SuperProvenanceWidgets: Tracking and Visualizing Analytic Provenance Across UI Control Elements

Antariksh Verma
The Hong Kong University of Science and Technology
Hong Kong, China
averma@connect.ust.hk

Kaustubh Odak
Amazon Web Services
Seattle, Washington, USA
kaustubhodak1@gmail.com

Arpit Narechania
The Hong Kong University of Science and Technology
Hong Kong, China
arpit@ust.hk

Figure 1: SuperProvenanceWidgets: a JavaScript library comprising UI controls and a *SuperWidget* showing an aggregated overview as well as a detailed temporal history of interactions both within and across UI controls.

CHI EA '26, Barcelona, Spain

ACM ISBN 979-8-4007-2281-3/26/04
https://doi.org/10.1145/3772363.3798409

Abstract

ProvenanceWidgets is an existing JavaScript library that tracks the recency and frequency of user interactions with individual UI controls (e.g., range sliders and dropdowns) and dynamically overlays this provenance onto them. In this work, we introduce **SuperProvenanceWidgets**, an extension to ProvenanceWidgets featuring a

new SuperWidget that similarly tracks and visualizes provenance but across multiple UI controls, enabling users to understand how, when, and whether different UI controls were used. Through three example usage scenarios, we demonstrate how this cross-control SuperWidget helps (a) audit and share analysis workflows, (b) surface and mitigate exploration biases, and (c) facilitate user interface design and personalization. We also perform a technical self-assessment using the Cognitive Dimensions of Notations to evaluate the library's usability for developers. SuperProvenanceWidgets is integrated into the ProvenanceWidgets library and is available as open-source software at **ProvenanceWidgets.github.io**, empowering developers to build advanced provenance applications.

CCS Concepts

• **Human-centered computing** → **Visualization toolkits**; **User interface toolkits**.

Keywords

Analytic Provenance, User Interface Controls, Visualization, Library

ACM Reference Format:

Antariksh Verma, Kaustubh Odak, and Arpit Narechania. 2026. SuperProvenanceWidgets: Tracking and Visualizing Analytic Provenance Across UI Control Elements. In *Extended Abstracts of the 2026 CHI Conference on Human Factors in Computing Systems (CHI EA '26), April 13–17, 2026, Barcelona, Spain.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3772363.3798409

1 Introduction and Background

Analytic provenance is the documented history of how users engage with and manipulate data visualizations [33, 35, 36]. Documenting this history is important because our memory has a limited capacity to remember and track our prior interactions with data. These limitations occur both in quantity and over time as memories decay [25, 27]. This creates a barrier to effective data exploration.

Consequently, provenance has been used to address this barrier in multiple ways. For instance, it supports sensemaking [32] and decision-making [26] with data. It helps evaluate visualization systems [7, 21], design adaptive systems [41], improve machine learning model performance [18], replay and replicate analytic workflows [4, 38], and generate summary reports [10, 23].

Provenance has also been shown to drive deeper analytic insights. For instance, it facilitates unique data discoveries [19, 44], improves user confidence [6] and inspiration [15] levels, and increases contextual awareness [40] and recall [15] of previously visited data. In some cases, it even causes surprise [29].

In terms of tooling, existing tools support provenance tracking across diverse domains, including data analysis platforms [33], code editors [43], computational notebooks [16, 20], workflow systems [4, 13], collaborative environments [2, 17, 37], websites [22], UI controls [30], and even video games [14, 24].

Most relevant to our work are software libraries that enable developers to build custom provenance-aware tools with cross-platform consistency [1, 3, 11, 30, 44]. For example, Trrack [11] is a JavaScript library for creating and tracking interaction histories in web applications, supporting action recovery, reproducibility, collaboration, and logging. ProvenanceWidgets [30] is another JavaScript library that tracks and dynamically overlays analytic provenance, in the form of the recency and frequency of user interactions, onto UI controls such as sliders and dropdowns. However, ProvenanceWidgets *independently* visualizes provenance on *individual UI controls*; there is no way to analyze the aggregated summary or the detailed temporal sequence and duration of interactions *across multiple UI controls*. For instance, when a user combines an input field and range slider with a checkbox group to identify trends across variables, developers using ProvenanceWidgets or Trrack must manually extract logs from each control and analyze them separately. This workflow is labor-intensive and redundant. So we ask: *How can we (better) track and visualize provenance* across *different UI controls?*

In response, we present **SuperProvenanceWidgets**, an extension to ProvenanceWidgets that tracks and visualizes provenance *across multiple UI controls*. This enables users to understand not just *what* controls were used, but *how*, *when*, *for how long*, and *in what sequence*—revealing the temporal relationships between interactions that individual control tracking obscures.

We achieve this through two key advancements to the ProvenanceWidgets framework. First, we introduce a new modular architecture based on the *Separation of Concerns* [12]. We restructured ProvenanceWidgets by de-coupling it into independent modules—separating provenance capture, visualization, and cross-control aggregation. This gives developers fine-grained control to customize and extend the library for diverse use cases. Second, we introduce *SuperWidget*, a novel multi-widget control that aggregates and visualizes *super-provenance* across controls. It reveals temporal sequences, interaction durations, and usage patterns that show how users orchestrate multiple controls during analysis.

We demonstrate the impact of these advancements through three usage scenarios: (a) auditing and sharing complex analysis workflows [8, 11], (b) surfacing and mitigating human [42] and exploration biases [28, 34], and (c) enabling user interface design [39] and personalization [41]. We also evaluate the library's developer usability through the Cognitive Dimensions of Notations [5]. SuperProvenanceWidgets is integrated into ProvenanceWidgets and is available as open-source software at **ProvenanceWidgets.github.io**, empowering developers to build advanced provenance applications.

2 SuperProvenanceWidgets

2.1 Design Goals

Our design of SuperProvenanceWidgets builds upon the goals of the original ProvenanceWidgets library, and is driven by five goals.

- G1 **Cross-widget provenance tracking.** The library should keep track of provenance within as well as across UI controls.
- G2 **Real-time visual overlays.** The library should visualize aggregated summaries as well as temporal histories of users' analytic provenance within and across multiple UI controls, in real time.
- G3 **Control selection.** The library should enable users to intuitively identify and navigate to individual UI controls.
- G4 **Action recovery.** The library should enable users to revert to previous states of the UI controls.
- G5 **Modular design.** The library should be composed of independent, composable units to enable developers to customize, extend, and integrate only the components they need.

2.2 Design Process

To achieve our design goals, we systematically explored visualization strategies for presenting cross-widget provenance (**G1**). We considered both **ex-situ**–wherein cross-provenance is shown in a separate widget–and **in-situ**–wherein cross-provenance is shown on each UI control directly–approaches. All this, while balancing navigation support (G3) with minimal interference during analysis.

2.2.1 Ex-situ Visualization. In ex-situ approaches, provenance data is displayed in a separate widget that users can access as needed. This approach also simplifies navigation to specific UI controls (**G3**). We prototyped two designs: *Sankey diagram* and *Gantt chart*.

Sankey diagram. Figure 2a shows this design where the height of each band corresponds to number of interactions, highlighting the *nestedness* of the flow. Sankey diagrams excel at revealing high-level patterns like which control combinations users favor most and how interactions branch across widgets. However, they obscure precise timing and make it difficult to recover exact UI states for replay.

Gantt chart. Figure 2b shows this design where the position of each bar illustrates the sequence of all interactions across widgets. Gantt charts clearly encode interaction timing, duration, and precise ordering, enabling action recovery by clicking any bar to restore that UI state. Their weakness is visual clutter when interaction volume is high, though this can be mitigated via filtering and zooming.

2.2.2 In-situ Visualization. In in-situ approaches, we overlay provenance directly onto UI controls for immediate context. We explored two designs (Figure 3): *widget background* and *interaction badges*.

Widget background. In this design, control backgrounds are shaded darker or lighter based on recency of last interaction. This provides instant "at-a-glance" feedback about which controls were most recently used without requiring extra screen space. However, it conveys no information about interaction frequency, duration, or sequence across multiple controls.

Interaction badges. In this design, numeric badges beside controls display interaction counts. This reveals usage frequency at a glance and works well for comparing relative activity across controls. However, badges create persistent visual clutter during active analysis and cannot show temporal relationships or precise timing between interactions.

We ultimately favored ex-situ Gantt charts over other designs While in-situ encodings provide immediate context, their persistent overlays interfere with primary analysis tasks. Sankey diagrams reveal useful high-level patterns but obscure precise timing and state recovery, whereas Gantt charts explicitly encode interaction sequence and duration along a continuous timeline to fully support temporal tracking and action recovery (**G2**, **G4**). We integrated this Gantt chart into a new *SuperWidget*, described next.

2.3 SuperWidget: an extension to ProvenanceWidgets

SuperProvenanceWidgets extends ProvenanceWidgets [30] by introducing the *SuperWidget* for cross-control provenance (G1). It maintains full backward compatibility, supporting all original UI controls: radio buttons, checkboxes, single sliders, range sliders, dropdowns, multiselects, and input text fields. The *SuperWidget* comprises two views:

Aggregate View. To provide a compact overview of cross-widget provenance activity, we designed an Aggregate View (Figure 4b). Each colored box (pink, green, red, etc.) represents one UI control (input box, checkbox group, radio group, single/multi-select dropdown, single/range slider). Colors are automatically assigned via the *d3-scale-chromatic*[1] JavaScript library for clear distinction.

Initially, unused controls show just a colored border with no fill (Figure 4a). As interactions occur, boxes grow in size proportional to total interaction count and reposition based on recency. Hovering reveals details like widget ID, interaction count, and last interaction timestamp (Figure 4c). Clicking navigates to the corresponding control (**G3**), bringing it into focus, while the overview supports real-time aggregated visualization (**G2**).

Temporal View. To visualize the provenance across widgets over time, we designed an overlaid popover below the SuperWidget, showing full interaction history (Figure 5b). It lists all widgets by ID alongside a Gantt chart where the x-axis represents interaction sequence. Each row compiles that widget's interactions as colored boxes, revealing timing and ordering information.

To connect SuperWidget colors with individual UI controls, we enhanced ProvenanceWidgets' provenance button (with footprint icon) with matching colored borders. On click, the button fills solid with the same color, enabling instant visual identification between SuperWidget boxes and their corresponding controls.

2.4 Architecture

Existing provenance libraries like ProvenanceWidgets [30] and others [9] often use monolithic architectures that bundle all functionality together, resulting in unnecessarily large libraries that developers must include entirely. We redesigned ProvenanceWidgets based on the *Separation of Concerns* principle [12]. This decomposes **SuperProvenanceWidgets** into three independent, composable modules that developers can mix and match (**G5**):

(1) **Compute Module.** The backend engine that captures user interactions, computes provenance statistics (recency, frequency, sequences), and stores data in memory for real-time access.
(2) **UI Module.** The frontend library providing pre-built SuperWidget components (Aggregate View, Temporal View) that connect to any Compute backend.
(3) **Scents Module.** Optional visualization extensions that overlay provenance cues (colored borders, badges) directly onto individual UI controls.

Each module operates independently but integrates seamlessly via well-defined interfaces. Developers can use just the Compute module for custom UIs, combine UI+Compute for complete solutions, or add Scents for in-situ visualization–resulting in significantly smaller bundles and greater flexibility.

[1] https://d3js.org/d3-scale-chromatic

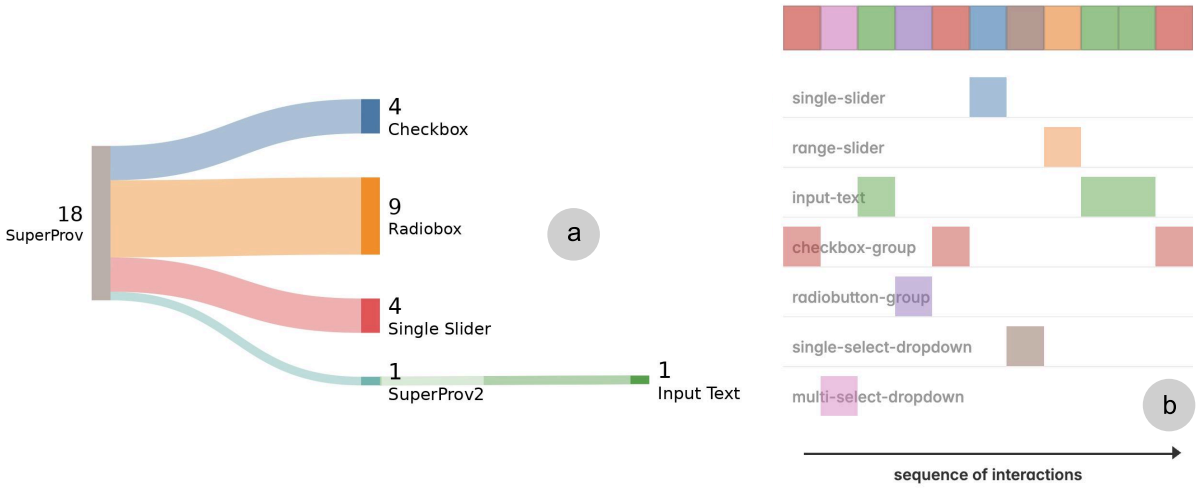

Figure 2: Different representations of user interactions (a) as a Sankey flow diagram and (b) as a Gantt chart.

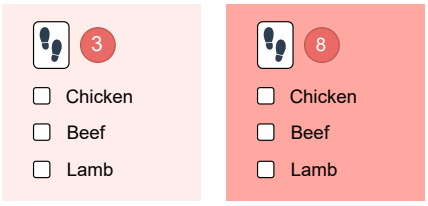

Figure 3: Two checkboxes with different interaction statuses and different visual properties.

2.5 Implementation

The **Compute** module for SuperProvenanceWidgets is implemented in TypeScript, and the **UI** and **Scents** are implemented in React.js; these modules can be used as *Web Components* making them framework-agnostic.

2.5.1 User Interface API. We declare our UI API below in terms of providers, hooks and components.

(1) **Providers.** Used to manage sharing of provenance-data across different components (**G1**).

```
<ProvenanceProvider>
    <App /> {/* Render root component */}
</ProvenanceProvider>
```

Additionally, wrapping the application with the provider adds hierarchy, so developers can wrap the portion of the application that needs use of provenance, as opposed to making it available throughout the application.

(2) **Hooks.** Used to invoke getters and setters for provenance-data. The hook we expose is `useProvenance`.

```
const [registeredComponents,
    setRegisteredComponents] =
    useProvenance()
```

The hook returns two objects: `registeredComponents` and `setRegisteredComponents`. The former is an array that stores the global provenance of every single widget, whereas the latter is a provenance modifier function.

Figure 4: Aggregate View: (a) empty interaction status, (b) interacted with components, (c) hovering over colored boxes

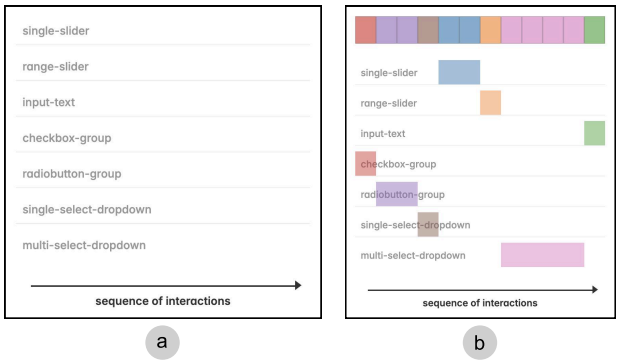

Figure 5: Temporal View: (a) empty interaction status, (b) interaction status after having interacted with multiple components

2.5.2 Scents API. The Scents API exposes a `<Bars />` component to render scent bars, with the following properties.

(1) **guidance**: Provenance data object containing aggregate and domain information used to populate and scale the bars.
(2) **orientationScheme**: A function that maps normalized values to colors for the bars based on the color domain.
(3) **barKeys**: Position bars along the secondary dimension (horizontal/vertical).
(4) **encodings**: An object specifying the visualization layout – orientation, positionDomain and colorDomain.
(5) **width** and **height**: SVG dimensions; dimensions are used for scaling bar lengths.

2.5.3 Compute API. The Compute API implements a single abstract *GenericProvenance* class, and then extends it into the following classes to expose type-level functionality.

(1) **NumericProvenance**: Adds min/max bounds to track the range of numeric data.
(2) **RangedProvenance**: Handles ranged slider interactions storing paired [lowValue, highValue] data with count tracking and edge boundary management for visualization.
(3) **SelectionProvenance**: Tracks multi-select interactions recording which items were selected/unselected at each timestamp, maintaining selection and interaction counts per item.
(4) **TextProvenance**: A type alias of GenericProvenance specialized for string values, used for text input tracking.
(5) **SuperProvenance**: Aggregates multiple widget provenances (sliders, dropdowns, checkboxes, etc.), allowing registration of multiple widgets and tracking overall interaction counts.

3 Usage Scenarios

We illustrate **SuperProvenanceWidgets**' capabilities through three scenarios that demonstrate how cross-control provenance can support common analysis needs.

3.1 Audit and share analysis workflows.

Imagine a data science team tasked with auditing a colleague's complex analysis session to ensure reproducibility and validate key decisions. Without aggregated provenance, reviewers must sift through raw logs or replay sessions manually, which is time-consuming and error-prone, especially when interactions span dozens of UI controls like sliders, dropdowns, and checkboxes.

With SuperProvenanceWidgets, developers can connect custom data sources to the Compute module, which automatically records every interaction across controls. The SuperWidget's Aggregate View immediately reveals dominant controls through large, colored boxes sized by interaction frequency and positioned by recency, while the Temporal View displays precise sequences and durations via intuitive Gantt bars. Team members can easily export this structured provenance–complete with widget IDs, timestamps, and counts–for asynchronous review, quickly confirming appropriate use of critical controls or spotting anomalies for discussion.

Figure 6: Scenario: (a) Widgets are out of viewport during a visual analysis session; (b) user navigates to specific widgets via the SuperWidget.

3.2 Surface and mitigate exploration biases.

Consider a visual analyst exploring a large dataset, such as sales records filtered by demographics and regions, but unintentionally fixating on familiar subsets due to viewport constraints or habitual control use. In traditional systems lacking cross-control summaries, this availability bias goes unnoticed, leading to skewed insights—like overlooking underrepresented regions or demographics—and potentially propagating errors into reports or models.

SuperProvenanceWidgets can empower users to detect such biases at a glance via the SuperWidget. Empty-filled boxes in the Aggregate View (Figure 6a) highlight untouched controls, such as out-of-viewport filters for gender or region; clicking them navigates directly to those widgets (Figure 6b), prompting balanced exploration. Developers can further leverage the API to reorder widgets by usage frequency, surfacing underused ones prominently; for instance:

```
const [registeredComponents,
    setRegisteredComponents] = useProvenance()
for (const [widgetId, widgetData] of
    registeredComponents) {
  // access widget-level data
}
```

This is particularly vital for real-world issues like gender bias, where users might filter to a single value without exploring alternatives, as the Aggregate View provides instant awareness of uneven interaction patterns across the full set of controls.

3.3 Facilitate user interface design and personalization.

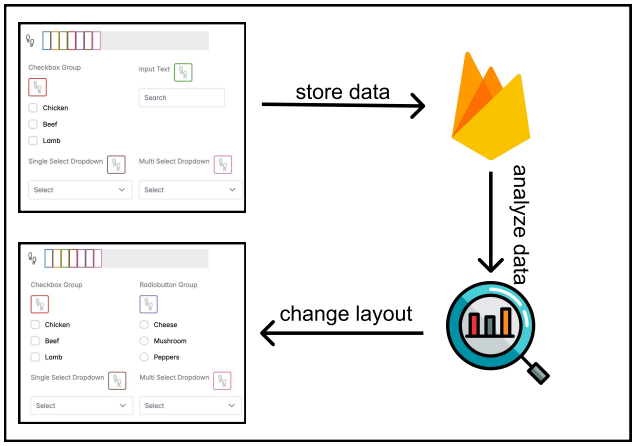

Figure 7: High-level process diagram for offline UI layout customization.

Picture a dashboard developer monitoring user sessions on an interactive analytics tool, noticing frustrating patterns like repeated scrolling or tab-switching between interdependent controls–such as color sliders paired with category checkboxes–for trend comparisons. Absent cross-control provenance, these inefficiencies remain hidden, resulting in suboptimal layouts that hinder productivity and user satisfaction.

The SuperWidget uncovers these patterns: frequent co-interactions appear as clustered color pairings in the Aggregate View and tight Gantt bar groups in the Temporal View. Developers can then implement dynamic personalization, auto-grouping high-co-use controls or bubbling low-usage ones to the top—using data streamed to Firebase for team analysis or processed in-memory for real-time tweaks (Figure 7). Options include manual weekly reviews or automatic in-session reordering, streamlining access and enhancing workflows for all users.

4 Evaluation: Cognitive Dimensions of Notation

We present a technical self-assessment of SuperProvenanceWidgets using the Cognitive Dimensions of Notations [5].

Consistency: SuperProvenanceWidgets exposes a common set of UI components, which behave consistently (described in Section 2.3). The notation is also consistent with the framework it has been implemented in (i.e. React.js), as well as being independent of base libraries (via Web Components).

Viscosity: The system's viscosity increases with lower abstraction. The **compute** module, being more primitive, requires more effort to work with than the **UI** module.

Abstraction Gradient: Following goal **G5**, the library spans multiple abstraction levels, from low-level **compute** to high-level **UI** components, enabling developers to choose the appropriate layer for their needs. The abstraction gradient flows from simple use-cases requiring least effort, or low viscosity, to more complex use-cases that span the full function of the library.

5 Limitations and Future Work

SuperProvenanceWidgets represents a significant step forward in web-based analytic provenance by enabling developers to track and visualize interactions across multiple UI controls in real-time. Through its modular architecture and innovative SuperWidget, the library addresses key challenges in analysis transparency, bias mitigation, and interface optimization, as demonstrated in our usage scenarios. However, several limitations present opportunities for future enhancement.

Limitations. While SuperProvenanceWidgets excels at capturing fine-grained interaction data within web-based UI controls, its current implementation has notable constraints. First, it focuses exclusively on client-side provenance tracking in JavaScript environments; it does not natively support server-side computations or integration with backend analytics pipelines, limiting its utility in full-stack applications. Second, the library assumes relatively low-to-moderate interaction volumes—high-frequency sessions with hundreds of rapid interactions may lead to visual clutter in the Temporal View's Gantt chart, despite filtering options. Third, while framework-agnostic via Web Components, adoption may require additional effort for non-React developers unfamiliar with its TypeScript-based APIs. Finally, provenance data remains in-memory by default, raising concerns for long sessions without explicit export mechanisms to persistent storage.

Future Work. Building on this foundation, several promising directions emerge. We plan to extend provenance capture to hybrid environments, integrating with server-side frameworks like Node.js or Python backends for comprehensive full-stack tracking. Adaptive visualization techniques—such as automatic clustering, zooming, or AI-driven summarization—could mitigate clutter in dense interaction histories. Real-time collaboration features, inspired by tools like Trrack, would enable shared SuperWidgets for team-based auditing. Additionally, leveraging the tracked provenance for proactive guidance, such as subtle nudges toward underused controls or bias alerts (e.g., extending prior work on analytic behavior awareness [28]), could transform passive tracking into active support for unbiased exploration. Finally, empirical user studies will validate the library's impact on real-world tasks, measuring outcomes like analysis reproducibility, bias reduction, and productivity gains.

6 Conclusion

SuperProvenanceWidgets is a JavaScript library of UI controls that tracks and visualizes users' analytic provenance within and across UI controls, empowering developers to build more transparent and adaptive guidance-enriched visual analytics tools [31]. It extends its predecessor library, ProvenanceWidgets, and is available as open-source software at **ProvenanceWidgets.github.io**.

References

[1] Wolfgang Aigner, Stephan Hoffmann, and Alexander Rind. 2013. Evalbench: A software library for visualization evaluation. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 41–50. https://doi.org/10.1111/cgf.12091

[2] Sriram Karthik Badam, Zehua Zeng, Emily Wall, Alex Endert, and Niklas Elmqvist. 2017. Supporting Team-First Visual Analytics through Group Activity Representations.. In *Graphics Interface*. 208–213. https://doi.org/10.5555/3141475.3141515

[3] Patrick Baudisch, Desney Tan, Maxime Collomb, Dan Robbins, Ken Hinckley, Maneesh Agrawala, Shengdong Zhao, and Gonzalo Ramos. 2006. Phosphor: explaining transitions in the user interface using afterglow effects. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*. 169–178. https://doi.org/10.1145/1166253.1166280

[4] Louis Bavoil, Steven P Callahan, Patricia J Crossno, Juliana Freire, Carlos E Scheidegger, Cláudio T Silva, and Huy T Vo. 2005. Vistrails: Enabling interactive multiple-view visualizations. In *VIS 05. IEEE Visualization, 2005.* IEEE, 135–142. https://doi.org/10.1109/VISUAL.2005.1532788

[5] Alan F Blackwell, Carol Britton, Anna Cox, Thomas RG Green, Corin Gurr, Gada Kadoda, Maria S Kutar, Martin Loomes, Chrystopher L Nehaniv, Marian Petre, et al. 2001. Cognitive dimensions of notations: Design tools for cognitive technology. In *Cognitive Technology: Instruments of Mind: 4th International Conference, CT 2001 Coventry, UK, August 6–9, 2001 Proceedings*. Springer, 325–341. https://doi.org/10.1007/3-540-44617-6_31

[6] Jeremy E Block, Shaghayegh Esmaeili, Eric D Ragan, John R Goodall, and G David Richardson. 2023. The Influence of Visual Provenance Representations on Strategies in a Collaborative Hand-off Data Analysis Scenario. *IEEE Transactions on Visualization and Computer Graphics* 29, 1 (2023), 1113–1123. https://doi.org/10.1109/TVCG.2022.3209495

[7] Zoya Bylinskii, Nam Wook Kim, Peter O'Donovan, Sami Alsheikh, Spandan Madan, Hanspeter Pfister, Fredo Durand, Bryan Russell, and Aaron Hertzmann. 2017. Learning Visual Importance for Graphic Designs and Data Visualizations. In *ACM UIST*.

[8] Steven P Callahan, Juliana Freire, Emanuele Santos, Carlos E Scheidegger, Cláudio T Silva, and Huy T Vo. 2006. VisTrails: visualization meets data management. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*. 745–747. https://doi.org/10.1145/1142473.1142574

[9] Akhilesh Camisetty, Chaitanya Chandurkar, Maoyuan Sun, and David Koop. 2019. Enhancing Web-based Analytics Applications through Provenance. *IEEE Transactions on Visualization and Computer Graphics* 25 (2019), 131–141. https://doi.org/10.1109/TVCG.2018.2865039

[10] Yang Chen, Scott Barlowe, and Jing Yang. 2010. Click2Annotate: Automated Insight Externalization with Rich Semantics. In *2010 IEEE Symposium on Visual Analytics Science and Technology*. IEEE, 155–162.

[11] Zach Cutler, Kiran Gadhave, and Alexander Lex. 2020. Trrack: A library for provenance-tracking in web-based visualizations. In *2020 IEEE Visualization Conference (VIS)*. IEEE, 116–120. https://doi.org/10.1109/VIS47514.2020.00030

[12] Edsger W. Dijkstra. 1982. *On the Role of Scientific Thought*. Springer New York, New York, NY, 60–66. https://doi.org/10.1007/978-1-4612-5695-3_12

[13] Yiren Ding, Jack Wilburn, Hilson Shrestha, Akim Ndlovu, Kiran Gadhave, Carolina Nobre, Alexander Lex, and Lane Harrison. 2023. reVISit: Supporting Scalable Evaluation of Interactive Visualizations. In *2023 IEEE Visualization and Visual Analytics (VIS)*. 31–35. https://doi.org/10.1109/VIS54172.2023.00015

[14] Anders Drachen. 2015. Behavioral telemetry in games user research. *Game user experience evaluation* (2015), 135–165. https://doi.org/10.1007/978-3-319-15985-0_7

[15] Cody Dunne, Nathalie Henry Riche, Bongshin Lee, Ronald Metoyer, and George Robertson. 2012. GraphTrail: Analyzing large multivariate, heterogeneous networks while supporting exploration history. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1663–1672. https://doi.org/10.1145/2207676.2208293

[16] Klaus Eckelt, Kiran Gadhave, Alexander Lex, and Marc Streit. 2023. Loops: Leveraging Provenance and Visualization to Support Exploratory Data Analysis in Notebooks. *OSF Preprint* (2023). https://doi.org/10.31219/osf.io/79eyn

[17] Tommy Elllkvist, David Koop, Erik W Anderson, Juliana Freire, and Cláudio Silva. 2008. Using Provenance to Support Real-Time Collaborative Design of Workflows. In *Provenance and Annotation of Data and Processes*.

[18] Alex Endert, Patrick Fiaux, and Chris North. 2012. Semantic Interaction for Sensemaking: Inferring Analytical Reasoning for Model Steering. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (2012), 2879–2888. https://doi.org/10.1109/TVCG.2012.260

[19] Mi Feng, Cheng Deng, Evan M. Peck, and Lane Harrison. 2017. HindSight: Encouraging Exploration through Direct Encoding of Personal Interaction History. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2017), 351–360. https://doi.org/10.1109/TVCG.2016.2599058

[20] Kiran Gadhave, Zach Cutler, and Alexander Lex. 2024. Persist: Persistent and Reusable Interactions in Computational Notebooks. In *Computer Graphics Forum*.

[21] Steven Gomez and David Laidlaw. 2012. Modeling Task Performance for a Crowd of Users from Interaction Histories. In *ACM CHI*.

[22] googleanalytics [n. d.]. Google Analytics. https://marketingplatform.google.com/about/analytics. Accessed: June 15, 2024.

[23] Samuel Gratzl, Alexander Lex, Nils Gehlenborg, Nicola Cosgrove, and Marc Streit. 2016. From Visual Exploration to Storytelling and Back Again. In *Computer Graphics Forum*.

[24] Troy Costa Kohwalter, Leonardo Gresta Paulino Murta, and Esteban Walter Gonzalez Clua. 2017. Capturing game telemetry with provenance. In *2017 16th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. IEEE, 66–75. https://doi.org/10.1109/SBGames.2017.00016

[25] Zhicheng Liu and Jeffrey Heer. 2014. The Effects of Interactive Latency on Exploratory Visual Analysis. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (2014), 2122–2131.

[26] Karthic Madanagopal, Eric D Ragan, and Perakath Benjamin. 2019. Analytic provenance in practice: The role of provenance in real-world visualization and data analysis environments. *IEEE Computer Graphics and Applications* 39, 6 (2019), 30–45. https://doi.org/10.1109/MCG.2019.2933419

[27] George A Miller. 1994. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *Psychological Review* 101, 2 (1994), 343.

[28] Arpit Narechania, Adam Coscia, Emily Wall, and Alex Endert. 2022. Lumos: Increasing Awareness of Analytic Behavior during Visual Data Analysis. *IEEE Transactions on Visualization and Computer Graphics* 28, 1 (2022), 1009–1018. https://doi.org/10.1109/TVCG.2021.3114827

[29] Arpit Narechania, Shunan Guo, Eunyee Koh, Alex Endert, and Jane Hoffswell. 2025. Utilizing Provenance as an Attribute for Visual Data Analysis: A Design Probe with ProvenanceLens. *IEEE Transactions on Visualization and Computer Graphics* (2025). https://doi.org/10.1109/TVCG.2025.3571708

[30] Arpit Narechania, Kaustubh Odak, Mennatallah El-Assady, and Alex Endert. 2024. ProvenanceWidgets: A Library of UI Control Elements to Track and Dynamically Overlay Analytic Provenance. *IEEE Transactions on Visualization and Computer Graphics* (2024).

[31] Arpit Ajay Narechania. 2024. *Designing, Developing, and Democratizing Guidance for Visual Analytics*. Ph. D. Dissertation. Georgia Institute of Technology.

[32] Phong H Nguyen, Kai Xu, Andy Bardill, Betul Salman, Kate Herd, and BL William Wong. 2016. SenseMap: Supporting Browser-based Online Sensemaking through Analytic Provenance. In *IEEE VAST*.

[33] Chris North, Remco Chang, Alex Endert, Wenwen Dou, Richard May, Bill Pike, and Glenn Fink. 2011. Analytic provenance: process+ interaction+ insight. In *CHI'11 Extended Abstracts on Human Factors in Computing Systems*. 33–36. https://doi.org/10.1145/1979742.1979570

[34] Jamal R Paden, Arpit Narechania, and Alex Endert. 2024. BiasBuzz: Combining Visual Guidance with Haptic Feedback to Increase Awareness of Analytic Behavior during Visual Data Analysis. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*. 1–7. https://doi.org/10.1145/3613905.3651064

[35] William A. Pike, John Stasko, Remco Chang, and Theresa A. O'Connell. 2009. The Science of Interaction. *Information Visualization* 8, 4 (2009), 263–274. https://doi.org/10.1057/ivs.2009.22 arXiv:https://doi.org/10.1057/ivs.2009.22

[36] Eric D Ragan, Alex Endert, Jibonananda Sanyal, and Jian Chen. 2016. Characterizing provenance in visualization and data analysis: an organizational framework of provenance types and purposes. *IEEE Transactions on Visualization and Computer Graphics* 22, 1 (2016), 31–40. https://doi.org/10.1109/TVCG.2015.2467551

[37] Ali Sarvghad and Melanie Tory. 2015. Exploiting analysis history to support collaborative data analysis. In *Proceedings of the 41st Graphics Interface Conference*. 123–130.

[38] Yedendra B Shrinivasan, David Gotz, and Jie Lu. 2009. Connecting the Dots in Visual Analysis. In *IEEE VAST*.

[39] Dan Siroker and Pete Koomen. 2015. *A/B testing: The most powerful way to turn clicks into customers*. John Wiley & Sons.

[40] Amy Skopik and Carl Gutwin. 2005. Improving revisitation in fisheye views with visit wear. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 771–780. https://doi.org/10.1145/1054972.1055079

[41] Andreas Walch, Michael Schwärzler, Christian Luksch, Elmar Eisemann, and Theresia Gschwandtner. 2020. LightGuider: Guiding Interactive Lighting Design using Suggestions, Provenance, and Quality Visualization. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2020), 569–578. https://doi.org/10.1109/TVCG.2019.2934658

[42] Emily Wall, Arpit Narechania, Adam Coscia, Jamal Paden, and Alex Endert. 2022. Left, Right, and Gender: Exploring Interaction Traces to Mitigate Human Biases. *IEEE Transactions on Visualization and Computer Graphics* 28, 1 (2022), 966–975. https://doi.org/10.1109/TVCG.2021.3114862

[43] Amelia Wattenberger. 2021. Footsteps for VS Code. https://marketplace.visualstudio.com/items?itemName=Wattenberger.footsteps.

[44] Wesley Willett, Jeffrey Heer, and Maneesh Agrawala. 2007. Scented Widgets: Improving navigation cues with embedded visualizations. *IEEE Transactions on Visualization and Computer Graphics* 13, 6 (2007), 1129–1136. https://doi.org/10.1109/TVCG.2007.70589