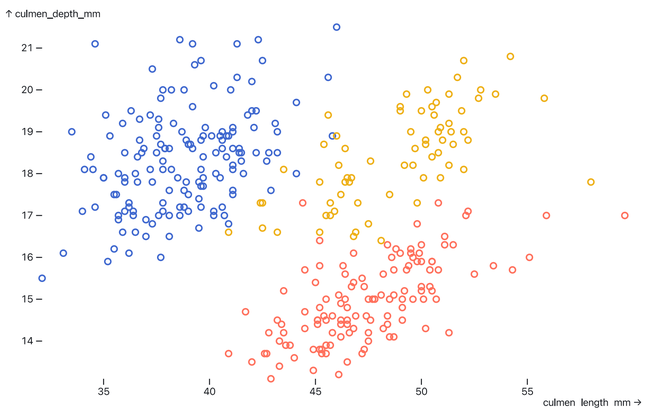
Extracting Visualization Workflows from Versioned Notebooks

Colin Brown, Hamed Alhoori, Maoyuan Sun, David Koop
Department of Computer Science, College of Liberal Arts & Sciences, Northern Illinois University

Motivation & Approach

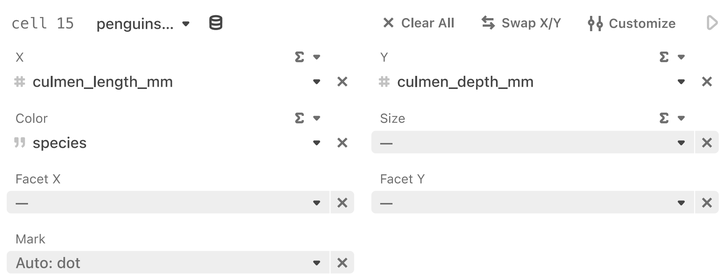
- Designing visualizations is an iterative process involving exploration of various visual encodings.
- We wanted to understand how users build on existing work, like templates, and explore different types of visualizations and encodings.
- We wanted to examine differences between users, frameworks, and settings.
- Exploratory notebooks and higher-level frameworks facilitate rapid iteration, allowing users to quickly test ideas and examine results.
- Changes in code-heavy visualization workflows can be difficult to understand and analyze.
- We decided to use publicly-available notebook version histories to observe how users build and refine visualizations over time.
- Observable [1] hosts over 100,000 publicly-available notebooks, many with visualizations, along with (partial) version histories.
- Our study compares the Observable Plot [2] and Vega-Lite [8] frameworks and the Observable Chart Cell wizard.
- The structure of Plot and Vega-Lite visualization code allowed us to build a truncated abstract syntax tree (AST) to analyze changes in Observable JavaScript (ojs) code cells, and we used this to summarize changes in visualizations.



Observable Plot

```
Plot.dot(penguins, {x: "culmen_length_mm",
                    y: "culmen_depth_mm",
                    stroke: "species"}).plot()
```

Observable Chart Cell



Vega-Lite

```
vl.markCircle({fill: null})
  .data(penguins)
  .encode(
    vl.x().fieldQ("culmen_length_mm").scale({'zero':false}),
    vl.y().fieldQ("culmen_depth_mm").scale({'zero':false}),
    vl.stroke().fieldN("species"),
  )
  .render()
```

Code/settings to produce similar plots in Observable Plot, Observable Chart Cell and Vega-Lite (shown plot uses Observable Plot).

Goals

- Understand the process of iterative design by examining user behavior in Observable Notebooks
- Understand how users reuse and adapt published notebooks for their own work
- Build programmatic methods to analyze large numbers of Observable Notebooks

Future Work

- Incorporate lower-level visualization frameworks like D3 into the analyses
- Examine the coupling between data wrangling tasks and visualization updates
- Classify different visualization changes based on the structures of underlying frameworks

Methods & Data

- We collected 173,211 publicly available notebooks from Observable over the past year.
- We identified Chart Cells via cell metadata and Observable Plot & Vega-Lite using static code analysis.

Library/Tool	# Notebooks	# Cells	# Cell-Versions
Observable Plot	22,751	83,005	752,041
Vega-Lite	10,046	53,777	319,783
Chart Cell	2,156	5,169	141,169

- We distinguished *meaningful forks* as those where users made changes after forking the notebook.
- We checked if users added new data to the notebook after forking.
- We computed how likely a user is to iterate on the same cell versus moving to another.
- We examined cloned cells by searching for the same code structure across the corpus.

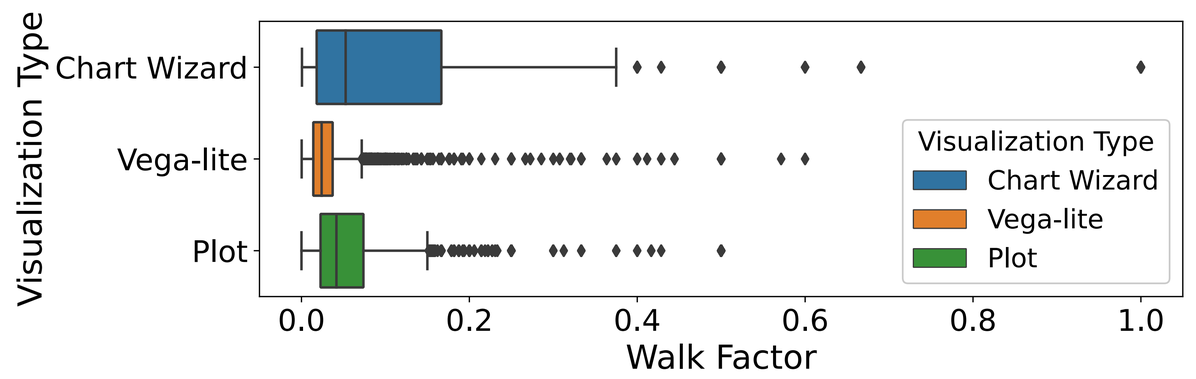
Results

Adapting Existing Visualizations

- Authors modifying forks added new files (data) to their fork only 51.7% of the time. This may indicate that users that find a promising example determine it does not fit their needs.
- We found 119,598 cells that were duplicated across Observable notebooks. The percentage relating to Plot (6.55%) and Vega-Lite (4.03%) were relatively low, but cells that contained fragments related to D3 (26.25%) comprised more than a quarter of the cloned cells.

Visualization Modifications

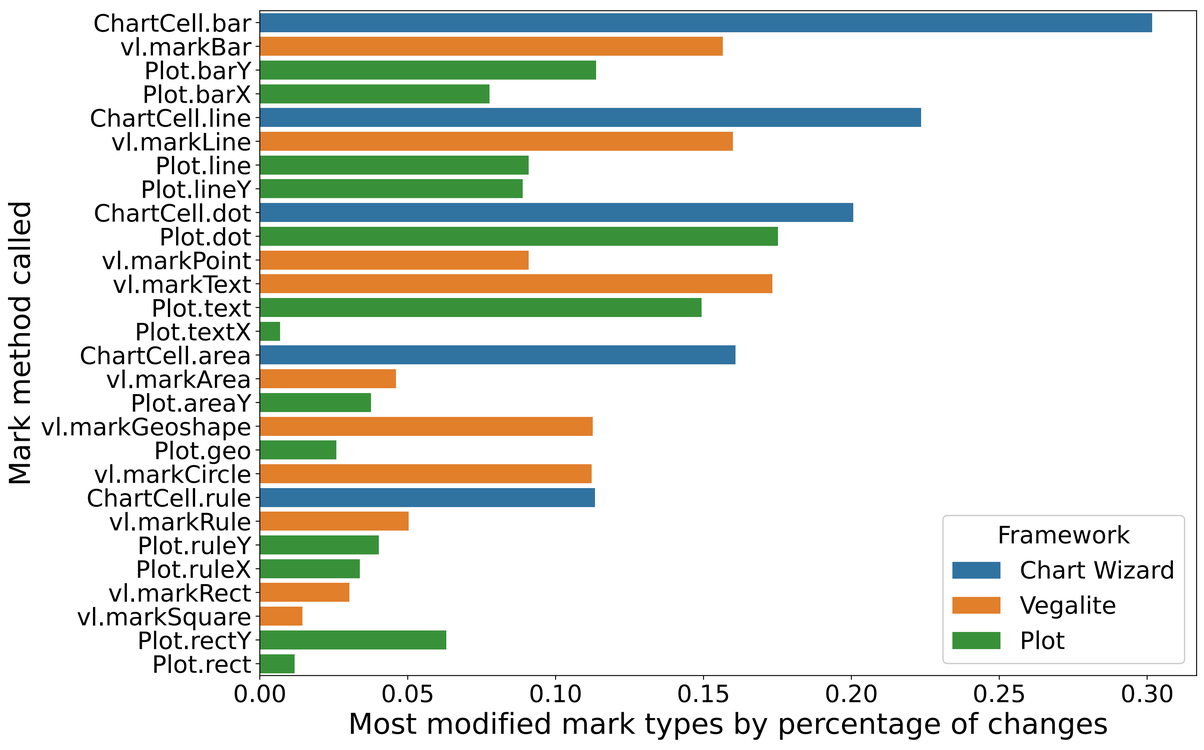
- We computed a *walk factor* that measures how often users stay in a cell (0) vs. move to another cell (1). A notebook's walk factor is computed as an average across all edits in the notebook. Notebooks with higher walk factors have users switching between cells more often than those with lower walk factors. We computed box plots for each visualization framework.



- Chart Cell wizard cells do not see many changes related to data values, and most options (color, fx, fy, mark, size, x, y) are rarely changed after being set.
- Vega-Lite cells have many properties; 95 of the 329 potential properties are changed on average more than 1.5 times while only 46 change are changed more than twice. Two frequently changed properties are axis and hover.

Framework Differences

- Mark types vary across the visualization frameworks, although there are some differences.
- Chart cells use bar marks most often; Plot has higher point mark use; and Vega-Lite leverages more line marks.



References

[1] Observable, 2024. observablehq.com [2] Observable Plot, 2024. observablehq.com/plot/ [3] H. K. Bako, A. Varma, A. Faboro, M. Haider, F. Nerrise, B. Kenah, J. P. Dickerson, and L. Battle. User-Driven Support for Visualization Prototyping in D3. In Intelligent User Interfaces, pp. 958–972. ACM, Sydney NSW Australia, Mar. 2023. doi: 10.1145/3581641.3584041 [4] M. Bostock, V. Ogievetsky, and J. Heer. D3 Data-Driven Documents. IEEE Trans. Visual. Comput. Graphics, 17(12):2301–2309, Dec. 2011. doi: 10.1109/TVCG.2011.185 [5] A. Head, E. L. Glassman, B. Hartmann, and M. A. Hearst. Interactive Extraction of Examples from Existing Code. In CHI, pp. 1–12. ACM, Montreal QC Canada, Apr. 2018. doi: 10.1145/3173574.3173659 [6] A. M. McNutt and R. Chugh. Integrated Visualization Editing via Parameterized Declarative Templates. In CHI, pp. 1–14. ACM, Yokohama Japan, May 2021. doi: 10.1145/3411764.3445356 [7] X. Pu and M. Kay. How Data Analysts Use a Visualization Grammar in Practice. In CHI, pp. 1–22. ACM, Hamburg Germany, Apr. 2023. doi:10.1145/3544548.3580837 [8] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-Lite: A Grammar of Interactive Graphics. IEEE Trans. Visual. Comput. Graphics, 23(1):341–350, Jan. 2017. doi: 10.1109/TVCG.2016.2599030