DataQuilt: Extracting Visual Elements from Images to Craft Pictorial Visualizations

Jiayi Eris Zhang, Nicole Sultanum, Anastasia Bezerianos, Fanny Chevalier

DataQuilt: Extracting Visual Elements from Images to Craft Pictorial Visualizations

Jiayi Eris Zhang[1] **Nicole Sultanum**[1] **Anastasia Bezerianos**[2] **Fanny Chevalier**[1]
[1] Department of Computer Science, University of Toronto
[2] Université Paris-Saclay, CNRS, Inria, LRI
{eriszhang | nicolebs | fanny} @dgp.toronto.edu anastasia.bezerianos@lri.fr

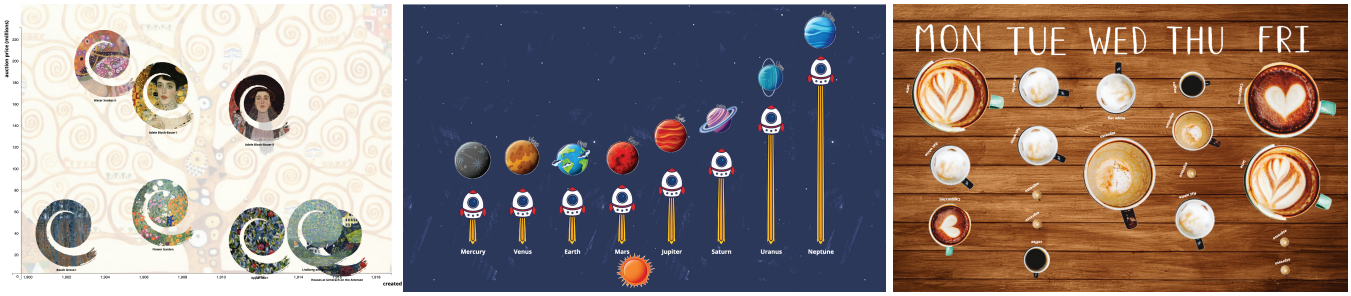

Figure 1. Examples of creations realized with DataQuilt, a new interactive authoring tool that allows authors to borrow visual and stylistic elements from raster images and re-purpose them to create custom, pictorial visualizations. Left: a scatterplot of famous paintings by Klimt, showing the date of creation (x-axis) against how much it was sold for in auction (y-axis). Each data point is a spiral-shaped glyph whose texture is mapped to the painting it represents, whereas the *Tree of Life* painting is used as a decorative background. Middle: a bar chart representing the distance from the sun for the planets of our solar system. Each data point is represented by a space rocket, whose exhaust flames are stretched according to the underlying data. Decorative glyphs (sun, planets) are used for further information and visual appeal. Right: A personal visualization depicting one's coffee intake over a week. The type of coffee (espresso, latte, etc.) is represented by different coffee cups, all extracted from photographs. The orientation of the handle represents the time, whereas size is proportional to the drink size and horizontal position corresponds to the day of the week.

ABSTRACT

Recent years have seen an increasing interest in the authoring and crafting of personal visualizations. Mainstream data analysis and authoring tools lack the flexibility for customization and personalization, whereas tools from the research community either require creativity and drawing skills, or are limited to simple vector graphics. We present DataQuilt, a novel system that enables visualization authors to iteratively design pictorial visualizations as collages. Real images (*e.g.*, paintings, photographs, sketches) act as both inspiration and as a resource of visual elements that can be mapped to data. The creative pipeline involves the semi-guided extraction of relevant elements of an image (arbitrary regions, regular shapes, color palettes, textures) aided by computer vision techniques; the binding of these graphical elements and their features to data in order to create meaningful visualizations; and the iterative refinement of both features and visualizations through direct manipulation. We demonstrate the usability of DataQuilt in a controlled study and its expressiveness through a collection of authored visualizations from a second open-ended study.

Author Keywords

pictorial visualization; creativity; graphic design; collage

CHI '20, April 25–30, 2020, Honolulu, HI, USA.

ACM ISBN 978-1-4503-6708-0/20/04 ...$15.00.
http://dx.doi.org/10.1145/3313831.3376172

> *"Did you know that everything you see and like can become* design material *for* your data*?"*
>
> — G. Lupi, S. Posavec [26]

INTRODUCTION

Giorgia Lupi and Stefanie Posavec's quote encourages us to be inspired by the patterns and images around us when constructing our visualizations. It is a call to the increasing number of novice visualization enthusiasts that actively engage with their own data, from data collection to visualization authoring [44]. While not necessarily artistically inclined, these casual users seek to craft creative, pleasing, relatable, and memorable visual representations that convey messages and present insights.

However, the needs of such visualization authors are not well aligned with the goals of traditional visualization tools that focus on exploration and analysis [21, 51]. Advanced visualization systems like Tableau [4] support traditional graphs and charts, such as scatterplots and bar charts, but lack flexibility when it comes to crafting creative, custom-made visualizations. Design tools such as sketching or vector graphics editors like Illustrator [1] enable such power but require a certain level of artistic ability and tool expertise to make a realization look compelling. Moreover, maintaining *data integrity* [22] between the graphical elements and the underlying data in such tools is most often a manual process, which is tedious, time consuming, and error prone [7, 8, 24].

Closer to Lupi and Posavec's vision are recent tools from the visualization community for creating custom-made visualizations. Greater expressiveness is achieved through sketch-

ing [21, 51], or access to non-traditional graphics and chart templates [35]. But they either require drawing skills to generate visually appealing results [21, 51], are restricted to vector graphics [22, 24, 35] or specific chart designs [47], or do not allow easy exploration of design alternatives [39].

We move closer to leveraging *"everything we see around us as design material"* for data visualizations, by using raster images as both *inspiration* and as a *source* of glyphs and visual features that can be mapped to data. Our approach supports *pictorial visualizations* — visual representations of data that transcend traditional visualizations, allowing for greater freedom of visual expression through a more experimental, organic approach to their graphical language. This includes custom-made infographics [14], handcrafted visualizations [25], and data art [46].

DataQuilt is an interactive authoring tool that allows visualization authors to borrow components and stylistic features from raster images (*e.g.*, sketches, paintings, photographs) and repurpose them as elements of graphical language, to encode and decorate data in a style inspired by the original image.

We want to empower casual users, i.e., typical office worker that are not experienced or skilled in digital illustration or visualization, to create expressive visualizations, that no other tool alone currently enables. With DataQuilt visualization authors can extract elements with various shapes to serve as creative building blocks for their visualization design (glyphs), as well as color palettes and textures which are then easily mapped to data. They can also try out different designs, by altering and interchanging these elements and their layout through simple drag-and-drop operations.

Our contributions include: (1) *A means to easily repurpose raster images for data visualization*: a semi-guided extraction of relevant features of an image (arbitrary regions, regular forms, color palettes, textures), aided by computer vision techniques, allows authors to easily extract features from images. These elements can be edited and refined through direct manipulations interactions, to turn them into reusable glyphs. (2) *Data binding*: multiple ways of binding the extracted glyphs' graphical properties to data can be used to create meaningful visualizations of different layouts. (3) *A streamlined iterative process*: we allow visualization authors to smoothly iterate and refine the extracted components and visualization design.

Our approach is validated in several ways. First, a series of workshops helped identify *design goals* for pictorial visualizations. Second, we evaluated the *usability* of the tool and the ability of novice users to use different features to author and revise visualizations, through a set of controlled tasks. Third, we explored the *expressiveness* of our tool and its potential to support pictorial visualization authoring in an a set of open-ended tasks and through a gallery of examples. Our results show that visually-rich creations can be generated and further refined in a few steps by novice users, highlighting that as a creativity tool, DataQuilt supports both easy entry for novice users, and the ability to generate image-based, sophisticated outcomes not otherwise possible with traditional tools [42]. Materials are available at `http://dataquilt.github.io`.

RELATED WORK

We review works that discuss enriched visualizations and tools facilitating their authoring, as well as the use of image manipulation and computer vision in visualization.

Value of visually enriched visualizations

Visualization authors who are well versed in Tufte's school of thought [45] would generally agree with the *data-ink* principle, that states that the amount of ink devoted to displaying data information should be maximized. Practical implications of this principle include perceptual effectiveness [17, 48], which tends to be the primary focus of most visualization designs. Visualizations that include embellishments (or 'chartjunk') are often seen as inferior from a pragmatic standpoint, because they are believed to be ineffective, if not counter-productive, in terms of visual effectiveness. Similarly, while appreciated for their aesthetic appeal, artistic visualizations remain relatively unexplored in visualization science [23, 46].

Tufte has a more nuanced take on the matter than generally believed; his book's epilogue states *"the principles should not be applied rigidly [...] it is better to violate any principle than to place graceless or inelegant marks on paper"*, suggesting that there is more to visualization design than the decoding of values and identification of patterns [14, 46]. Greater aesthetic quality of a visual representation can indeed play a role in information intake [9], by increasing engagement [19, 29], comprehension [6] and memorability [10, 11]. The popularity of artistic projects such as Dear Data [25], the countless custom-made personal visualizations that can be found online [44], and the long-running success of venues such as the IEEE VIS Arts Program also suggest that there is great value in less pragmatic-focused, more artistic-based visualizations.

Visualization authoring tools

In line with a historical focus on visualizations designed primarily for analytical purposes, mainstream tools such as Tableau [4] and PowerBI [2], and research prototypes such as Voyager [50] excel at facilitating the production of traditional charts; however, these tools lack the flexibility to support the authoring of unconventional, creative visualizations, like the pictorial visualizations enabled by DataQuilt (Fig. 1).

In contrast, programming toolkits and libraries enable virtually endless expressivity. Widely adopted examples include d3 [12], Vega [5], Vega-lite [40], ggplot [49] and Processing [3]. While powerful, these require to use a textual (imperative or declarative) language, which demands a high level of programming skills to create custom visualizations. Instead, we strive to empower casual users with no coding experience to create their own visualizations easily.

Design tools such as Illustrator [1] also enable high flexibility, but provide poor support for preserving data-to-visual integrity, which has been consistently reported to be tedious, and error prone in previous studies [7, 8, 22]. To achieve both effectiveness in creating graphics bound to data, while achieving visual aesthetics of superior quality, practitioners typically have resorted to a segmented workflow combining both classes of tools [7], which can hinder the creative workflow.

Our approach lies in between these two extremes, enabling to reduce the gap between free-form visual expression enabled by unconstrained design tools, and data-driven constraints and guides that allow to preserve visual-to-data integrity. We strive to best support the creative process for casual users, with a tool that has both a *low threshold* (how difficult it is to learn how to use the system), and a *high ceiling* (how much can be done using the tool) [30, 42].

There has been a growing interest in the visualization community to achieve a similar goal, which resulted in tools that Satyanarayan et al. refer to as *visual builders* [39]:

Infonice [47] allows PowerBI users to customize glyphs via the Infographics Design plug-in to create isotypes-like charts, but requires familiarity with PowerBI to start with. Lyra [38] and Charticulator [35] provide users with fine control over the characteristics of pre-defined visual marks and layouts, and allow high expressivity by supporting glyph composition and collections. However, they are restrictive in the workflow, as all of the visual content is generated via the specification of visual rules in different panels of complicated interfaces [39]. For instance, in neither of these tools can the user interchange glyphs without starting a visualization from scratch, hindering experimentation. Nor easily use different glyphs to encode different categories, hindering the creation of custom designs. There are ways around the latter, e.g., the user can create multiple pipelines in Lyra or PlotSegments in Charticulator, but this requires the creation of several independent visualizations that then need to be to manually combined and synchronized.

Liu et al. [24] take a different approach, by considering the visualization as free-form vector graphics that do not follow a particular template or rules. They introduce the notion of *lazy data binding*, that consists of applying data encodings only when necessary. They propose Data Illustrator, which allows for greater visual expression and flexibility than Lyra and Charticulator via its support for the creation of custom vector shapes and direct manipulation of graphical elements in the canvas. But as the previous tools, the interface can be intimidating to novice users, and while it is possible to refine glyphs in a dedicated glyph editor, again only one glyph design can be applied across all data points. And again substituting a glyph design with another existing glyph requires re-creating the visualization from scratch.

We build on the approach of lazy binding, but strive for a more flexible workflow: as the authors of the above systems acknowledge, "While [Lyra, Charticulator and DataIllustrator] present several different entry points to author a given design, once one starts down a particular path it can be very difficult to make a change and often involves starting from scratch" [39]. In contrast, we facilitate quick and easy iterative design by allowing users to easily substitute glyphs and bind/unbind visual variables with simple drag-and-drop operations.

Data-Driven Guides [22], precursor of Data Illustrator, are another example of a powerful tool for lazy data binding. Virtual guides, whose properties are bound to data, are used to guide the drawing of vector shapes by measuring and snapping to the guides. One nice property is that they allow for the re-purposing of existing custom vector graphics, whose elements can be easily resized according to data. This contrasts with previously mentioned tools, that require users to design glyphs from scratch, using simple geometrical shapes. Data-Driven Guides are however restricted to size, width and length visual properties. We go several steps further, by: allowing users to borrow arbitrary-shaped, closed glyphs, textures and colour palettes from any visual source; and supporting more bindings, including texture, and repeating a sub-portion of a glyph.

While they enable some degree of expressivity, most of the above tools remain limited to specific set of operations on vector graphics. DataInk [51] and DataSelfies [21] leverage and integrate sketching instead, as a more direct way of composing free-form graphical elements that would form the basis for glyph-based visualizations. DataToon [20] also uses sketch-based input for creating comic visualizations. These works, however, require a significant degree of artistic skill and creativity to get started. We take a different approach, and allow the user to borrow elements from any existing image, and re-purpose them to create pictorial visualizations.

In summary, there has been immense progress in recent years, to bridge the gap between creating beautiful, custom-made visuals and support data-driven visual authoring. Our work adds to these efforts, by contributing an *image-based visualization authoring tool* with a rich and flexible data binding interaction paradigm. Our approach enables expressive visualization design, by borrowing visual elements from existing art and photographs, and easily iterating over visualization design in a unified and simplified workflow consolidated into a cohesive authoring tool.

Visualization and image processing

Unlike other authoring systems, DataQuilt builds on the idea of borrowing and collaging visual elements from existing raster images. This approach involves the use of image manipulation and computer vision techniques to support the semi-guided extraction and manipulation of visual elements. Computer vision methods have been previously used in visualization, though for different purposes such as automatically extracting and retargeting colour mappings [41, 34], and visualization encodings [33, 28] from raster images. Image manipulations have also been applied in visualization, for instance in Transmogrifiers [13] to distort regions of an image into more meaningful visualizations.

We are also starting to see the rise of advanced techniques for the automated design of infographics and visualizations. These include automated generation of simple infographics from predefined styles adapted to data facts in a textual form [18], the automated extraction of a timeline visual template from a raster image in order to extend it [16], or the manipulation of mountain photos as line charts [32].

The above approaches consist of automatically creating visualizations, therefore removing the user from the creative process. In contrast, we integrate semi-automated image manipulation techniques, such as contour removal [37] and smooth texture repetition [15], as a support for casual users when authoring expressive, image-based, pictorial visualizations.

Figure 2. Examples from the workshop visualizations: (1-3) original image sources; (a-f) participant-produced visualizations.

REPURPOSING IMAGE COMPONENTS

Our goal is to help casual users and enthusiasts create visualization collages, by repurposing components from raster images (paintings, photographs, sketches) as inspiration and as source material for crafting custom-made pictorial visualizations. We expect this creative process to be highly iterative, requiring trial and error to reach a desired aesthetic outcome. We thus started out with the following high-level design goals:

- Lowering creativity barriers by allowing authors to repurpose elements from raster images as visualization components, instead of creating them from scratch or being limited to a few pre-set marks (**G1**);
- Support persistent data binding between elements and data while still allowing for changes to the visual elements (e.g., resizing) and the visualization (e.g., re-layout) (**G2**);
- Facilitate fast iterations and smooth transitions between the manipulation of the visual elements and that of the visualization (e.g., swap glyphs), in a fluid workflow (**G3**).

These goals empower casual users, with no experience in coding or using image editing tools, to create custom-made pictorial visualizations that no other tool currently supports, and experiment with different visualization designs.

We set out to understand what components from existing images people may want to use when constructing new visualizations, and how they would put them together. To this end we conducted two workshops that informed the remaining design goals for our approach.

Workshops on visualization collages

We are inspired by Lupi and Posavec's workshops and their book series, where artistic images as well as the world around us can act as inspiration for visualization design [25]. We conducted two 1.5-hour long workshops with 20 people in our institution. Participants entailed HCI and computer graphics, but no experience in creating personalized visualizations. We provided them with: (i) a spreadsheet featuring a dataset of characters from Star Wars, with 18 data rows and 8 dimensions (both numerical and categorical). And (ii) sample images, including abstract paintings and image sets with clear and well-separated shapes (Fig. 2(1-3)). Participants were asked to craft on paper creative visualizations by "stealing and stitching" graphical elements from the sample images, and later indicating what elements they were inspired from.

We collected 69 sketches, analyzed to inform aspects of the sample images that inspired or were used in the visualizations. Figure 2 shows visualization examples and original samples that inspired them.

We observed that participants often extracted *closed shapes* from the original image that acted as inspiration in their visualizations, both from the image set and the abstract paintings (see arrows in Fig. 2). We refer to these forms as *glyphs* henceforth. The shape of these glyphs was either used as-is or their visual aspects were adjusted to encode different information, e.g., color adjustment on Figure 2f.

One common data mapping participants performed was to adjust the size of the glyph with respect to a numerical dimension in the dataset. We observed three variants. In the first, the glyph was uniformly scaled (like Figure 2f). In the second, the glyph was replicated, similar to [47], to form bars in a bar chart visualization (Fig. 2b). In the third, again for bar charts, the height of the glyph was stretched (Fig. 2a).

We noted that for the two bar chart scaling types, participants sometimes applied the repetition and stretching to only part of the glyph (Fig. 2a), indicating that the notion of glyph is broader than a simple data element: *part of the glyph is decoration* and stays constant, and *part of it encodes data* and is stretched or repeated. This division was also seen in other types of mapping (beyond scale). For example part of the glyph maintained the original color and another part was colored based on a specific data property (Fig. 2f). Or when additional visual elements were added on the glyph and mapped to data properties (Fig. 2c and Fig. 2d).

Color or texture pattern inspiration, in particular from abstract images was another common element. Participants would decorate their visualization marks (bars or circles) with textures or colors captured from the image. In some cases textures were edited to encode another dimension or differentiate data points, e.g., slight changes in their color (Fig. 2e). We also noted cases where these colors and textures were applied to marks that did not follow the traditional shapes (e.g., bars or circles), but were rather more fluid forms (Fig. 2e).

These observations led to three additional design goals:

- Allow authors to easily extract both closed forms and fluid arbitrary regions from images to create glyphs (**G4**);
- Allow for flexible but consistent glyph manipulation. In particular, allow to keep parts of the glyph unchanged across data points, and map data attributes to sub-regions or sub-elements of the glyph (**G5**);
- Extract texture and color patterns that can be applied to different visualization marks (**G6**).

PROOF-OF-CONCEPT PROTOTYPE: DATAQUILT

Guided by the design goals, we designed DataQuilt, a proof-of-concept prototype to illustrate our approach (see Fig. 3). DataQuilt comprises of three main components: (a) a set of *libraries* containing the *glyphs*, *colour palettes* and *textures* extracted from source images; (b) an interactive *data table*, with its data binding options (see Fig. 5), and (c) the *main canvas* where the visualization is created. We will illustrate how the features of the tool enables a creative workflow through a usage scenario motivated by sessions with participants.

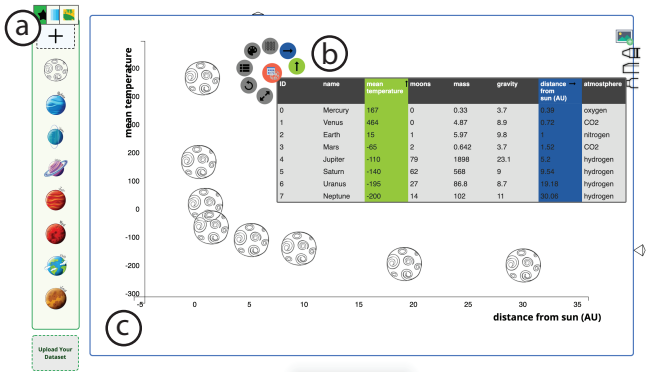

Figure 3. DataQuilt allows to repurpose visual and stylistic elements extracted from source images, stored in the *libraries* (a), by binding them to the data through drag-and-drop interactions with the *interactive data table* (b). These result in a data-driven pictorial visualization displayed on the *main canvas* (c). Here, a planet glyph is bound to the data, the x- and y-position are mapped to distance and temperature respectively.

Context

Elsa, an elementary school teacher, wants to discuss planetary data in her class. She has a dataset of the 8 planets in our solar system, and their properties, such as the mean temperature, distance to the Sun, and main gas in the atmosphere. She wants to create a set of compelling visualizations to engage her students. Elsa opens DataQuilt and loads her dataset, which updates the data table (Fig. 3b). She wants to show how proximity with the sun impacts temperature, via a scatterplot.

Glyph creation and binding - G4

To represent the planets, Elsa thinks about using a generic, yet expressive illustration, as the focus of this first activity is on relationships between, and not as much the planets themselves. She had a mosaic of hand-drawn space elements that she thinks would be perfect. To create the *glyph* that will represent the data points, Elsa clicks on the "+" button from the glyph library (Fig. 3a), which triggers a view of the image collection. She adds, then selects her image. DataQuilt automatically opens the *glyph extraction panel*, in which she roughly traces a contour around a planet to extract it, using the lasso (Fig. 4a). She knows she does not need to be too precise since DataQuilt automatically removes the background. Upon release, a *glyph editing panel* pops up, where Elsa can further edit her selection (Fig. 4b), which she decides not to. She saves, and the glyph is automatically added to the *glyph library*.

As the library stores several glyphs, Elsa can select one and bind it to data by dragging it from the glyph library to the *data binding* icon at the corner of the data table. This results in the

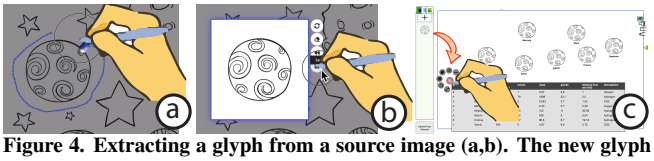

Figure 4. Extracting a glyph from a source image (a,b). The new glyph is bound to data by dragging from the glyph library to the data table (c).

canvas to be automatically populated with an instance of the glyph per data element (Fig. 4c). The data binding icon then reveals a menu of possible data mappings (Fig. 5 left).

Visual-to-data mapping - G2

To create her scatterplot, Elsa applies data-driven constraints to her visualization (i.e., lazy data binding [24]). She starts by dragging the x-axis *visual variable icon* from the menu, and dropping it to the header of the column to map the "distance from sun" data dimension to the x-axis (Fig. 5). This automatically re-positions all of the glyphs so that their horizontal location corresponds to the underlying data. An axis is also displayed. Similarly, she maps the "temperature" column to the y-axis, turning the visualization into the desired scatterplot (Fig. 3). Visual cues, i.e. colours, indicate what mappings are currently active for reference.

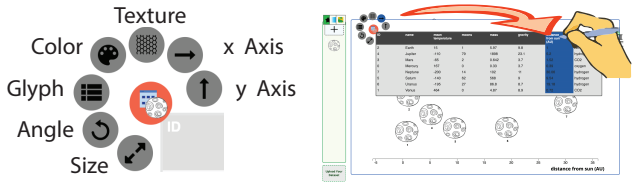

Figure 5. Visual-to-data mapping can be added by dragging a visual variable icon from the menu, and dropping on a column header. Glyphs are automatically updated to comply to the mapping. Here x-axis is mapped, causing relocation of glyphs. An axis is also drawn for reference.

Iterative design - G3

After saving her scatterplot, Elsa wants to edit her design into a visualization featuring other planetary characteristics. She starts by unmapping the y-axis visual variable, by simply dragging it out of the column header. She still wants planets organized along the x-axis, but this time, by temperature. To do so, she drags the x-axis icon from the column it is currently mapped to, and drops it on the other column, which automatically updates the x-positions of the elements accordingly. Other ways she can experiment with her design is to swap the mapped glyph out with a new one from the library, while maintaining the chosen data mappings.

Data-driven manipulation constraints - G2

DataQuilt allows for freeform, direct manipulations of the the data glyphs on the canvas, so long as the changes do not violate visual-to-data integrity. To facilitate the user's workflow, data-driven constraints are used to guide the user's manipulations throughout. For instance, when neither of the x or y-axis is mapped, data glyphs can be freely moved around the canvas. When the x-axis is mapped, the data glyph motion is constrained vertically so as to preserve a meaningful x-positing with regard to the underlying data (Fig. 6). Elsa experiments with layout, using a combination of the horizontal alignment ruler guide (right side of the canvas) and direct manipulation to spread the elements vertically.

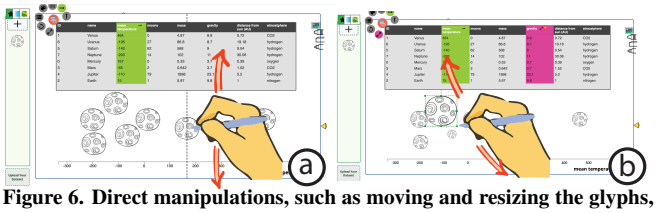

Figure 6. Direct manipulations, such as moving and resizing the glyphs, are constrained according to the visual-to-data mappings. Left: glyphs whose x-position is bound to data can only be moved vertically. Right: resizing one of the glyphs propagates to other glyphs when size is bound to data so as to preserve a meaningful relative size across elements.

Next, she maps the "gravity" dimension to size. This results in some glyphs to become too big to Elsa's liking: she experiments by manually resizing one of the glyphs via direct manipulation, which propagates to other glyphs, whose relative size is adjusted to remain truthful to the data (Fig. 6b).

Mapping different glyphs to categories - G1, G3

Elsa thinks about a more meaningful representation, where the students will be able to relate temperature to atmosphere composition. For this, she would like to use a different glyph for each of the gases in her dataset. She starts by extracting one glyph per gas, adding to the library of glyphs. Elsa drags the "Glyph" visual variable to the atmosphere column header so as to map a different glyph per unique value of this attribute. An interactive legend is automatically pulled on the side of the table, showing each unique value, and the associated glyph, currently the hand-drawn planet. Elsa drags glyphs from the library to the boxes in the legend to associate different glyphs to unique values (Fig. 7). This creates a rich representation that combines different visual elements for each category. She now can see a nice trend, where H_2-dominant planets tend to be colder than others, and saves the visualization.

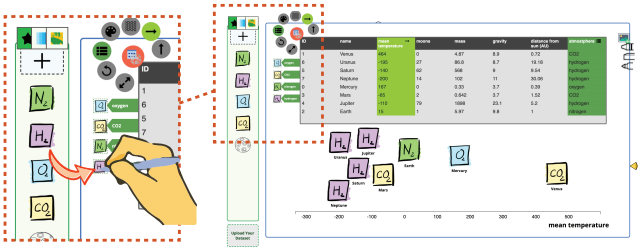

Figure 7. Using the Glyph visual variable, different glyphs can be associated to different categorical values via drag-n-drop operations.

Texture and colour mapping - G3, G6

Other elements of a raster image, such as color and texture, can also be extracted and used in the visualization. Elsa thinks of gravity and mass dimensions. She wants a more playful visualization than a scatterplot, and thinks of an illustration that evokes planets and weights. Assisted by the background removal tool, she extracts the glyph and binds it, overriding the default glyph; she maps size to mass and gravity to y-axis. She likes the result, but wants the glyphs to more explicitly represent each planet. She decides to map a different filling texture for each of the planets so that the glyphs resemble more the actual underlying planets.

She maps the texture variable to the name, which pulls a legend similar as that for assigning different glyphs. She navigates to the texture library, and extracts texture patches following a similar workflow as that for extracting glyphs; the only difference is that textures can only be squared patches. She assigns the texture corresponding to each planet and obtains an original visualization with textured glyphs (Fig. 8a).

Figure 8. Textures (a) and colors (b) are mapped to data values via drag-and-drop interactions from the libraries to the visual variable legend.

Colors can also be assigned to numerical and categorical data attributes. DataQuilt supports both the automated and manual extraction of color palettes from images. Elsa creates such a palette and maps it to temperature. In this case, she only needs to specify a color for the min and the max values, DataQuilt generates a continuous gradient (Fig. 8b. When assigned to a categorical attribute, the tool randomly picks a different colour for each of the unique values. Individual colors can be pinned; new color assignments are generated by clicking on the "reassign" icon, in a similar fashion as in Color Builder [43].

Stretching glyphs - G5

Elsa wants to create a bar chart of distance from the sun. She swaps the main glyph with a newly extracted rocket glyph. To create it, she first maps distance to size, resulting in proportionally scaling all of the rockets. She accesses the subsettings of the size mapping tool via the column header, and experiments with the stretching tool (Fig. 9).

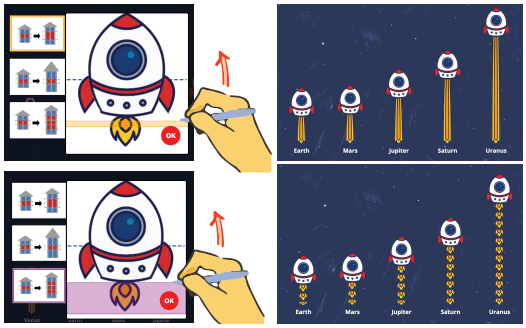

Figure 9. Parts of glyphs can be stretched (a) or replicated (b) as a function of the underlying data.

She starts by stretching the rocket blast using the first option. She specifies the particular vertical segment of the glyph that she wants stretched, by adjusting the zone in the panel (Fig. 9, top). she only selects a portion of the flame, so as to keep the rest of the glyph unchanged. Elsa is curious to see whether replicating the rocket blast would be more compelling. She goes back in the settings, selects the last option this time and selects the whole flame to be repeated (Fig. 9, bottom).

Decorative elements

To further embellish her plots, Elsa also added a background image, and decorative elements to her composition, by simply dragging and dropping glyphs to the canvas and skething annotations with the pencil tool (e.g., Fig. 1, middle). She now has a collection of compelling visualizations for her class.

Implementation

DataQuilt is a web-based Javascript application, using React.js for the front-end and D3.js, Flask and Python for the data binding and manipulation, and image processing back-end. Computer vision techniques (mostly OpenCV) include GrabCut [37] and Canny edge detector [15] to assist with glyph extraction, and k-means clustering for color palette extraction.

VALIDATION

Like other prior works in that space, we did not formally compare our approach to other systems [36, 39], given that there are no other works that allow to easily create pictorial visualizations from raster images. Rather, we follow a methodology that aligns with prior literature [36] and evaluate 1) the *usability* of our tool, through a lab study with guided reproduction tasks; 2) its *expressiveness*, in a follow-up, free-form study. We also illustrate the capability and interactions of DataQuilt with 3) a gallery of visualizations, and 4) authoring scenarios (see Figs. 1,11 and http://dataquilt.github.io).

Usability study

We evaluated the fluidity of the workflow to create custom visualizations, i.e., how easy it is to go from extracting elements from images to binding these elements to data, and back.

Procedure and Tasks

Participants filled out a demographics questionnaire, and completed a tutorial consisting of a system walkthrough slideshow together with an open exploration session for them to experiment with the tool features, for as long as desired.

Participants were then instructed to replicate target visualizations (Fig. 10), borrowing elements from images. The dataset (common to all tasks) consisted of 9 fruit juices, and their features (*fruit type*, *juice type*, *calories*, *sugar* and *vitamin c*). We designed the tasks to cover the core features of DataQuilt. In task **(T1)**, subjects started with a blank slate, allowing us to capture how they created a visualization from scratch. In tasks **(T2-3)**, they were asked to *modify* their last result to recreate a new target. The transitions to **(T2-3)** allowed us to observe how participants *iterated* on a design. Participants were asked to think aloud as they performed the tasks.

Participants then filled out a usability questionnaire, and participated in a semi-structured interview. The study took about 1h to complete (tutorial lasted 20-40 min). All sessions were screencast and audio recorded. Participants received a $15 Amazon gift card in compensation for their time.

Participants and apparatus

Ten participants completed the study: two illustrators (P1, P9), one program manager (P8) and graduate students. Table 1 summarizes their demographics and expertise. Participants operated the tool using a mouse on a MacBook Pro (2.5 GHz) and external monitor at a 2560x1440 resolution.

Results

All participants were able to complete the tasks with no or minimal guidance within an average 4min (**T1**, min: 3, max: 6), 7min (**T2**, min: 3, max: 10), and 3min (**T3**, min: 1, max: 4). Below we report on main insights combining interviews, a usability questionnaire and our own observations.

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
Age	35	26	26	22	50	25	25	28	35	29
Gender	M	M	M	F	M	F	F	F	F	M
data charts	3	2	4	4	4	4	3	5	4	4
vector graphics	4	1	2	2	2	3	4	3	4	3
digital illustration	5	2	5	2	3	3	4	3	4	3
visualization tools	3	4	3	5	4	4	3	4	3	3
vis. programming	2	3	4	1	4	4	4	2	2	5

Table 1. Participants' frequency of dealing with "data charts" (1: never, 5: always), and experience with "vector graphics software", "image manipulation tools", "data visualization tools" and "data visualization programming" (1: not at all experienced, 5: very experienced).

Simplistic, in a good way. All participants commented on the *"straightforward"*, *"intuitive"*, and *"easy"* nature of the tool, which was mainly attributed to the simplicity of the drag'n'drop operations, and the stripped-down interface design. Computer vision techniques for extracting glyphs and palettes were also mentioned (P6, P7). Participants commented that they would not want to add any features or otherwise the tool would become cumbersome or confusing (P1, P2, P6, P8, P9), praising the current approach: there *"weren't too many options thrown at me" (P1)*, *"there wasn't a lot of steps [...] and it was very clear how to do it"* (P6).

Non-trivial start. Some participants were confused by where to start when working on **T1**. P5 wanted to build an axis first, which is not possible until a glyph is bound to the table: *"Why should I start with an empty canvas? [...] There should be already glyphs lying around. Maybe a default glyph."* (P5). Two other participants also clicked on the binding button of the table before binding a glyph, as if to reveal the menu. This confusion may have also been due to having to start by binding a single glyph to the whole table, before other glyphs could be mapped to different data categories: *"It's weird to just have to pick one and then doing the mapping after. [...] It was super easy, but cognitively, it felt a little odd."* (P6)

Different workflows. We observed that every participant followed a different series of operations when creating and editing a visualization. Regardless of their strategy (remove all bindings that are no longer needed first, vs. iteratively changing, one aspect of the binding at a time), participants were able to fluidly modify the desired part of the visualization without having to plan out, start over, or being constrained in any way.

Glyphs vs. texture. One point of friction was caused by the mixing up of glyph and texture. While most participants found the separation in three libraries (glyph, colour, texture) to be *"super intuitive"* (P9), 4 had a hard time separating these concepts (P1, P5, P6, P10). The confusion partly came from the fact that glyphs that are not abstract shapes are already textured: *"I like the workflow of applying a texture to a glyph, but it is a bit challenging to differentiate between glyphs and textures. If you think about glyphs, generally it is a shape, but when you see it is already textured, I don't think of them as I can apply a texture on them"* (P10). P10 actually completed **T1** by texturing an apple shape with different fruit textures, which did not work very well because of the apple silhouette.

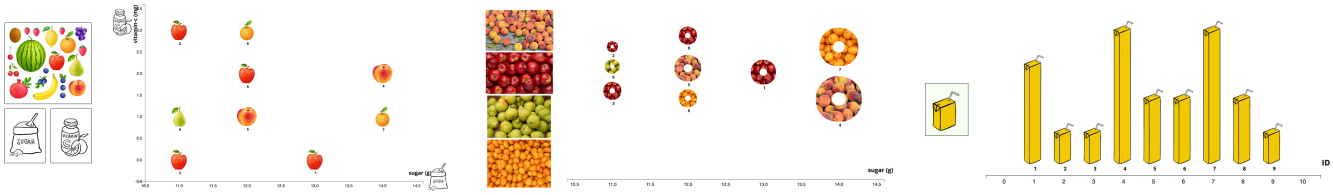

Figure 10. Source images and target visualizations to replicate for the three tasks in our usability study.

Stretch and scale. Another point of confusion was found during **T3**, as many participants were unsure of how to map the height of the juice boxes to calories. Four participants (P2, P4, P6, P8) attempted to map the y-axis to calories to obtain the bar chart, as *"when you think of a general bar chart you think of x- and y- axes"* (P6). Eight participants paused for a few seconds, having to remember the steps to access the stretching options, as size was not a trivial choice for them: *"Size and height were two different things for me"* (P6).

This said, participants found the results of the stretching tool to be *"very inspiring"* (P7), one commenting that *"this is the best feature"* (P8), and one that *"I never have encountered before"* (P9). While operating the tool was generally found easy, it did require participants *"to use some brain power"* (P1), so as *"to decipher which one I want to use"* (P2) as the icons were not explicit enough. The fact that there were a limited number of options, and the workflow was generally smooth, one participant *"would just try until it worked"* (P9). Having different, possibly animated, icons, or even a preview of the distortion, was suggested to improve discoverability.

A safe environment, conducive to experimentation. Participants felt confident using the system overall (2 "strongly agree", 6 "agree", 2 "neutral"), and when unsure about what visual variable to use, did not hesitate to experiment, letting the tool guide them by not allowing meaningless mappings such as size (continuous) and fruit type (categorical): *"I didn't try to think too much"* (P9) . P1 mentioned that *"they aren't any dark corners, where [they are] too scared to go there"*. The easy (un)mapping, direct manipulation and generally streamlined and simple workflow were all seen as strengths, facilitating exploration and experimentation. *"It's easy to play with the numbers, and also the glyphs and that's very fun"* (P7)

Approachable while enabling great creative power. When asked about the potential of the general idea of borrowing elements from existing raster images, participants unanimously commented that it was very valuable, and that DataQuilt was filling a gap. They commented: *"It's giving me freedom that I haven't had before. It's quite a lot."* (P5), *"I think you can be a lot more creative"* (P8), *"I see no downsides and only benefits to doing that"* (P6). Comparing to other tools, they qualified our approach as enabling: in other tools *"there is no way you can create something like this"* (P3); while being much simpler, referring to others as having *"a segmented process [where] you lose a lot of the creativity"* (P8). Overall, all participants saw value in creating the types of artifacts that DataQuilt supports, providing example of useful scenarios where such visuals would be valuable, including education for children, and for lay people.

Suggestions for improvement. Participants also made multiple suggestions for improvements. While most recommended against adding more features to not clutter the interface, recurring comments included: the possibility to snap glyphs to axes when they are used as a label (as in **T1**); the possibility to hide the data table; a wider selection of guiding-selection tools for the glyph extraction (e.g., rectangle, circle selection); ways to keep part of the texture consistent across data points and change others; and finally more ways of interacting directly from the glyph, as e.g., accessing underlying values and binding tools directly from a glyph.

Artificial task. We designed artificial tasks, evaluating how participants would work to obtain pre-defined visualizations. However, this imposes a specific sequence of actions (to perform the study steps), which may differ to what they would do when creating a visualization they had envisioned themselves. P9 commented: *"I had to reverse engineer everything. If I was working on my own visualization, I may go completely differently about the tool"*.

Expressiveness

We conducted a second, free-form study, where participants were asked to craft any visualization using DataQuilt. We invited back participants from the first study (including a pilot subject) that had expertise in graphical design, i.e., anyone who rated themselves as "(very) experienced" in using vector graphics software or digital image manipulation tools.

Procedure, apparatus and participants

A computer with a Wacom Cintiq pen-and-touch screen tablet was made available for the participants to use any time, and for as long as they wanted, for a period of five days after the first study. We provided starter datasets to ignite creativity, but also encouraged participants to use any datasets they wanted. For the source images, we asked them to use any free-of-right resource they wanted. Working sessions were video recorded. At the end of the study period we conducted a semi-structured interview asking participants about the general impressions and creative process using DataQuilt. Participants were compensated an $15 Amazon gift card.

Four of the five invitees took part in the follow-up study (P1, P3, P7, hereafter referred to as D1, D2, D3 respectively, and one of the pilot participants (referred to as D4). D1 and D4 came in multiple times to experiment with the tool, whereas D2 and D3 created visualization(s) in a one-go session, after having worked on building a collection of source images and brainstorming ideas on their own. D4 is a female graduate student, age: 23 (for others, please refer to Table 1).

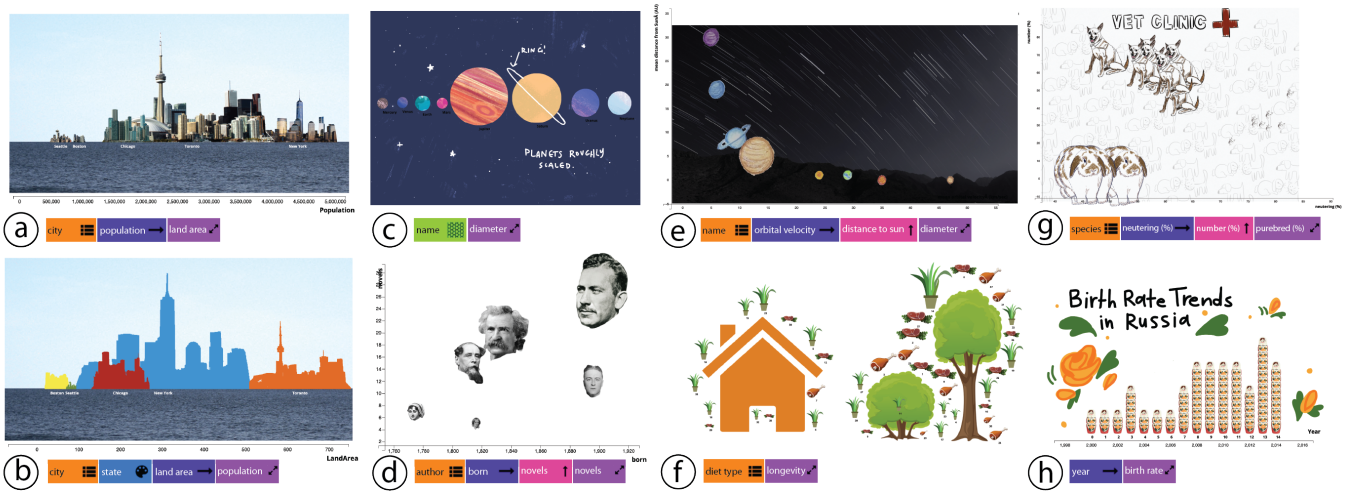

Figure 11. Pictorial visualizations and bindings created by participants in the Expressiveness study, covering a variety of datasets on (a) (b) North American cities, (c) (e) planets , (d) writers , (f) wild + domestic animals, (g) veterinary visits, and (h) birth rates in Russia.

Results

Figure 11 shows a gallery of sample creative visualizations authored by our participants. The rest can be accessed at http://dataquilt.github.io. Below we report major insights gained through interviews and analysis of artifacts.

Swift iterations favour exploratory creativity. Participants appreciated the ability to rapidly iterate over different designs, and took the time to experiment with different data bindings before committing to a final idea: *"You can make a choice about what you want to see and then get pretty immediate feedback of what those choices look like (..). And then just as easily you can drag (what) you bound out."* (D3). This led to an increased chance to stumble upon interesting perspectives on the data and its presentation, which guided most of what they chose to report, e.g., D2's rationale for one of his designs (Fig. 11(e)): *"I did not think in advance about this curve that we ended up getting. [...] I didn't think I would see this pattern. So I did not plan the whole thing in advance but when I was just experimenting with putting the different variables on the two axes, I realized 'oh, it gives me a cool plot'".*

Realizing a vision. Participants had prior exposure to DataQuilt, and knowing its range of capabilities guided a particular set of design choices: *"My whole visualisation was based on being inspired by the tool (...) [tiling/stretching] are the features I found most exciting and thus why I played with them the most."* (D4). This said, participants were satisfied with their end results, and found that they were able to fulfill their (very diverse) personal goals, even when they were less concrete, e.g.: *"In the end the visualisation ended up pretty much as expected"* (D4) and *"[My second visualization design] was more like free form and I think the system actually helped me explore that space pretty well"* (D2).

Temporary mappings as a creative aid. The visualization in Fig. 11(f) represents animals and their characteristics, organized manually around decorative glyphs: *domesticated* animals close to a house), *wild* animals close to trees. When asked about what the process was to place the elements, the participant indicated that they referred to the ID's of each data point to retrieve their *domesticated* value. We suggest a more efficient way to perform this task: colour could have been used as a temporary encoding to distinguish between the two categories, just for the time when layout was performed, to only be unmapped later to come back to original glyphs.

Limitations. Working in "uncharted territory" revealed additional limitations of the system. First, participants pointed to a few challenges they faced working with images, e.g. spotty background removal for noisy images (D1), which involved some pre-processing of images with external tools like Photoshop before importing them into DataQuilt (D1, D2). As for data-driven manipulations, participants suggested: extending the data binding features to include spatial clustering (D2, D4); partial application of colour and texture to glyphs (D2, D4); and more flexible X/Y axis binding to accommodate non numerical dimensions and potential overlap with scaling (D2, D3, D4). In particular, participants wished for more expressive power with their favourite feature — glyph scaling and tiling — including the ability to tile arbitrary regions (D1, D4), expand in various directions (D2, D3) and scale different glyphs (D3).

Usage beyond study. While they did not see themselves using DataQuilt for formal occasions, participants commented on its appeal for informal presentations (D2, D4) and for visualizing personal data (D1, D3, D4). As for the possibility of sustained usage, D4 said: *"The thing that brings me back to this tool is maintaining data integrity, and knowing that I could swap out the data at any point and not have to re-do the graph".*

DISCUSSION AND LIMITATIONS

DataQuilt was enthusiastically received by participants, who felt it gave them creative power that existing visualization authoring tools lack (**G1**). They stressed the simplicity of the interface, that allows to integrate data-driven representations and free-form graphics extracted from images (**G2**), in a fluid workflow enabling iterative, exploratory creativity (**G3**). Their comments and successful tasks demonstrate the *low threshold* in learning to use the system, while the diverse free-form designs they generated demonstrate its *high ceiling* [30, 42].

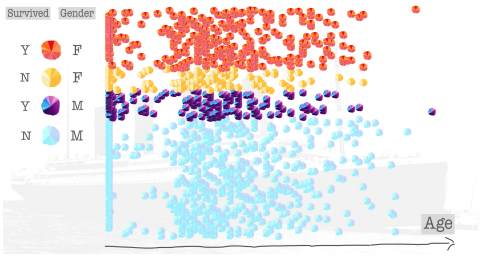

Figure 12. A visualization of gender and survival status of Titanic passengers [31], featuring 1309 individual data points.

A combination of several non-trivial, but well integrated features were highlighted. These include image processing techniques for extracting glyphs and color palettes (**G4**), support of different workflows, and direct manipulation actions for data-binding that encouraged experimentation. This simplicity of binding was identified as a potential aid during the creative process, but not necessarily in the end result, for example to keep track of which elements are in which group to guide a free-form layout, and then remove the mapping.

Some participants did not know where to start their exploration. Currently our tool requires a single glyph to be bounded before a visualization is constructed. Our decision was explicit, in order to avoid premature commitment to a layout and allow users to experiment. It is possible that the confusion is due to participants trying to replicate a given visualization rather than crafting their own inspired by images, on which the likely starting point would be glyph creation and loading. Nonetheless, we plan to pre-bind a "default" glyph and have a default presentation of data points, as suggested by participants. Our tool supports the extraction of both free-form and closed-form glyphs (**G4**) as well as textures and color palettes (**G6**). As the original glyphs are pre-textured (with the fill pattern extracted from the original image), some participants had trouble understanding that they could keep their contour and re-color or re-texture them. This requires us to rethink how to visually convey the two aspects of the glyph (form and fill) and organize the glyph library accordingly.

One of our design goals was to allow users to keep parts of the glyph unchanged across data points, and map data attributes to sub-elements of the glyph (**G5**). We partly support this in the form of stretching and replication of a sub-region of the glyph. We can envision extending this to other properties, keeping part of the glyph consistent across data points and re-texture, re-color, or rotate another part based on the data mapping.

The decision to group scaling operations together (**G5**) caused some confusion. Participants had trouble remembering that the region-stretching and repetition tool were under the size mapping. In the next iteration of the tool we will differentiate the notion of length (stretch / repeat) and scale.

Vector graphics are often preferred for infographics and data visualization for their cleanness and their high quality no matter how they are rescaled and distorted. In contrast, raster images are limiting in several ways: computation is greatly slowed down, as the processing of pixel-based images is much more costly than that of vector graphics, and can result in unflattering outputs as glyphs undergo deformations.

As in other creativity tools, we assume a clean dataset, without missing or erroneous values, of a rather small scale [39]. Currently, DataQuilt can comfortably accommodate dataset of a few hundreds items (Fig. 12, http://dataquilt.github.io).

Finally, our tool enables the creation of custom visualizations, adding elements such as textures, differently shaped glyphs, and embellishments. As such, it gives ample room for users to create visualizations that may qualify as chartjunk [45], if embellishments are not purposefully used to aid communication [6]. To mitigate the risks, we took special care to promote good visualization practices as much as possible. For example, it is not possible to map shape or texture to numerical dimensions, as they are not recommended for numerical values [27]. Our colour tool behaves differently when associated to categories (discrete palette) vs. numerical (generation of a gradient). And when glyphs are mapped to an axis they remain constrained by it, providing both freedom and constraint to preserve data-to-visual integrity.

CONCLUSION

Inspired by a vision of data ubiquity and freedom of expression, this work presented DataQuilt, a tool to support the creation of rich and organic visualizations, by borrowing and repurposing elements from raster images. This makes visualization authoring accessible to a broader audience that is not necessarily artistically inclined, nor versed in image editing software and visualization programming.

Our design process was heavily user centered: DataQuilt was informed by formative findings from two data driven workshops, and was validated in both a usability study and a creative, free-form exercise study. We found that the tool can be easily appropriated by novices and is expressive enough to allow for significant diversity in visualization designs. More importantly, we found that the tool supported quick exploratory manipulations of designs, which has been noted by some participants as being conducive to creativity.

From emerging gaps we have also outlined an agenda for future research. We underscore the importance of improving the expressiveness of bindings to size (beyond vertical bar charts), color (e.g., applying color to parts of the glyph, instead of painting the entire region a flat color) and of positioning (e.g., for spatial clustering). More generally, findings from the formative assessment and evaluation studies revealed great user interest in powerful authoring tools for custom visualizations, that remain nonetheless easy to use and conductive to design refinement and experimentation in order to support creativity. On this last note, we conclude how we started, with a quote.

> *"To see your world through a new lens, where everything and anything can be a creative starting point for play and expression."*
>
> — G. Lupi, S. Posavec [26]

ACKNOWLEDGMENTS

We thank Giorgia Lupi and Stefanie Posavec for inspiration, Nathalie Riche and DGP lab members for their input. This work was partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

REFERENCES

[1] 2019. Adobe Illustrator. www.adobe.com/Illustrator. (2019). Accessed: 2019-09-11.

[2] 2019. Microsoft PowerBI Software. https://powerbi.microsoft.com. (2019). Accessed: 2019-12-15.

[3] 2019. Processing. https://processing.org/. (2019). Accessed: 2019-12-15.

[4] 2019. Tableau Software. www.tableau.com. (2019). Accessed: 2019-09-11.

[5] 2019. Vega: A Visualization Grammar. http://trifacta.github.io/. (2019). Accessed: 2019-12-15.

[6] Scott Bateman, Regan L Mandryk, Carl Gutwin, Aaron Genest, David McDine, and Christopher Brooks. 2010. Useful junk?: the effects of visual embellishment on comprehension and memorability of charts. In *Proceedings of the 2010 CHI Conference on Human Factors in Computing Systems (2010)*. ACM, 2573–2582. DOI:http://dx.doi.org/10.1145/1753326.1753716

[7] Alex Bigelow, Steven Drucker, Danyel Fisher, and Miriah Meyer. 2014. Reflections on how designers design with data. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*. ACM, 17–24. DOI: http://dx.doi.org/10.1145/2598153.2598175

[8] Alex Bigelow, Steven Drucker, Danyel Fisher, and Miriah Meyer. 2016. Iterating between tools to create and edit visualizations. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2016), 481–490. DOI: http://dx.doi.org/10.1109/TVCG.2016.2598609

[9] Rita Borgo, Alfie Abdul-Rahman, Farhan Mohamed, Philip W Grant, Irene Reppa, Luciano Floridi, and Min Chen. 2012. An empirical study on using visual embellishments in visualization. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (2012), 2759–2768. DOI: http://dx.doi.org/10.1109/TVCG.2012.197

[10] Michelle A Borkin, Zoya Bylinskii, Nam Wook Kim, Constance May Bainbridge, Chelsea S Yeh, Daniel Borkin, Hanspeter Pfister, and Aude Oliva. 2015. Beyond memorability: Visualization recognition and recall. *IEEE Transactions on Visualization and Computer Graphics* 22, 1 (2015), 519–528. DOI: http://dx.doi.org/10.1109/TVCG.2015.2467732

[11] Michelle A Borkin, Azalea A Vo, Zoya Bylinskii, Phillip Isola, Shashank Sunkavalli, Aude Oliva, and Hanspeter Pfister. 2013. What makes a visualization memorable? *IEEE Transactions on Visualization and Computer Graphics* 19, 12 (2013), 2306–2315. DOI: http://dx.doi.org/10.1109/TVCG.2013.234

[12] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D3: Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics (Proc. InfoVis)* (2011). DOI:http://dx.doi.org/10.1109/TVCG.2011.185

[13] John Brosz, Miguel A Nacenta, Richard Pusch, Sheelagh Carpendale, and Christophe Hurter. 2013. Transmogrification: causal manipulation of visualizations. In *Proceedings of the 26th annual ACM Symposium on User Interface Software and Technology*. ACM, 97–106. DOI: http://dx.doi.org/10.1145/2501988.2502046

[14] Lydia Byrne, Daniel Angus, and Janet Wiles. 2015. Acquired codes of meaning in data visualization and infographics: beyond perceptual primitives. *IEEE Transactions on Visualization and Computer Graphics* 22, 1 (2015), 509–518. DOI: http://dx.doi.org/10.1109/TVCG.2015.2467321

[15] John Canny. 1986. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6 (1986), 679–698. DOI: http://dx.doi.org/10.1109/TPAMI.1986.4767851

[16] Zhutian Chen, Yun Wang, Qianwen Wang, Yong Wang, and Huamin Qu. 2019. Towards Automated Infographic Design: Deep Learning-based Auto-Extraction of Extensible Timeline. *IEEE transactions on visualization and computer graphics* 26, 1 (2019), 917–926. DOI: http://dx.doi.org/10.1109/TVCG.2019.2934810

[17] William S Cleveland and Robert McGill. 1984. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *J. Amer. Statist. Assoc.* 79, 387 (1984), 531–554. DOI: http://dx.doi.org/10.1080/01621459.1984.10478080

[18] Weiwei Cui, Xiaoyu Zhang, Yun Wang, He Huang, Bei Chen, Lei Fang, Haidong Zhang, Jian-Guan Lou, and Dongmei Zhang. 2019. Text-to-Viz: Automatic Generation of Infographics from Proportion-Related Natural Language Statements. *IEEE transactions on visualization and computer graphics* 26, 1 (2019), 906–916. DOI: http://dx.doi.org/10.1109/TVCG.2019.2934785

[19] Steve Haroz, Robert Kosara, and Steven L Franconeri. 2015. Isotype visualization: Working memory, performance, and engagement with pictographs. In *Proceedings of the 2015 CHI Conference on Human Factors in Computing Systems (2015)*. ACM, 1191–1200. DOI:http://dx.doi.org/10.1145/2702123.2702275

[20] Nam Wook Kim, Nathalie Henry Riche, Benjamin Bach, Guanpeng Xu, Matthew Brehmer, Ken Hinckley, Michel Pahud, Haijun Xia, Michael J McGuffin, and Hanspeter Pfister. 2019a. DataToon: Drawing Dynamic Network Comics With Pen+ Touch Interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (2019)*. ACM, 105. DOI: http://dx.doi.org/10.1145/3290605.3300335

[21] Nam Wook Kim, Hyejin Im, Nathalie Henry Riche, Alicia Wang, Krzysztof Gajos, and Hanspeter Pfister. 2019b. DataSelfie: Empowering People to Design Personalized Visuals to Represent Their Data. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (2019)*. ACM, New York, NY, USA, Article 79, 12 pages. DOI: http://dx.doi.org/10.1145/3290605.3300309

[22] N. W. Kim, E. Schweickart, Z. Liu, M. Dontcheva, W. Li, J. Popovic, and H. Pfister. 2017. Data-Driven Guides: Supporting Expressive Design for Information Graphics. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (Jan 2017), 491–500. DOI: http://dx.doi.org/10.1109/TVCG.2016.2598620

[23] Robert Kosara. 2007. Visualization criticism-the missing link between information visualization and art. In *2007 11th International Conference Information Visualization (IV'07)*. IEEE, 631–636. DOI: http://dx.doi.org/10.1109/iv.2007.130

[24] Zhicheng Liu, John Thompson, Alan Wilson, Mira Dontcheva, James Delorey, Sam Grigg, Bernard Kerr, and John Stasko. 2018. Data Illustrator: Augmenting Vector Design Tools with Lazy Data Binding for Expressive Visualization Authoring. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (2018)*. ACM, New York, NY, USA, Article 123, 13 pages. DOI: http://dx.doi.org/10.1145/3173574.3173697

[25] Giorgia Lupi and Stephanie Posavec. 2016. *Dear Data*. Chronicle books.

[26] Giorgia Lupi and Stephanie Posavec. 2018. *OBSERVE, COLLECT, DRAW! - A Visual Journal* (1 ed.). Princeton Architectural Press.

[27] Jock Mackinlay. 1986. Automating the Design of Graphical Presentations of Relational Information. *ACM Transactions on Graphics* 5, 2 (April 1986), 110–141. DOI:http://dx.doi.org/10.1145/22949.22950

[28] Gonzalo Gabriel Méndez, Miguel A Nacenta, and Sebastien Vandenheste. 2016. iVoLVER: Interactive visual language for visualization extraction and reconstruction. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (2016)*. ACM, 4073–4085. DOI: http://dx.doi.org/10.1145/2858036.2858435

[29] Andrew Vande Moere and Helen Purchase. 2011. On the role of design in information visualization. *Information Visualization* 10, 4 (2011), 356–371. DOI: http://dx.doi.org/10.1177/1473871611415996

[30] Brad Myers, Scott E Hudson, Randy Pausch, and Randy Pausch. 2000. Past, present, and future of user interface software tools. *ACM Transactions on Computer-Human Interaction (TOCHI)* 7, 1 (2000), 3–28. DOI: http://dx.doi.org/10.1145/344949.344959

[31] The University of Sheffield. 2019. Datasets for Teaching. https://www.sheffield.ac.uk/mash/statistics/datasets. (2019). Accessed: 2019-09-11.

[32] Ji Hwan Park, Arie Kaufman, and Klaus Mueller. 2018. Graphoto: Aesthetically Pleasing Charts for Casual Information Visualization. *IEEE computer graphics and applications* 38, 6 (2018), 67–82. DOI: http://dx.doi.org/10.1109/MCG.2018.2879066

[33] Jorge Poco and Jeffrey Heer. 2017. Reverse-Engineering Visualizations: Recovering Visual Encodings from Chart Images. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 353–363. DOI: http://dx.doi.org/10.1111/cgf.13193

[34] Jorge Poco, Angela Mayhua, and Jeffrey Heer. 2017. Extracting and retargeting color mappings from bitmap images of visualizations. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (2017), 637–646. DOI: http://dx.doi.org/10.1109/tvcg.2017.2744320

[35] Donghao Ren, Bongshin Lee, and Matthew Brehmer. 2018a. Charticulator: Interactive construction of bespoke chart layouts. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (2018), 789–799. DOI: http://dx.doi.org/10.1109/tvcg.2018.2865158

[36] Donghao Ren, Bongshin Lee, Matthew Brehmer, and Nathalie Henry Riche. 2018b. Reflecting on the Evaluation of Visualization Authoring Systems: Position Paper. In *2018 IEEE Evaluation and Beyond-Methodological Approaches for Visualization (BELIV)*. IEEE, 86–92. DOI: http://dx.doi.org/10.1109/BELIV.2018.8634297

[37] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. 2004. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM Transactions on Graphics (TOG)*, Vol. 23. ACM, 309–314. DOI: http://dx.doi.org/10.1145/1186562.1015720

[38] Arvind Satyanarayan and Jeffrey Heer. 2014. Lyra: An interactive visualization design environment. In *Computer Graphics Forum*, Vol. 33. Wiley Online Library, 351–360. DOI: http://dx.doi.org/10.1111/cgf.12391

[39] Arvind Satyanarayan, Bongshin Lee, Donghao Ren, Jeffrey Heer, John Stasko, John Thompson, Matthew Brehmer, and Zhicheng Liu. 2019. Critical reflections on visualization authoring systems. *IEEE transactions on visualization and computer graphics* (2019). DOI: http://dx.doi.org/10.1109/TVCG.2019.2934281

[40] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2016. Vega-lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2016), 341–350. DOI: http://dx.doi.org/10.1109/tvcg.2016.2599030

[41] Manolis Savva, Nicholas Kong, Arti Chhajta, Li Fei-Fei, Maneesh Agrawala, and Jeffrey Heer. 2011. Revision: Automated classification, analysis and redesign of chart images. In *Proceedings of the 24th annual ACM Symposium on User Interface Software and Technology*. ACM, 393–402. DOI: http://dx.doi.org/10.1145/2047196.2047247

[42] Ben Shneiderman. 2007. Creativity support tools: Accelerating discovery and innovation. *Commun. ACM* 50, 12 (2007), 20–32. DOI: http://dx.doi.org/10.1145/1323688.1323689

[43] Maria Shugrina, Wenjia Zhang, Fanny Chevalier, Sanja Fidler, and Karan Singh. 2019. Color Builder: A Direct Manipulation Interface for Versatile Color Theme Authoring. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (2019)*. ACM, 456. DOI:http://dx.doi.org/10.1145/3290605.3300686

[44] Alice Thudt, Charles Perin, Wesley Willett, and Sheelagh Carpendale. 2017. Subjectivity in personal storytelling with visualization. *Information Design Journal* 23, 1 (2017), 48–64. DOI: http://dx.doi.org/10.1075/idj.23.1.07thu

[45] Edward Tufte. 1983. *The Visual Display of Quantitative Information*. CT: Graphics Press, Cheshire. DOI: http://dx.doi.org/10.1515/9783050093833-035

[46] Fernanda B Viégas and Martin Wattenberg. 2007. Artistic data visualization: Beyond visual analytics. In *International Conference on Online Communities and Social Computing*. Springer, 182–191. DOI: http://dx.doi.org/10.1007/978-3-540-73257-0_21

[47] Yun Wang, Haidong Zhang, He Huang, Xi Chen, Qiufeng Yin, Zhitao Hou, Dongmei Zhang, Qiong Luo, and Huamin Qu. 2018. InfoNice: Easy creation of information graphics. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (2018)*. ACM, 335. DOI: http://dx.doi.org/10.1145/3173574.3173909

[48] Colin Ware. 2012. *Information visualization: perception for design*. Elsevier.

[49] Hadley Wickham. 2016. *ggplot2: elegant graphics for data analysis*. Springer.

[50] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2016. Voyager: Exploratory Analysis via Faceted Browsing of Visualization Recommendations. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* (2016). DOI: http://dx.doi.org/10.1109/TVCG.2015.2467191

[51] Haijun Xia, Nathalie Henry Riche, Fanny Chevalier, Bruno De Araujo, and Daniel Wigdor. 2018. DataInk: Direct and Creative Data-Oriented Drawing. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (2018)*. ACM, New York, NY, USA, Article 223, 13 pages. DOI: http://dx.doi.org/10.1145/3173574.3173797