DataGarden: Formalizing Personal Sketches into Structured Visualization Templates

Anna Offenwanger, Theophanis Tsandilas, and Fanny Chevalier

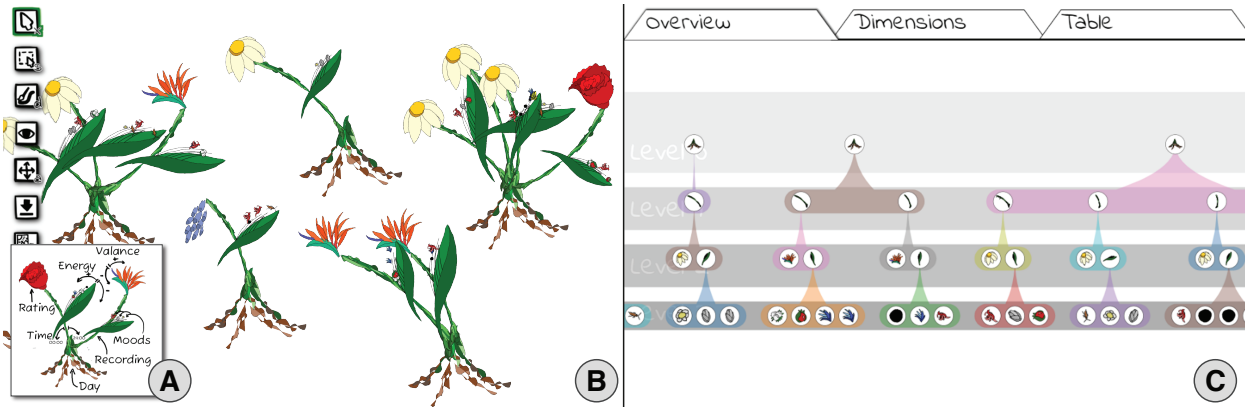

Fig. 1: DataGarden supports sketching personal, expressive designs and formalizing these as structured visualization templates. To express (A) a visualization design idea, a user sketches a few representative glyphs in (B) the canvas, making their vision explicit. DataGarden provides the means to structure the freeform sketch into a visualization template by (C) capturing implicit style and explicit data mappings via user interaction and machine support. 'Real' data can then be fed to the template. This featured visualization is generated from a template created with the tool. For additional examples, see: https://datagarden-git.github.io/datagarden

Abstract— Sketching is a common practice among visualization designers and serves an approachable entry to data visualization for non-experts. However, moving from a sketch to a full fledged data visualization often requires throwing away the original sketch and recreating it from scratch. Our goal is to formalize these sketches, enabling them to support iteration and systematic data mapping through a visual-first templating workflow. In this workflow, authors sketch a representative visualization and structure it into an expressive template for an envisioned or partial dataset, capturing implicit style as well as explicit data mappings. To demonstrate our proposed workflow, we implement DataGarden and evaluate it through a reproduction and a freeform study. We investigate how DataGarden supports personal expression and delve into the variety of visualizations that authors can produce with it, identifying cases that demonstrate the limitations of our approach and discussing avenues for future work.

Index Terms—Personal visualization, Visualization template, Sketch input, Sketch-based visualization, Visualization by-example

1 Introduction

Sketching is a common practice among visualization designers, and is also an approachable entry to visualization for individuals interested in personal data. Designers use sketches as a *"device to help us make our vision explicit"* [43], to externalize envisioned data mappings and visuals before getting into the reeds of data cleaning, and sometimes even before data is available [2]. Sketching and other forms of visual-first data visualization authoring workflows are also being encouraged as an approachable medium to get non-experts engaged in data visualization [34]. However, these sketches often remain just sketches. To move from a sketch to a full fledged data visualization requires throwing away the original sketch and laboriously redrawing it from scratch, or changing mediums and recreating it in existing visualization software. Can we instead formalize these visualization sketches to enable them to support iteration and systematic data mapping?

- *Anna Offenwanger and Theophanis Tsandilas are with the Université Paris Saclay, CRNS, Inria, LISN.*
  E-mail: {anna.offenwanger | theophanis.tsandilas}@lisn.upsaclay.fr
- *Fanny Chevalier is with the Departments of Computer Science and Statistical Sciences at the University of Toronto.*
  E-mail: fanny@dgp.toronto.edu

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

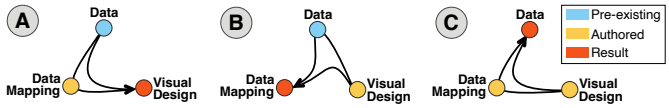

Fig. 2: (A) Traditional data-driven workflows assume pre-existing data, with authors specifying data mappings to create visuals. (B) Alternative visual-first workflows start with creating visuals, which are then linked to pre-existing data to form data mappings [21, 65]. (C) Our workflow enables authors to sketch visuals, generate a data schema from their envisioned data mappings, and record data from their sketches.

Freeform sketching gracefully blends personal aesthetic taste and data analytics [31]. Examples include Lupi and Posavec's hand-drawn visualizations from *Dear Data* [33, 34] and Bremer and Wu's *Data Sketches* [3], featuring creative, informative, and visually unique data representations. The expressive and flexible nature of sketching makes it a powerful yet highly-accessible medium that anyone can harness [2, 32]. Its flexibility in particular makes it well suited for crafting personal visualizations, whose design is limited only by one's imagination.

Perhaps more importantly, a freeform visual-first approach frees authors from many constraints imposed by existing visualization workflows. Systems often assume pre-existing data, and prioritize data-driven workflows at the expense of creative freedom (Figure 2.A). To achieve more expressiveness, approaches that focus on graphics, layout, and aesthetics have been proposed, with some integrating vector-graphics editing [21, 37, 41, 70] and sketching [65] within the authoring workflow. In these approaches, visual authors start by laying graphics on a canvas, and then bind the visual properties to the underlying

data—a process referred to as lazy data binding (Figure 2.B). While such workflows offers greater flexibility, data binding still assumes an existing dataset with which to reconcile the graphics representation. This imposes an artificial limitation to the creation process [51], which our work aims to address.

We propose a workflow in which authors formalize a personal sketch into a structured visualization template through specifying data mappings (Figure 2.C). Authors express a representative visualization for an envisioned or partial dataset by giving examples of the range of values, mappings, and the overall layout through sketch-based interaction, and then specify the intended data mappings. We aim to preserve the unique qualities of hand-drawn visualization, such as small variations in shapes and color, unorthodox layouts and glyphs, and idiosyncratic sketching styles. To maintain the advantage of sketching in that it is free from busywork, we also seek to semi-automate the sketch structuring process. Semi-automation is preferable to full automation because while sketch recognition can significantly reduce the workload of instrumenting hand-drawn sketches, fully automated methods often cannot handle ambiguity in visuals, and author input is needed to specify, for example, which visual properties are representational and which are aesthetic. We therefore seek to have interaction and machine support work in tandem, showing the author the structure inferred by the system, while also enabling the author to modify and expand on the structure. This workflow aligns with the visual-first approach common to designers [2], but we believe it is also applicable to non-experts as a tangible way to engage with data visualization [12, 13].

In order to demonstrate and evaluate our proposed workflow, we implemented DataGarden, a sketch-based authoring system supporting a small subset of data visualizations we call 'plant-like' visualizations, characterized by a hierarchical structure resembling tree branches and leaves. This specialized focus allows us to study our visualization template creation workflow through a more constrained yet still expressive and highly idiosyncratic family of representations. We describe how an author can create a custom template in DataGarden through an illustrative scenario (Section 4) and describe the computational methods that support our workflow (Section 5). We evaluate how DataGarden supports individuals to use and establish an understanding of the sketch-based templating workflow through the system and tutorial we provide (Section 6.1). We also assess how this workflow supports the creation of custom visualizations with four visualization experts (Section 6.2). We complement the two studies with a gallery of visualizations crafted using the system (Section 6.3). Drawing from our findings, we discuss how DataGarden supports personal expression, and delve into the variety of visualizations that authors can produce with it. We also identify cases which demonstrate the limitations of our approach and discuss avenues for future work (Section 7).

2 Related Work

DataGarden's design draws upon an extensive body of prior research on sketch-based data visualization and visualization authoring tools.

2.1 Visualization Sketching

Functions of visualization sketching. Professionals commonly transition between sketches on paper and computers to develop new visualization designs [11]. Lupi [32] explains that before working with actual data, sketches enable them to *"visualize the architecture of the infographics and cultivate ideas for shaping the data visually,"* while later, sketching with data can *"help raise new questions about the data itself."* Bremer emphasizes the benefits of sketching *"to discover and remove thinking errors"* when externalizing a concept on paper [3].

Bigelow et al. [2] who observed how designers work with data confirm this sketching practice. However, sketching is also readily accessible to non-designers as a medium for swiftly exploring personal data representations [60], maintaining visual diaries in everyday life [34], or introducing children to visualization and data science [18]. DataGarden targets users who may not be experts in design but are eager to explore creative ways to collect and represent their own data.

Sketch-based visualization systems. Information visualization research has studied sketch-based systems that cover a wide spectrum of data-oriented activities. SketchStory [24], SketchInsight [25], SketchSliders [58], and ActiveInk [44] focus on data-exploration tasks, while VoyagerInk [22] targets data-analysis tasks when analysts annotate their graphs and their notes. These systems often combine sketching with pen and touch interaction.

Other research efforts have been dedicated to visualization authoring tasks. Early systems like NapkinVis [6] primarily aimed at inferring common chart types, such as scatter plots and bar charts, from rough sketches. More recent work, however, studies sketching as a means of crafting personal data representations and narratives. Many of these approaches draw inspiration from projects like DearData [33]. For instance, DataInk [65] allows users to create artistic glyphs on custom layouts defined through sketching. Dear Pictograph [45] extends this concept to immersive environments, while DataQuilt [69] combines sketch input with image processing of photographs or illustrations. DataSelfie [20] enables users to sketch custom data representations, which can then be linked with data collected through questionnaires.

Another line of research focuses on sketch-based interaction for data storytelling and creating new visualization design, including authoring of data comics [19], expressive timelines of personal data [38], immersive representations of scientific data [17, 53], and personal representations of algebraic expressions [49].

While DataGarden shares similar motivations with existing sketch-based authoring systems like DataInk [65] and DataSelfie [20], it distinguishes itself through its support for flexible, hierarchical structures and its unique authoring workflow centered around generalizable visualization templates, as elaborated upon later in this section.

Structuring sketches. When users can sketch freely on a canvas, structuring sketches can be difficult. Computer graphics algorithms can assist this process, for example, by grouping together individual strokes [28, 29] and parameterizing these groups [39]. The HCI community has also investigated interaction mechanisms for managing sketch recognition and semantics. In Musink [59], for example, a music composer can sketch strokes that express a custom representation, but the actual structure of these strokes and their semantics are defined at a later stage through an interface that combines automatic recognition and manual control. In WritLarge [66], users can interactively specify which portions of their sketches they want to be recognized and then transform strokes in flexible and easily reversible representations that define structure and semantics at different levels. DataGarden incorporates techniques and concepts from this entire line of research.

2.2 Visualization Creation Workflows

Graphics-driven workflows. Visualization systems traditionally follow a data-driven workflow which Bigelow et al. [2] observe restricts the creative freedom of visual thinkers interested in crafting novel data representations with a focus on graphics design, layout, and aesthetics. The visualization literature has explored various paths to address this problem. One approach is the *ink-data settling* procedure [52], developed within the context of scientific visualization, which automatically infers associations between data and illustrative marks drawn by an artist. Systems like Gold [37], Lyra [50], Charticulator [41], and Data-Driven Guides [21] integrate vector-graphics editing tools into visualization authoring workflows. These systems enable authors to draw their data marks and glyphs before binding their parameters to the data. Data Illustrator [30] introduces further flexibility through a *lazy data binding* mechanism, where data bindings only partially constrain the visual objects. This allows authors to directly manipulate non-bound visual properties or remove bindings to explore alternatives.

While these systems offer greater flexibility, Satyanarayan et al. [51] note that they still assume that authors have a *"desired visualization in mind"* and *"a dataset in the appropriate format."* These assumptions do not always align with some visualization creation workflows, especially when the data or schema is not available upfront.

Addressing personal data collection practices, DataSelfie [20] eliminates the need for pre-existing datasets. However, its applicability is

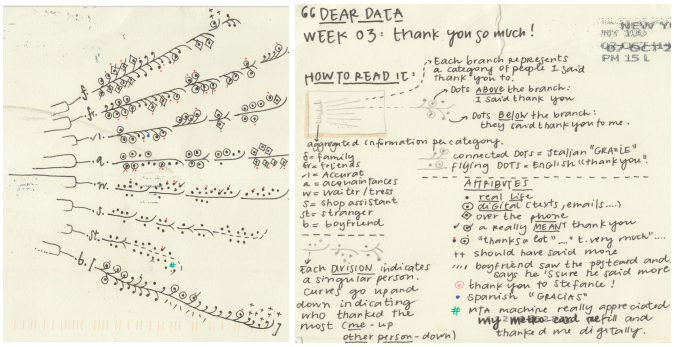

Fig. 3: Lupi's visualization (cropped) of "thank yous" (Week 3) and their notes on the data mappings [33]. DataGarden helps authors formally define the data mappings of sketched visualizations and use them to convert sketches into generalizable visualization templates.

limited to visualizing responses to questionnaires and requires authors to design the questionnaire before drawing visuals. Tailored towards design-oriented workflows, StructGraphics [57] offers a more versatile data-agnostic approach by deriving the data schema directly from the visualization structure. DataGarden expands on this approach to support personal representations for users who are not necessarily experts, focusing particularly on helping them structure their freeform sketches.

Visualization by example. Once created, DataGarden templates serve as examples for generating new visualization instances. *By-example* or *by-demonstration* visualization workflows have been used in past research to generate charts from a partial example [37], to infer visualization grammars [62], or to transform visualizations for data exploration purposes [48]. Unlike DataGarden, which supports custom sketch-based data representations, these approaches were limited to a small set of traditional charts such as bar charts and scatter plots, and typically required a well-formatted input dataset.

Input visualizations. Traditionally, data visualizations are visual representations of pre-existing datasets. Nevertheless, as Huron and Willet [14] and, more recently, Bressa et al. [4] explain, a visualization can also serve as the medium of direct data collection. The authors identify various instances of *input visualizations*, including those designed for personal reflection [55], personal planning [40], and engaging in public debates [23]. Sketching is commonly used for adding annotations [22,38] or data hunches [27] over existing data marks. DataGarden templates take this a step further by allowing users to input new data points directly through sketching on the canvas.

3 Approach and Design Considerations

We target supporting authoring of personal, expressive visualizations in scenarios where the actual data may initially be unavailable or only partially accessible [51], such as in the case of an individual planning what personal data to collect. Creating a new visualization design always requires establishing a data schema and defining the visual encodings that map the drawn elements to data values, and vice versa. We propose to support such scenarios by enabling users to author a visualization *template* through freeform sketching. This template describes the graphical structure of the visualization, and the mappings between the visualization structure and the data schema. It can then be used as an archetype for entering new data points through sketching or generating new visualization instances using entirely new data. Our approach is guided by the following design considerations.

D1: Support By-Example Visualization Specification

Since data may not be available or not yet be formatted or digitized [2], the visualization authoring workflow should start from the drawing canvas as a constraint-free and expressive medium [3, 34]. We take a lazy-data-binding-like approach [30] where visualization authors can start by creating visual elements and defer data mapping to later in the process. Borrowing ideas from StructGraphics [57], our goal is to deduce the data tables directly from the visualization. Thus, we treat the sketched elements as an *input visualization* [4, 14], where authors have the capability to input data by directly drawing data points onto the drawing canvas itself. Combined together, these approaches enable a flexible visual-first workflow tailored to sketching, where users can draw new visual elements, create new hierarchy levels, and arrange the visualization structuring at any point in the process.

D2: Preserve the Idiosyncratic Qualities of the Sketch

Sketching is often viewed as a swift and informal method of creation. However, a sketch inherently carries the distinctive signature of its creator. In the context of DataGarden, the focus is on scenarios involving personalized data visualization, where the user who crafts the visualization is also its primary consumer. Consequently, our aim is to uphold the idiosyncratic aesthetics of the sketch. This is not only to enhance support for memory functions [61], but also to encourage user engagement with the visualization [63] and strengthen the system's role as a tool for personal expression.

D3: Support Interactive Sketch Recognition

Working with free-from sketches is challenging because unstructured sketches are difficult for a system to recognize. We employ a *semi-structured delayed interpretation* [59] approach where sketch recognition can be partial and unfolds progressively across multiple semantic levels. Individual strokes are grouped, acting as the basic visualization elements, which then compose the larger visualization structure. Their visual properties (e.g. color, angle) are interlinked through this structure. Subsequently, the visualization author can assign semantics by binding these elements to data dimensions.

The transition from sketches to structured visualization templates requires close interaction between the human and the machine, supporting partial and iterative specification essential for visualization novices [10]. The visualization authors must communicate their visual intent and the intended data abstraction, while the machine must convey its understanding of the visualization structure and its modifications.

D4: Facilitate Smooth Transitions between Representations

The creation of visualization templates demands authors to engage with various visualization representations, each catering to different aspects of the creation workflow, including drawing, structuring at different semantic levels, and specifying data dimensions. Throughout this process, it is crucial to enable authors to seamlessly navigate across semantic levels [66], enabling them to assess and refine recognition without being constrained to a specific sequence of actions. These representations should therefore be fully synchronized, incrementally changing as the author edits the different representations.

However, aiding authors in deducing mappings between these representations is equally crucial. We draw inspiration from the interaction model of Histomages [7], where the pixels of an edited image correspond to those in its histogram, allowing for their selection and modification in either synchronized view. In DataGarden, the elementary components preserved across views are not pixels but strokes and their groups that compose the visualization glyphs.

4 DataGarden

To scope our exploration, we focus on a particular class of visualizations, borrowing from the metaphor of a *garden* from older work in personal informatics [9] to characterize a custom representation of a personal data collection. DataGarden's representations draw inspiration from hierarchical plant-like visualizations featured in the Dear Data dataset [33] (see Figure 3). These representations offer flexible, organic layouts that can be easily extended, such as by directly sketching free-from branches or custom-shaped glyphs that represent individual data points. The unrestricted nature of such layouts mirrors the nature of many personally significant data contexts, which are often messy, idiosyncratic, and evolving [54, 55].

DataGarden's user interface is composed of two fully synchronized components (Figure 1): the *sketching canvas*, where glyphs are drawn and laid out (Figure 1A), and the *structure inspector*, which includes three complementary views of the visualization structure (Figure 1B): (1) The *overview* shows the visualization hierarchy as deduced by the

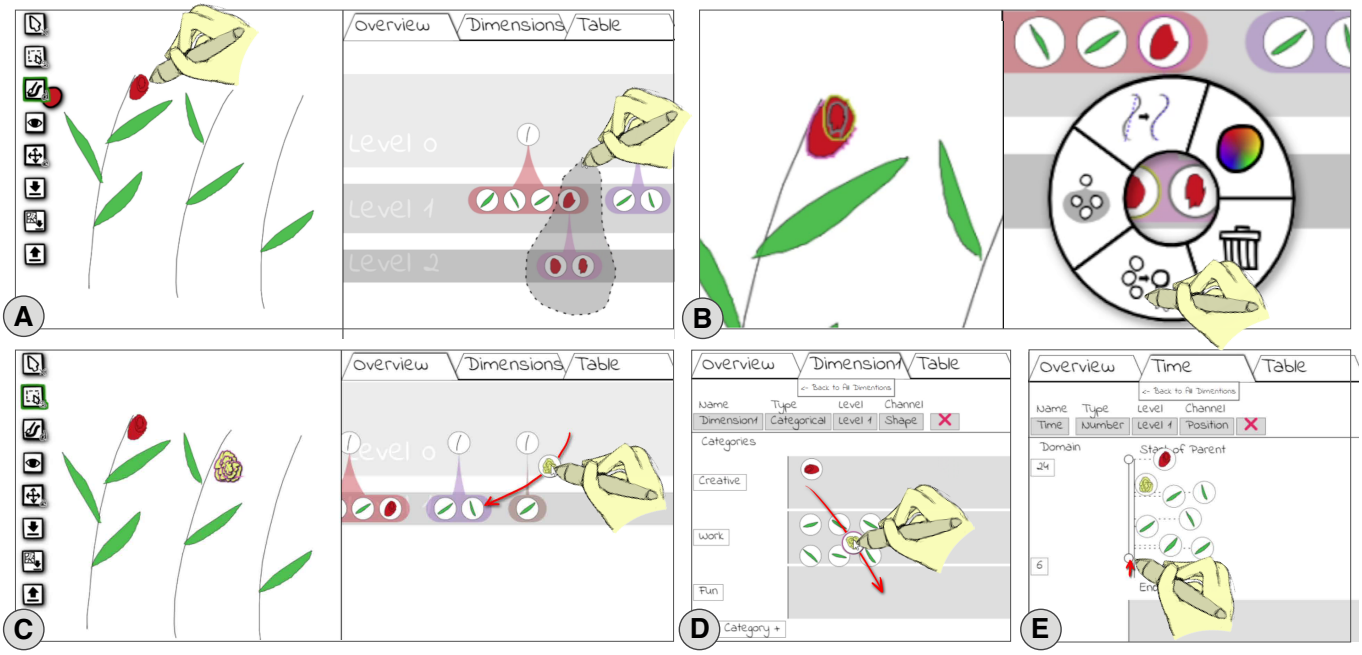

Fig. 4: A visualization authoring workflow (A) Mary begins by sketching her design concept. DataGarden supports customized hand-drawn glyphs that can be composed of many strokes, like the rose Mary draws on the *sketching canvas*. To support structuring the strokes into glyphs, DataGarden includes polymorphic selection, so Mary can select her strokes in the canvas or in any of the *structure inspector* views. Here, she does so in the *overview*. (B) The selection and highlighting propagates across all the views so Mary can be sure she has the right strokes selected. Mary merges the strokes into a single glyph using the merge option in the context menu. (C) The structure inspector views support reviewing and correcting the systems inferred structure. Mary notices her carnation is not parented correctly, so she drags it to the correct child group. (D) Mappings are specified through direct manipulation. Mary drags the carnation to its own category in her dimension. (E) For continuous channels like position, the elements are displayed along a linear projection. Mary specifies that her data range covers a portion of the channel range.

system (Figure 4A) and allows the author to directly manipulate glyphs to make corrections (Figure 4A-C). (2) The *dimensions view* provides tools for defining data dimensions and establishing mappings between the dimensions and the visual channels of the glyphs (Figure 4D-E). DataGarden supports mappings from categorical and continuous data to identity and magnitude visual channels. And (3) the *table view* shows the data tables derived from the dimensions and mappings specified by the author (Figure 6). The data tables are interactive, so the author can change values and append new rows, thereby changing and generating new instances of glyphs and therefore visualizations on the canvas.

We walk through DataGarden's workflow principle through a usage scenario outlined in Figure 4 and the video figure in our supplemental materials. This workflow is supported by several automatic classification algorithms described in Section 5.

4.1 Sketching the Visualization Template

The visualization author, Mary, has decided to create a visualization of her activities to reflect on when she finds time for her creative pursuits. She wants to break down her activities by day and be able to see how many of her activities are work-related or recreational versus actually creative. She decides to draw each day as a separate plant, and have multiple leaves and flowers on each plant to show her activities.

DataGarden's visual-first workflow (**D1**) means Mary can start sketching her idea without needing to reference her activity data (Figure 4A-left). To defer to the author's style (**D2**), DataGarden performs minimal beautification, and complex shapes are supported through the ability to compose them from multiple strokes, either layering by drawing them in order from bottom to top (Figure 4A) or "oversketching" lines (Figure 5) as traditionally used by artists [28, 29].

4.2 Structuring the Visualization Hierarchy

While Mary sketches on the canvas, DataGarden attempts to infer her intended glyphs and hierarchical structure, communicating this structure in its overview (**D3**). DataGarden does not require the visualization to be complete to start structuring the glyphs and may misinterpret the author's intent. Mary notices that while the system has correctly inferred the hierarchy of leaves, the strokes of the rose have not been grouped into a single glyph (Figure 4A). Mary uses the lasso tool to select the strokes in the overview, or alternatively on the canvas (**D4**), and applies a merge from the context menu (Figure 4A-B). Later, when a newly drawn flower is not parented correctly, she corrects the visualization hierarchy (**D3**) by dragging her glyph in the overview (Figure 4C). Glyphs are represented here as directly manipulable icons, helping Mary associate them with their original canvas representation (**D4**).

4.3 Specifying Data Dimensions

Mary switches to the dimensions view to externalize the data mappings that she has in mind. Mary wishes to map each branch to a different day, the position of each leaf node to time, and the shape to a different type of activity, i.e. "Creative" (roses), "Fun" (carnations), and "Work" (leaves). She considers the six visual channels supported by DataGarden (shape, color, position, size, angle, and label), and begins by creating a categorical dimension bound to shape for her activity mapping. DataGarden attempts to automatically cluster the glyph shapes into the intended number of categories (**D3**). As clustering is often difficult or ambiguous, Mary can specify the number of categories, assign them labels, and correct the classification of individual glyphs by dragging their icon to the right category (Figure 4D). DataGarden enables Mary to map data dimensions to glyphs appearing at any level of the visualization hierarchy. For example, Mary maps days to individual branches using the "label" channel of the top level. DataGarden can then infer that the branches' day values apply to their children.

Mary then decides to create a continuous numeric dimension to represent time within a day and bind it to the position channel of her glyphs. She specifies the extreme values of time (e.g., from 0 to 24) as well as their range along the *spine* of the branch (Figure 4E).

For continuous channels like position, size, and angle, DataGarden infers the visual value from the sketch, along with the dependency tree for relative values like position. If the inferred values are incorrect, the author can use DataGarden's tools to reveal the *spine structure* of the glyphs on the canvas (Figure 5). The author can then manually correct the spine of a glyph (**D3**) by redrawing it, or by requesting the system

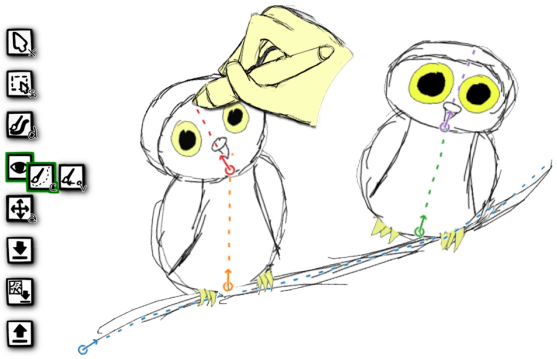

Fig. 5: A visualization with a primary stem (perch), two branches (bodies), and two leaves (heads), showing the individual spines (dashed lines), roots (circles) and angles (arrows). Users can freehand draw spines for glyphs to specify the layout relations between parent and child glyphs.

to automatically infer a new spine (discussed in Section 5). Similarly, the glyphs' inferred angle can be manually changed by placing the root (which serves as the base of the rotation) and manipulating the direction of the representative arrow (Figure 5).

4.4 Interlinked Views and Two-way Data Mapping

With a dimension specified, DataGarden infers the data values currently represented in the canvas, and displays them in the table view (Figure 6), thereby providing the data schema for this visualization (**D1**).

To support both manipulation and comprehension across different views (**D4**), DataGarden offers polymorphic selection tools [1]. The selection tools can be applied to objects in any view, and the selection is populated across all the views (Figure 4A-B). Mary can therefore select items from the table (Figure 6) to ensure that her glyphs are being represented correctly and there are no errors in the structure, or the data mappings (**D3**). In addition, to further support forming the conceptual link between the different template abstractions, DataGarden also supports animated transitions between the structure views. If Mary switches between the dimension view and the overview, the nodes will shift between their respective positions in each view (**D4**).

The binding between the table and the canvas extends beyond the selection. Updates to the table values will update the glyph sets and their visual attributes. Updates to the glyphs will likewise update the table values. Therefore, when Mary has collected and formatted her data to match the table, she can input her dataset to create one or many complete versions of the visualization (Figure 6).

With DataGarden, Mary can thus author a simple instance of the visualization which captures all the properties of the data and mappings, by drawing just a few exemplar glyphs and specifying their structure within a hierarchical layout (**D1**). Once done, she can apply these template specifications to automatically scale this to a larger dataset (Figure 6). Conversely, Mary can use the template as input visualization [14], drawing a new branch every day, and documenting her activities during that day by sketching new flowers and leaves.

5 Technical Implementation

DataGarden is a Web application implemented with JavaScript using the D3 library to support tree (*Overview*) and animated force-directed layouts (*Overview* and *Dimensions* view). DataGarden implements heuristics and algorithms for automatic inference and classification, including for stroke merging, glyph parenting, assigning spines, generating new elements, and revising the layout. In this section, we detail the structure of DataGarden visualizations and the techniques used to support sketch recognition and template-based visualization generation.

5.1 DataGarden Visualization Structure

The core of a DataGarden visualization is its glyphs. A glyph can have an arbitrary shape, is composed of one or many strokes with optional color, and is parameterized through a root, spine, and angle. Glyphs are hierarchically nested, so each glyph can have multiple children, positioned along its spine. For example, the branch in Figure 5 is

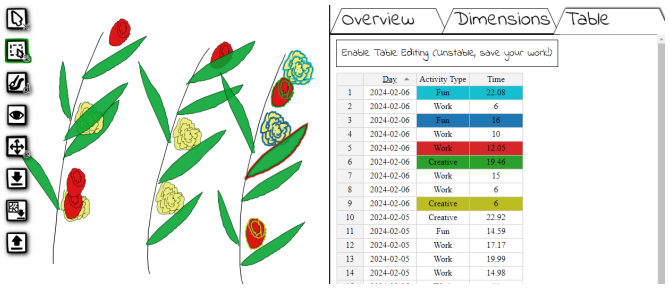

Fig. 6: With her template specified, Mary can add rows to the data table to create the final version of the visualization. The selection propagation can support her in inspecting her data mappings.

the parent of the two owl bodies, which in turn are the parents of the respective owl heads. Each glyph's root defines the base of the glyph around which it is rotated. The root is projected onto the closest point on the parent's spine to determine the glyphs relative distance along its parent. The root also determines the orientation of the glyph's own spine, the end of the spine closest to the root being considered as the start. We designed this structure to enable support for five visual channels (position, angle, size, color, shape) chosen from those accepted as most effective for continuous and/or categorical data [36]. We included a sixth channel value label, which is not a visual channel but rather a means to specify that data is represented by glyph presence.

DataGarden departs from visualization conceptualization of representative authoring systems [51] by treating all visual elements as the same sort of object, a glyph, rather than distinguishing between axes and leaf nodes. By doing this, DataGarden can easily support arbitrarily deep trees, and a flexible workflow where authors are not required to specify whether a glyph is meant to be a leaf node. Since each hierarchy level contains a different type of glyphs, DataGarden's interface allows authors to map different data dimensions to each level, e.g., the angle of the stems in Figure 1 are mapped to time whereas the angles of their child leaves are mapped to energy. DataGarden's design also departs from tree visualization grammars [26]: instead of using a global coordinate system, the spine of each glyph defines the 1D curvilinear position coordinates for its own children.

5.2 Automatic Template Structuring

To enable automatic template structuring, we opted to develop simplistic 'good enough' algorithms that would reduce the burden of the visualization structuring enough to enable us to demonstrate the notion of template creation. We expect that future work will be able to substantially improve on the algorithms outlined here.

Stoke merging is conducted based on line overlap. If n points of the new stroke fall within a δ distance of an existing stroke, we assume they should belong to the same glyph. This heuristic is primarily accurate for overdrawing scenarios [29, 39]; in other instances like in Figure 4A, the system will often fail to identify parts of the flower as a single glyph. We experimented with a more sophisticated algorithm, StripMaker [28], but as the current implementation is not designed to work in real-time, more work is required to integrate it into a workflows like this.

Glyph parenting is performed by comparing the size of the new stroke relative to the size and distance of the surrounding strokes. This assumes a) that people draw parent items before drawing their children, and b) that parent items will be larger than their children. Since in plant-like visualizations these two prerequisites are generally met, these heuristics function well in many cases.

To calculate distance and relative angle from parents to children, we need a simplified path representation of glyphs, which we call a spine. In order to calculate spines, we integrated StrokeStrip [39]. StrokeStrip was designed to create a clean spline from multiple overdrawn strokes, and is therefore suitable for generating a spine for any line-like glyph. However, the algorithm does not work for non-linear glyphs such as the plant roots in Figure 1. As its original implementation does not run in a browser, we connect it to DataGarden's Web interface through a server. We also provide a backup heuristic which takes the two most

distant points on the glyph, projects all points onto that axis in order to bucket them, and then takes the average of the buckets to create a spine.

For clustering and classification, we use basic k-means and k-nearest neighbors algorithms over the pixels of miniature versions of the glyphs in their upright orientation, where DataGarden initially tries to infer an optimal k by using the silhouette score [47] of the Euclidean distance. We apply clustering and classification to both shape and color, allowing DataGarden to support collections of colors that can then be bound to the same categorical value.

5.3 Visualization Generation

To show how the visualization structuring results in a usable template, we developed a naive algorithm to generate a full visualization from a DataGarden template and a dataset. For each row in the table, DataGarden will create a new glyph by copying one of the corresponding shapes from the template (or a glyph picked randomly from that of the right level if shape is not bound). If a parent glyph does not yet exist, it will copy a new parent from the template. If bound to data, the glyph's size, angle, color, and position will be updated. If not, glyphs will remain unchanged and be distributed evenly, retaining their relative angle, along their parent's spine.

6 Evaluation

In line with past evaluation approaches [42], we conducted a reproduction and a free-from user study to evaluate DataGarden. A visualization gallery provides additional examples that demonstrate the expressive potential of our approach.

6.1 Reproduction Study

Our reproduction study evaluated whether new users could establish a functional understanding of the visual-first workflow, as well as how DataGarden's interaction and automatic recognition supports and fails to support them in creating visualization templates.

6.1.1 Method

Participants. As the visual-first workflow can be applicable to both visualization experts and non-experts, we aimed to recruit participants with a range of visualization experience. We recruited 12 participants (6 men, 6 women) from our university campus and a student residence. Three participants reported no previous experience with data visualization, nine reported using data visualization occasionally or frequently for analysis. Of those, three reported designing new data visualizations occasionally or frequently.

Apparatus. Participants were seated in a quiet room and interacted with a (1920 × 1080) Wacom Cintiq 16 pen display connected to a Dell laptop running Windows 11 with DataGarden on Google Chrome. A slide deck with task instructions and a 7-minute video tutorial of DataGarden were presented on a separate computer screen, next to the pen display. For the preparation task, participants were provided with a paper worksheet and colored pencils.

Task. Participants were asked to create a DataGarden template for a visualization visually inspired by Giorgia Lupi's visualization from Week 27 of Dear Data [33] (Figure 8-Reference). The visualization captures activities (either work or personal) with different outcomes (productive or unproductive) during a single week. The participants had to implement three categorical data dimensions (*Day*, *Type* of activity, and its *Outcome*) and one continuous (*Time*). They were required to establish mappings with visual channels at various hierarchy levels: each day had to be mapped to a unique branch, the position of leaves in a branch had to be mapped to a different time of the day, while the color and relative angle of each leaf had to be mapped to the type and outcome of an activity. The participants were asked to envision using their template in a scenario in which they would track their activities, in coordination with a bunch of friends.

Procedure. After completing a background questionnaire, the participants were briefed on the scenario, the data dimensions, and the mappings of the target visualization template. They were then given a paper worksheet with a copy of the visualization that served as inspiration (Figure 8-Reference), and were asked to locate the leaves corresponding to three data entries. This preparation task ensured that participants understood the visualization and the data mappings.

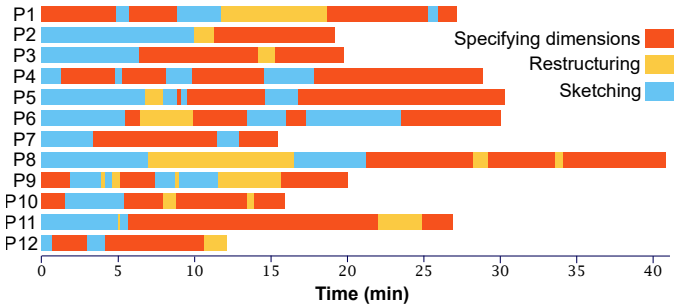

Fig. 7: Timelines for the main task of the reproduction study, showing how the 12 participants iterated between sketching branches, leaves, and applying color ●, editing the visualization structure, including the hierarchy, spines, and angles ●, and specifying data dimensions ●.

After following the video tutorial and familiarizing themselves with DataGarden's user interface, participants were then instructed to use the system to complete the main task while thinking aloud. During this process, the experimenter intervened only when participants were stuck or engaged in lengthy unnecessary actions.

The participants were given an interview where they were prompted to articulate their comprehension of the system's sketch structuring concepts, such as the visualization hierarchy, the data dimension types, and the visual channels. Sessions lasted 47 to 94 min (Median = 65.5).

Data collection. We recorded audio, and the screen of the pen display. We analysed audio transcripts together with the screen recordings.

6.1.2 Results

Some participants such as P10 chose to stick closely to the example, whereas others such as P1 took a creative and idiosyncratic approach (Fig. 8). We refer readers to our supplementary material for detailed results. Overall, most participants were able to reason with abstract data, grasped the concept of a visualization template, and correctly completed the task.

Authoring workflows. Participants took a variety of approaches to creating their visualization (Figure 7). Some drew the visualization elements in one go, before creating dimensions (P2, P3, P8, P10, P11). Others took an iterative approach (P1, P4, P5, P6, P7, P9, P12), creating dimensions between the activities of drawing branches, drawing leaves, and coloring leaves.

Some participants adjusted the hierarchical structure (P1, P2, P5, P8, P9, P11) and spatial interpretation (spines/angles) (P9), during or after drawing while others noted and corrected hierarchy (P6) and spatial interpretation (P3, P8, P10, P11, P12) issues when they bound the elements to dimensions. Some participants (P4, P7) made no corrections as the system interpreted their sketches to their satisfaction.

Participants were successful at reasoning in terms of creating a template by-example, with no data table to refer to per se. Some evidently created arbitrary, yet well-chosen data items on the fly (e.g., Figure 8-P9), whereas others more closely tried to reproduce data from the reference (e.g., Figure 8-P10). A few participants (P1, P9, P10) found DataGarden's unconventional workflow to be confusing: *"I'm doing data. I'll write down the data and then I'll do the drawing, but it's the reverse, so it's kind of tricky"* (P1). P9 also found that *"it's weird because it's both creating the data and doing the mapping at the same time. [...] As a designer, I would always work with a sample of data"* but added that this habit *"might be a deformation from working with computer scientists all the time and working with data sets."*

Dealing with sketch recognition issues. Ten of the 12 participants used DataGarden's tools to correct the system's inferred structure: seven participants restructured the tree hierarchy, six corrected the leaf angles, and one revised the spines. Corrections to the tree were mostly minor, except for P6 and P8. P6, who reported no previous experience with

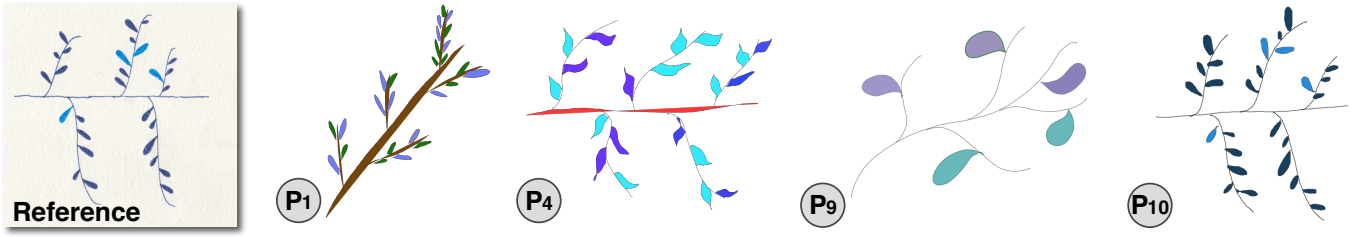

Fig. 8: Visualization from Dear Data [33] Week 27 by Georgia Lupi used as inspiration for the reproduction study (reference), and selection of DataGarden templates authored by participants.

data visualization, sketched axes and labels over their branches. Since DataGarden's prototype does not support free annotations, its sketch recognition mechanisms resulted in a messy tree. The participant tried to correct the problem: *"I don't know why I'm grouping them to the line. I just feel that it would be good to connect everything"* (P1). However, cleaning up the hierarchy proved to be challenging. Because the tree was not structured properly, the dimension settings did not make sense, so the participant was unable to recover and complete the task for all four data dimensions. In contrast, P8 was able to correct the issue but noted that it might pose more challenges for users without a technical background: *"I think because of my computer science background, I understand the meaning of normal and projections, so maybe for artists, maybe they won't be clear about this"* (P8).

Finally, participants made frequent use of cross-view highlighting to verify how the system structured their sketches, such as to verify glyphs when specifying mappings (6 participants) and their mapped positions (9 participants), their angles (8 participants), as well as the resulting table (5 participants). However, some participants complained that it was often difficult to distinguish which element was highlighted (P5, P11) and that highlighting colors were *"distracting"* (P7).

6.2 Free-form Study with Visualization Experts

To collect critical insight about the extent to which our approach (mis)aligns with experts' mental models, and prompt creative ideas that push the system's boundaries, we conducted a complementary study with data visualization experts on an open-ended task.

6.2.1 Method

Participants. We recruited four visualization researchers (E1–E4) from our direct network (convenience sample) across Europe and North America. The participants included two Computer Science faculty members, one senior research scientist in industry, and one post-doctoral fellow, with one participant having 5-10 years of visualization experience, and the remaining three having over 10 years of experience.

Apparatus. The study was conducted online, and participants used their personal equipment, typically consisting of a computer equipped with a mouse or trackpad, and Google Chrome to access DataGarden.

Task. We asked the experts to devise a template example that would test the limits of DataGarden, straying from canonical examples to avoid potential overfitting. While participants were invited to provide feedback on usability issues, it was explicitly stated that the study's objective was to gather their impressions on the concept of formalizing visualization design through template sketching, and to assess DataGarden's workflow as a possible instantiation of this concept.

Procedure. Participants connected to a video-conference call and filled out a questionnaire asking about demographics data and familiarity with sketch-based and hierarchical visualizations. Participants were then asked to watch a video tutorial (same as in our first study), and familiarize themselves with DataGarden's interface with a concrete example (the one used for the reproduction task). Once comfortable with how to operate the tool, they were asked to use it to design their own visualization. Since learning issues were not the focus of this study, participants were free to seek assistance from the experimenter to execute their vision.

After exploring and iterating on their template, participants were asked to discuss the concepts embodied in DataGarden, their impressions about the workflow, as well as ideas building on this approach. Each session lasted approximately 1.5 to 2 hours. Participants were offered a gift card for their time, although most declined it.

Data collection and analysis. Audio and video of the participant's shared screen were recorded. We conducted a thematic analysis of transcripts and screen recordings, using an affinity diagramming approach.

6.2.2 Results

The experts worked on a diverse set of visualization styles, covering nested layouts (E1, E2), relative color filling (E2), and traditional charts (E3). Figure 9 presents polished versions of the templates envisioned by the experts, as finalized by the experimenter.

Current sketching practice. All participants reported sketching on paper or whiteboard as part of their everyday practice to *"just brainstorm or ideate"* (E1), or to *"help people to understand what's going to happen"* (E3). Their process involves sketching from real or made up data. For instance, E2 described generating *"partial and even fictional representations"* when brainstorming a design, and for tasks such as iterating on a figure, *"it's more based on real data, but it's still not precise and not complete [...] I cherry pick some data points that gives some variety, but also gives an idea of the overall distributions."*

Sketching to externalize a data structure. E1 and E2 commented on the fact that while they did not start from an existing data table for their visualization example, the data structure was at least partially formed in their mind: *"you have a very good mental model of what the data looks like"* (E2). E1 reported that *"the data exists implicitly in your head"* and compared this workflow with the writing process: *"You know what you want to say [...] You don't know how to get there yet. But the process of writing allows you to better formalize that knowledge in your head."* As E4 further explained: *"I'm thinking about my data while I'm drawing. And I think this is really interesting [...]"*

Working with abstract vs. concrete data values. In most cases, the experts worked with unspecific, or even placeholder values, as they verbalized data dimensions and externalized specific instances on the canvas: *"I can give them random names, it doesn't really matter. Name0, Name1, that works."* (E1), and *"I'm not really specific on any of my real data [...] Let's just make it up"* (E4). E3 commented that working without real values as in DataGarden, *"I think I can be more expressive, I can be more creative."*

While working with true data values was not deemed necessary, the process of sketching still prompted the participants to think about plausible data ranges and distributions. For instance, E1 quickly felt the need to work with a concrete, yet arbitrary selection of existing songs. Similarly, after sketching a few circles to represent countries, E4 turned to the data table to "fix up" the values to match reality closer. Having the sketched template reflect real data was also noted as an important milestone for evaluating whether a design would scale: *"I would really like adding more data and see [because] when you scale it, you realize it just doesn't work right? It's overlapping 20 values."* (E3).

Specifying visualizations through templates. The experts appreciated the directness of creating representative examples on the canvas as a mechanism for specifying a target representation – *"I think it's really cool that it is generating the entries based on my sketch [...] like the more fun way of specifying a visualization"* (E1). Reporting on their strategies for creating templates, E2 explained: *"I draw multiple so*

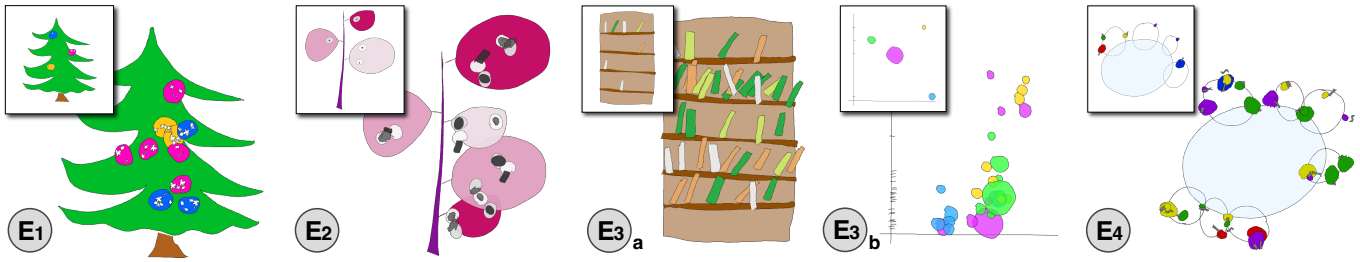

Fig. 9: Results from the expert study. Visualizations generated from templates (insets) created by participants – the templates were reproduced and further optimized by the authors. E1: Christmas carols represented as Christmas balls, whose color encodes the genre, and position in the tree the overall ranking (higher in the tree, higher ranked); snowflakes within the balls represent people rating the song (same shape = same person). E2: a whimsical activity tree whose branches represent different research (dark purple), teaching (pink), and service (light pink) projects; with each project comprising tasks (gray bubbles) with associated level of completion (dark gray filling). E3.a: Books (to) read on a given month (i.e. shelf); the tilting of the book indicates the number of pages read, and color encodes rating (low, medium, high). E3.b: A bubble chart inspired by GapMinder [46], showing countries' life expectancy (x) against income (y); the paper authors extended to add size to encode population, and color for continent. E4: A journey around the globe, where the central sphere represents Earth, each bubble around corresponds to a country where the participant lived (size = length of stay), and bubbles around countries indicate activities which the participant liked/disliked (color).

that I capture variation in the data," later adding: *"You draw a big one and a small one. You draw variation of color, these kind of things, to show a bit the the spread that you might have in your data set, and then it informs some meaningful specification based on that."* E3 described a similar conceptualization of templates: *"To me, a template is just a collection of all the extreme versions of all the marks for continuous data, and an enumeration of all of them when it's categorical."*

However, experts were somewhat frustrated that the system could not extrapolate beyond extreme (magnitude) and specific (identity) values that their marks captured. For instance, E1 and E3 proposed that the system should infer complete visual palettes from partial specifications, reducing manual effort: *"Say that I want five colors. I would not want to draw five elements [...] You want to draw one shape and say: "map a color" and just do it all at once and have a color"* (E1). Participants also expressed a need for intelligent mechanisms for shapes: *"It would be helpful that [the system] interpolates between the shapes trying to have something in between and maybe create more shapes like this. Yeah, I wish we had more possibilities to increasing diversity"* (E3).

Specifying custom dependencies and constraints. Besides explicit data-to-visual mapping rules, the experts expressed a desire to formalize other implicit graphical constraints. For example, E1 wished to indicate that color does not encode data: *"I want them to be randomized. Maybe I want them to be whimsical, right?"* The same expert also wanted the snowflakes to remain within their nesting snowball (Figure 9-E1). E2 sought to create a filling mark whose size is relative to the size of its parent to indicate a ratio (Figure 9-E2), while E4 wanted to describe property dependencies between parent and children: *"If I have something of a large workload, it would have to increase the size of the parent as well"* (E4).

Finally, the experts envisioned graphical specifications that could express implicit degrees of freedom and design constraints driven by aesthetic considerations. Interestingly, some experts had specific expectations regarding the level of intelligence the system should exhibit in populating the canvas with glyphs whose positions were not explicitly specified. They encountered instances of both success, e.g., books neatly placed on shelves (Figure 9-E3), and failure, e.g., uneven distribution of circles around the Earth (Figure 9-E4).

Iterating between visualizations and their templates. All participants alternated between sketching and specifying mappings, and then generating new visualization instances by adding new data. According to E4, *"it's easy to explore new ideas because you would just sketch"*, while E3 compared it favourably against coding, where *"alternatives are difficult to test."*

We observed that a clearer separation of the initial minimalist template from the more complete visualization might be beneficial. Participants experimented with applying the template to actual data and then reverting back to the template. While maintaining integrity between the canvas and the data table was acknowledged as a strength – *"the act of generating that table is, I think, one of the most powerful things about this tool"* (E1) – the process became more challenging when adding additional data to generate more glyphs: *"I would do a separation between my actual values and the values I use for the template [...] especially because when I add the data, then it becomes much more difficult to edit"* (E3).

6.3 Gallery

To evaluate the expressive capabilities of DataGarden, we curated a gallery of visualizations [1]. The gallery includes simple visualizations with two or three hierarchical levels that showcase the spectrum of visuals achievable with our approach, and others that demonstrate how drawing freeform spines (e.g. Figure 5) can support flexible layouts. It also includes two visualizations that test the scalability of our approach. Figure 1 illustrates encoding seven visual channels into multistroke glyphs. Another visualization encodes a year-long dataset of work activities. With larger datasets, we found creating the final version required iterating between tweaking the template and testing with data, as the full datasets had too many glyphs to directly manipulate.

We also reproduced three representative visualizations from Dear Data [33]: Posavec's Week 1, and Lupi's (Figure 3) and Posavec's Week 3 visualizations. Reproducing Posavec's Week 3 was straightforward as its structure aligned with DataGarden's canonical example, featuring a hierarchy with data entries placed on branches by time. Posavec's Week 1 and Lupi's Week 3 could only be partially reproduced due to their stacked branches, which are not explicitly supported.

7 Discussion and Future Work

Our evaluation confirms the effectiveness of freeform sketching of visualization templates as a means of personal and creative expression. We reflect on both the technical and conceptual contributions as well as the limitations inherent in our approach.

7.1 Support for Personal Expression

The two user studies and our gallery provide variety of expressive and idiosyncratic instances of visualizations. Our participants' comments also provide evidence that DataGarden preserves individual style and supports personal expression (**D2**). However, our sessions with experts also highlight several limitations of our current implementation.

Layouts and links. DataGarden is specifically designed for hierarchical structures. Its support for infinitely nested glyphs using spines distinguishes it from similar systems, such as DataInk [65], whose flat glyph design cannot deal with hierarchical representations, such as those in Lupi's visualization in Figure 3. However, as DataGarden does not currently incorporate intelligent layout mechanisms, it does not fully support the rich layouts specified in hand drawn sketches, and the generated distribution occasionally results in overlap, requiring tedious manual correction. Additionally, new glyph instances are simple copies

[1] https://datagarden-git.github.io/datagarden

of template glyphs, which while this retains the authors style, does not have the same organic appeal of fully hand drawn visualizations. We encourage future work to explore algorithms to achieve both organic layout and organic glyph variation. Promising avenues for the latter could include generative AI variations [68] or techniques like Parametric Drawing Tools [15]. In addition, several examples in Dear Data [33] include free-shaped links between glyphs. Those features require further support for sketch recognition and structuring, as well as developing new data mapping functions.

Axes, labels, legends. To interpret a DataGarden visualization, viewers may rely on their own memory (if they are the actual authors) or refer to the visualization specifications in the structure inspector. However, P6's attempt to sketch axes and labels directly onto the canvas underscores the need for additional, more direct mechanisms. Embedding interpretation elements such as axes, labels, and legends into sketch-based visualizations remains an unexplored area of research, posing numerous challenging questions. For instance, should authors be provided with tools to draw these elements alongside their visualization templates?

Beyond hierarchical visualizations. DataGarden demonstrates how our workflow supports the design of hierarchical visualizations, but its primary goal is to serve as a probe for thinking through how to support a visual-first, machine-supported approach to structuring sketches for visualization authoring. Our evaluation showed that DataGarden can support more than plant-based visualizations (e.g. bubble chart, Figure 9E3b), but it takes a certain mindset to make these designs hierarchy-compliant. For an application to directly support more types of visualizations the underlying structures (e.g. child vs container relations) need to match those of the desired visualization. As the number of possible underlying structures increases, so does the ambiguity of the authors intended structure. Designing interactions to resolve this ambiguity remains an open challenge for future work.

7.2 Getting the Idea Across with Few Examples

DataGarden enables visualization authors to use partial or fictional data to convey the essence of their design through a few examples. This approach presents both opportunities and challenges.

Working with partial or fictional data. Some participants found working without data liberating, as they could focus on the design rather than the data (**D1**, **D2**). This observation echoes visual artists who describe sketching as a means to *"focus on just getting the idea across, instead of obsessing and fine tuning every detail"* [3]. We also observed that sketching without data challenged some participants' habits and mindset. Further, we learned from the visualization experts that while making data "real" is not necessary, doing so regularly during the process helps ground one's thinking and test the design. Therefore, we see value in approaching the design problem from both directions. An area for future research is to explore dedicated support for effective switching between partial and complete or between envisioned and real data during visualization design. For instance, experts suggested keeping templates distinct from their real data instantiations, or providing previews of populated instances of a partial template design.

Specifying visualizations through examples. Our participants found value in creating a data schema using examples and were quick at picking up strategies to fully specify data ranges. However, they wanted the system to take our approach further through mechanisms for interpolating and extrapolating glyphs. Such mechanisms would be useful for template iteration, as well for generating visualizations that support more organic shapes. Potential solutions in this direction may include skeleton-based deformation [21], by-example shape synthesis techniques [16, 56], or stable diffusion [64]. Future work needs to explore whether such approaches could fit to an interactive authoring workflow, while still supporting authors' idiosyncratic drawing style.

7.3 Extracting and Formalizing Intent from Sketches

The driving concept of DataGarden, along with its core technical challenge, is to turn unstructured sketches into visualization templates. While DataGarden demonstrates the potential and challenges of generating sketch-based visualizations from templates, offering a complete and robust solution to the problem was beyond the scope of this work. We reflect on challenges involved in inferring the authors' intent, or conversely, in helping authors convey their intent to the system.

Support for explicit and implicit visual specifications. DataGarden allows authors to specify their data mappings by directly manipulating glyphs in the *Dimensions* view (**D4**). Participants appreciated this approach, especially for encoding the same data value with similar yet nonidentical items, adding a nice touch of diversity (**D2**). There are, however, other decisions which merit further examination, such as enabling authors to constrain their layouts (e.g., plot glyphs within an area, align glyphs, avoid collision) or specify high-level design constraints (e.g., randomize colors such that the final result is whimsical).

Beyond the explicit mappings, participants were many times pleased at the decisions made by the simple heuristics (**D3**) to fill-in the gaps for aesthetic choices (e.g., books did not defy laws of physics), but we also observed instances where they wished the system was smarter or had a better "design taste." More work is needed to achieve greater user control while maintaining the ease of freehand sketching. Offering functionality for explicit specifications not directly derived from data (e.g., for alignment [8]) could provide users with more control but may require additional effort. On the other hand, employing more advanced sketch recognition methods (e.g., using image embeddings for clustering [67]) or integrating tacit knowledge of good design practice [35] could improve the inference of design specifications but would still require authors to manually correct inference errors.

Gaps in the interpretation of free sketches. When designing a visualization system that supports partial specification, we have to accept that all sketch recognition algorithms will, at some point, make mistakes. We chose to address this challenge through direct manipulation tools that complement automatic recognition (**D3** and **D4**). Still, we observed that this approach may not suffice. If the user's mental model of what the system can interpret from a free sketch is erroneous—for example P6 assumed that DataGarden could distinguish between sketched visualization glyphs and annotations which was not the case—then, additional guidance is needed to make the direct manipulation effective. Additionally, as the size of the visualization scales, the interfaces for doing direct manipulation can become unwieldy. While DataGarden demonstrates the potential for the visual-first workflow, additional work is still needed to make visual data encoding explicit through interaction, and to manage such interactions at higher levels of complexity.

7.4 Risks and limitations

Visualization misuse. Personal visualizations like the plant-like ones we study may not necessarily result in functional (e.g., supporting data analysis) or "good" visualizations in terms of accurately conveying data. Our work targets personal expression through creative visualization but does not guard against potential misuse. In future research, it may be valuable to explore the integration of guidance for visual artists new to data visualization, particularly regarding common ethical and informational threats in visualization design [5].

Study participants. While participants in our evaluation had diverse expertise and experience with data visualization, they were recruited from our personal network (free-form study) and local student pool (reproduction study). Future studies should strive for broader diversity.

8 Conclusion

We studied a sketch-based structuring approach for authoring personal plant-like data visualizations. We introduced DataGarden, which implements a sketching-to-data approach to visualization creation. DataGarden enables authors to interactively structure their sketches into generalizable visualization templates that maintain a bidirectional connection between the sketched representations and underlying data tables. Results from our two user studies and accompanying gallery demonstrate the creative and expressive potential of our approach. However, they also point to limitations and future opportunities, such as expanding support for more diverse sketch representations, capturing users' implicit graphical constraints, and integrating advanced sketch recognition and shape synthesis techniques.

Acknowledgements

We would like to extend special thanks to Chenxi Liu and Silvia Lopez for the work on integrating StrokeStrip [39] and investigating StripMaker [28]. The work is supported by the CNRS-University of Toronto Ph.D. Mobility Funding program, the French National Research Agency under grant ANR-21-CE33-0002 GLACIS, and NSERC under grant RGPIN-2018-05072.

Supplementary Material

The supplementary materials, including the code of DataGarden, tutorials, the gallery, and the study apparatus for both studies can be found at datagarden-git.github.io/datagarden, also available as a zip file at osf.io/adymb.

References

[1] M. Beaudouin-Lafon and W. E. Mackay. Reification, polymorphism and reuse: three principles for designing visual interfaces. In *Proceedings of the working conference on Advanced visual interfaces*, pp. 102–109, 2000. doi: 10.1145/345513.345267 5

[2] A. Bigelow, S. Drucker, D. Fisher, and M. Meyer. Reflections on how designers design with data. In *Proc. AVI*, 2014. doi: 10.1145/2598153.2598175 1, 2, 3

[3] N. Bremer and S. Wu. *Data Sketches: A journey of imagination, exploration, and beautiful data visualizations*. AK Peters/CRC Press, 2021. 1, 2, 3, 9

[4] N. Bressa, J. Louis, W. Willett, and S. Huron. Input visualization: Collecting and modifying data with visual representations. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24, article no. 499, 18 pages. Association for Computing Machinery, New York, NY, USA, 2024. doi: 10.1145/3613904.3642808 3

[5] A. Cairo. *How charts lie: Getting smarter about visual information*. WW Norton & Company, 2019. 9

[6] W. O. Chao, T. Munzner, and M. van de Panne. Rapid pen-centric authoring of improvisational visualizations with NapkinVis. In *Proc. InfoVis Posters*, 2010. 2

[7] F. Chevalier, P. Dragicevic, and C. Hurter. Histomages: fully synchronized views for image editing. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, 6 pages, p. 281–286. Association for Computing Machinery, New York, NY, USA, 2012. doi: 10.1145/2380116.2380152 3

[8] M. Ciolfi Felice, N. Maudet, W. E. Mackay, and M. Beaudouin-Lafon. Beyond snapping: Persistent, tweakable alignment and distribution with stickylines. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pp. 133–144, 2016. doi: 10.1145/2984511.2984577 9

[9] S. Consolvo, D. W. McDonald, T. Toscos, M. Y. Chen, J. Froehlich, B. Harrison, P. Klasnja, A. LaMarca, L. LeGrand, R. Libby, I. Smith, and J. A. Landay. Activity sensing in the wild: a field trial of ubifit garden. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, 10 pages, p. 1797–1806. Association for Computing Machinery, New York, NY, USA, 2008. doi: 10.1145/1357054.1357335 3

[10] L. Grammel, M. Tory, and M.-A. Storey. How information visualization novices construct visualizations. *IEEE transactions on visualization and computer graphics*, 16(6):943–952, 2010. doi: 10.1109/TVCG.2010.164 3

[11] S. Heller and R. Landers. *Raw Data: Infographic Designers' Sketchbooks*. Thames & Hudson, 2014. 2

[12] S. Huron, S. Carpendale, A. Thudt, A. Tang, and M. Mauerer. Constructive visualization. In *Proceedings of the 2014 conference on Designing interactive systems*, pp. 433–442. ACM, Vancouver BC Canada, June 2014. doi: 10.1145/2598510.2598566 2

[13] S. Huron, Y. Jansen, and S. Carpendale. Constructing Visual Representations: Investigating the Use of Tangible Tokens. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2102–2111, Dec. 2014. Conference Name: IEEE Transactions on Visualization and Computer Graphics. doi: 10.1109/TVCG.2014.2346292 2

[14] S. Huron and W. Willett. Visualizations as data input? In *IEEE Conference on Visualization and Visual Analytics (alt.VIS)*, 2021. doi: 10.11575/PRISM/39323 3, 5

[15] J. Jacobs, J. Brandt, R. Mech, and M. Resnick. Extending manual drawing practices with artist-centric programming tools. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pp. 1–13, 2018. doi: 10.1145/3173574.3174164 9

[16] E. Kalogerakis, S. Chaudhuri, D. Koller, and V. Koltun. A probabilistic model for component-based shape synthesis. *ACM Trans. Graph.*, 31(4), article no. 55, 11 pages, jul 2012. doi: 10.1145/2185520.2185551 9

[17] D. F. Keefe, D. Acevedo, J. Miles, F. Drury, S. M. Swartz, and D. H. Laidlaw. Scientific sketching for collaborative vr visualization design. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):835–847, 2008. doi: 10.1109/TVCG.2008.31 2

[18] G. Khan. *Drawing Data with Kids: Cultivating Data-Literacy: A Screen-Free Journey Through the Art of Visualization for Kids*. Gulrez Khan, 2023. 2

[19] N. W. Kim, N. Henry Riche, B. Bach, G. Xu, M. Brehmer, K. Hinckley, M. Pahud, H. Xia, M. J. McGuffin, and H. Pfister. DataToon: Drawing dynamic network comics with pen+ touch interaction. In *Proc. ACM CHI*, 2019. doi: 10.1145/3290605.3300335 2

[20] N. W. Kim, H. Im, N. Henry Riche, A. Wang, K. Gajos, and H. Pfister. Dataselfie: Empowering people to design personalized visuals to represent their data. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, 12 pages, p. 1–12. Association for Computing Machinery, New York, NY, USA, 2019. doi: 10.1145/3290605.3300309 2

[21] N. W. Kim, E. Schweickart, Z. Liu, M. Dontcheva, W. Li, J. Popovic, and H. Pfister. Data-driven guides: Supporting expressive design for information graphics. *IEEE transactions on visualization and computer graphics*, 23(1):491–500, 2016. doi: 10.1109/TVCG.2016.2598620 1, 2, 9

[22] Y.-S. Kim, N. Henry Riche, B. Lee, M. Brehmer, M. Pahud, K. Hinckley, and J. Hullman. Inking your insights: Investigating digital externalization behaviors during data analysis. In *Proc. ACM ISS*, 2019. doi: 10.1145/3343055.3359714 2, 3

[23] L. Koeman, V. Kalnikaité, and Y. Rogers. "everyone is talking about it!": A distributed approach to urban voting technology and visualisations. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, 10 pages, p. 3127–3136. Association for Computing Machinery, New York, NY, USA, 2015. doi: 10.1145/2702123.2702263 3

[24] B. Lee, R. H. Kazi, and G. Smith. SketchStory: Telling more engaging stories with data through freeform sketching. *IEEE TVCG (Proc. InfoVis)*, 19(12), 2013. doi: 10.1109/TVCG.2013.191 2

[25] B. Lee, G. Smith, N. H. Riche, A. Karlson, and S. Carpendale. SketchInsight: Natural data exploration on interactive whiteboards leveraging pen and touch interaction. In *Proc. IEEE PacificVis*, 2015. doi: 10.1109/PACIFICVIS.2015.7156378 2

[26] G. Li, M. Tian, Q. Xu, M. J. McGuffin, and X. Yuan. Gotree: A grammar of tree visualizations. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, 13 pages, p. 1–13. Association for Computing Machinery, New York, NY, USA, 2020. doi: 10.1145/3313831.3376297 5

[27] H. Lin, D. Akbaba, M. Meyer, and A. Lex. Data hunches: Incorporating personal knowledge into visualizations. *IEEE TVCG (Proc. VIS)*, 29(1), 2023. doi: 0.1109/TVCG.2022.3209451 3

[28] C. Liu, T. Aoki, M. Bessmeltsev, and A. Sheffer. Stripmaker: Perception-driven learned vector sketch consolidation. *ACM Transactions on Graphics (TOG)*, 42(4):1–15, 2023. doi: 10.1145/3592130 2, 4, 5, 10

[29] C. Liu, E. Rosales, and A. Sheffer. Strokeaggregator: Consolidating raw sketches into artist-intended curve drawings. *ACM Transaction on Graphics*, 37(4), 2018. doi: 10.1145/3197517.3201314 2, 4, 5

[30] Z. Liu, J. Thompson, A. Wilson, M. Dontcheva, J. Delorey, S. Grigg, B. Kerr, and J. Stasko. Data Illustrator: Augmenting vector design tools with lazy data binding for expressive visualization authoring. In *Proc. ACM CHI*, 2018. doi: 10.1145/3173574.3173697 2, 3

[31] G. Lupi. Data humanism, the revolution will be visualized. 1

[32] G. Lupi. Sketching with data opens the mind's eye. *National Geographic*, 2015. 1, 2

[33] G. Lupi and S. Posavec. *Dear Data*. Princeton Architectural Press, 2016. `https://www.dear-data.com/by-week`. 1, 2, 3, 6, 7, 8, 9

[34] G. Lupi and S. Posavec. *Observe, Collect, Draw!: A Visual Journal : Discover the Patterns in Your Everyday Life*. Princeton Architectural Press, 2018. 1, 2, 3

[35] 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 visualiza-*

tion and computer graphics, 25(1):438–448, 2018. doi: 10.1109/TVCG.2018.2865240 9

[36] T. Munzner. Marks and channels. *Visualization Analysis and Design*, pp. 94–114, 2014. 5

[37] B. A. Myers, J. Goldstein, and M. A. Goldberg. Creating charts by demonstration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '94, 6 pages, p. 106–111. Association for Computing Machinery, New York, NY, USA, 1994. doi: 10.1145/191666.191715 1, 2, 3

[38] A. Offenwanger, M. Brehmer, F. Chevalier, and T. Tsandilas. Timesplines: Sketch-based authoring of flexible and idiosyncratic timelines. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):34–44, 2024. doi: 10.1109/TVCG.2023.3326520 2, 3

[39] D. Pagurek van Mossel, C. Liu, N. Vining, M. Bessmeltsev, and A. Sheffer. Strokestrip: Joint parameterization and fitting of stroke clusters. *ACM Transactions on Graphics*, 40(4), 2021. doi: 10.1145/3450626.3459777 2, 5, 10

[40] S. Projects. bit planner. `https://specialprojects.studio/project/bit-planner`. 3

[41] D. Ren, B. Lee, and M. Brehmer. Charticulator: Interactive construction of bespoke chart layouts. *IEEE TVGC (Proc. InfoVis)*, 25(1), 2018. doi: 10.1109/TVCG.2018.2865158 1, 2

[42] D. Ren, B. Lee, M. Brehmer, and N. H. Riche. Reflecting on the evaluation of visualization authoring systems: Position paper. In *Proc. IEEE Evaluation and Beyond (BELIV)*, 2018. doi: 10.1109/BELIV.2018.8634297 6

[43] J. C. Roberts, C. J. Headleand, and P. D. Ritsos. *Five design-sheets: creative design and sketching for computing and visualisation*. Springer, 2017. doi: 10.1007/978-3-319-55627-7 1

[44] H. Romat, N. Henry Riche, K. Hinckley, B. Lee, C. Appert, E. Pietriga, and C. Collins. Activeink: (th)inking with data. In *Proc. ACM CHI*, 2019. doi: 10.1145/3290605.3300272 2

[45] H. Romat, N. Henry Riche, C. Hurter, S. Drucker, F. Amini, and K. Hinckley. Dear pictograph: Investigating the role of personalization and immersion for consuming and enjoying visualizations. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, 13 pages, p. 1–13. Association for Computing Machinery, New York, NY, USA, 2020. doi: 10.1145/3313831.3376348 2

[46] H. Rosling. Data - gapminder.org. `http://www.gapminder.org/data/`, 2012. 8

[47] P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987. doi: 10.1016/0377-0427(87)90125-7 6

[48] B. Saket, H. Kim, E. T. Brown, and A. Endert. Visualization by demonstration: An interaction paradigm for visual data exploration. *IEEE transactions on visualization and computer graphics*, 23(1):331–340, 2016. doi: 10.1109/TVCG.2016.2598839 3

[49] N. Saquib, R. H. Kazi, L.-y. Wei, G. Mark, and D. Roy. Constructing embodied algebra by sketching. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, article no. 428, 16 pages. Association for Computing Machinery, New York, NY, USA, 2021. doi: 10.1145/3411764.3445460 2

[50] A. Satyanarayan and J. Heer. Lyra: An interactive visualization design environment. In *Computer graphics forum*, vol. 33, pp. 351–360. Wiley Online Library, 2014. doi: 10.1111/cgf.12391 2

[51] A. Satyanarayan, B. Lee, D. Ren, J. Heer, J. Stasko, J. Thompson, M. Brehmer, and Z. Liu. Critical reflections on visualization authoring systems. *IEEE TVCG (Proc. InfoVis)*, 26(1), 2019. doi: 10.1109/TVCG.2019.2934281 2, 3, 5

[52] D. Schroeder, D. Coffey, and D. Keefe. Drawing with the flow: A sketch-based interface for illustrative visualization of 2d vector fields. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*, pp. 49–56, 2010. doi: /10.2312/SBM/SBM10/049-056 2

[53] D. Schroeder and D. F. Keefe. Visualization-by-sketching: An artist's interface for creating multivariate time-varying data visualizations. *IEEE transactions on visualization and computer graphics*, 22(1):877–885, 2015. doi: 10.1109/TVCG.2015.2467153 2

[54] J. Snyder, E. Murnane, C. Lustig, and S. Voida. Visually encoding the lived experience of bipolar disorder. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, 14 pages, p. 1–14. Association for Computing Machinery, New York, NY, USA, 2019. doi: 10.1145/3290605.3300363 3

[55] A. Thudt, U. Hinrichs, S. Huron, and S. Carpendale. Self-reflection and personal physicalization construction. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, 13 pages, p. 1–13. Association for Computing Machinery, New York, NY, USA, 2018. doi: 10.1145/3173574.3173728 3

[56] Y. Tian, A. Luo, X. Sun, K. Ellis, W. T. Freeman, J. B. Tenenbaum, and J. Wu. Learning to infer and execute 3d shape programs. In *International Conference on Learning Representations*, 2019. 9

[57] T. Tsandilas. StructGraphics: Flexible visualization design through data-agnostic and reusable graphical structures. *IEEE TVCG (Proc. VIS)*, 27(2), 2021. doi: 10.1109/TVCG.2020.3030476 3

[58] T. Tsandilas, A. Bezerianos, and T. Jacob. SketchSliders: Sketching widgets for visual exploration on wall displays. In *Proc. ACM CHI*, 2015. doi: 10.1145/2702123.2702129 2

[59] T. Tsandilas, C. Letondal, and W. E. Mackay. Musink: composing music through augmented drawing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, 10 pages, p. 819–828. Association for Computing Machinery, New York, NY, USA, 2009. doi: 10.1145/1518701.1518827 2, 3

[60] J. Walny, S. Huron, and S. Carpendale. An exploratory study of data sketching for visual representation. *Computer Graphics Forum (Proc. EuroVis)*, 34(3), 2015. doi: 10.1111/cgf.12635 2

[61] J. D. Wammes, M. E. Meade, and M. A. Fernandes. The drawing effect: Evidence for reliable and robust memory benefits in free recall. *Quarterly Journal of Experimental Psychology*, 69(9):1752–1776, 2016. PMID: 26444654. doi: 10.1080/17470218.2015.1094494 3

[62] C. Wang, Y. Feng, R. Bodik, A. Cheung, and I. Dillig. Visualization by example. *Proc. ACM Program. Languages*, 4(POPL), article no. 49, 28 pages, dec 2019. doi: 10.1145/3371117 3

[63] J. Wood, P. Isenberg, T. Isenberg, J. Dykes, N. Boukhelifa, and A. Slingsby. Sketchy rendering for information visualization. *IEEE TVCG (Proc. InfoVis)*, 18(12), 2012. doi: 10.1109/TVCG.2012.262 3

[64] J. Wu, J. J. Y. Chung, and E. Adar. viz2viz: Prompt-driven stylized visualization generation using a diffusion model. *ArXiv*, abs/2304.01919, 2023. doi: 10.48550/arXiv.2304.01919 9

[65] H. Xia, N. Henry Riche, F. Chevalier, B. De Araujo, and D. Wigdor. DataInk: Direct and creative data-oriented drawing. In *Proc. ACM CHI*, 2018. doi: 10.1145/3173574.3173797 1, 2, 8

[66] H. Xia, K. Hinckley, M. Pahud, X. Tu, and B. Buxton. Writlarge: Ink unleashed by unified scope, action, & zoom. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, 14 pages, p. 3227–3240. Association for Computing Machinery, New York, NY, USA, 2017. doi: 10.1145/3025453.3025664 2, 3

[67] J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, 10 pages, p. 478–487. JMLR.org, 2016. 9

[68] X. Xu, Z. Wang, G. Zhang, K. Wang, and H. Shi. Versatile diffusion: Text, images and variations all in one diffusion model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7754–7765, 2023. doi: 10.1109/ICCV51070.2023.00713 9

[69] J. E. Zhang, N. Sultanum, A. Bezerianos, and F. Chevalier. DataQuilt: Extracting visual elements from images to craft pictorial visualizations. In *Proc. ACM CHI*, 2020. doi: 10.1145/3313831.3376172 2

[70] J. Zong, D. Barnwal, R. Neogy, and A. Satyanarayan. Lyra 2: Designing interactive visualizations by demonstration. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):304–314, 2020. doi: 10.1111/cgf.12391 1