Dominik Moritz — Research Statement

While computers can help us manage data, human judgment and domain expertise is what turns it into understanding. Meeting the challenges of increasingly large and complex data requires methods that **richly integrate the capabilities of both people and machines**. In response to these challenges, my systems research combines methods from *visualization*, *data management*, *human-computer interaction*, and *programming languages* to enable effective methods for data analysis and communication.

My thesis contributes new languages and models for visualization design that power **interactive systems for scalable data exploration**. **Vega-Lite** [1] is a high-level declarative language for rapidly creating interactive visualizations, while also providing a convenient yet powerful representation for tools that generate visualizations [5,6]. **Draco** [2] is a model of visualization design that extends Vega-Lite with shareable design guidelines, formal reasoning over the design space, and visualization recommendation. **Falcon** [3] and **Pangloss** [4] contribute techniques for scalable interaction and exploration of large data volumes by making principled trade-offs among people's latency tolerance, precomputation, and the level of approximation. A recurring strategy across these projects is to leverage an understanding of people's tasks and capabilities to inform system design and optimization.

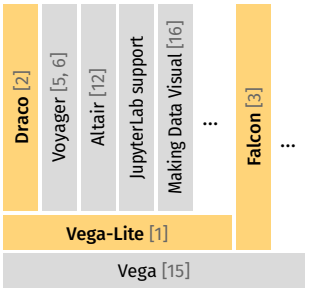

Figure 1: The systems I developed as part of my thesis (highlighted) are part of an ecosystem of systems that build on each other. Vega and Vega-Lite provide a platform for other applications and research projects. I contributed to all systems in this figure.

Vega-Lite, Draco, and Falcon are available as open-source projects with significant adoption. For example, Vega-Lite has over 100,000 downloads per month on NPM. My systems are part of an ecosystem of tools (Figure 1) used by the Python and JavaScript data science communities. By releasing my systems as open source and actively maintaining them, I not only make them available to analysts but also broaden the scope and impact of my research. My systems form the basis of many research papers, teaching curricula, and grant proposals, both in my group and across universities all over the world.

Vega-Lite: A Language for Interactive Visualization

The vast majority of visualizations are authored through end-user applications such as spreadsheets or business intelligence tools. Many of these tools lack consideration of perceptual principles or fall short of fully supporting an expressive range of graphics. To raise the abstraction level of visualization, I co-created Vega-Lite [1] as a **foundation for writing programs that generate visualizations**, for example in the Voyager visualization recommendation browser [5,6,7]. Vega-Lite is a declarative high-level format for representing and reasoning about interactive, multi-view visualizations. By deferring execution concerns to the runtime, designers can focus on design questions rather than implementation details. The declarative specification facilitates systematic enumeration of the design space, retargeting to different platforms, reuse, and automatic optimization of the execution.

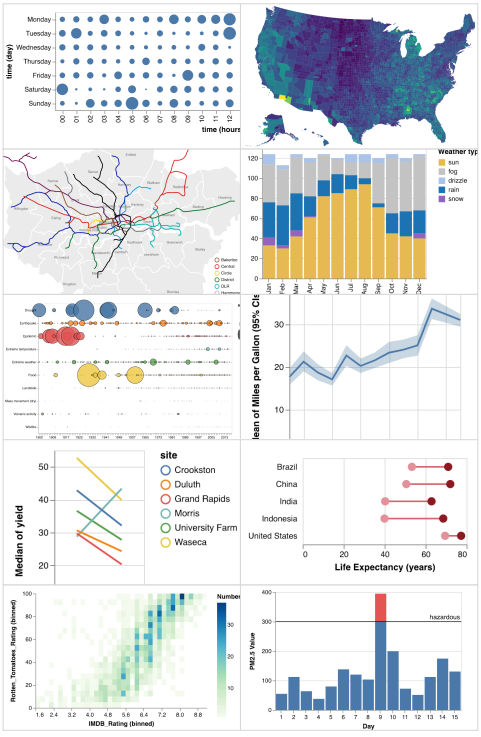

Figure 2: With Vega-Lite, analysts and designers can create an expressive range of charts including interactive multi-view visualizations.

Vega-Lite is designed for statistical graphics and—compared to the lower-level Vega [15] that it compiles to—trades off general expressivity for orders of magnitude shorter specifications. A chart is specified as a set of encodings that map data fields to properties (e.g., color or size) of graphical marks (e.g., points or bars). By combining these basic building blocks, users can create an expressive range of graphics (Figure 2). To keep specifications concise, users can omit low-level details such as axes and scales from their specifications. The gap between the high-level abstractions and the low-level execution leads to ambiguity. The Vega-Lite compiler resolves this ambiguity with carefully designed rules. Vega-Lite uses a declarative model for visual encoding, providing a balance of expressive power and usable, domain-specific constructs. This approach provides an abstraction where people can rapidly create visualizations in the midst of an analysis session and where the runtime system can automatically optimize how data is processed to render the chart.

In contrast to prior declarative visualization languages, Vega-Lite introduces a view algebra for combining basic plots into more complex multi-view displays, and a new selection abstraction for declarative specification of interaction techniques [1]. This novel grammar of interactions has enabled analysts to produce and modify interactive graphics

with the same ease with which they construct static plots. For example: before Vega-Lite, implementing the crossfiltering interaction shown in Figure 3 required a custom library with its own idiosyncratic API. Today, we can define the interactive chart in Vega-Lite with only 35 lines of JSON.

Vega-Lite has been widely adopted as a language to specify visualizations in the data science community; it is an official plotting format in JupyterLab and has been popularized via a Python wrapper called Altair [12]. The Altair team describes Vega-Lite as "perhaps the best existing candidate for a principled lingua franca of data visualization." Vega-Lite has been used in a book for practitioners [16], to teach visualization design (e.g., at UW, Stanford, CMU, Michigan), and in research projects both as a tool (e.g., for literate programming environments [17]) and as an implementation-independent language (e.g., for augmented/virtual reality [18]).

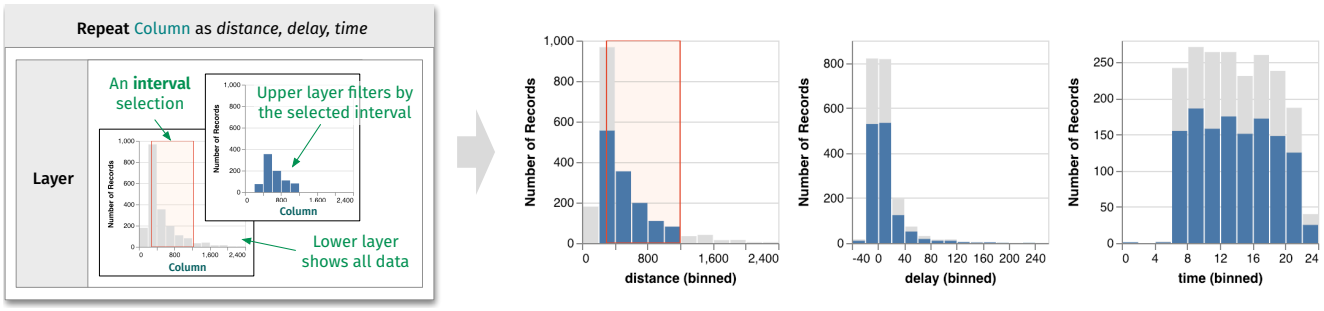

Figure 3: In Vega-Lite, Repeating a layered histogram with a selection produces coordinated histograms. The lower layer (grey) shows the unfiltered data while the upper layer (blue) shows the data in the selected range.

Modeling Visualization Design in Draco

To create effective visualizations, designers must consider the data domain and perceptual principles for design. Existing work has attempted to formalize this design knowledge as logical rules. Even though these rules are informed by perceptual research, they are engineered by hand, often only cover a few chart types, and are not reused or automatically validated. Moreover, the primary approaches used to apply these rules are now decades old, based on static models and greedy optimization methods. Rather than building idiosyncratic representations of design knowledge for individual systems, I seek to **make formal models of design knowledge a shared resource that can be extended, tested, and provide the base for future research**.

I am realizing this vision in Draco [2], a formal model that represents visualizations as sets of logical facts (i.e., specifying choices of dataset, mark type, and visual encoding channels) and expresses best practices and trade-offs among design guidelines as a collection of hard and soft constraints over these facts (e.g., one might prefer bars to start at zero). I base the visualization description language on the Vega-Lite grammar and extended it to express characteristics about the data (e.g., cardinality, skew) and task (e.g., summary, value). The constraints express preferences validated in perceptual experiments and general visualization design best practices. I also developed a machine learning approach based on learning-to-rank to derive the weights of soft constraints from experimental data.

As a visualization tool, Draco automates the tedious and repetitive parts of authoring visualizations. Draco can automatically synthesize effective designs from partial specifications as queries over the space of visualizations (Figure 4). I model the input query as additional constraints and use Draco to systematically enumerate the visualizations that do

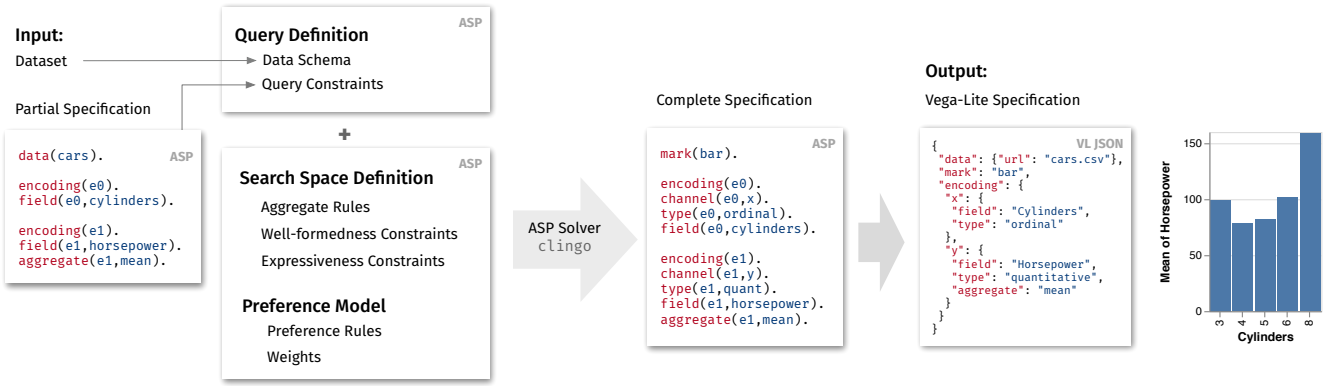

Figure 4: Draco compiles a user query to a set of rules and combines them with the existing knowledge base (search space definition + preferences) into an Answer Set Program (ASP). It then calls the Clingo solver to obtain an optimal solution and translates it to Vega-Lite.

not violate the hard constraints and find the most preferred visualizations according to the soft constraints. I formalize the problem of finding appropriate encodings as finding optimal completions of partial inputs, which provides well-defined semantics. A constraint solver with efficient domain-independent search algorithms replaces the otherwise necessary custom enumeration and scoring logic.

While Draco can synthesize visualizations in automated design tools, its applications go far beyond. Using constraints, we can **take theoretical design knowledge and express it in a concrete, extensible, and testable form**. At the recent IEEE VIS conference, Draco received the best paper award as it provides a platform for systematic discussions about visualization design. Draco formally describes trade-offs among design guidelines. Researchers can now experiment with different trade-offs to improve the science of visualization. They can systematically sample and enumerate the design space and concretely compare design models. Tool builders can use evolving knowledge bases and benefit from efficient search algorithms provided by modern constraint solvers. Using the implementation-independent language of constraints to model design knowledge could accelerate the transfer of research into practical tools.

Scalable Visual Data Analysis with Falcon and Pangloss

As the scale at which analysts need to work has outpaced the tools they use, we are challenged to create new tools that do not overwhelm people or their computational resources. Vega-Lite and Draco can help address the first issue. With Vega-Lite, we can describe interactive visualizations for large and complex datasets. With Draco, we can describe design guidelines for visualizations that guide analysts towards visual encodings that show important patterns and outliers regardless of the scale of the data (e.g., [13]). The challenge for the visual analysis system is to manage the amount of data and computation while remaining responsive.

The Vega-Lite runtime can leverage that specifications and execution are separate to partition the dataflow and push expensive computations into a scalable backend system [8]. While this approach supports static visualizations well, current data processing tools are often insufficient for interactive visual analysis. Delays in interactive visualization exploration systems break perceived correspondence between actions and response, reduce engagement, and lead to fewer observations made [19]. Poor support for interactive exploration has the potential to skew attention toward "convenient" and familiar datasets, along with the implied selection biases.

To support interactive analysis at scale, I developed **prefetching and indexing methods for low-latency interaction across linked multi-view visualizations**. I implemented these ideas in Falcon [3], a web-based visualization system (Figure 5). Falcon models and optimizes a user's session with client-side state rather than treating every query as an independent request. In a session, Falcon leverages the fact that users typically only interact with a single view at a time. When the user interacts with a view, Falcon creates an index that supports interactions with that view. Falcon can calculate the index from an aggregate query in a database system. As the user moves the brush in the view, Falcon computes the necessary data for all other views in constant time. When the user interacts with a different view, Falcon reindexes the data, which incurs a short delay. We accept this trade-off since users are more sensitive to latencies in brushing interactions than delays after switching views [19]. Falcon further trades-off initial accuracy of the visualizations for faster view switching. By applying these principled trade-offs, Falcon sustains real-time interactivity at 50fps for brushing and linking interactions among multiple visualizations of billion-record datasets—all while running the visualization in a standard browser. Falcon advances the state of the art in two ways. First, I exploit that not all interactions are equally latency sensitive to develop user-centered indexing strategies for a scalable interactive system. Second, unlike previous systems that required custom data structures, Falcon can use existing database systems to help create its index.

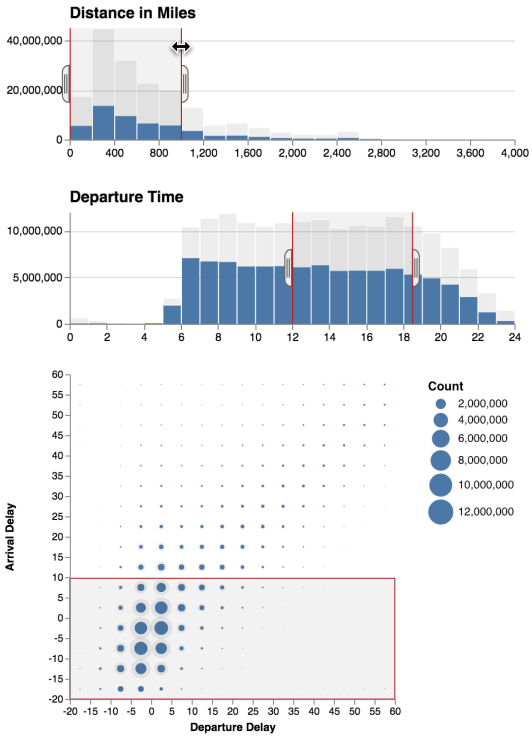

Figure 5: Visualizations in Falcon showing a dataset of 180 million flights in a web browser. The brushes select short (top) afternoon flights (middle) that have at most a 10-minute arrival delay (bottom). The views update in real-time when the user draws, moves, or resizes any brush.

Prefetching and indexing techniques, like the ones implemented in Falcon, work best when queries take a few seconds. However, for petabyte-scale datasets, even scanning a large dataset may take

minutes and inhibit interactive exploration. In this situation, users are given a choice: they can wait for the system to complete long-running queries or rely on an approximation based on a sample of the data. While attractive, approximate values can be—by their nature—incorrect. In exploratory visualization, an analyst might see dozens of visualizations; they are almost guaranteed to encounter a visualization where the errors are outside the predicted bounds. In my thesis, I address this issue with *optimistic visualization* [4]. In an optimistic visualization system, an analyst begins by constructing a **fast, approximate query**. Computation of the **precise results is placed in the background**, allowing analysts to continue their exploration without watching for updates. When the query is complete, the system invites the analyst to verify their observations (Figure 6). We implement this approach in Pangloss and discuss design implications [4, 10]. A laboratory study and three case studies at Microsoft showed that optimistic visualization can meet analysts' needs for both speed and precise results. Optimistic visualization gives people confidence in working with approximate results and paves the way towards broader adoption of approximate methods in exploratory analysts.

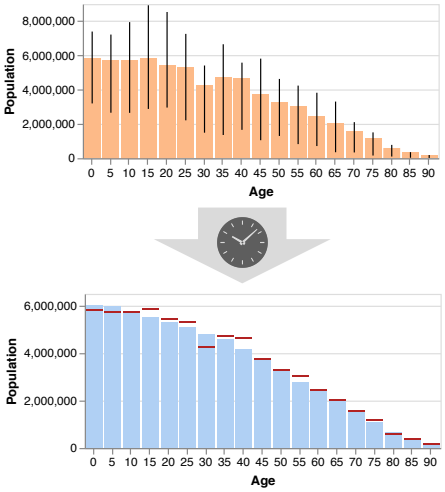

Figure 6: In optimistic visualization, users can immediately work with approximate results (top) while precise answers are computed in the background. When results arrive, they can check their observations (bottom).

Future Research

Analysts want to interact with their tools as if computers were infinitely fast but increasingly large data forces them to deal with idiosyncratic APIs or accept slowdowns. I envision a future where analysts can turn their data into understanding regardless of its structure or size. Systems should automatically ensure that interactive visual analysis interfaces are appropriate for the amount and distribution of the data. Fulfilling this vision requires innovations in different areas of computer science: visualization, data management, HCI, and PL. These traditionally separate concerns interact in complex ways. For example, an expressive visualization algebra that a database can optimize is considered a grand challenge of scalable visualization. To unify the various components (e.g., automated reasoning, full-stack optimization, and making approximation accessible) in a single system, we have to explore the design space and make principled trade-offs informed by the complementary strengths and weaknesses of computers and people. My future research will contribute systems that use automated reasoning over domain-specific representations of data analysis to inform how to efficiently run data science pipelines and enhance our ability to analyze and communicate data.

Automated Reasoning over Visualization Knowledge

While Draco currently supports automated reasoning for individual static charts, a large swath of the visualization design space remains uncharted. I am working on multiple projects that aim to extend Draco to interactive multi-view charts and integrate richer task models. To manage Draco's expansion in scope and complexity, I plan to develop interactive systems that enable quick adaptation of the knowledge base to an organization's needs. By integrating Draco into existing analysis tools like Altair [12], I can collect user actions to continuously improve Draco's suggestions.

As we increase automation and provide computational guidance, we also need to preserve a balance between automation and autonomy of the analyst. Only then can we get the benefits of scale and not lose the benefits of human expertise and intuition. I want people to stay in control. My systems should always explain their recommendations and let people override any decisions made by the machines. For example, a design recommender should warn designers against misleading plots (similar to a linter or spell checker) and propose alternatives with explanations to educate novices and improve visualization literacy.

Draco draws a new frontier of research that I plan to expand in the coming years. Studying how people use visualization models opens opportunities to make models more personal and adapt to specific domains. Draco's formal model of visualization design facilitates structured explorations of the design space of visualizations. Studies of human perception can not only inform the visualization model in Draco, but I am particularly excited about going the other way and using Draco's reasoning power to identify holes in our knowledge of expressive design. By formalizing the results of perceptual studies, we can assess whether they are in conflict with or subsumed by existing knowledge. Based on these results, active learning techniques can inform experiments that close gaps in our knowledge.

Full-Stack Optimization for Interactive Data Systems

Traditionally, the different stages of a data processing pipeline have been optimized as independent components, with optimizations focused on either the user interface or the data processing system. However, many optimizations—such

as prefetching and indexes per view in Falcon—are born out of a holistic consideration of front-end and back-end concerns together. I plan to develop and evaluate data science systems that integrate various optimization strategies such as indexing, prefetching, perceptually motivated cost models, and algorithms that optimally place data and computation on the client or the server [8]. By leveraging ambiguity and reasoning about declarative specifications of the data transformations and visual encodings such as Vega-Lite, systems could apply the optimizations automatically.

An essential aspect of this research is to integrate various optimizations into one system so that we can systematically evaluate design trade-offs. One of these trade-offs is to choose appropriate sampling strategies, data transformations, and visual encodings based on the analyst's goals. Data-intensive analysis routinely produces perceptually overwhelming visualizations and exceeds the capabilities of existing visual analytics tools (e.g., a scatterplot with a billion points). A recommendation system like Draco can suggest a scalable alternative (e.g., a heatmap) informed by details about the data that are typically not relevant to the analyst's goals (e.g., the data distribution can affect querying). The analyst can then interact with their data without having to worry about idiosyncrasies of the querying system.

To evaluate how new systems perform in practice, I will continue to seek out collaborations with data-driven domain scientists (e.g., [14]). I will use these interactions to gain insights into what tools, languages, and interactive systems are likely to have the greatest impact.

Living with Approximation and Uncertainty

As we continue to integrate more automation into analysis and process data in complex pipelines with many steps, the provenance of data becomes opaque to the analyst. Analysts often do not know how their data has been processed and how errors might have accumulated. As a result, the quality and uncertainty in data and derived models becomes unclear. I want to develop means to track the provenance of data and convey uncertainty in complex pipelines. For example, an interactive analysis system could automatically suggest appropriate uncertainty visualizations or identify potentially confounding factors. With improved means to handle uncertainty—in particular ways to limit the uncertainty about uncertainty—analysts can better assess data quality and may be satisfied with less data.

Even after analysis, data scientists need to accurately present the uncertainty in their models to decision makers. Journalists often credit effective communication of uncertainty of quantitative information as one of the biggest challenges they face. I also want to explore how insights from uncertainty visualization in analysis—like the ones I used during my Ph.D. [4, 10, 9]—can inform how we communicate uncertainty to the general public. I believe that by raising visualization and data literacy around uncertainty, we might help people become more informed and engaged citizens.

References

1. Vega-Lite: a Grammar of Interactive Graphics. *Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, Jeffrey Heer.* InfoVis 2016. **Best Paper Award**. `https://vega.github.io/vega-lite/`
2. Formalizing Visualization Design Knowledge as Constraints: Actionable and Extensible Models in Draco. *Dominik Moritz, Chenglong Wang, Greg L. Nelson, Halden Lin, Adam M. Smith, Bill Howe, Jeffrey Heer.* InfoVis 2018. **Best Paper Award**
3. Falcon: Balancing Interactive Latency and Resolution Sensitivity for Scalable Linked Visualizations, *Dominik Moritz, Bill Howe, Jeffrey Heer*. CHI 2019.
4. Trust, but Verify: Optimistic Visualizations of Approximate Queries for Exploring Big Data. *Dominik Moritz, Danyel Fisher, Bolin Ding, Chi Wang*. CHI 2017.
5. Voyager: Exploratory Analysis via Faceted Browsing of Visualization Recommendations. *Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, Jeffrey Heer*. InfoVis 2015. **Invited to SIGGRAPH 2016 as 1 of 4 top TVCG papers**.
6. Voyager 2: Augmenting Visual Analysis with Partial View Specifications. *Kanit Wongsuphasawat, Zening Qu, Dominik Moritz, Riley Chang, Felix Ouk, Anushka Anand, Jock Mackinlay, Bill Howe, Jeffrey Heer*. ACM SIGCHI 2017.
7. Towards A General-Purpose Query Language for Visualization Recommendation. *Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, Jeffrey Heer*. HILDA at SIGMOD 2016.
8. Dynamic Client-Server Optimization for Scalable Interactive Visualization on the Web. *Dominik Moritz, Jeffrey Heer, and Bill Howe*. DSIA at VIS 2015.
9. Value-Suppressing Uncertainty Palettes. *Michael Correll, Dominik Moritz, Jeffrey Heer*. CHI 2018.
10. What Users Don't Expect about Exploratory Data Analysis on AQP Systems. *Dominik Moritz, Danyel Fisher*. HILDA at SIGMOD 2017.
11. Lessons from Pangloss: User Encounters with Uncertainty. *Dominik Moritz, Danyel Fisher*. Uncertainty workshop at CHI 2017.
12. Altair: Interactive Statistical Visualizations for Python. *Jacob VanderPlas, Brian E. Granger, Jeffrey Heer, Dominik Moritz, Kanit Wongsuphasawat, et. al.* JOSS 2018
13. Visualizing a Million Time Series with the Density Line Chart. *Dominik Moritz, Danyel Fisher*. arXiv 2018.
14. Exploring neighborhoods in large metagenome assembly graphs reveals hidden sequence diversity. *C. Titus Brown, Dominik Moritz, Michael O'Brien, Felix Reidl, Taylor Reiter, Blair Sullivan.* bioRxiv 2018.
15. Reactive Vega. *Arvind Satyanarayan, Ryan Russell, Jane Hoffwell, Jeffrey Heer.* InfoVis 2015.
16. Making Data Visual. Danyel Fisher, Miriah Meyer. O'Reilly Press 2018.
17. Design Exposition with Literate Visualization. *Jo Wood, Alexander Kachkaev and Jason Dykes.* Infovis 2018. Best Paper Honourable Mention
18. A Toolkit for Building Immersive Data Visualizations. *Ronell Sicat, et.al.* Infovis 2018.
19. The Effects of Interactive Latency on Exploratory Visual Analysis. *Zhicheng Liu, Jeffrey Heer.* Infovis 2014