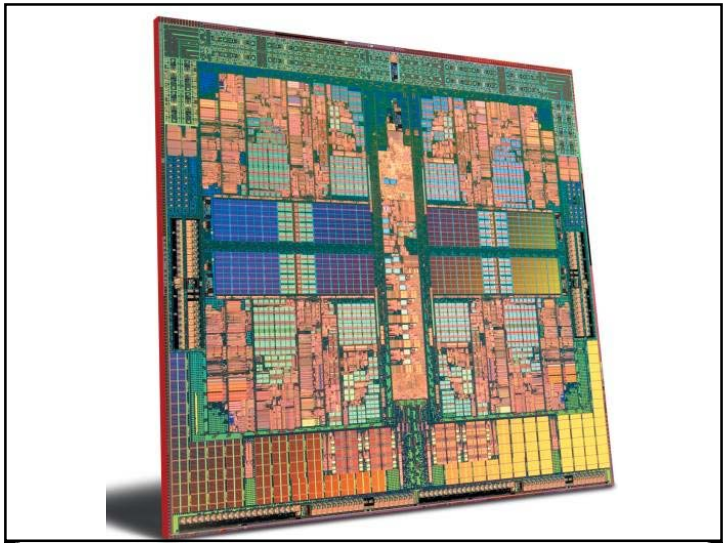
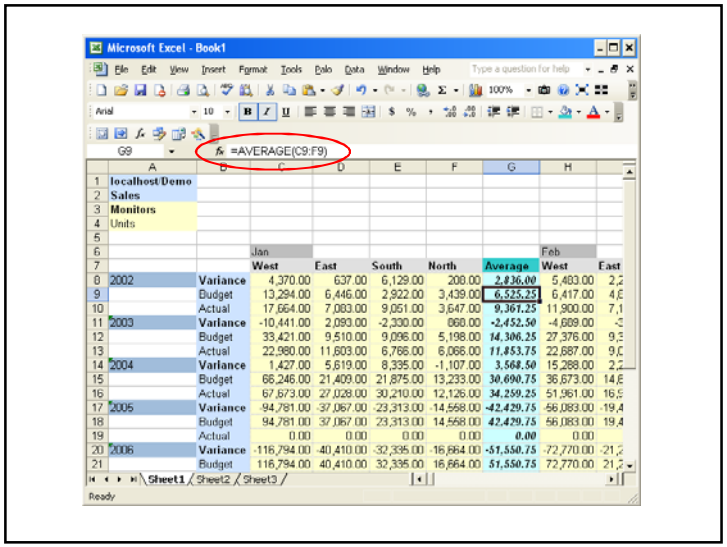
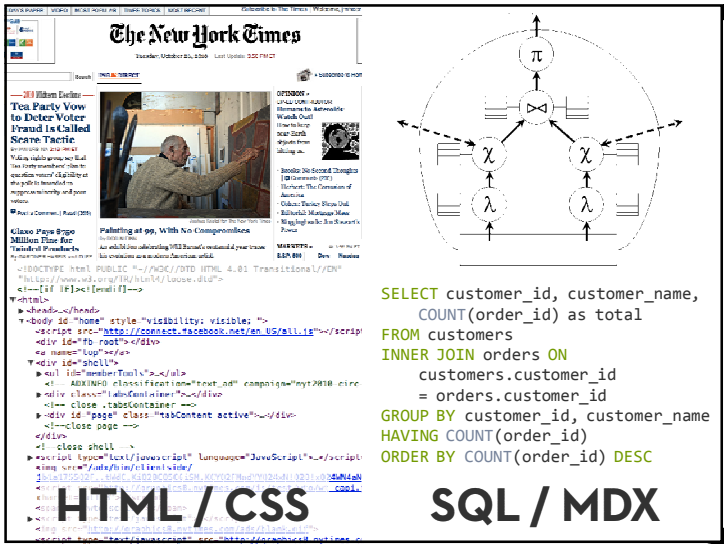
domain specific languages for

Interactive Data Analysis

Jeffrey Heer Stanford University








```
SELECT customer_id, customer_name,
    COUNT(order_id) as total
FROM customers
INNER JOIN orders ON
    customers.customer_id
    = orders.customer_id
GROUP BY customer_id, customer_name
HAVING COUNT(order_id)
ORDER BY COUNT(order_id) DESC
```

SQL / MDX

End-User Programmers

People who write programs, but *not* as their primary job function.

Instead, they must write programs in support of achieving their main goal, which is something else, such as accounting, designing a web page, doing office work, scientific research, etc.

Myers, Ko & Burnett 2006

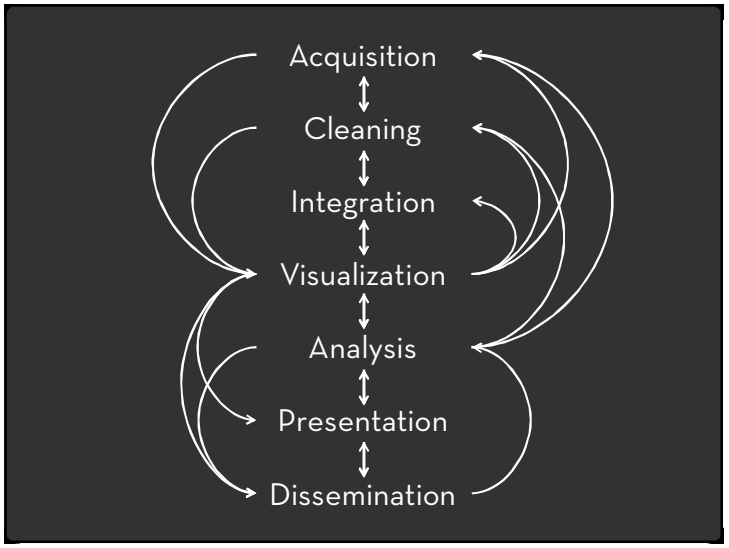
End-User Programming Methods

Domain Specific Languages
Keyword Programming
Programming-by-Demonstration
Visual Programming

Today:

A DSL for making interactive visualizations (*Protovis*)

An interactive visual tool for making statements in a DSL (*Wrangler*)



How do people create visualizations?

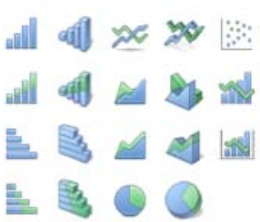
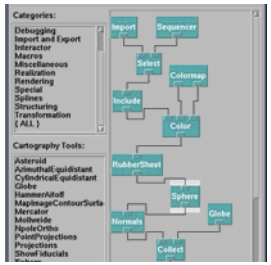


Chart Typology
Pick from a stock of templates
Easy-to-use but limited expressiveness
Prohibits novel designs, new data types



Component Architecture
Permits more combinatorial possibilities
Novel views require new operators,
which requires software engineering.

Efficiency ↑ ↓ **Expressiveness**

Chart Typologies
Excel, Many Eyes, Google Charts

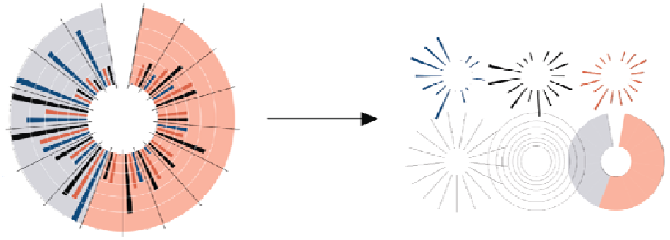
Visual Analysis Languages
Tableau VizQL, ggplot2, HiVE

?

Component Model Architectures
Improvise, Prefuse, Flare

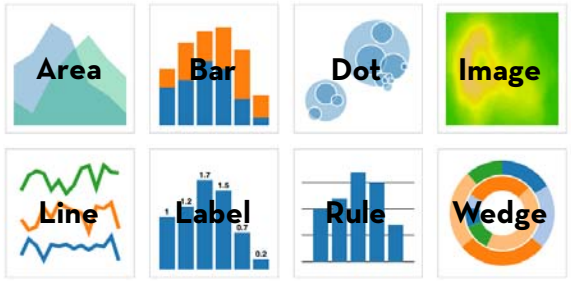
Graphics APIs
OpenGL, Java2D, GDI+, Processing

Protovis: A Declarative Language for Visualization



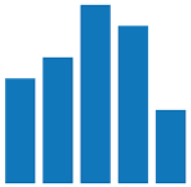
A graphic is a composition of data-representative marks.

with **Mike Bostock** & **Vadim Ogievetsky**



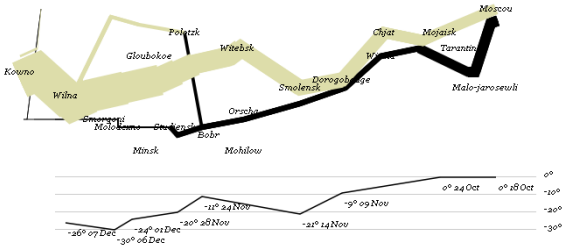
Protovis

Create customized visualizations using a declarative specification language.



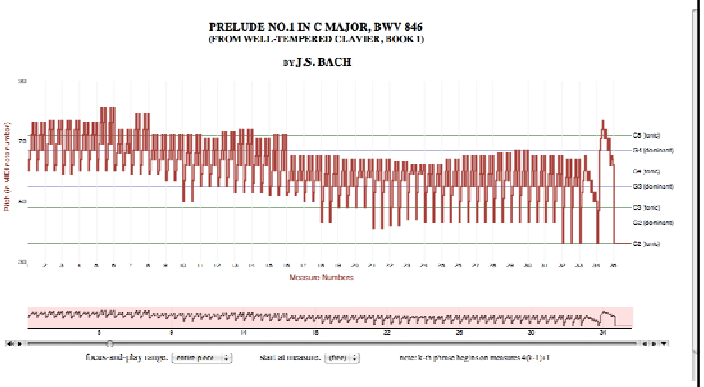
```
var vis = new pv.Panel();
vis.add(pv.Bar)
  .data([1, 1.2, 1.7, 1.5, .7])
  .bottom(10)
  .width(20)
  .height(function(d) d * 70)
  .left(function()  this.index * 25 + 20);
vis.render();
```

Protovis (http://protovis.org) – Declarative Visualization Specification

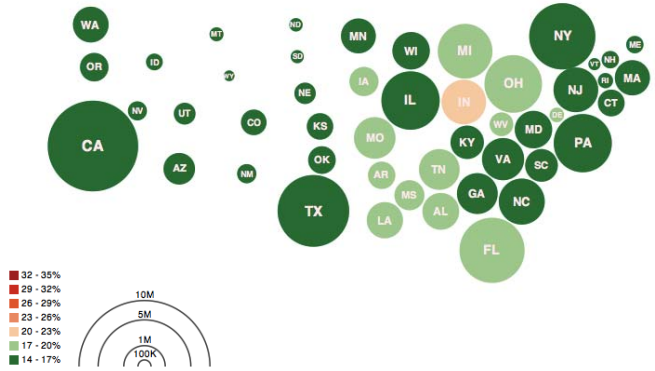



```
var army = pv.nest(napoleon.army, "dir", "group");
var vis = new pv.Panel();

var lines = vis.add(pv.Panel).data(army);
lines.add(pv.Line)
  .data(function() army[this.idx])
  .left(lon).top(lat).size(function(d) d.size/8000)
  .strokeStyle(function() color[army[panelIndex][0].dir]);

vis.add(pv.Label).data(napoleon.cities)
  .left(lon).top(lat)
  .text(function(d) d.city).font("italic 10px Georgia")
  .textAlign("center").textBaseline("middle");
```

```
vis.add(pv.Rule).data([0,-10,-20,-30])
  .top(function(d) 300 - 2*d - 0.5).left(200).right(150)
  .lineWidth(1).strokeStyle("#ccc")
  .anchor("right").add(pv.Label)
    .font("italic 10px Georgia")
    .text(function(d) d+"°").textBaseline("center");

vis.add(pv.Line).data(napoleon.temp)
  .left(lon).top(tmp) .strokeStyle("#0")
.add(pv.Label)
  .top(function(d) 5 + tmp(d))
  .text(function(d) d.temp+"° "+d.date.substr(0,6))
  .textBaseline("top").font("italic 10px Georgia");
```

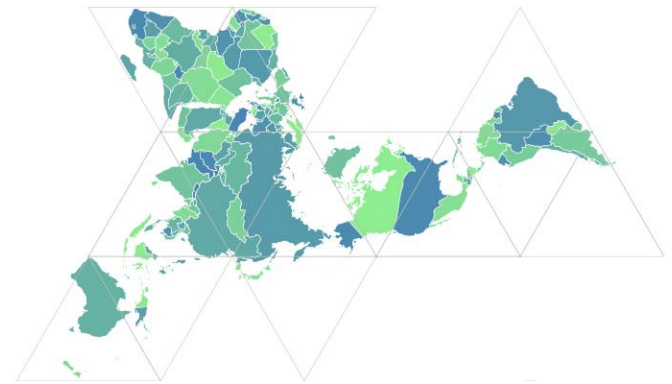


Bach's Prelude #1 in C Major | Jieun Oh

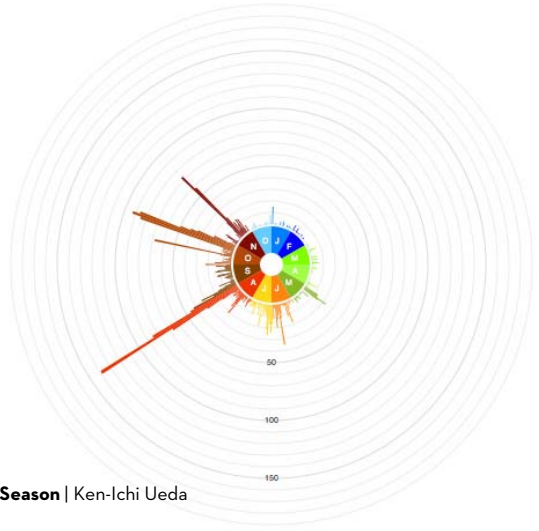


Obesity Map Vadim Ogievetsky

Obesity Map Vadim Ogievetsky



Dymaxion Maps | Vadim Ogievetsky



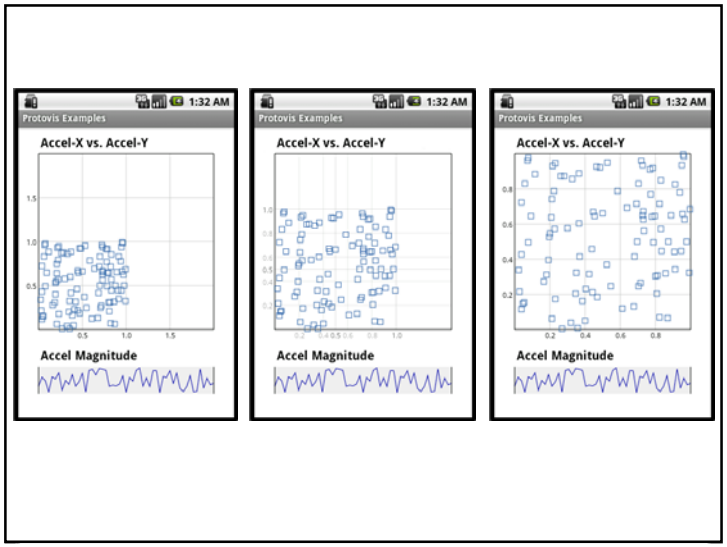
FlickrSeason | Ken-Ichi Ueda

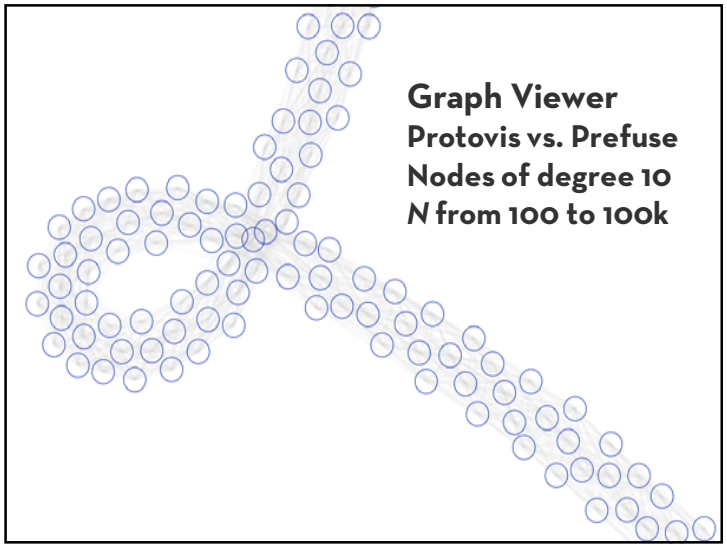
Exploiting Declarative Specification

Protovis has led to faster designs, less code
- Job Voyager: 5x less code, 10x less dev time
- Over 40,000 downloads and widely in use

Multiple implementations: JavaScript & Java

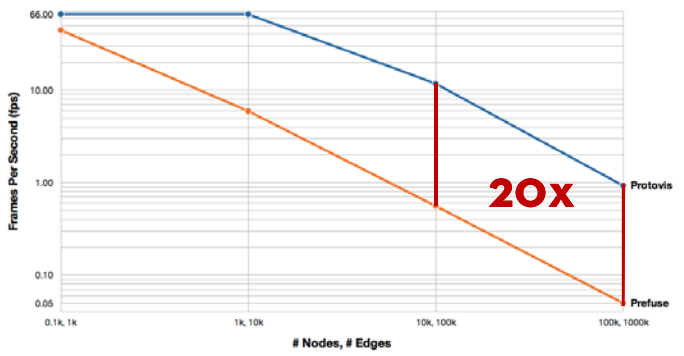
Behind-the-scenes optimization & parallelization
- 20x scalability over prior systems (in Java)





Graph Viewer
Protovis vs. Prefuse
Nodes of degree 10
***N* from 100 to 100k**

Interactive Graph Layout Performance



Design Process

Determine the domain entities and operators

Iterative development with domain expert (me)

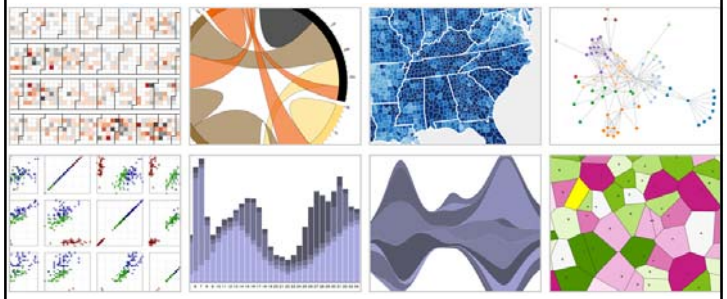
Generate alternative designs
  Write hypothetical code; compare & contrast

Minimize surface area

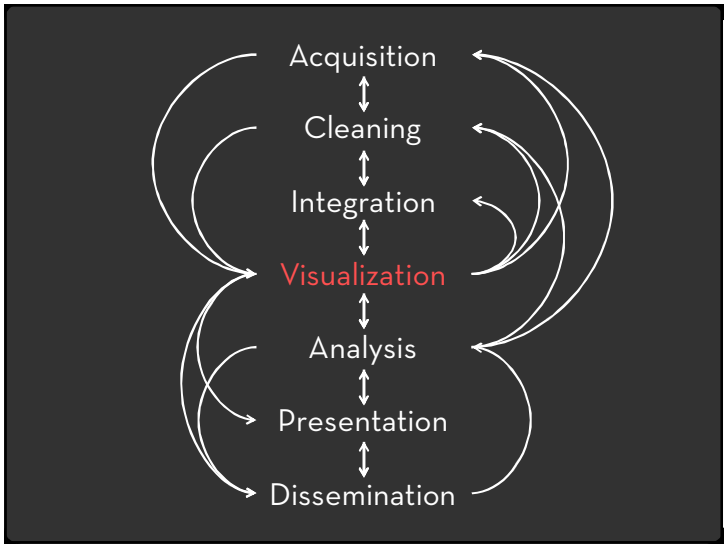
Expressiveness, efficiency, accessibility

Cognitive Dimensions of Notation [Green et al]

d3.js Data-Driven Documents



with **Mike Bostock** & **Vadim Ogievetsky**



Bureau of Justice Statistics - Data Online
http://bjs.ojp.usdoj.gov/

Reported crime in Alabama

Year	Population	Property crime rate	Burglary rate	Larceny-theft rate	Motor vehicle theft rate
2004	4525375 4029.3	987	2732.4	309.9	
2005	4548327 3900	955.8	2656	289	
2006	4599030 3937	968.9	2645.1	322.9	
2007	4627851 3974.9	980.2	2687	307.7	
2008	4661900 4081.9	1080.7	2712.6	288.6	

Reported crime in Alaska

Year	Population	Property crime rate	Burglary rate	Larceny-theft rate	Motor vehicle theft rate
2004	657755 3370.9	573.6	2456.7	340.6	
2005	663253 3615	622.8	2601	391	
2006	670053 3582	615.2	2588.5	378.3	
2007	683478 3373.9	538.9	2480	355.1	
2008	686293 2928.3	470.9	2219.9	237.5	

Reported crime in Arizona

Year	Population	Property crime rate	Burglary rate	Larceny-theft rate	Motor vehicle theft rate
2004	5739879 5073.3	991	3118.7	963.5	
2005	5953007 4827	946.2	2958	922	
2006	6166318 4741.6	953	2874.1	914.4	
2007	6338755 4502.6	935.4	2780.3	786.7	
2008	6500180 4087.3	894.2	2605.3	587.8	

Reported crime in Arkansas

Year	Population	Property crime rate	Burglary rate	Larceny-theft rate	Motor vehicle theft rate
2004	2750000 4033.1	1096.4	2699.7	237	
2005	2775708 4068	1085.1	2720	262	
2006	2810872 4021.6	1154.4	2596.7	270.4	
2007	2834797 3943.5	1124.4	2574.6	246.5	
2008	2855390 3843.7	1182.7	2433.4	227.6	

Reported crime in California

Year	Population	Property crime rate	Burglary rate	Larceny-theft rate	Motor vehicle theft rate
2004	35842038	3423.9	686.1	2033.1	704.8
2005	36134147	3321	692.9	1915	712
2006	36457549	3175.2	676.9	1831.5	666.8
2007	36553215	3032.6	648.4	1784.1	600.2
2008	36756666	2940.3	646.8	1769.8	523.8

Reported crime in Colorado

Year	Population	Property crime rate	Burglary rate	Larceny-theft rate	Motor vehicle theft rate
2004	4601821 3918.5	717.3	2679.5	521.6	

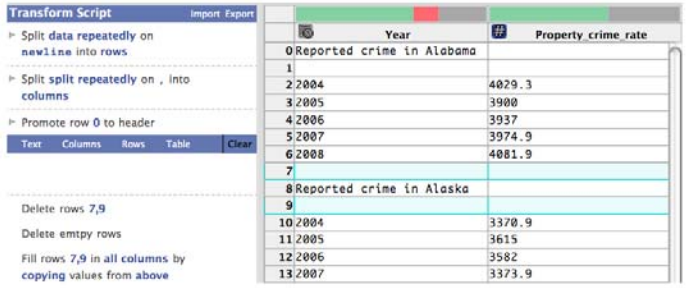
Data Wrangling (n):

A process of iterative data exploration and transformation that enables analysis.

The goal of wrangling is to make data *useful*:

- Map data to a form readable by downstream tools (database, stats, visualization, …)
- Identify, document, and (where possible) address data quality issues.

DataWrangler



with **Sean Kandel**, Philip Guo, Ravi Parikh, Andreas Paepcke & Joe Hellerstein

From UI to running code…

```
split('data').on(NEWLINE).max_splits(NO_MAX)
split('split').on(COMMA).max_splits(NO_MAX)
columnName().row(0)
delete(isEmpty())
extract('Year').on(/.*/).after(/in /)
fill('extract').method(COPY).direction(DOWN)
delete('Year starts with "Reported crime in"')
columnName('extract').to('State')
```

Wrangler in 2 Parts...

1. Declarative data transformation language
   Tuple mapping – split, merge, extract, delete
   Reshaping – e.g., fold, unfold (cross-tabulation)
   Lookups and joins – e.g., FIPS code to US state
   Sorting, aggregation, etc.

   Informed by prior work in databases:
   Potter's Wheel & SchemaSQL

Wrangler in 2 Parts...

1. Declarative data transformation language
   +
2. Mixed-initiative interface for data transforms
   Select data elements of interest
   Suggest applicable transforms
   Enable rapid **preview and refinement**

Transform Suggestion

Interaction
↓
Infer Operands
↓
Generate Transforms
↓
Rank Transforms
↓
Present Top-N

Transform Suggestion

Interaction
↓
Infer Operands
↓
Generate Transforms
↓
Rank Transforms
↓
Present Top-N

- Text Selection
- Text Editing
- Row Selection
- Column Selection
- Transform Menu
- Click Data Quality Meter

Transform Suggestion

Interaction
↓
Infer Operands
↓
Generate Transforms
↓
Rank Transforms
↓
Present Top-N

Map user input to transform operands.

Example: text highlight maps to row, column, and text selections.

Inferred text selections include string indices and regular expressions.

Text Selection Inference

```
Series Id: LNU02000000
```
-> ^ STR WS STR SYM WS STR NUM $

```
Series Id: LNU02000000
```
MATCH Indices 11-22
MATCH LNU02000000
MATCH LNU NUM
MATCH STR NUM
AFTER : WS

Transform Suggestion

Interaction
↓
Infer Operands
↓
Generate Transforms
↓
Rank Transforms
↓
Present Top-N

Map user input to transform operands.

Example: text highlight maps to row, column, and text selections.

Inferred text selections include string indices and regular expressions.

Transform Suggestion

Interaction
↓
Infer Operands
↓
Generate Transforms
↓
Rank Transforms
↓
Present Top-N

Enumerate transforms that accept inferred operands as input.

Set unmatched params to default values.

Apply filter heuristics: No-ops, delete-all, and overly sparse outputs.

Transform Suggestion

Interaction
↓
Infer Operands
↓
Generate Transforms
↓
Rank Transforms
↓
Present Top-N

Sort transforms by:
- Selection status
- Specification difficulty
- Frequency in corpus
- Description length

Transform Suggestion

Interaction
↓
Infer Operands
↓
Generate Transforms
↓
Rank Transforms
↓
Present Top-N

Comparative Evaluation

Compared Wrangler performance to Excel with 3 data cleaning tasks on small data sets.

Median completion time for Wrangler at least twice as fact in all tasks ($p < 0.001$).

Suggestions and visual previews used heavily.

Conclusions

Performance, Portability, Productivity
Expressiveness, Efficiency, Accessibility

DSLs should support domain reasoning
- ... by end user programmers
- ... by optimizing compilers
- ... by development tools

All three might influence DSL design.

Future work: models for DSL program inference.