

CS 8803-MDS

Human-in-the-loop Data

Analytics

Lecture 25

11/21/22

Logistics

What's happening in the rest of the semester:

- 11/23 (Wed): No class (Thanksgiving)

- 11/28 (Mon): additional OH during class time

- 11/30 (Wed): draft paper (due 4PM) and peer review (in class)

- 12/05, 12/07: final presentation (in class)

- 12/09 (Fri): final paper (due 5PM)

Grading breakdown

Project: 50%

- ~~• Proposal: 5%~~
- ~~• Progress Report: 5%~~
- ~~• Evaluation Plan: 5%~~
- Draft Paper + Peer Review: 5%
- Final Presentation: 10%
- Final Paper: 20%

Draft Paper

Due 11/30 @ 4PM

- Paper outline:
 - section and subsection header, main figures
- Draft paper:
 - expect some contents in the methods and evaluation sections
 - other sections can stay in outline form

Final Presentation

12/05, 12/07 in class

- 10 min + 2min questions
- 6 groups per class, randomized order (TBA next week)
- Attendance for both sessions are mandatory
- Grading will be based on course staff evaluation + peer evaluation

Final Paper

Due 12/09 @ 5PM

- ≥ 4 pages for one-person teams, ≥ 6 pages for everyone else
- no more than 8 pages
- Link to code
- Team Dynamics Assessment Form

Today's class

Falx: Synthesis-Powered Visualization Authoring

Authors: Vishnu

Reviewer: Sankalp

Archaeologist: Yiheng

Falx: Synthesis-Powered Visualization Authoring

Chenglong Wang, Yu Feng, Rastislav Bodik, Isil Dillig, Alvin Cheung, Amy J. Ko

Presented by: Vishnu Krishnan



What is Visualization?

What is Visualization?

Visualizations are mappings from **data columns** to **geometric properties**.

- *“The Grammar of Graphics” [Wilkinson 1999]*

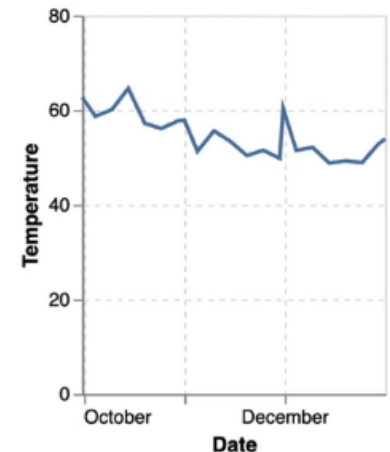
What is Visualization?

Visualizations are mappings from **data columns** to **geometric properties**.

- *“The Grammar of Graphics” [Wilkinson 1999]*

Date	Temperature
2011-10-01	62.7
2011-10-05	58.7
2011-10-10	60.1
...	...
2011-12-30	52.9

`line(x ← Date,
y ← Temperature)`



Visualization in Practice

Can you find the problem?

Date	New York	San Francisco
2011-10-01	63.4	62.7
2011-10-05	64.2	58.7
2011-10-10	71.2	60.1
...
2012-09-30	62.3	55.1

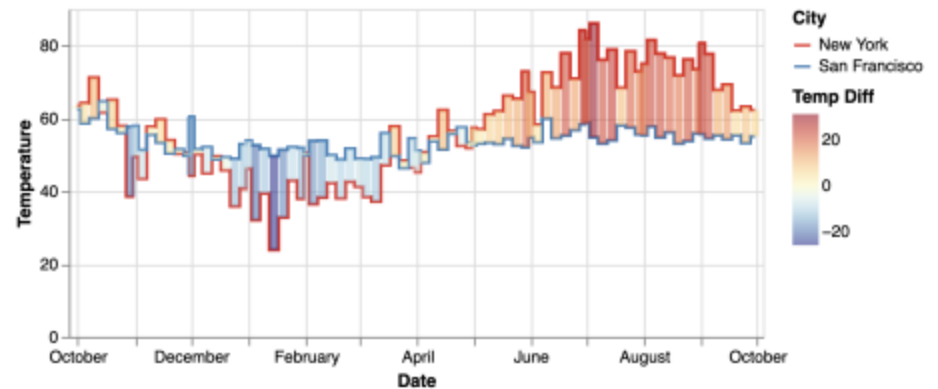


Figure 3: A visualization that compares New York and San Francisco temperatures between 2011-10-01 and 2012-09-30.

Visualization in Practice

Problem – There exists a mismatch between the design and the input data layout -> The input data layout requires data transformations.

Date	New York	San Francisco
2011-10-01	63.4	62.7
2011-10-05	64.2	58.7
2011-10-10	71.2	60.1
...
2012-09-30	62.3	55.1

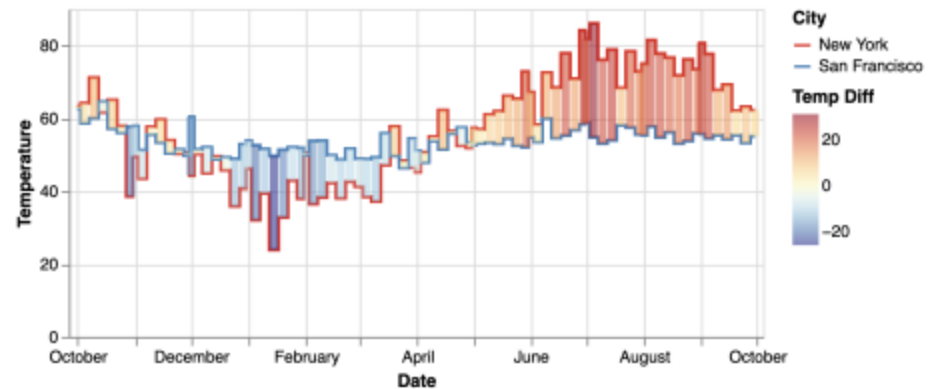
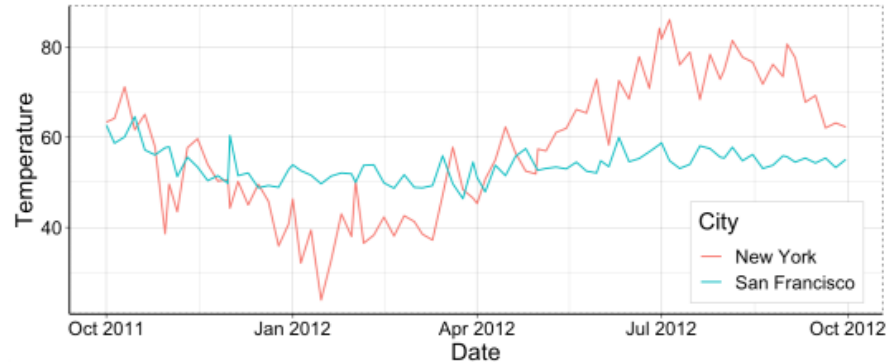


Figure 3: A visualization that compares New York and San Francisco temperatures between 2011-10-01 and 2012-09-30.

Usage Scenario

User Experience in R -



(a) A line chart that shows temperature trends.

Layer – 1 transformation and visualization

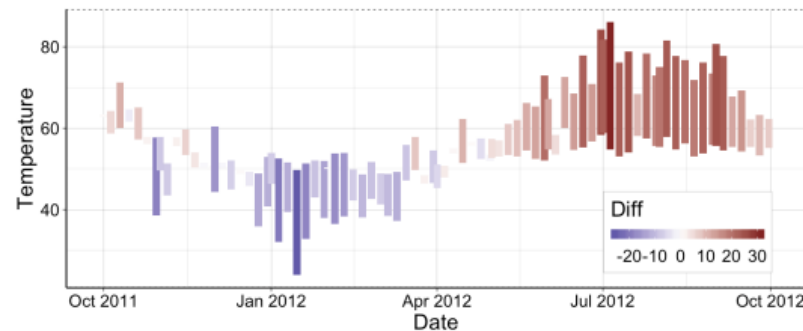
```
df1 <- pivot_longer(data = df,  
  cols = ("New York", "San Francisco"),  
  names_to = "City", values_to = "Temperature")
```

Date	City	Temperature
2011-10-01	New York	63.4
2011-10-01	San Francisco	62.7
...
2012-09-30	San Francisco	55.1

```
plot1 <- ggplot(data = df1) +  
  geom_line(aes(x = `Date`, y = `City`,  
    color= `Temperature`, group = `Temperature`))
```

Usage Scenario

User Experience in R -



(b) A bar chart that visualizes temperature difference.

Figure 4: Two visualizations created in R that compare New York and San Francisco temperatures.

Layer – 2 transformation and visualization

```
df2 <- mutate(df, Diff = `New York` - `San Francisco`)
plot2 <- ggplot(df2) +
  geom_rect(aes(xmin = `Date`, xmax = `Date`,
               ymin = `New York`, ymax = `San Francisco`,
               fill = `Diff`))
```

Usage Scenario

User Experience in Falx – Visualization by Example

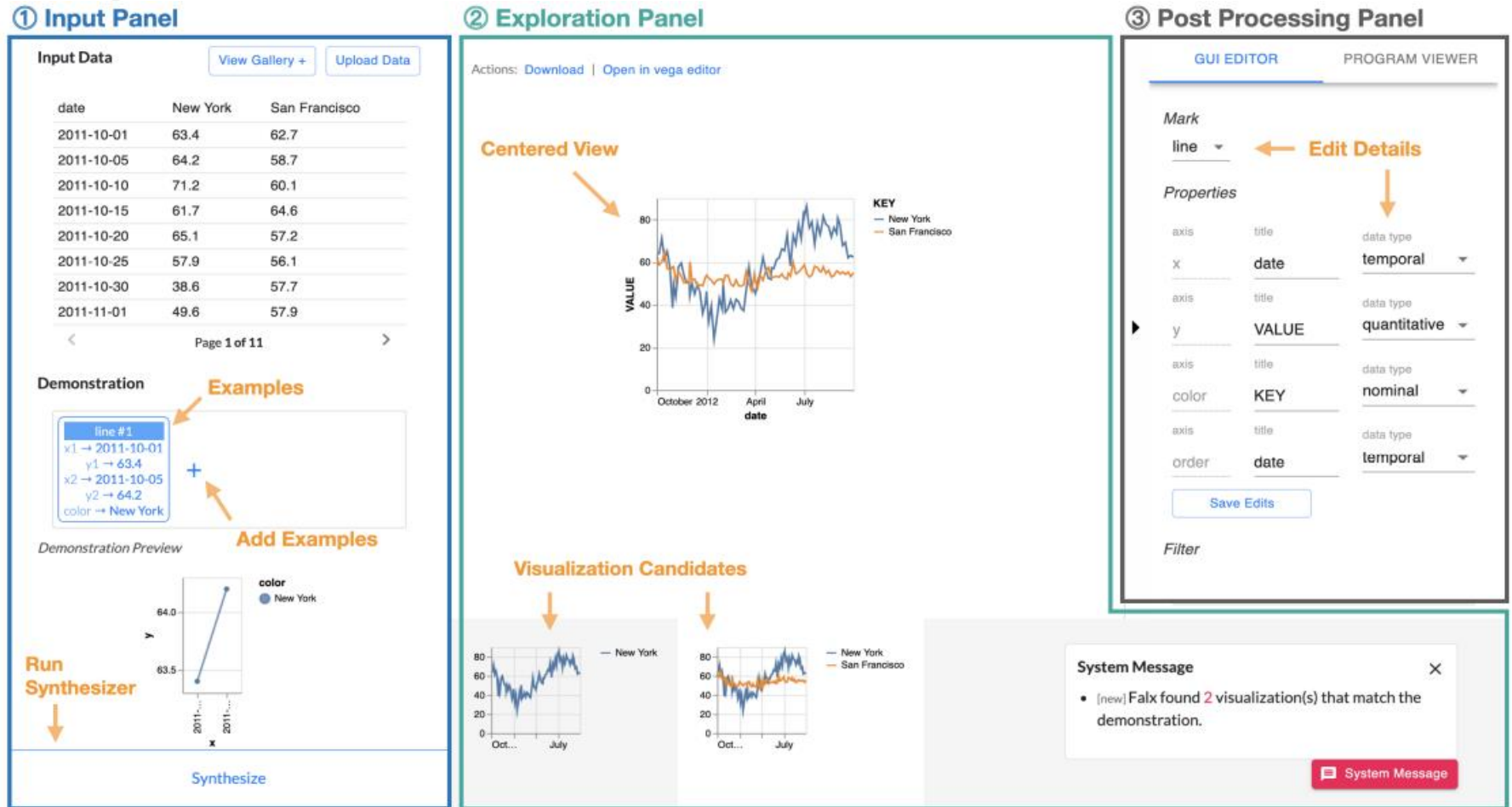


Figure 5: Falx interface has three panels: (1) Data analysts import data and create examples in the input panel. (2) Analysts explore and examine synthesized visualizations in the exploration panel. (3) Analysts edit visualization details in the post processing panel.

Usage Scenario

User Experience in Falx – Visualization by Example

User creates a Line object.

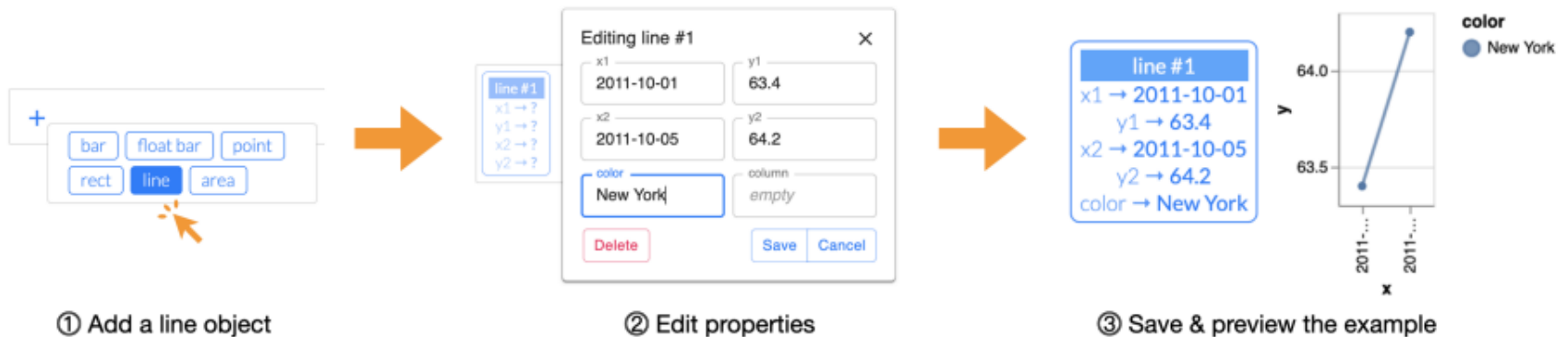


Figure 6: Amelia creates a line segment to demonstrate the visualization task.

Similarly, the user creates a bar object.

Usage Scenario

User Experience in Falx – Visualization by Example

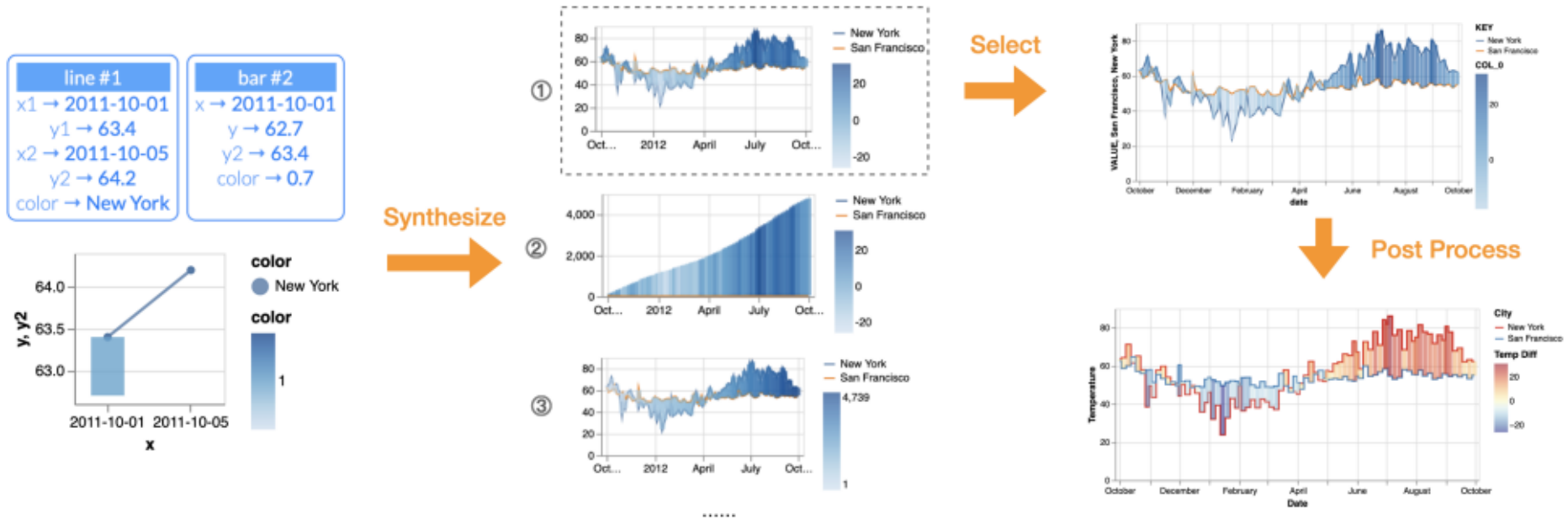


Figure 7: Amelia's interaction with Falx to create the second layer visualization.

System Architecture - Background

Program Synthesis –

- Programming-by-example (PBE) is a branch of program synthesis that aims to synthesize programs that satisfy input-output examples provided by the user
- **Common technique** - enumerative search using some cost metric
- **Cost metric can be** a statistical models that estimate likelihood of the program being correct or even models that measures simplicity of programs.

System Architecture

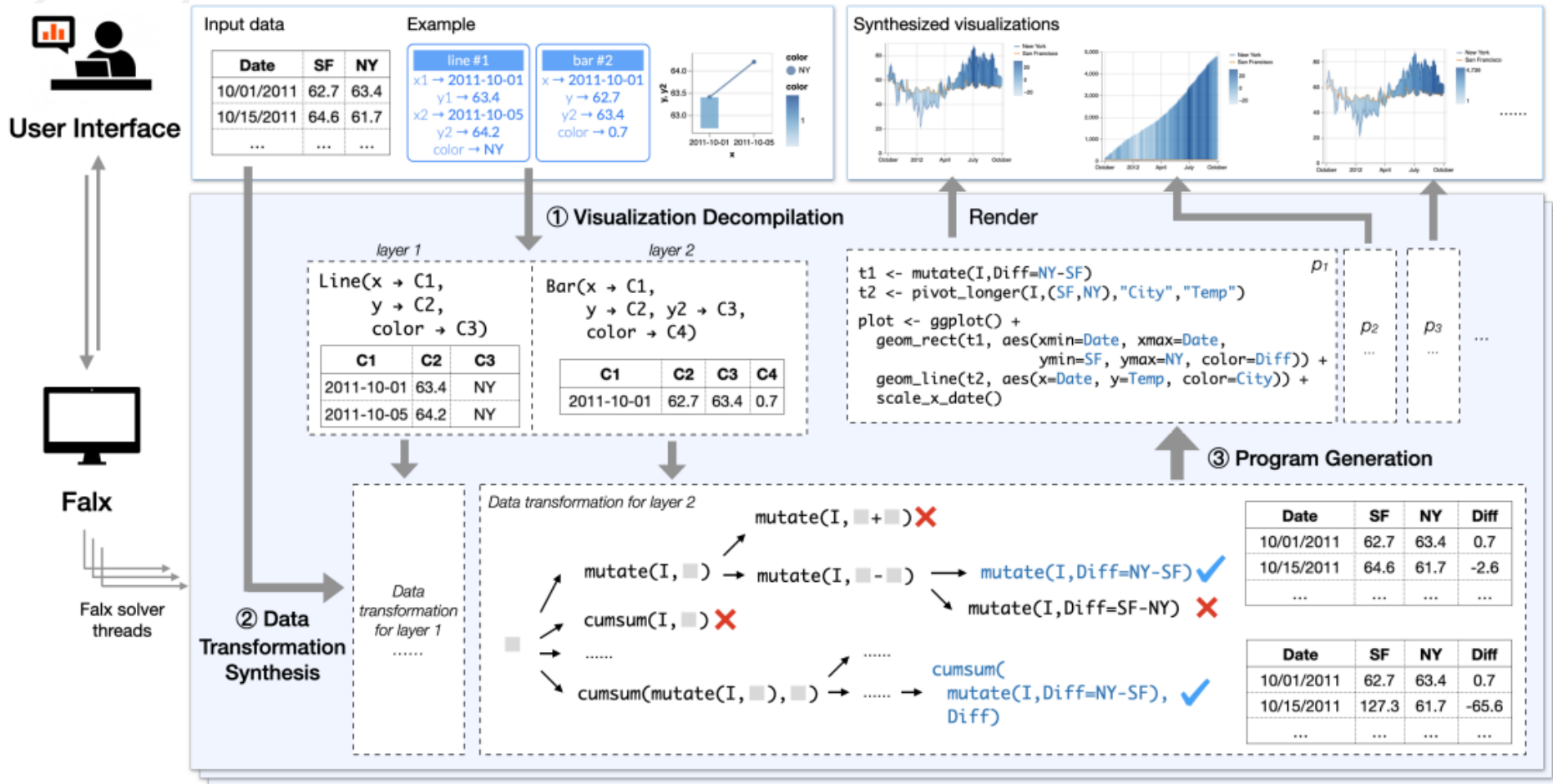


Figure 8: The architecture of the Falx system. Each solver thread synthesizes visualizations that match user examples in three steps: (1) visualization decompilation, (2) data transformation synthesis, and (3) program generation.

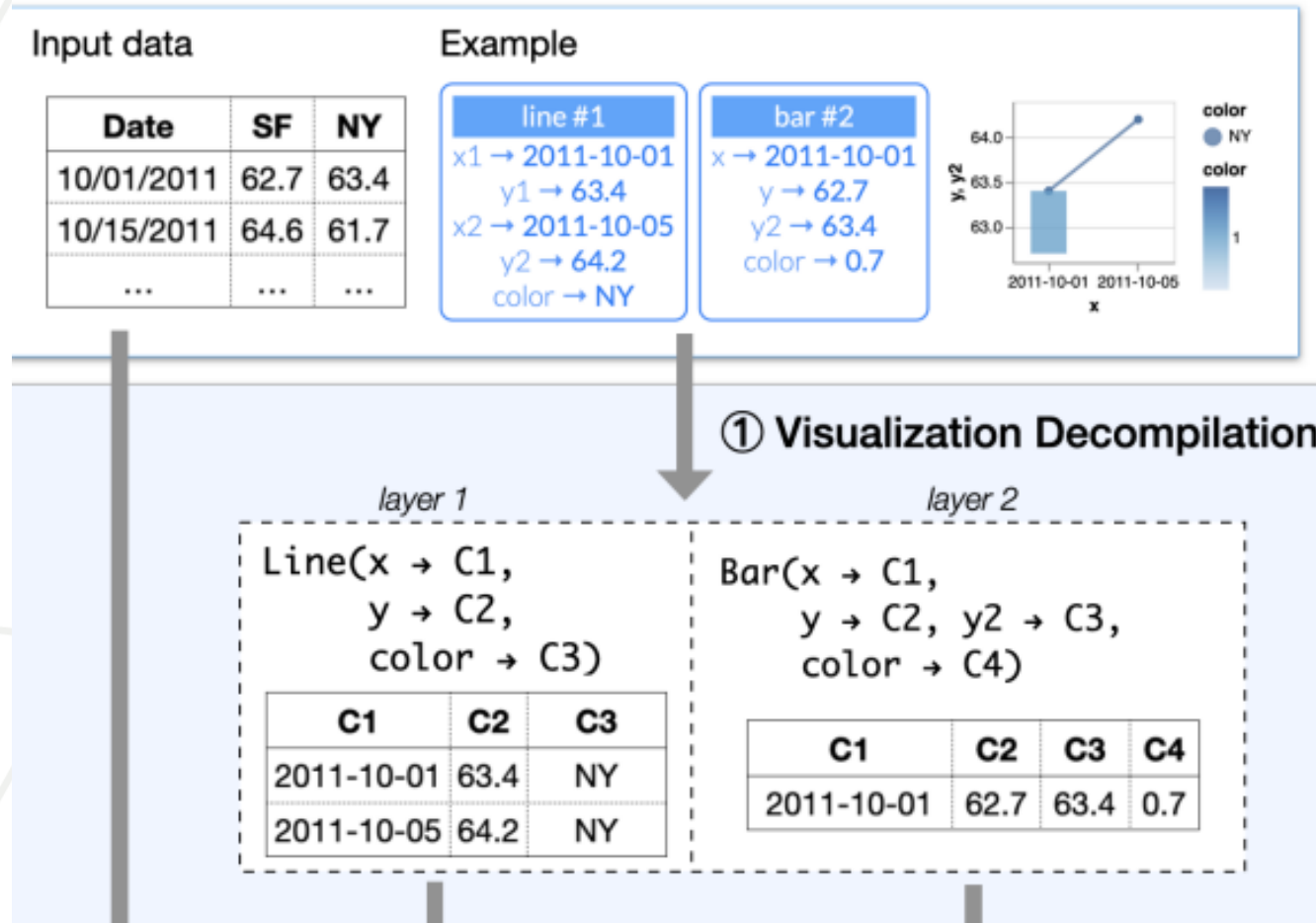
System Architecture

Steps 1 – Visualization Decomposition

- Infers visualization layers from the user example.
 - Partitions based on geometric type and property
 - Creates 1 visualization layer for each group
- Creates one basic visualization and an example table.
 - example table contains the same number of columns as the number of visual channels in this layer
 - visualization is specified as encodings(map columns to visual channels)
- Fills table with values for each example table.

System Architecture

- Steps 1 – Visualization Decomposition



System Architecture

Steps 2 - Data Transformation Synthesis

- For each example table, the synthesizer aims to synthesize a transformation program P_t , which can transform the input table into a table that contains the example table

$$T \subseteq P_t(T_{\text{in}}).$$

- Constructs sketches of transformation programs and iteratively expands search tree and fills arguments in these partial programs.
- Uses deduction pruning to prune infeasible partial programs for efficiency. (dramatically reduces search space)

System Architecture

Steps 2 - Data Transformation Synthesis

- Concrete programs encountered are added to candidate pool and are sent to the post processor.
- Terminates on search space exhaustively searched or search time budget reached.

Type	Operator	Description
Reshaping	pivot_longer	Pivot data from wide to long format
	pivot_wider	Pivot data from long to wide format
Filtering	select	Project the table on selected columns
	filter	Filter table rows with a predicate
Aggregation	group	Partition the table into groups based on values in selected columns
	summarise	For every group, aggregate values in a column with an aggregator
	cumsum	Calculate cumulative sum on a column for each group
Computation	mutate	Arithmetic computation on selected columns
	separate	String split on a column
	unite	Combine two string columns into one with string concatenation

Figure 9: Data transformation operators supported in Falx. For clarity, we omit the parameters of each operator.

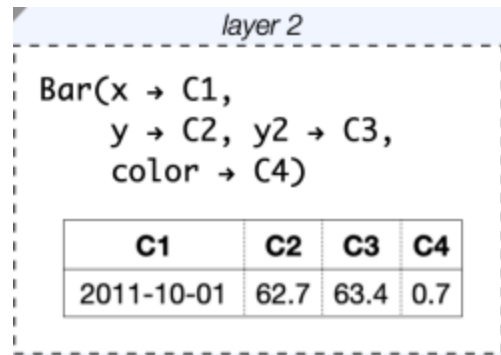
System Architecture

Steps 2 - Data Transformation Synthesis

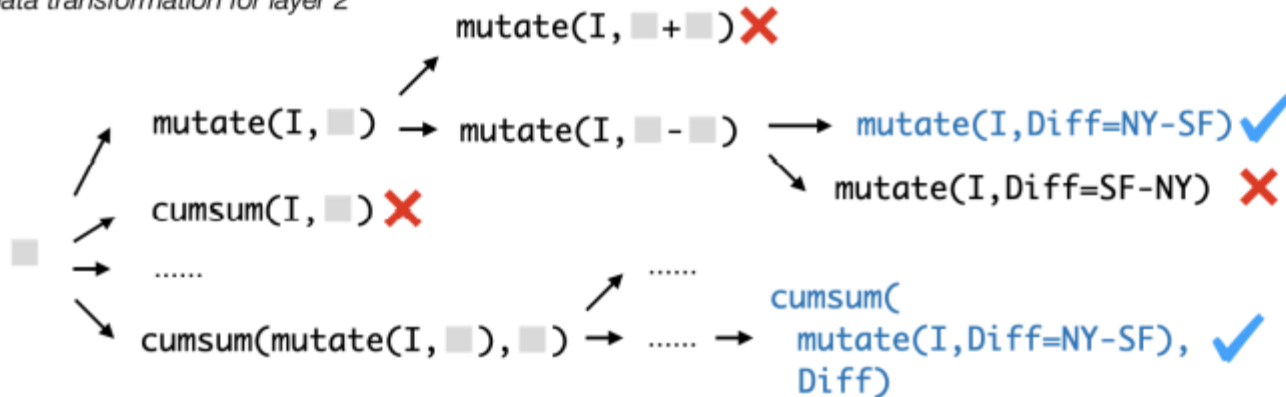
- Optimizations –
 - Falx memorizes abstract interpretation results for partial programs to allow reusing them whenever possible. (abstract interpretation programs run expensive operators like aggregation and pivoting on big tables.)
 - Uses different Falx solver threads with different starting program sketches to search different portions of the search space in parallel allowing a larger pool findings. Made use of timeout to enable responsiveness.

System Architecture

Steps 2 - Data Transformation Synthesis



Data transformation for layer 2



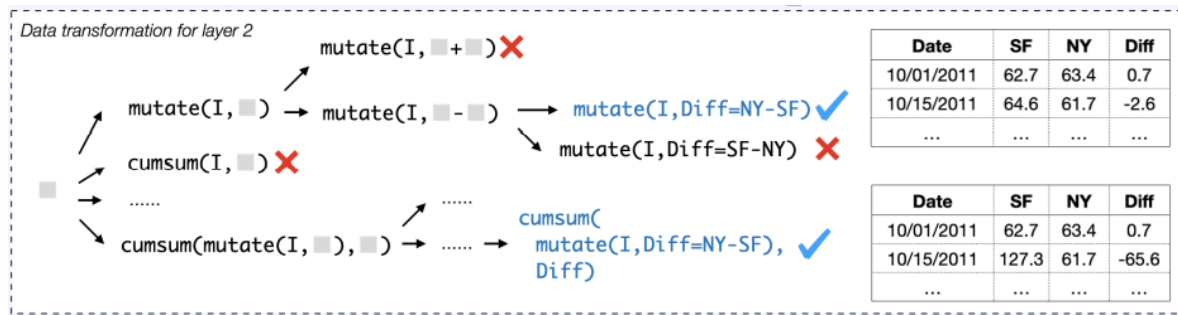
Date	SF	NY	Diff
10/01/2011	62.7	63.4	0.7
10/15/2011	64.6	61.7	-2.6
...

Date	SF	NY	Diff
10/01/2011	62.7	63.4	0.7
10/15/2011	127.3	61.7	-65.6
...

System Architecture

Steps 3 - Processing Synthesized Visualizations

- Generates visualizations by combining the visualization program generated in step 1 with table transformation programs generated in step 2.



System Architecture

Steps 3 - Processing Synthesized Visualizations

- For each data transformation program, Falx applies the table transformation program on the input data to obtain a transformed output and unifies the output table schema with the schema in the visualization program.
- **Instantiates previously** omitted visualization details(scale, domain, etc) and compiles the visualization program into Vega-Lite (or R) script through syntax-directed translation.
- Finally, Falx groups and ranks the visualizations based on the complexity of the programs (numbers of expressions).

Evaluation: User Study

Participants – 2 groups

Falx

16 participants (10 M, 5 F, 1 Unknown, Ages 23-51)

6 experienced(10+) , 8 moderate (1-10), 2 beginner (0)

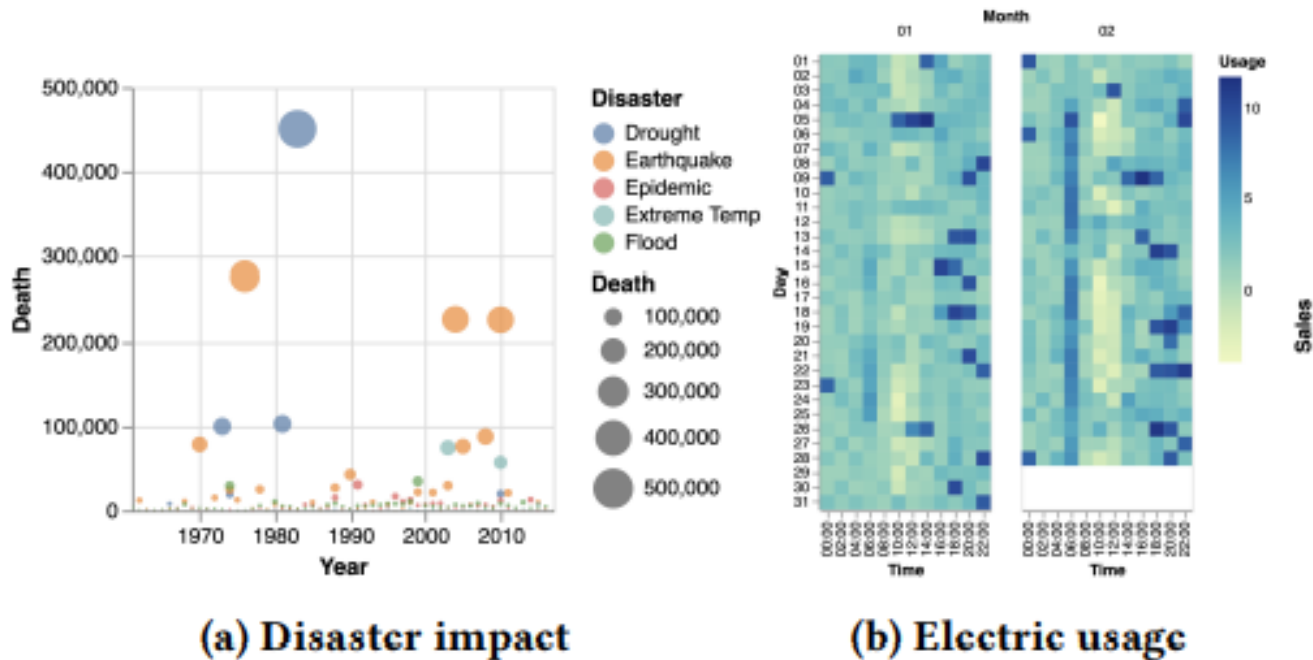
Baseline

17 participants(12 M, 4 F, Ages 19-60)

8 experienced(10+) ,9 moderate (1-10), knew R

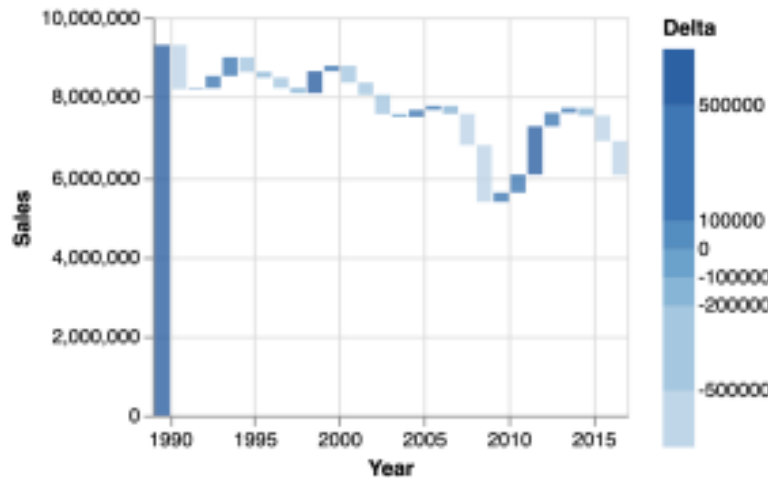
Evaluation: User Study

Conducted user study with 4 different scenarios –

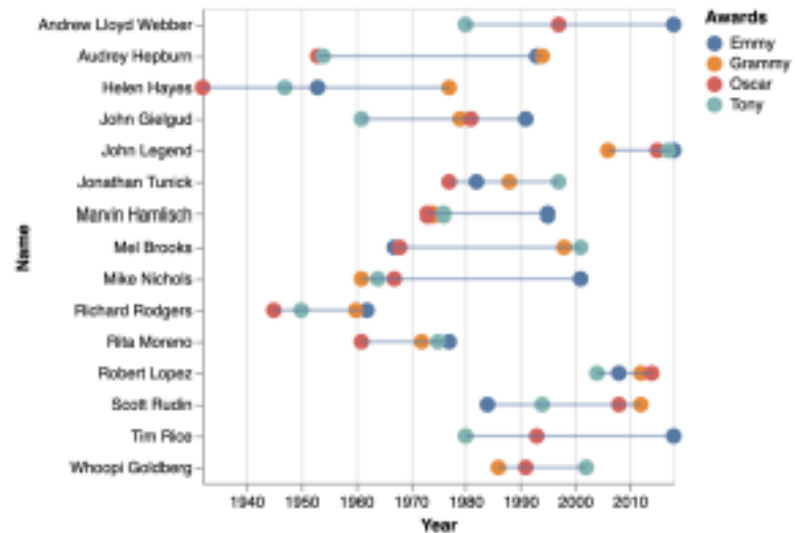


Evaluation: User Study

Conducted user study with 4 different scenarios –



(c) Car sales



(d) Movie awards winners

Evaluation: User Study

- We chose R as the baseline tool due to its popularity among data analysts and its ability to support both data transformations and visualizations.
- We provided as input a table that can be directly imported into the tools
- Explicitly described visualization designs to the participants in text the same context.
- Outcomes –
 - Correct
 - Wrong
 - Give up

Evaluation: User study results

Task	R (N = 17)		Falx (N = 16)	
	n	%	n	%
Disaster Impact	16	94.1%	14	87.5%
Electric Usage	13	75.6%	14	87.5%
Car Sales	5	29.4%	11	68.8%
Movie Awards	14	82.4%	16	100%

Figure 11: The number and percentage of participants correctly finished each study task.

statistically significant difference in the completion rate in the car sales visualization ($p < 0.05$)

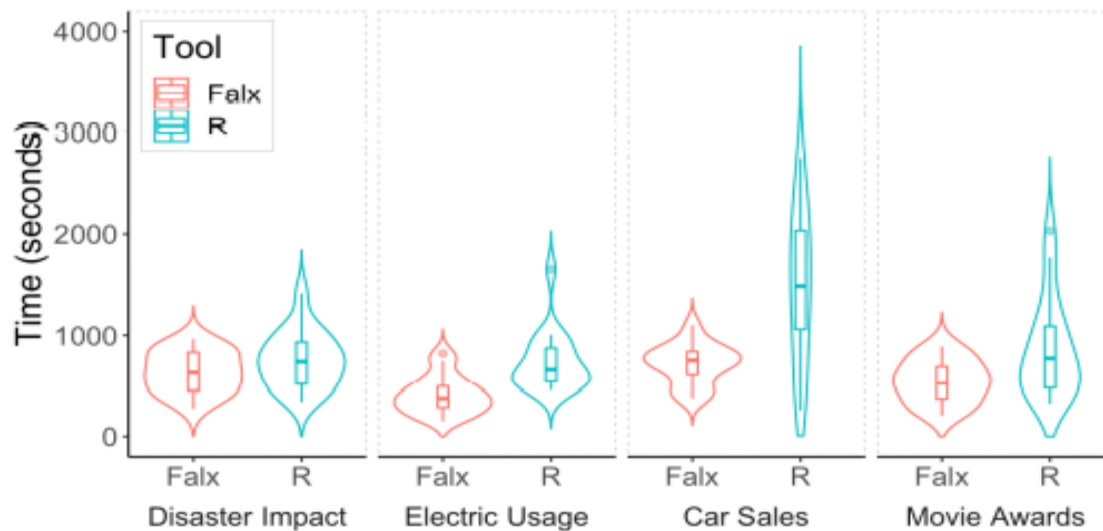


Figure 12: Violin plot showing the amount of time participants spent on each task for both Falx and R study groups.

Falx users performed better and more consistently

Evaluation: User study results

- Using Wilcoxon rank sum test with Holm's sequential Bonferroni procedure for p value correction significant improvement in user efficiency for car sales visualization and electric usage visualization
- While Falx participants were also generally faster in the other two tasks no significant difference.

Qualitative Interview Results

Task Experience -

- Finding the right visualization function
- Data transformation
- Learning to create expressive visualizations.

Workflow Implications –

- Create visualizations for discussions and presentations
- Prototyping complex analysis
- Reduce team collaboration effort

Limitations –

- “very high standard visualizations”
- “deep integration with other tools” – data cleaning

Conclusion

Falx users were able to effectively adopt Falx to solve visualization tasks that they could otherwise cannot solve, and in some cases, they do so more quickly.

Future Work

Visualization Learning – Help beginner analysts learn

Bootstrapping Complex Data Analysis - Falx could expose synthesized programs during the synthesis process and allow users to steer the synthesis process to better disambiguate results.

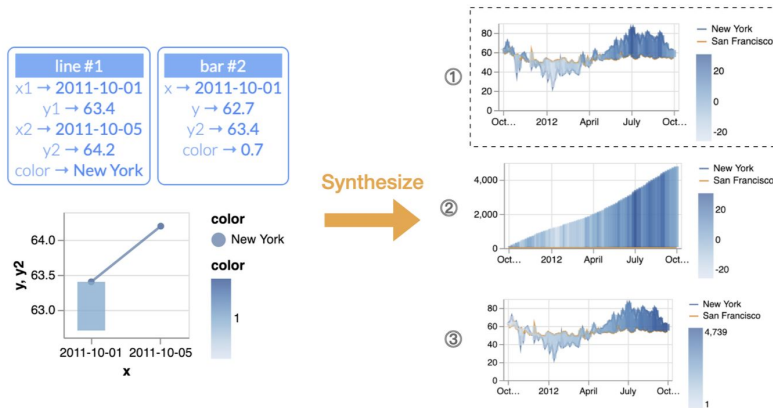
Falx: Synthesis-Powered Visualization Authoring

Wang et.al. (CHI 2021)

Reviewer role : Sankalp

Summarizing...

- Extremely common scenario: input data format — mismatch — intended visualization
- Expertise in data transformation tool/language needed to remove mismatch
- Falx : a tool that takes a small example visualization as input, and shows candidate visualizations
- Automates data transformation steps and visualization program creation steps



- Flow:
 - Example visualization -> visualization program + example table T
 - Start from input table T_{in} , find a sequence of operations P_t s.t. $P_t(T_{in})$ contains T
 - If candidate sequence found, combine it with visualization program and recommend to user

Strong Points

- Extremely well written paper and easy to follow. Section 2 (Usage Scenario) provides great introduction
- Use of common running example throughout paper -> few cognitive context switches needed
- **Relevance of problem:** I could immediately relate to it. Data transformation is not fun!
- **Reusability:** Even if input comes with different layout, process of demonstrating a visualization example remains same
- **Quick ramp up:** Multiple participants in user study mentioned that they learnt to use Falx quickly
- Approach of “X by example” seems to have established validity across domains:
 - Programming-by-example systems (PBE)
 - Query-by-example (QBE) systems for querying relational databases
- Seems to keep responsiveness in mind. Has used time budgets of 5-20 seconds.
- Power of technique well-illustrated. Converts construction problem to demonstration + verification problem (easier)
- User study well-designed to test the claim of “enables novice data analysts (beginners in R) to create visualizations fast”. Uses statistical significance tests while discussing results.

Opportunities for improvement

- After reading paper, no idea where approach fails
 - Paper talks of computational complexity blowup, but don't know when / at what point / in what scenario
 - A plot on “number of operators in data transformation” versus “time to generate sequence” would have helped
 - User study examples do not require too many transformations, can't glean information from this
- Data transformation synthesizer (step 2 in 3-step process) uses efficient algorithm to search for data transformation programs
 - Would have appreciated more examples on pruning. Only one concrete example given (prune cumsum operator path since leads to increase in # columns)
- Is the system customizable enough? Meaning, is the set of operators in tidyverse library to construct data transformation pipeline complete? Not sure, and not clarified in paper
- Power users of R (one of two groups in the user study) could have been asked to then use Falx and see their impressions on it

Overall evaluation?

Accept!

A well-written paper identifying a relevant problem and reusing techniques from multiple disparate fields to solve it.

Thank you!

**(also to Professor Rong and Kaushik for a well
conducted course! I enjoyed it :))**

The background is a dark blue gradient. On the left, there is a large, semi-transparent circular image of a circuit board. Overlaid on this and the background are several geometric shapes: a blue parallelogram and a light green parallelogram in the upper left, and a grey, 3D-like circuit board pattern in the upper right.

Falx: Synthesis-Powered Visualization Authoring

Archaeologist: Yiheng Mao



Short summary of the paper

The paper presents a synthesis-powered visualization tool called Falx, which enable users to create a visualization design and receive suggestions for the design using visual encoding examples without the need to manually specifying the visualization or spending significant effort on data transformation.

Empirical studies show users were able to effectively adopt Falx to solve visualization tasks that were very difficult, and in some cases, do so in greater efficiency.

Timeline of synthetic visualization studies

Dec 2019

Visualization by example



April 2020

Scout: Interface Layout
Exploration



June 2020

Query execution engines with
mutations



May 2022

Synthesizing analytical SQL
queries from computation
demonstration



May 2021

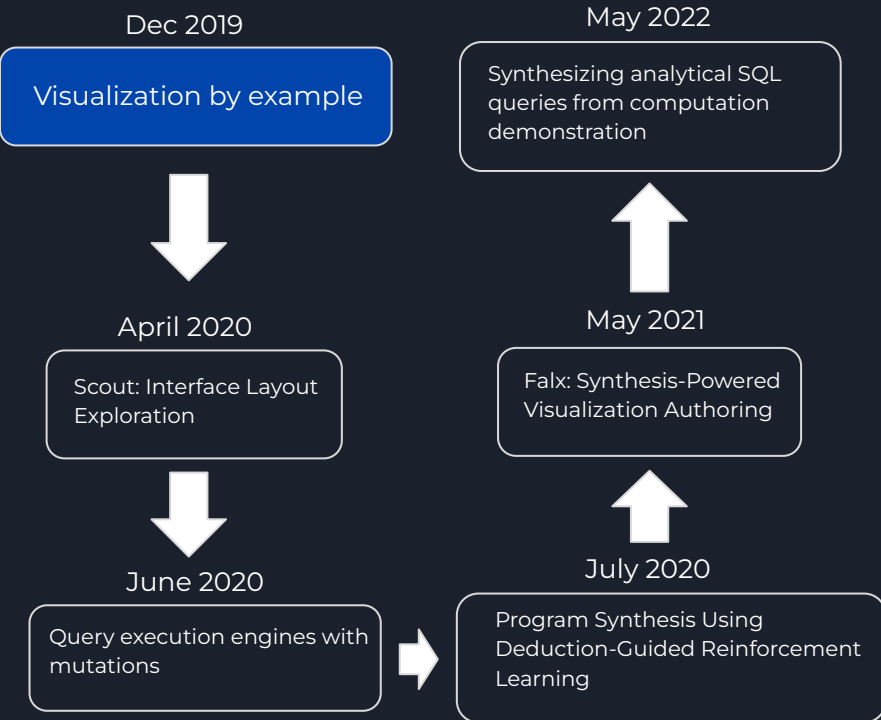
Falx: Synthesis-Powered
Visualization Authoring



July 2020

Program Synthesis Using
Deduction-Guided Reinforcement
Learning

Visualization by example

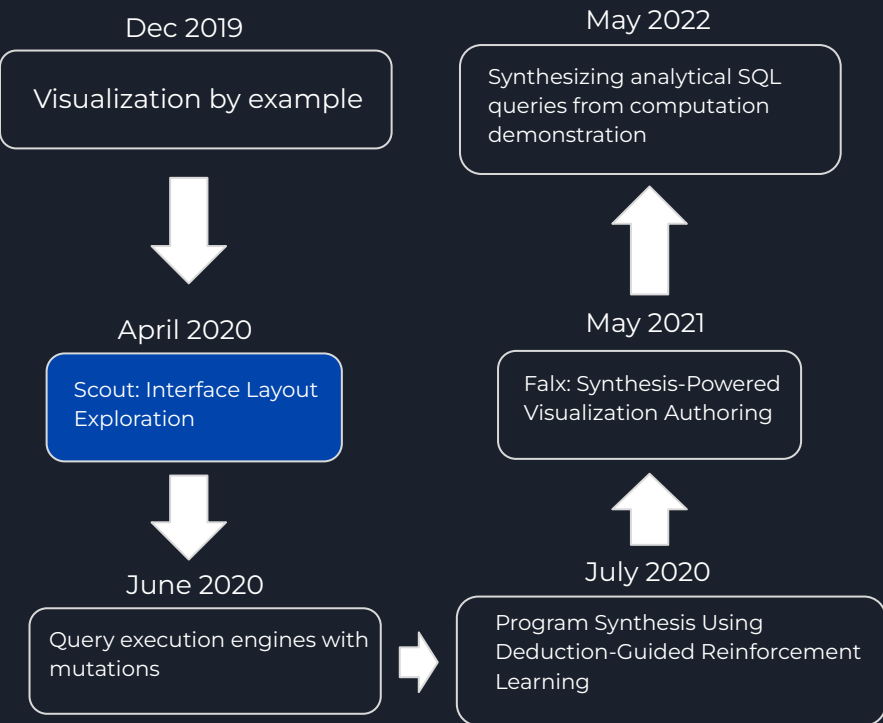


From a program synthesis perspective, automating visualization tasks poses two challenges: First, because many visualization tasks require data wrangling in addition to generating plots from a given table, we need to decompose the end-to-end synthesis task into two separate sub-problems. Second, because the intermediate specification that results from the decomposition is necessarily imprecise, this makes the data wrangling task particularly challenging.

This paper implements a visualization-by-example approach in a tool called Viser, that address these problems by developing a new compositional visualization-by-example technique that decomposes the end-to-end task into two different synthesis problems over different domain-specific languages and leverages bi-directional program analysis to deal with the complexity that arises from having an imprecise intermediate specification.



Scout: Interface Layout Exploration

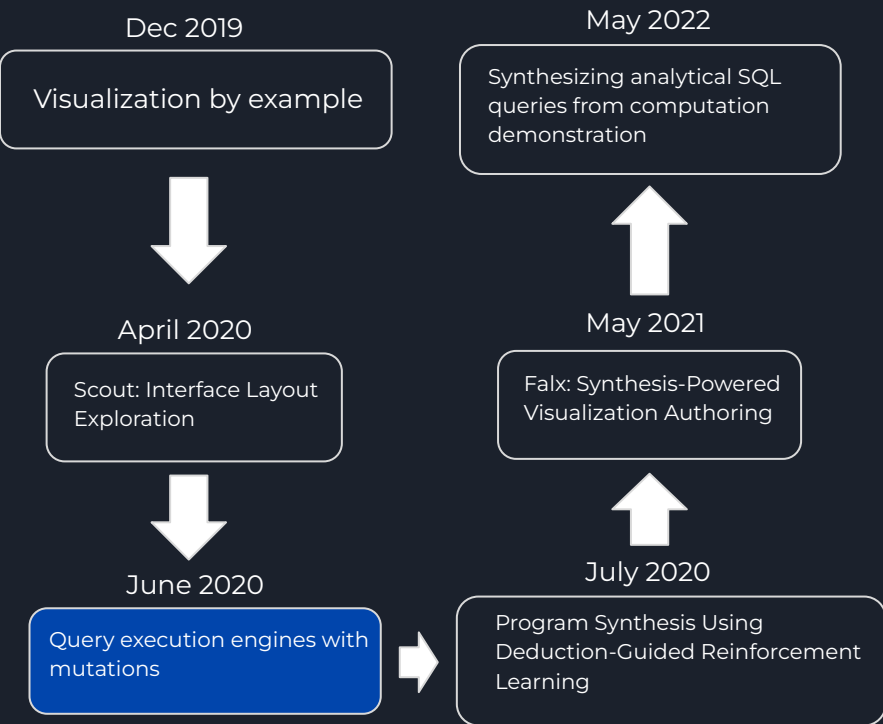


This paper presents Scout, a system that helps visualization designers to rapidly explore alternatives through mixed-initiative interaction with high-level constraints and design feedback.

Prior constraint-based layout systems use low-level spatial constraints and generally produce a single design. Scout introduces high-level constraints based on design concepts (e.g., semantic structure, emphasis, order) and formalizes them into low-level spatial constraints that a solver uses to generate potential layouts.



Testing query execution engines with mutations



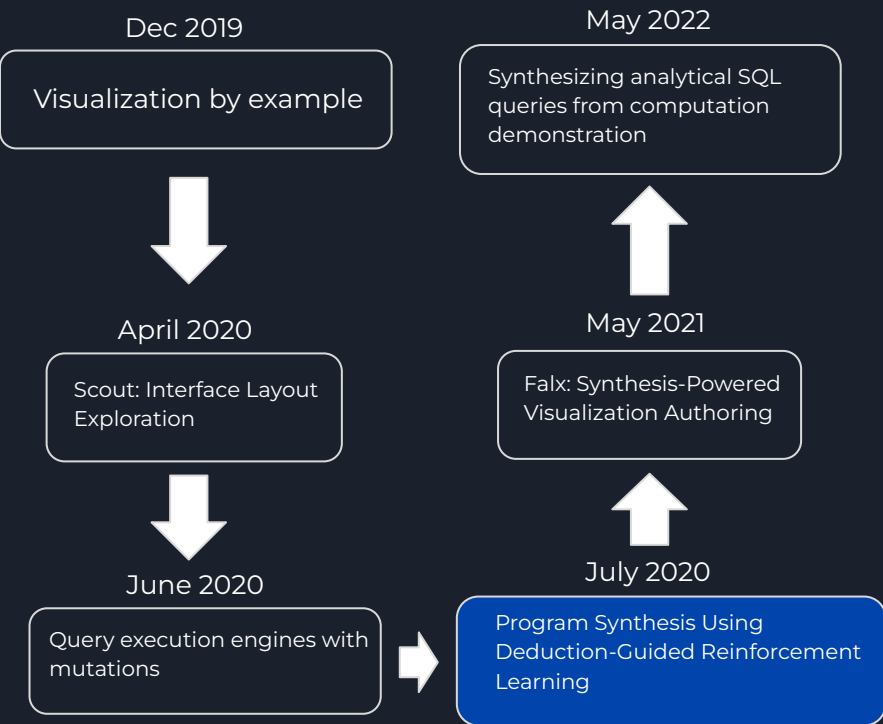
One challenge query optimizer engines face is that the high cost of testing query execution engines often prevents developers from making fast iteration during the development process, which can increase the development cycle or lead to production-level bugs.

To address this challenge, the authors propose a tool called MutaSQL, that can quickly discover correctness bugs in SQL execution engines.

MutaSQL generates test cases by mutating a query Q over database D into a query Q' that should evaluate to the same result as Q on D . MutaSQL then checks the execution results of Q' and Q on the tested engine.



Program Synthesis Using Deduction-Guided Reinforcement Learning

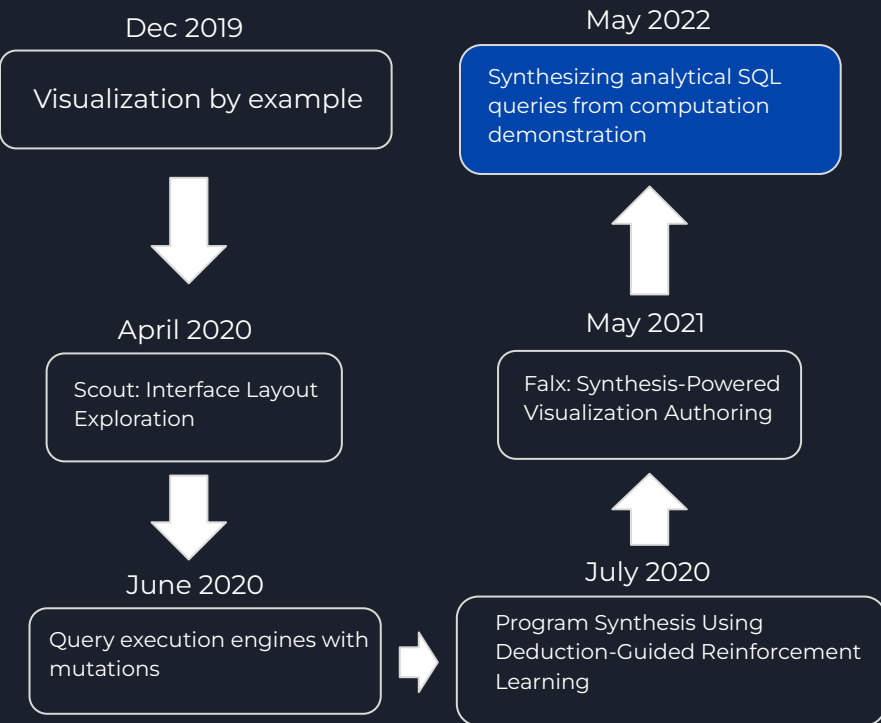


This paper presents a new program synthesis algorithm based on reinforcement learning. Given an initial policy (i.e. statistical model) trained off-line, the algorithm uses this policy to guide its search and gradually improves it by leveraging feedback obtained from a deductive reasoning engine.

Specifically, the authors formulate program synthesis as a reinforcement learning problem and propose a new variant of the policy gradient algorithm that can incorporate feedback from a deduction engine into the underlying statistical model.



Synthesizing analytical SQL queries from computation demonstration



The partitioning and grouping operators in analytical SQL could be challenging for novice users. Unfortunately, programming by example, shown effective on standard SQL, are less attractive because examples for analytical queries are more laborious to solve.

To make demonstrations easier to author, they designed a new end-user specification implemented as Sickie, that allows the user to demonstrate the task using a (possibly incomplete) cell-level computation trace, which, through a new abstraction-based synthesis algorithm, allows the system to prune the search tree.



Thank you!

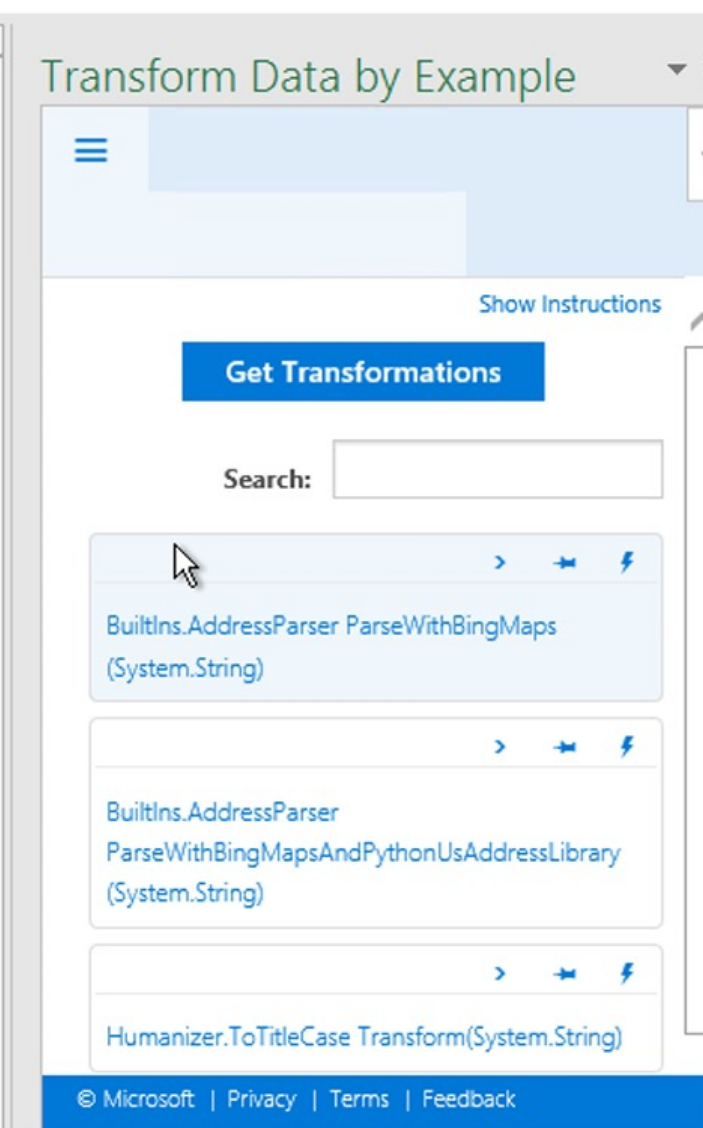
Any questions?

Additional reading

Transform-data-by-example (TDE): an extensible search engine for data transformations

- Support data transformation functions crawled from GitHub

C	D
Address	Output
4297 148th Avenue NE L105, Bellevue, WA 98007	Bellevue, WA, 98007
2720 N Mesa St, El Paso, 79902, USA	El Paso, TX, 79902
3524 W Shore Rd APT 1002, Warwick,02886	Warwick, RI, 02886
4740 N 132nd St, Omaha, 68164	Omaha, NE, 68164
10508 Prairie Ln, Oklahoma City	Oklahoma City, OK, 73162
525 1st St, Marysville, WA 95901	Marysville, CA, 95901
211 W Ridge Dr, Waukon,52172	Waukon, IA, 52172
1008 Whitlock Ave NW, Marietta, 30064	Marietta, GA, 30064
602 Highland Ave, Shinnston, 26431	Shinnston, WV, 26431
840 W Star St, Greenville, 27834	Greenville, NC, 27834



The data visualization landscape

Visualization Language

Vega-lite, SQL-like, Query-by-example, Keywords, Drag-and-drop

Efficient

Data Visualization

In-memory DB, Prefetching,
Approximate, Progressive

Smart

Data Visualization

Similarity, Deviation, Behavior,
Personalized, Perception

Data Storage

RDBMs, Hadoop, Cloud, Spreadsheets, Flat files

Discussion

What're your favorite papers in this section and what have you learned from the paper?

- Explanation: MacroBase, Slice Finder, Domino
- Recommendation: SeeDB, Lux, MCP
- Interfaces: Vega-lite, Qetch, Falx

Next class

What's happening next:

11/23 (Wed): No class (Thanksgiving)

11/28 (Mon): Additional OH during class time

11/30 (Wed): Draft paper (due 4PM) and peer review (in class)

Happy Thanksgiving!