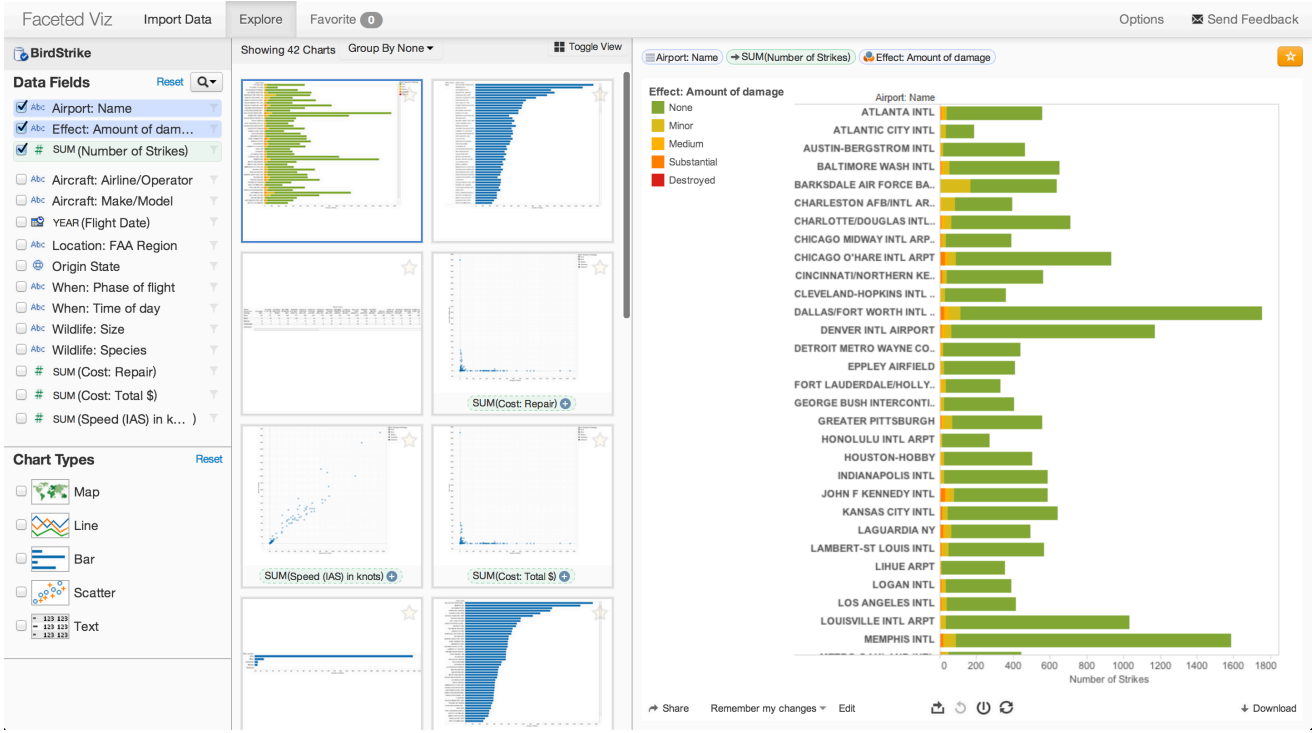
**Analysis tools:** R, Python, NOAA Ferret, NCAR Command Language, AWK, sed, etc.

Residents of the Mesopelagic Zone

Scattering Intensity



Kanit “Ham” Wongsuphasawat




# Accelerating Data Analysis with Visualization Recommendations


Kanit “Ham” Wongsuphasawat

with Anushka Anand, Jock Mackinlay, Jeffrey Heer,
Zening Qu, Dominik Moritz, and Jeff Snyder

Çağatay Demiralp


# Çağatay Demiralp

Former Postdoc with Jeff Heer

Research Staff Member at IBM T. J. Watson

Worried NYT Reader*

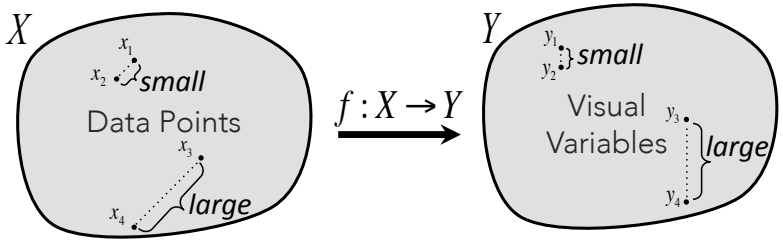
* picture is representative

# FOUNDATIONS

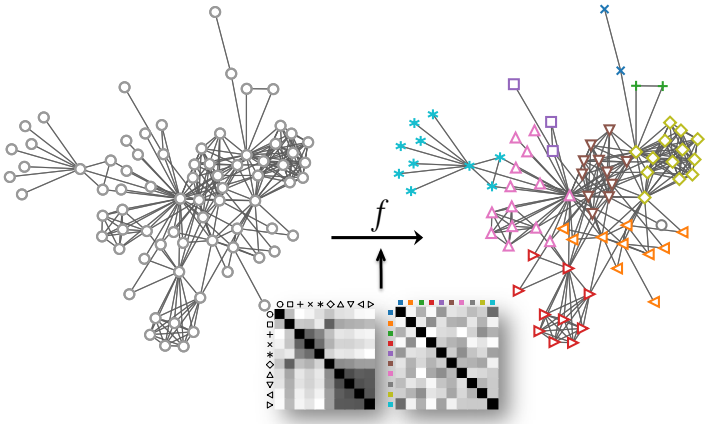
What is a good visualization?

Can we automatically generate & evaluate good visualizations?

**NEED OPERATIONAL MODELS TO INCORPORATE EMPIRICAL PERCEPTION!**




visual embedding

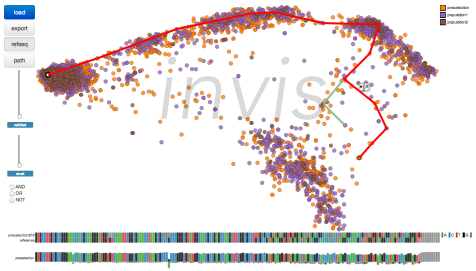



perceptual kernels

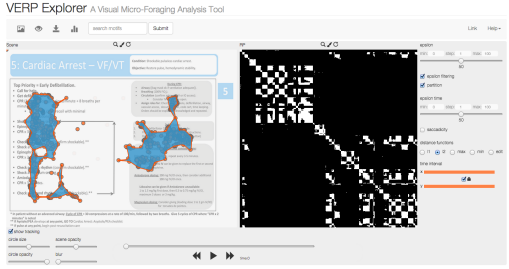
# DOMAIN TOOLS

How can we interactively explore thousands of genomes?

How can we analyze eye movements to debug visual design?




*invis*




verp