INKLING: Exploring How Students Design Visualizations with Integrated Sketching, Natural Language, and Code Editing

Sadie Fisher* Austin P. Wright† Ayaan M. Kazerouni‡

California Polytechnic State University
San Luis Obispo, CA

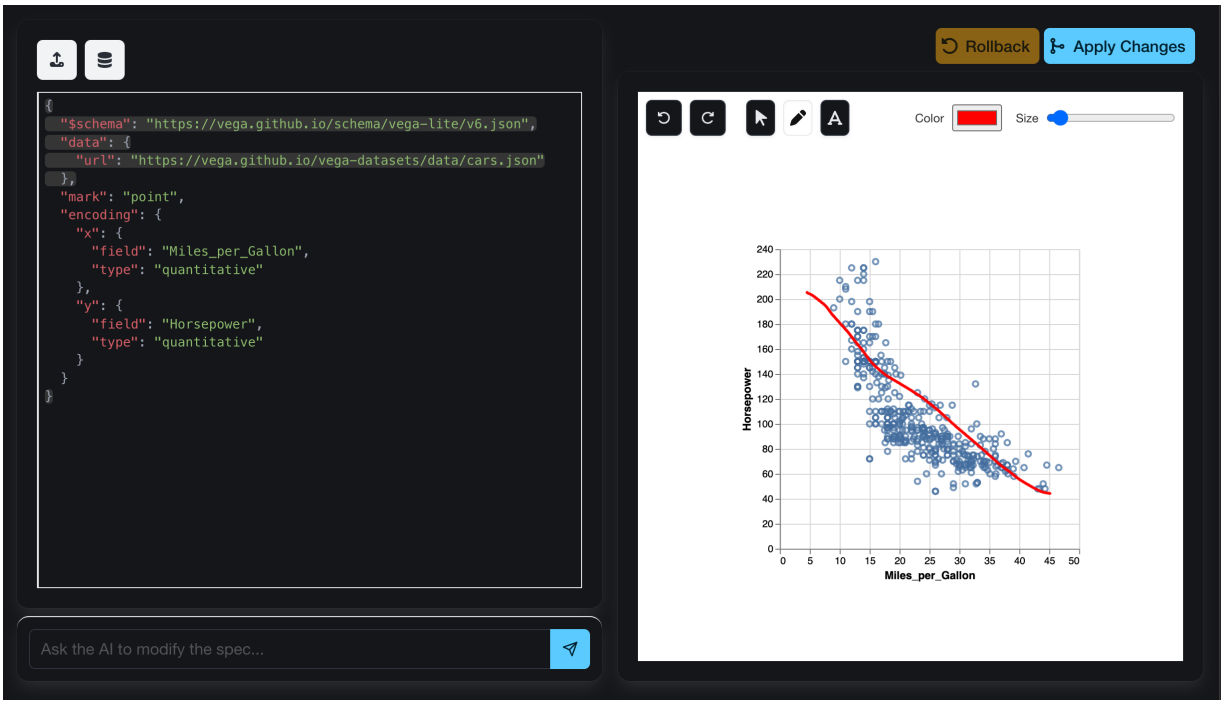

Figure 1: The INKLING interface, containing a Vega-Lite code editor (middle left), an input box for natural language instructions (bottom left), and a canvas for sketching and annotating (right). After first inputting a dataset, the user can begin designing with any of the three modalities, and freely switch between them as they refine the data visualization.

ABSTRACT

Visualization authors can use a number of tools to design and author data graphics, including those with alternative input modalities. For example, sketching is useful for exploring ideas, natural language for expressing intents that are not easily visually expressible, and direct code editing for precise refinement. Prior work exploring these modalities has tended to one modality at a time. However, a data visualization designer may benefit from the ability to fluidly move between modalities for different stages of the design process. We present INKLING, a prototype multimodal editor for data visualization in which users can combine sketching, natural language instructions, and direct code editing to design visualizations that are bound to their data. We conducted pilot user studies with six visualization novices to gain an initial insight into how they used the three modalities to complete assigned visualization tasks. This work is meant to inform further larger-scale empirical studies into the effectiveness of multimodal visualization editing. Our qualitative analysis of screen recordings and audio transcriptions revealed consistent patterns of usage of the three interaction modes.

Index Terms: Data visualization, multimodal editing.

1 INTRODUCTION

Data visualization has long been used to help make sense of and communicate insights from large amounts of data. Visualizations today are created not only by specialists, but also by students or domain experts like journalists wanting to present data at work or for personal purposes [7, 10]. While efforts have been made to improve visualization interpretation skills (e.g., [13, 6]), there remains a lack of pedagogical resources for visualization construction [7].

Huron et al. [10] outline three design challenges for universalizing visualization: keeping tools simple, enabling expressive freedom, and incorporating data dynamics. Existing solutions for designing visualizations typically address only one or two of these challenges at a time. Current approaches generally fall into three categories [10, 26]: template-based tools like Excel, programmatic systems, or environments that allow hand-drawn sketching. Template-based tools are simple to use, allowing users to quickly generate visualizations through a menu of predefined chart types. However, this restricts a user's ability to create custom visualiza-

*e-mail:sfishe10@calpoly.edu
†e-mail:awrigh20@calpoy.edu
‡e-mail:ayaank@calpoly.edu

tions for their particular analytical needs. Programmatic systems such as those based on grammars of graphics (e.g., [19]), offer greater expressive power by allowing users to declaratively compose primitives like marks, encodings, and layers to create custom visualizations. But this flexibility comes at the cost of a steeper learning curve. Finally, sketching is widely used during early-stage ideation because it is quick and highly expressive [3, 4], allowing users to freely explore visualization ideas without technical constraints. But sketches are inherently static and disconnected from underlying data, and users must manually re-create sketches using formal tools, introducing friction between ideation and implementation. This fragmentation can increase cognitive effort and makes it difficult to preserve design intent throughout the process.

We propose INKLING (Fig. 1), a multimodal visualization authoring system that integrates three authoring modalities: code editing, hand-drawn sketching, and natural language instructions. A user can freely move between or combine modalities based on familiarity or context, increasing expressivity and reducing the cognitive load associated with visualization design [3, 16].

While prior work has explored visualization authoring through different modalities, they have tended to assume a dominant mode of interaction, whether it be direct manipulation [3], coding [19], or prompting through text or speech [20, 27]. In the spirit of Huron et al.'s "constructive visualization" [10], INKLING acknowledges that visualization design is a multi-faceted task, where a designer may use sketching to explore ideas, language to express intent, and code for precision and refinement. Additionally, visualizations in INKLING are grounded in the user's dataset and an editable and exportable code representation.

We conducted six pilot user studies with visualization novices to gain an initial understanding of the limitations and affordances of our prototype. This workshop paper presents our initial evaluation along with plans for improvements and further research.

We found that participants used each modality for distinct purposes: sketching to construct and modify visual structure, natural language instructions to specify transformations that were difficult to express through sketches, and code editing to inspect and refine the visualization specifications produced using the other two modalities. This project contributes a novel visualization editing tool, an initial understanding of how and when authors employ the three modalities during the design process, and future design directions for combined visualization authoring inputs.

2 RELATED WORK

Visualization authors have a number of means through which they may create visualizations, each with its own strengths and weaknesses [10, 26]. *Template-based environments* like Excel are simple to use but limited to specific chart types. *Programmatic tools* like Vega-Lite [19] give the author more control but demand a steeper learning curve. *Natural language interfaces* like VisTalk [27] can help flatten these learning curves, but are still challenging if the designer needs to compose primitives like marks, encodings, layers, and multi-view layouts into custom displays that are not easily described in natural language. Finally, the fluid and expressive nature of *sketching* allows the designer to directly communicate the visual structure they envision, and allows them to quickly and cheaply explore many designs in parallel [4].

The value of sketching as an input modality has been recognized both within [3] and without [4, 9] the data visualization domain. In 2011, Browne et al. [3] presented SketchVis, which aimed to integrate sketching into the data analysis process. Using SketchVis, a user could sketch chart elements (like axes and legends) that were then bound to their real data to produce data visualizations. Users appreciated the rapid data exploration afforded by sketching, but Browne et al. acknowledge the limitations that SketchVis supported only two chart types (barcharts and scatterplots) and the sketch input modality, which limited what users could accomplish quickly. Huang et al. [9] presented SketchGPT, a multimodal interaction paradigm for touchscreen devices that couples sketch-based gestures and speech with system interfaces. Their evaluation suggested that users found the combination of sketching and speech to be easy to use, particularly for "graphical contents or elements that were hard to refer to [through speech alone]" [9, Section 5.5].

Another line of research investigates natural language interfaces for authoring visualizations. Setlur et al. [20] introduced Eviza, a natural language interface that employs a grammar-based probabilistic method to analyze queries posed by a user, and updates a visualization accordingly. Like in Huang et al.'s SketchGPT [9], Eviza demonstrated that speech-based interactions were rapid and intuitive, but posed limitations when it came to complex interactions. Similarly, Wang et al. [27] proposed an interface (VisTalk) to translate user utterances into structured visualization editing actions. Their user study found that participants viewed natural language interaction as easier to learn and more efficient than traditional menu-based interfaces, particularly for users with less visualization expertise.

The works discussed thus far have identified the potential of sketch- and natural language-based interactions for authoring visualizations, while also noting the limitations of using *only* one modality or the other. They have echoed Oviatt et al.'s recommendation for human-centered design, namely to model users' natural multimodal communication patterns [16, 17]. The SketchGPT [9] project showed multimodal interactions (including sketching) to be promising in general. However, these lines of work have so far remained separate: prior sketch-to-vis work (e.g., [3, 18, 23]) has not explored integration with other modalities like natural language and direct code editing.

Multimodal inputs can also positively impact visualization novices' learning of visualization tools. Multimodal inputs allow a user to express design intent through multiple representations. For instance, using the taxonomy described by Wu et al. [30], they may use *symbolic-mathematical* representations like code, *visual-graphical* representations like sketches, or *textual-verbal* representations like natural language instructions. Prior work has reported the pedagogical benefits of *multiple external representations* [1, 11] in STEM disciplines like chemistry [5, 30] and computing [21]. When a phenomenon or system is presented to a learner through multiple representations, it facilitates the formation of connections between concepts that might not otherwise occur through a single representation. Additionally, multiple representations and input modalities can reduce cognitive load by engaging multiple processors in working memory at once [2, 16]. For example, Sibia et al. [21] found that learners were able to better grasp relationships between abstract data structures, memory and pointers, and the code they were tracing when presented with multiple synchronized representations as they stepped through the code. Similarly, designing and iterating on custom visualizations through multiple inputs (representations) could prove conducive to transfer learning between modalities, e.g., for a novice's learning of a visualization grammar like Vega-Lite through sketching.

Tools such as INKLING can represent a step toward Huron et al.'s "constructive visualization" [10] in which novices as well as experts have the ability to design custom visualizations with simple, expressive, and dynamic tools that harness existing competencies.

3 DESIGN GOALS

Based on prior work and our goals, we focused on three primary goals in designing INKLING, each with associated preliminary targets and evaluation criteria.

Design Goal 1: Multimodality

We sought to design a tool in which users can fluidly move between modalities as desired. In particular, users unfamiliar with some modes (such as students who have yet to master a visualization grammar, or users with limited experience using conversational AI systems as a design tool) should be able to produce valid visualizations using the other modes, requiring that each mode be independently functional. Additionally, if users have some variable degree of experience in all three modes, they should be able to combine them in whatever way feels natural for each part of the authoring process, requiring that each mode be always available. Finally, when considering these two components of multimodality together, INKLING should be able to facilitate learning from one mode to another by allowing users to easily see the effect of making changes in one mode within the context of each of the others, thus allowing existing competencies to help "fill in the gaps" in other less developed competencies.

Design Goal 2: Iterative Refinement

Visualization design is never done in practice through a single fixed operation [15], and thus INKLING must support iterative refinement. Indeed, one key benefit of sketching is the ability to quickly explore designs, and a "one and done" approach in which a user draws one sketch, and they must interact with the high-fidelity data visualization thereafter would significantly impoverish this ability. The same holds for natural language instructions.

To this end, we aim for the user to be able to use any of the modalities—or a combination therein—to *edit* an existing visualization, i.e., a visualization that was created using any of the modalities can also be edited using any of the modalities. That is, they should be able to iterate by directly editing code, issuing natural language instructions, or sketching on top of a rendered visualization. In the case of direct code editing, this is straightforward—the new visualization specification is the result of the edits. In the case of the other two modalities, the user's design intents must be interpreted and incorporated into the existing visualization specification.

Design Goal 3: Groundedness

Our final design goal is that all visualizations produced using INKLING should be *specifications operating on valid data bindings*.

Prior drawing tools for visualization authoring have historically been "static", i.e., not bound to real data [10], or limited to a few specific chart types [3]. For sketching and natural language instructions to work as sibling modalities to direct code editing, they must also be bound to the user's data, in the same way that, for instance, a Vega-Lite specification is bound to a dataset. To this end, INKLING requires a user to specify the dataset with which they are working, and all expressions of design intent (through code edits, sketches, or natural language) are evaluated against the variables in that dataset.

Identifying user intention through combinations of sketches and natural language instructions has been shown to be promising through recent advances in generative artificial intelligence [9, 22]. However, with the use of generative AI comes the risk of peddling plausible falsehoods. Therefore, another component of this design goal is that we never use generative AI to directly produce a visualization. Rather, we use it to interpret sketches or natural language to produce a visualization specification that is bound to the user's real data. Another benefit of this approach is that resulting visualizations can be exported and edited using other tools of the user's choosing.

4 System Design

We implemented a prototype visualization editor, INKLING, which allows authors to design data visualizations through any combination of code editing, sketching, and natural language instructions. The user begins by inputting a dataset whose schema is used to keep all model-assisted outputs grounded in real data. The editor contains three main components: a *code editor*, a *canvas*, and an input for *natural language instructions*. All three inputs are available to the user at all times.

The code editor provides editing support for the Vega-Lite visualization grammar [19]. We selected Vega-Lite because it is declarative, supports multi-view composition, and is widely used.[1]

The canvas area plays two roles: that of *sketchpad* and *visualization display*. The author can use the canvas to sketch an initial design and then send it to a server, where a pipeline of generative AI agents interprets the sketch and returns a corresponding Vega-Lite specification that implements the user's sketch. The system compiles the Vega-Lite specification and renders the resulting visualization in the canvas area. The user can also use sketching in the canvas area for *iterative refinement*; they can make annotations on the currently-displayed visualization and go through the same procedure to receive an updated Vega-Lite specification. Each time they apply changes, the canvas contents (sketch marks and, if present, the rendered visualization) are sent to the server for processing, whereupon a new Vega-Lite specification is returned. At all times, the specification is available for direct editing by the user, which would trigger a fresh render in the canvas.

Finally, the user can input natural language instructions, such as "change the color of the line to blue" or "make a graph of profit over time". Such instructions are mapped to variables in the dataset if appropriate.

The visualization rendered in the canvas is *only ever* the result of a Vega-Lite specification based on the user's inputted dataset (i.e., the figure itself is never directly generated).

The author can use any of the three modalities (or combinations therein) for design and refinement. Prompt chaining is employed to interpret the user's input. This involves breaking down large tasks into smaller sub-tasks, and feeding the results of each subtask to the next. This avoids giving a single agent a large task that it must decompose by itself; generative AI agents tend to perform better on complex tasks when they are broken up into checkable subtasks [28, 31]. Each operation (sketching from scratch, annotating on top of a rendered visualization, and sending natural language instructions) follows its own pipeline. Calls to the model include the canvas contents (figure + annotations), the Vega-Lite specification currently in the editor, and the schema from the user's dataset. For example, the process for interpreting an annotated visualization follows these steps:

1. Compare the image to the Vega-Lite specification; determine at a high level what additions or edits have been made in the image that are not present in the spec.
2. Use those descriptions to make corresponding modifications to the spec.
3. Make sure all field names match a column name from the dataset. If not, substitute it with the closest match.
4. Iteratively perform the following checks on the modified spec. To avoid lengthy or infinite loading times, this step concludes when either the spec passes the checks, or three iterations have been performed.
    (a) Does the updated spec accurately represent the annotated visualization?
    (b) Are all components that are present in the sketch also present in the spec, and vice versa?
    (c) Are there any issues that will prevent the visualization from displaying properly?

[1] At the time of this writing, Vega-Lite has 5.4k stars on GitHub and is included by default in the popular visualization platform Observable.

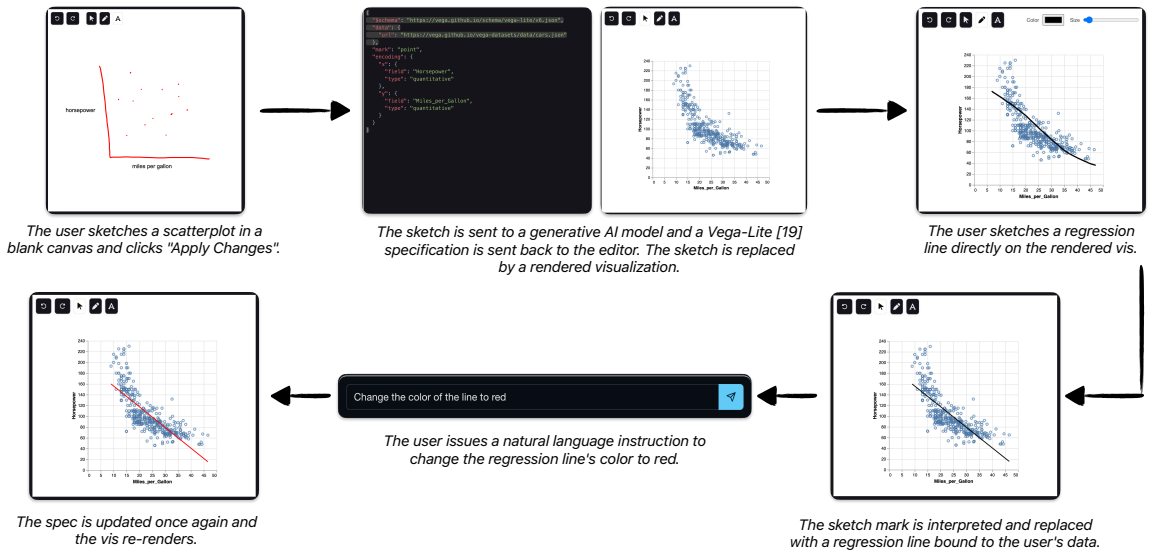

Figure 2: Example INKLING workflow. Each time the user applies a change (a sketch, annotation, or textual instruction) the Vega-Lite specification is updated accordingly, triggering a fresh rendering of the new visualization. The visualization specification is also directly editable.

5 PILOT EVALUATION

We conducted user studies to get initial feedback on our prototype. We qualitatively observed participants' interaction patterns with the three modalities, as well as the reasoning behind their interactions, gathered through a think-aloud protocol. We looked for the following information:

- What modality or modalities did participants tend to employ for each operation?
- How exactly did participants use those modalities? For example, if they tried to change the colors of markings through sketching, did they do so by coloring over the markings or by drawing a legend?
- Why did participants choose one modality over another?

5.1 Recruitment

We recruited six participants. First, we recruited participants from the pool of current and former takers of an upper-division elective course on data visualization in the computer science department at our university. Students in that course learn about visualization fundamentals such as the grammar of graphics and receive exposure to tools like Vega-Lite. Our expectation was that this relative familiarity with Vega-Lite would ensure that the code editing modality was not completely ignored. Additionally, by reaching out to our networks, we recruited a computer science graduate student (who was unfamiliar with Vega-Lite) and three individuals who had little or no experience with Vega-Lite and data visualization concepts. The spread of expertise is valuable given that the goal of this project is to assist data visualization novices, including those who have never used visualization libraries like Vega-Lite. Our participant pool is summarized in Tab. 1.

5.2 Procedure

We conducted user studies in which participants used our prototype to work through assigned tasks while thinking out loud. Participants were provided with a touchscreen device running INKLING, a stylus, and a keyboard. Each session was screen- and audio-recorded, and the procedure was approved by our Institutional Review Board.

Each session proceeded as follows. First, the participant took a short survey comprising questions about demographics and Likert-type questions about their experience with data visualization and Vega-Lite, as well as their trust in generative AI. The demographic questions were necessary to gauge representation in the group of participants, and the questions about trust in generative AI (taken from Hoffman et al. [8], Table 8) were necessary because users' perceptions of it are relevant to the ways they approach a tool that uses generative AI. All questions were optional.

After the survey, participants were given a demonstration of the features of the interface and could ask clarifying questions. Each participant was given the same demonstration.

Then the participants were given a set of tasks to work through. They were told that they could take as many steps as they wanted in order to reach the solution, and they were instructed to use whichever modality felt most intuitive for what they were trying to accomplish. The interviewer inputted the corresponding datasets into the tool for each task, and the datasets were also displayed for the participants' reference. The tasks are described below.

In this pilot evaluation, we aimed for a variety of *convergent* visualization authoring tasks, i.e., tasks meant to arrive at a predefined end goal. Task 1, which asked participants to replicate an existing visualization from scratch, let us observe how they approached various parts of the process from start to finish. Tasks 2 and 3, which incorporated smaller, specific sub-tasks, let us observe how users tackled specific types of operations.

Task 1. The system was loaded with a `stock-price` dataset, then participants were shown the visualization in Fig. 3 and asked to replicate it.

Task 2. The system was loaded with a `weather` dataset, and participants were given the following instructions in sequence, i.e., each subsequent instruction followed successful completion of the previous instruction. We wanted to simulate a process of iterating on a prototype visualization.

2.1 Make a scatterplot of max temperature over time.
2.2 Add a line showing the average max temperature over time.
2.3 Make the points more transparent.

Table 1: Participants' backgrounds, experience with data visualization and Vega-Lite, and trust in AI.

ID	Gender	Race/Ethnicity	Major	*I am comfortable designing data visualizations.*	*I am comfortable authoring data visualizations with Vega-Lite.*	Taken Data Vis Course	Trust in AI: Low (1) – High (5) [8]
P1	Man	White	Computer Science	Neutral	Somewhat Disagree	Yes	1.875
P2	Man	Asian/White	Computer Science	Neutral	Strongly Disagree	No	2.625
P3	Man	White	Architecture	Somewhat Agree	Strongly Disagree	No	2.75
P4	Woman	Asian	Computer Science	Somewhat Agree	Somewhat Agree	Yes	2.75
P5	Woman	Hispanic or Latino/White	Architecture	Neutral	Strongly Disagree	No	2.375
P6	Prefer not to answer	Hispanic or Latino	Computer Science	Somewhat Agree	Somewhat Agree	No	2.75

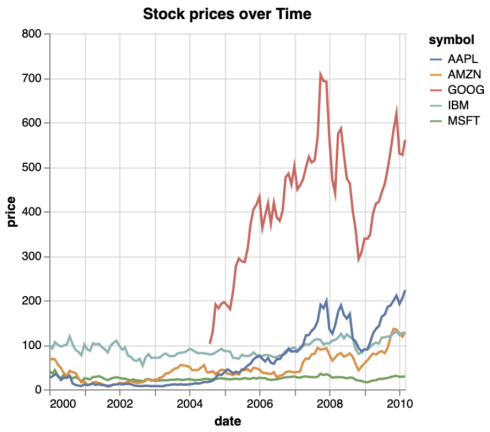

Figure 3: The visualization that participants were asked to recreate in Task 1.

Task 3. Participants were given a Vega-Lite specification that produced the visualization in Fig. 4:

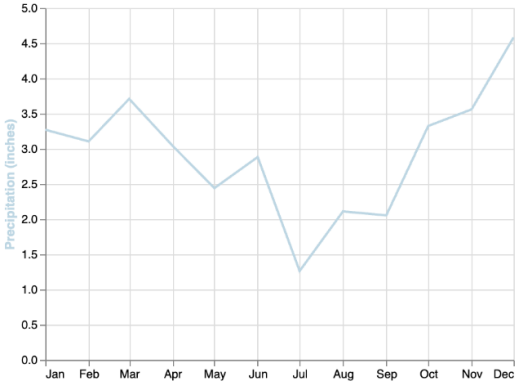

Figure 4: The starting visualization given to participants in Task 3.

Then they were given the following modification instructions in sequence:

3.1 Filter the graph to only include data from New York.
3.2 Set the precipitation scale's domain to range between 1 and 5.
3.3 Add a line showing wind over time. Display the wind scale on the right y-axis.
3.4 Change the color of the line and y-axis title for wind to orange.
3.5 Display both lines (precipitation and wind) as smooth/curved.
3.6 Below the plot, add a horizontal bar chart showing how many records exist for each type of weather for New York.
3.7 Move the bar chart to the top, above the plot.

5.3 Data Analysis

This section presents a description of participants' strategies to complete the assigned tasks. We analyzed each screen recording with its transcript and identified which modality or combinations of modalities participants chose to use for various tasks and their success. We felt it was not necessary to compile a formal codebook due to the small sample size, and because our goal was to gain an initial understanding of user interactions with the interface, as a precursor to further research.

5.4 Preliminary Findings

The primary goal of our pilot evaluation was to learn how authors used code editing, sketching, and natural language instructions while constructing and refining data visualizations. We identified some common ways in which participants interacted with the three modalities when using our prototype to work through data visualization tasks.

5.4.1 Task 1: Time series with a color encoding

All six participants started **Task 1** (recreating a provided visualization) by sketching the visualization's structure. It is possible they were driven to do this by the perceived novelty of the interaction; P1 and P3 said as much. Fig. 5 shows all six initial sketches along with a description of their sequence of interactions that led to successfully completing the task. Only P2's initial sketch led to the required final visualization. Most of the other participants verbally stated they wanted to "*see what happens and take it from there*" with the initial sketch. When the sketch did not produce the expected final result, P1 continued annotating (they drew a legend and wrote a title), and P3–P6 issued a single natural language instruction (e.g., P4 wrote "*Change the graph to show five lines of the five stock values over time*"). P3 and P5 finished by using the canvas to add the required chart title.

5.4.2 Task 2: Scatterplot with opacity setting and layered mean line

Task 2.1 asked the participants to create a scatterplot of monthly maximum temperature over time. Four participants again used sketching while participants P2 and P6 used textual instructions. P2 notably entered textual instructions as a comment in the code. Then, using natural language, they instructed the AI to fill out the code based on the comment; the resulting changes matched their intention.

Task 2.2 asked participants to overlay a monthly average line on the scatterplot. P1, P3, and P5 all began with sketching, while the other participants began with natural language instead. P1 made two attempts at overlaying a line using a sketch mark, but the model

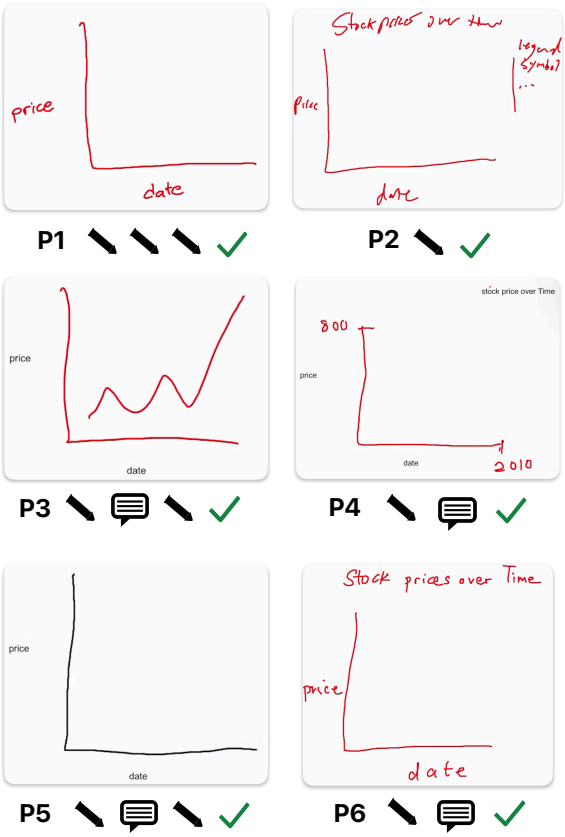

Figure 5: Initial sketches for Task 1. Only P2's first sketch succeeded immediately. All other initial sketches led to the vis shown in Fig. 6. The sequences of icons represent the series of sketch (✎) or natural language (💬) instructions the participant used before successfully completing the task (✓).

applied an incorrect transform each time (inferred incorrectly from the shape of the line drawn). P1 finally issued a natural language instruction specifying the desired transform (mean) and was successful. Participants P3 and P5 were hesitant to draw the line at first because they were unsure what it would look like, but ultimately they chose to see if the AI could determine what they were trying to draw. Importantly, their eventual figures did *not* result in a *mean* aggregation, but rather a LOESS transform. They did not notice since it looked visually close to what was expected, and they did not inspect the newly added code. The successful participants all started with or resorted to natural language for **Task 2.2** (with P6 stating not wanting the line to be misinterpreted as a motivation).

For **Task 2.3** (point opacity), all participants used a natural language instruction. P2 wanted to learn the Vega-Lite grammar during the session, and attempted to guess the code required first (they were close; they named the field "transparency" instead of "opacity"), but used natural language when that was not successful.

5.4.3 Task 3: Modifying a line chart by filtering, adding a layer, and changing a scale domain

Task 3 started with a time series of monthly precipitation amounts and asked the participant to make a series of modifications.

Four out of six participants (P1, P3, P4, P5) used a single natural language instruction to complete **Task 3.1** (filtering the weather

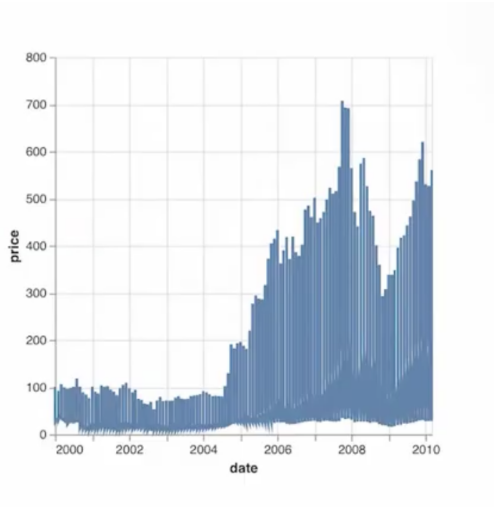

Figure 6: The model's interpretation of all the initial sketches for Task 1 except for P2's.

dataset to only show New York's weather). P5 commented that using natural language was helpful "*for all of the back-end stuff that you don't really manipulate visually, or process visually.*" P2 tried to make changes to the provided Vega-Lite and when that was not successful, also issued a natural language instruction. P6 made two attempts to achieve the desired filter through sketching. First, they wrote "New York" under the rendered visualization, which was interpreted as a title. Then, they wrote "show New York data only" by hand in the canvas, which was successful.

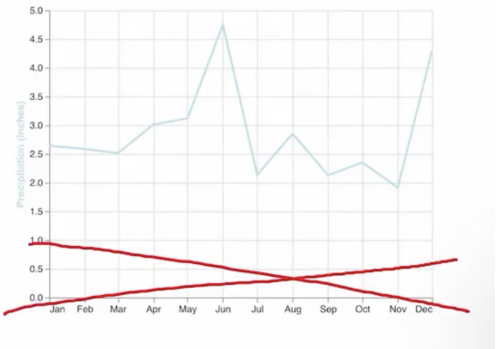

Figure 7: P5's solution to Task 3.2. They crossed out a section of the chart to communicate their intent to start the *y* axis at 1.

Task 3.2 (change the *y* axis domain from 0–5 to 1–5) saw some variety of strategies attempted. P1 and P2 both tried to directly edit the code with limited success (due to lack of Vega-Lite knowledge), and then switched to natural language. P3 did not edit the code, but they did first look through the code for the word "domain"—it was not present, so they issued a natural language instruction. P4 straightforwardly used natural language. Finally, P5 and P6 used the canvas to cross out the bottom portion of the graph (P5) or *y* axis (P6) to indicate that they wanted the domain to start at 1 instead of 0; Fig. 7 shows P5's attempt. For P5, this was not successful, and they fell back to natural language. For P6, the *y* axis scale was adjusted to start at approximately the minimum data value, which was not exactly what was asked for. So, they drew a new *y* axis with handwritten tick labels in the canvas, and were successful in

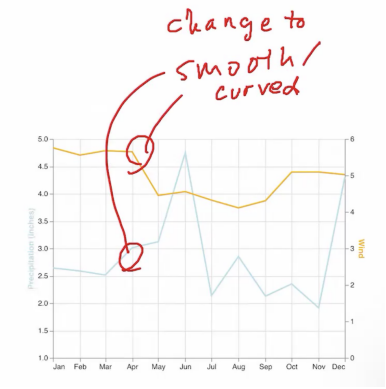

Figure 8: P6 wrote in the canvas to express desired changes.

achieving the desired domain.

For **Task 3.3** (layering in a line chart for wind speed), P1, P3, and P5 sketched a new trend line, a dual *y* axis, and the handwritten title "wind" on the existing visualization, and were successful at this task. P5 used natural language to adjust the title of the resulting additional *y* axis. The other three participants all used natural language to add the layered chart (with P2 again first attempting direct code edits, then falling back to natural language).

For **Task 3.4** (line and axis title color), all but P1 directly edited the code, using code from the original Vega-Lite chart as a reference. P1 used natural language to edit the axis title color.

Nearly all participants used natural language for **Task 3.5** (smoothing the lines in Fig. 4). P6 circled the sharp peaks on a line graph and wrote "change to smooth/curved" (Fig. 8)—a successful strategy.

For **Task 3.6** (concatenate a horizontal barchart below the line chart), nearly all participants used sketching. A couple of participants first drew a vertical bar chart and then used natural language to swap the axes. P2 only used natural language.

Finally, **Task 3.7** (moving the barchart above the line chart) was completed using a number of strategies. P1 and P3 used natural language immediately; P2 attempted to edit the code and then fell back to natural language; P4 (who had taken the course in which Vega-Lite was taught) successfully edited the code; and P5 and P6 both attempted to use sketching to communicate their intent. As in **Task 3.2**, P5 was unsuccessful: they drew a circle around the barchart and an arrow pointing above the line chart. This did not succeed, so they fell back to natural language. As in **Task 3.2**, P6 made a similar attempt, but provided more information to the model: they circled the barchart, drew an arrow, *and* hand-wrote "move above" in the canvas. This strategy was successful.

5.4.4 Trust in AI

Pre-interview survey responses suggested that participants held reservations about genAI. None indicated that they felt it was *predictable*, *reliable*, and *accurate*, and none responded more strongly than *neutral* to the statement "*I like using genAI for decision making*". Five out of six participants indicated wariness of genAI, and half "*somewhat agreed*" that they felt confident that genAI works well; the rest were "*neutral*" or "*somewhat disagreed*".

These results align with our observations that participants expected to make multiple attempts when using the model-assisted components. While working through tasks, participants often expressed that they were using the sketch or natural language tools to get to a starting point, and then they expected to make modifications from there. For example, when using natural language instructions to make the scatterplot in **Task 2.1**, P6 explained, "*I'll modify it if it's not to my liking.*" Reflecting on the changes made by the AI

Table 2: Modality usage by task. *Preferred* is the first modality the participant attempted for that task. *Fallback* pairs indicate a participant switched modalities after an unsuccessful attempt. Parenthesized numbers are the frequencies with which each pattern occurred.

✎= sketching 💬= natural language ⟨/⟩= code editing

Task	Preferred modality	Fallback
1	✎ (6)	✎→💬 (4)
2.1	✎ (4), 💬 (2)	–
2.2	✎ (3), 💬 (3)	✎→💬 (1)
2.3	💬 (5), ⟨/⟩ (1)	⟨/⟩→💬 (1)
3.1	💬 (4), ⟨/⟩ (1), ✎ (1)	⟨/⟩→💬 (1)
3.2	⟨/⟩ (2), 💬 (2), ✎ (2)	⟨/⟩→💬 (2), ✎→💬 (1)
3.3	✎ (3), 💬 (2), ⟨/⟩ (1)	⟨/⟩→💬 (1)
3.4	⟨/⟩ (5), 💬 (1)	–
3.5	💬 (5), ✎ (1)	–
3.6	✎ (5), 💬 (1)	✎→💬 (2)
3.7	💬 (2), ⟨/⟩ (2), ✎ (2)	⟨/⟩→💬 (1), ✎→💬 (1)

throughout their interview, P3 said, "*if they weren't what I wanted, I would be able to be more specific myself because I don't expect a level of specificity like that.*" Participants also frequently inspected the code after applying a sketch or instructions, even if they were unfamiliar with the language. After writing "*filter to only use NY data*" on the canvas to accomplish **Task 3.1**, P2 examined the code and verified that a filtering operation had been added. They also expressed that they felt there was a "*layer of abstraction and a lack of transparency*", and that it would be helpful if the changes made by the AI were highlighted and explained.

6 Discussion

We discuss some implications of our results and highlight avenues for discussion at the EduVis workshop and further research.

6.1 Patterns of Multimodal Editing

Observed interaction patterns are summarized in Tab. 2. Participants preferred sketching for tasks that were easily represented visually, like drawing a chart from scratch or moving, removing, or modifying visual elements. P4 said, "*I really appreciated [the sketching feature] because I feel like I made sure that I knew what I wanted before just kind of copy and pasting [the instructions] in.*"

Natural language was popular for changes that could not be easily communicated with sketching or direct code edits, or tasks for which participants were unsure how to represent the desired change through sketching or code (e.g., filtering data, changing the *y* axis range, and smoothing a jagged line mark). Natural language was the most common *fallback* modality when the user was unsuccessful with their preferred modality (Tab. 2).

Finally, direct code editing was typically used for small adjustments and to verify changes made by the model. Although only four participants were experienced coders (P1, P2, P4, and P6) and only two indicated that they were comfortable with using Vega-Lite (P4 and P6), all participants referenced the code editor multiple times during their sessions. Participants would often make a change through sketching or natural language instructions and then inspect the code to confirm that the expected modifications had occurred. This was especially common for tasks that involved adjusting specific values in the code, e.g., point opacity in **Task 2.3**. After the AI updated the code, P2, P4, and P5 again examined the code and located the newly added "opacity" value, and P4 and P5 adjusted it to their liking. Similar patterns of keyword-seeking emerged in later tasks involving modifications to colors or data filters (e.g., to confirm that a natural language filter instruction took hold). P5 commented, "*I really enjoy the ability to draw it out...and then being able to look through the code and manipulate certain aspects of it.*"

P1 similarly noted, "*In a perfect world, for anything small like that, I'd want to just go in and do it myself.*"

One caveat was that participants generally found the code editor somewhat difficult to use due to lack of a mouse or trackpad.

Participants tended to work iteratively, often using the model-assisted tools (sketching or natural language instructions) to establish a starting point rather than expecting them to produce exactly what they wanted. They would then refine the result or undo changes if necessary. In many instances, participants made comments such as, "*Let's just apply [the chat message or canvas changes] and see where it starts.*" P1 noted that, in their experience, AI systems performed better when asked to make smaller changes, which may have contributed to this behavior.

6.2 Implications for Learning

Our eventual goal is to use INKLING in the teaching and learning of data visualization design and implementation. While we leave for future work empirical study of INKLING's impact on visualization learners, in this section we explore the issue from a theoretical perspective. INKLING's support for multiple representations of a user's design intents has the potential to strengthen competencies associated with each modality [1, 11]. For example, a user who is not yet familiar with Vega-Lite may benefit from using a familiar action like sketching to express a visual structure, and seeing the Vega-Lite code that implements that structure.

We note that the positive effects of multiple representations on learning are not a given and may be influenced by a number of factors. In her DeFT framework, Ainsworth [1] suggests that multiple representations have three key functions to aid in learning: to complement, to constrain, and to construct deeper understanding.

Representations in INKLING *complement* each other. Sketches are uniquely suited to communicating visual structures [3], natural language instructions appear well-suited to describing instructions that are difficult to depict visually ([9], Sec. 5.4), and code editing was used to verify and refine generated outputs (Sec. 5.4). In our user studies, we also observed participants issuing atomic instructions through combined modalities, e.g., sketching and natural language (Fig. 8), or code and natural language (Sec. 5.4).

Representations in INKLING can also *constrain* each other. Consider again the multimodal instruction issued by P6, where they circled points in trend lines and wrote by hand "*change to smooth*" in the canvas. The same instruction might have contained ambiguities in natural language—for example, the instruction "make the lines smooth" might raise the question "which lines"? There was no such ambiguity in the multimodal instruction.

Finally, representations in INKLING can help the learner *construct* understandings that would have been difficult to achieve with a single representation. For example, we observed instances of participants using sketching or natural language to generate a Vega-Lite specification, and then refining the visualization by inspecting and editing the specification directly. Importantly, we observed this behavior with both Computer Science (P2) and Architecture (P3) students, neither of whom had taken the data visualization course in which Vega-Lite was taught. This can be empowering for a visualization creator whose goal is to *create visualizations* and not to *learn a visualization specification language*, but who nevertheless would benefit from the flexibility afforded by such languages.

The DeFT framework outlines benefits and risks of multiple representations that engage different processors in working memory. Some research suggests that this helps distribute cognitive load and form connections between concepts [2, 16, 21], while others suggest that the heterogeneity can make it difficult for learners to see relationships between different representations [1]. In the case of INKLING, we expect the multiple modalities to be a boon, in particular because of its rapid feedback cycle [12, 25]. When a user makes a sketch mark or issues a natural language instruction, they are immediately presented with the updated Vega-Lite code and resulting visualization, making an explicit connection between their chosen modality/representation and the specification.

However, as we saw in Sec. 5.4, with the ambiguity of sketching also comes the risk of a false sense of success. Two participants used sketching to produce a visualization with an incorrect data transform without noticing. The mistake would have been clear had they attended to the code changes produced by the AI model. This is an important risk factor to keep in mind for tool designers. Simply displaying the multiple representations is not sufficient; the connections between them must also be made readily apparent. We discuss possible tool improvements in the next section.

6.3 Future work

Larger-scale user studies. We conducted a small scale pilot with six participants whose goal was to inform further research. We will continue this work by conducting empirical evaluations of INKLING with a larger and more diverse participant pool, measuring success on visualization design and implementation tasks using existing assessments [7, 6]. We will also study impacts on non-visualization-expert users' tendency to reach for custom visualizations as ways to communicate data in their domain of expertise. These evaluations will be conducted in undergraduate data visualization courses at our university and through crowd-sourced user studies.

These studies will be preceded by improvements to INKLING based on the user studies described in this paper.

Tool improvements. Although our initial set of user studies proved promising in the sense that users liked and successfully used INKLING to complete visualization tasks, they also revealed several areas for improvement. First, not all tasks were successfully completed using the user's chosen modality of sketching. Further refinement of the pipeline of agents used to interpret sketch marks is underway. Next, we plan to make improvements to the code editor so that experienced users are not faced with a relatively impoverished editing experience compared to other editing tools. These improvements include improved error messages and suggestions based on the Vega-Lite specification schema.

Many interactions in INKLING are supported by generative AI. Our participants were cautious about relying entirely on AI-generated modifications, which appeared to encourage verification behaviors (e.g., reading the generated code), repeated attempts, and requests for additional transparency regarding what changes had been made. They frequently inspected the generated Vega-Lite specifications after making changes through sketching or natural language instructions, even if they did not directly edit the code. This behavior suggests that they valued the code representation as a source of verification and trust. Several participants expressed a desire to see which portions of the Vega-Lite specification were modified by the system in response to sketch or natural language inputs, and to receive explanations describing the reasoning behind those modifications. We plan to explore ways to make this connection more easily perceptible, starting with highlighting diffs when the visualization specification is updated by the generative AI model. We expect that this would have avoided the incorrect final visualizations from two participants in **Task 2**.

Beyond. We are planning numerous refinements to INKLING in addition to the suggestions that were surfaced through the pilot study. One is our choice of visualization grammar. We currently target Vega-Lite for reasons described in Sec. 4, but the authoring approach is not intrinsically tied to it. We may instead target a language-agnostic intermediate representation, like the one supported by Draco 2 [14], which can be compiled to a different grammar of choice (e.g., Vega-Lite, Altair [24], or ggplot2 [29]).

We also plan to use INKLING as a research platform to learn about the kinds of inputs that users prefer for communicating design intents regarding interactions and animations.

References

[1] S. Ainsworth. DeFT: A conceptual framework for considering learning with multiple representations. *Learning and Instruction*, 16(3):183–198, June 2006. doi: 10.1016/j.learninstruc.2006.03.001 2, 8

[2] A. Baddeley. Working memory: looking back and looking forward. *Nature Reviews Neuroscience*, 4(10):829–839, Oct. 2003. doi: 10.1038/nrn1201 2, 8

[3] J. Browne, B. Lee, S. Carpendale, N. Riche, and T. Sherwood. Data analysis on interactive whiteboards through sketch-based interaction. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '11, p. 154–157. Association for Computing Machinery, New York, NY, USA, 2011. doi: 10.1145/2076354.2076383 2, 3, 8

[4] M. B. Calavia, T. Blanco, A. Serrano, A. Biedermann, and R. Casas. *Think-Sketch-Create: Improving Creative Expression Through Sketching*, p. 1585–1597. Springer International Publishing, 2022. doi: 10.1007/978-3-031-15928-2_138 2

[5] A. Gautam, W. Bortz, and D. Tatar. Abstraction Through Multiple Representations in an Integrated Computational Thinking Environment. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, SIGCSE '20, pp. 393–399. Association for Computing Machinery, New York, NY, USA, 2020. doi: 10.1145/3328778.3366892 2

[6] L. W. Ge, Y. Cui, and M. Kay. CALVI: Critical Thinking Assessment for Literacy in Visualizations. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–18. ACM, Hamburg Germany, Apr. 2023. doi: 10.1145/3544548.3581406 1, 8

[7] L. W. Ge, Y. Cui, and M. Kay. Avec: An assessment of visual encoding ability in visualization construction. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, CHI '25. Association for Computing Machinery, New York, NY, USA, 2025. doi: 10.1145/3706598.3713364 1, 8

[8] R. R. Hoffman, S. T. Mueller, G. Klein, and J. Litman. Measures for explainable ai: Explanation goodness, user satisfaction, mental models, curiosity, trust, and human-ai performance. *Frontiers in Computer Science*, Volume 5 - 2023, 2023. doi: 10.3389/fcomp.2023.1096257 4, 5

[9] Z. Huang, C. Gao, Y. Shan, H. Hu, Q. Li, X. Deng, C. Ma, Y.-K. Lai, Y.-J. Liu, F. Tian, G. Dai, and H. Wang. Sketchgpt: A sketch-based multimodal interface for application-agnostic llm interaction. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*, UIST '25, p. 1–18. ACM, 2025. doi: 10.1145/3746059.3747598 2, 3, 8

[10] S. Huron, S. Carpendale, A. Thudt, A. Tang, and M. Mauerer. Constructive visualization. In *Proceedings of the 2014 Conference on Designing Interactive Systems*, DIS '14, p. 433–442. Association for Computing Machinery, New York, NY, USA, 2014. doi: 10.1145/2598510.2598566 1, 2, 3

[11] R. Kozma. The material features of multiple representations and their cognitive and social affordances for science understanding. *Learning and Instruction*, 13(2):205–226, Apr. 2003. doi: 10.1016/S0959-4752(02)00021-X 2, 8

[12] B. C. Kwon, W. Javed, N. Elmqvist, and J. S. Yi. Direct manipulation through surrogate objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 627–636. ACM, Vancouver BC Canada, May 2011. doi: 10.1145/1978942.1979033 8

[13] S. Lee, S.-H. Kim, and B. C. Kwon. VLAT: Development of a Visualization Literacy Assessment Test. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):551–560, Jan. 2017. doi: 10.1109/TVCG.2016.2598920 1

[14] D. Moritz, C. Wang, G. L. Nelson, H. Lin, A. M. Smith, B. Howe, and J. Heer. Formalizing Visualization Design Knowledge as Constraints: Actionable and Extensible Models in Draco. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):438–448, 2019. doi: 10.1109/TVCG.2018.2865240 8

[15] T. Munzner. A Nested Model for Visualization Design and Validation. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):921–928, Nov. 2009. doi: 10.1109/TVCG.2009.111 3

[16] S. Oviatt. Human-centered design meets cognitive load theory: designing interfaces that help people think. In *Proceedings of the 14th ACM international conference on Multimedia*, pp. 871–880. ACM, Santa Barbara CA USA, Oct. 2006. doi: 10.1145/1180639.1180831 2, 8

[17] S. Oviatt, A. Arthur, and J. Cohen. Quiet interfaces that help students think. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, pp. 191–200. ACM, Montreux Switzerland, Oct. 2006. doi: 10.1145/1166253.1166284 2

[18] J. Poco and J. Heer. Reverse-engineering visualizations: Recovering visual encodings from chart images. *Computer Graphics Forum*, 36(3):353–363, 2017. doi: 10.1111/cgf.13193 2

[19] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):341–350, 2017. doi: 10.1109/TVCG.2016.2599030 2, 3

[20] V. Setlur, S. E. Battersby, M. Tory, R. Gossweiler, and A. X. Chang. Eviza: A natural language interface for visual analysis. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16, p. 365–377. Association for Computing Machinery, New York, NY, USA, 2016. doi: 10.1145/2984511.2984588 2

[21] N. Sibia, J. Wen, A. Richardson, Y. Jain, A. Zavaleta Bernuy, B. Simion, A. Petersen, C. Nobre, and M. Liut. From State to Structure: Towards Abstraction Support in CS2. In *Proceedings of the 25th Koli Calling International Conference on Computing Education Research*, pp. 1–12. ACM, Koli Finland, Nov. 2025. doi: 10.1145/3769994.3769998 2, 8

[22] T. Sivertsen, N. Singh, and J. C. Davis. Sage: Structured agentic graph editing for software diagrams, 2026. 3

[23] Z. Teng, Q. Fu, J. White, and D. C. Schmidt. Sketch2vis: Generating data visualizations from hand-drawn sketches with deep learning. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, p. 853–858. IEEE, Dec. 2021. doi: 10.1109/icmla52953.2021.00141 2

[24] J. VanderPlas, B. Granger, J. Heer, D. Moritz, K. Wongsuphasawat, A. Satyanarayan, E. Lees, I. Timofeev, B. Welsh, and S. Sievert. Altair: Interactive statistical visualizations for python. *Journal of Open Source Software*, 3(32):1057, 2018. doi: 10.21105/joss.01057 8

[25] B. Victor. Learnable programming. Online essay, Sept. 2012. Accessed 2026-07-01. 8

[26] B. Victor. Drawing dynamic visualizations. Talk recorded at the Stanford HCI Seminar, Feb. 2013. Video available at `https://vimeo.com/66085662`; addendum at `http://worrydream.com/DrawingDynamicVisualizationsTalkAddendum`. Accessed 2026-06-29. 1, 2

[27] Y. Wang, Z. Hou, L. Shen, T. Wu, J. Wang, H. Huang, H. Zhang, and D. Zhang. Towards natural language-based visualization authoring. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):1222–1232, 2023. doi: 10.1109/TVCG.2022.3209357 2

[28] J. Wei, X. Wang, D. Schuurmans, M. Bosma, b. ichter, F. Xia, E. Chi, Q. V. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds., *Advances in Neural Information Processing Systems*, vol. 35, pp. 24824–24837. Curran Associates, Inc., 2022. 3

[29] H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. 8

[30] H.-K. Wu and S. Puntambekar. Pedagogical Affordances of Multiple External Representations in Scientific Processes. *Journal of Science Education and Technology*, 21(6):754–767, Dec. 2012. doi: 10.1007/s10956-011-9363-7 2

[31] T. Wu, M. Terry, and C. J. Cai. Ai chains: Transparent and controllable human-ai interaction by chaining large language model prompts. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22. Association for Computing Machinery, New York, NY, USA, 2022. doi: 10.1145/3491102.3517582 3