GALVIS: Visualization Construction through Example-Powered Declarative Programming

Leixian Shen
Tsinghua University
Beijing, China
slx20@mails.tsinghua.edu.cn

Enya Shen
Tsinghua University
Beijing, China
shenenya@tsinghua.edu.cn

Zhiwei Tai
Tsinghua University
Beijing, China
tzw20@mails.tsinghua.edu.cn

Yun Wang
Microsoft Research Asia
Beijing, China
wangyun@microsoft.com

Yuyu Luo
Tsinghua University
Beijing, China
luoyy18@mails.tsinghua.edu.cn

Jianmin Wang
Tsinghua University
Beijing, China
jimwang@tsinghua.edu.cn

ABSTRACT

Declarative programmatic approaches are an essential modality for data visualization construction. Despite the powerful customization ability, declarative programming requires users to create charts from scratch, thus building a well-designed visualization is an effort-consuming process. In this paper, we propose leveraging examples to alleviate the problem. The use of examples plays a vital role in visualization design. Users can be allowed to browse through designs for inspiration and adapt them for their own visualizations. In this demo, we directly leverage the entire Vega/Vega-Lite example galleries as chart templates and introduce an authoring pipeline to conveniently instantiate templates with the user's data for extensible programmatic modifications. Finally, we build GALVIS, an example-powered declarative programming tool for visualization construction, enabling efficient declarative programming and retaining the full spectrum of Vega/Vega-Lite characteristics.

CCS CONCEPTS

• **Human-centered computing → Visualization systems and tools**.

KEYWORDS

Visual Analysis, Authoring, Example, Declarative Programming

ACM Reference Format:
Leixian Shen, Enya Shen, Zhiwei Tai, Yun Wang, Yuyu Luo, and Jianmin Wang. 2022. GALVIS: Visualization Construction through Example-Powered Declarative Programming. In *Proceedings of the 31st ACM Int'l Conference on Information and Knowledge Management (CIKM '22), Oct. 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3511808.3557159

1 INTRODUCTION

The high demand for visual data analysis has nourished a remarkable series of visualization authoring tools both in industry (e.g., Tableau and Microsoft Power BI) and academia (e.g., Vega-Lite[1], Data illustrator[10], and Voyager[36]). They significantly reduce a user's efforts in creating visualizations. Among the tools, declarative programmatic approaches are an essential modality and are widely used to author data-oriented visualizations (e.g., Vega[26], Vega-Lite[25], ggplot2[35], and Atom[18]). Compared with interactive systems[21–23, 36, 38], declarative programmatic approaches support remarkable expressive chart designs and allow low-level modifications. However, writing declarative textual specifications is still a troublesome and time-consuming process for most developers. This modality assumes that users want to generate visualizations that they already have in mind but users may not know how to start. Moreover, it requires users to be familiar with the grammar and write code line by line from scratch.

To reduce the complexity of programming, we believe that learning and adapting examples is an effective solution. Examples serve a critical role in creative design practice[7] and are widely used in numerous general authoring tools that provide visualization services. Satyanarayan et al.[24] critically reflect on existing interactive authoring tools and note that these systems aim to author rather than design visualizations since they assume that users come with "a specific chart design in mind". The use of examples can avoid the blank canvas problem and guarantee ease of use[24]. Moreover, a vision for a visualization authoring system is "allowing users to browse through designs for inspiration, or adapt them for their own visualizations"[27]. However, this vision has not been applied to power declarative programming for visualization authoring.

In this paper, we build GALVIS ("GAL" is short for "Gallery"), a novel example-powered declarative programming tool for data visualization authoring, which directly leverages the entire Vega/Vega-Lite example galleries as chart templates, and the templates can be conveniently instantiated with the user's data for extensible interactive modifications. The design of GALVIS meets various challenges, such as how to enable users to conveniently browse through designs for inspiration, how to integrate programming approaches without invading the syntax, how to retain the full spectrum of Vega/Vega-Lite characteristics, and how to allow users to flexibly modify the generated textual specification of visualizations. To address these issues, as shown in Figure 1, we introduce a pipeline to effectively generate visualizations with the user's data from chart examples. Data is first decoupled from chart examples with the help of Vega-Link (Section 3.1) to generate templates (A). After loading data, the user can browse the examples for inspiration.

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10.
https://doi.org/10.1145/3511808.3557159

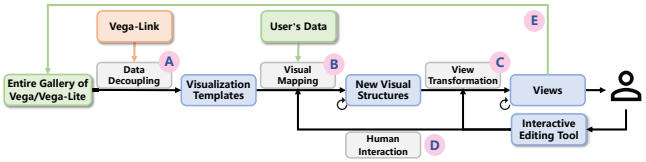


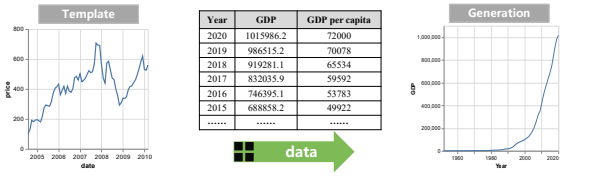
Figure 1: Visualization authoring pipeline from examples.

Figure 2: User interface of GALVIS.

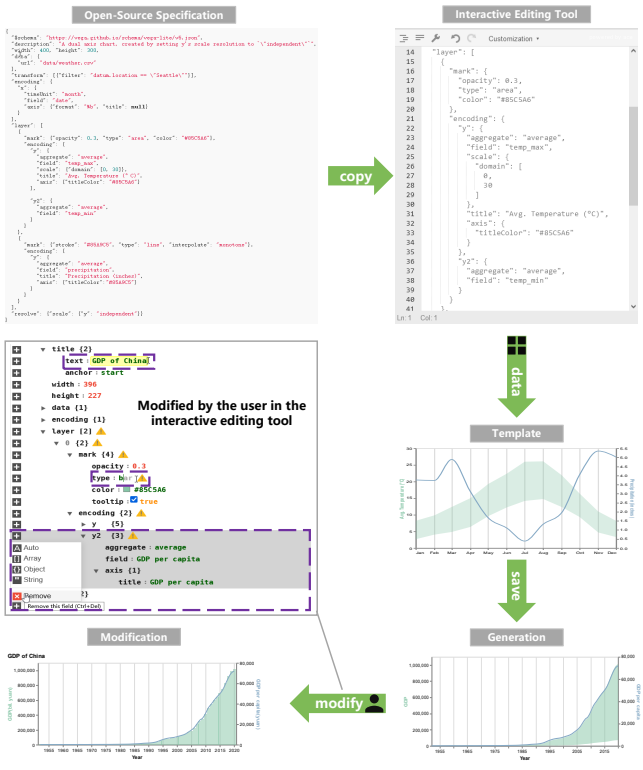
When the user drags a template and drops it onto the assembly area (D), the template is instantiated, and the system can automatically map the data variables into the template (B). The specification code is loaded in the interactive editing tool for extensible modifications (D). After that, various view transformations can be performed on the assembly area (C). The generated charts can also be reused as new templates (E).

2 DEMONSTRATION

David is a professional data scientist who uses Vega and Vega-Lite for visual analysis. He has recently used GALVIS for visualization authoring. David has an open-ended task that requires him to explore the movies[9] and Hollywood stories[8] datasets. As shown in Figure 2, after loading the data, data variables marked with data types (temporal, nominal, quantitative, and ordinal) are presented in (V1). David has no idea how to start the analysis, so he browses the templates in GALVIS for inspiration. In (V2), a wide array of templates are categorized hierarchically, and each template is represented as a thumbnail with a short description. When browsing the "Repeat & Concatenation" category, the marginal histograms catches his eyes, so he drags the template and drops it on the assembly area (V3). GALVIS automatically maps the dataset's variables to the template and generates a visualization. David is satisfied with the generated marginal histogram and drags it to the corner. Then, he finds that the scatterplot with mean and standard deviation overlay in the "Error Bars & Error Bands" category is interesting and drags the template. The generated chart displays two variables, *Audience Score* and *Worldwide Gross*, on the x-axis and y-axis, and the text specification is loaded in the template editor (V4). David wants to add a color channel with the *Lead Studio*, when he modifies the code, an autocompletion widget appears to help him complete input (like in Figure 5 (a)), and corresponding changes are reflected on the visualization in real-time. After that, when browsing the "Bar Chart" category, David gets inspired and wants to draw a bar colored by *Genre* and labeled with *IMDB Ratings*. He starts with a



(a) Visualization generation without any user modifications



(b) Import a new template and create a chart with minor user modifications

Figure 3: Convert examples to customized visualizations.

similar bar chart template with labels and modifies the specification with the interactive template editor. David finds that using GALVIS largely reduces the cost of time compared with writing code from scratch. After creating the three charts, David freely resizes, positions, and constructs their layouts on the assembly area with the help of the extension editor (V5), which allows users to fine-tune auxiliary properties (e.g., data sources, chart size, positions, margin, etc.). Finally, David saves (💾) and previews (🖵) the dashboard by clicking the function buttons in the tool panel (V6).

David has another task to analyze the GDP data[17]. He selects a line chart template and generates a visualization without any modifications (Figure 3 (a)). This chart can show the trend of GDP, but David thinks it is a bit monotonous. He remembers that he previously encountered a dual-axis chart when surfing the Internet, so he copies the code. As shown in Figure 3 (b), he first uploads the dataset to GALVIS and pastes the code in the template editor. Then he clicks the upload chart (☁) button to save the focused chart as a personal template. After that, he uploads the GDP data and drags the personal template to the assembly area. However, the generated

Table 1: Three types of template in GALVIS.

Template Type	Specification	Default number	Description
Common	Vega-Lite	178	common statistical graphics
Senior	Vega	89	relatively complicated but powerful
Personal	Vega/Vega-Lite	X	the user's customized charts

* X indicates that the user's customized charts can be reused as templates.

chart is a little strange as the area chart displays two variables on the y-axis (The template originally shows maximum and minimum temperatures in a weather dataset), so he uses the template editor to remove the "y2" encoding channel and changes the mark to "bar". As a result, the modified chart can clearly display the GDP and GDP per capita information. Finally, Davids makes some textual changes to the title and axis.

3 GALVIS

This section will discuss challenges we encounter when designing GALVIS and how we address the problems.

3.1 Framework

Conceptual frameworks play a significant role in visualization creation. Inspired by related works[15, 23, 34, 38], we choose Vega and Vega-Lite to specify visualizations, which are widely used, expressive enough, and more intuitive to understand. In addition, the original textual specifications of examples are associated with old data, which are not convenient for editing by the user. To templatize Vega/Vega-Lite-based charts and link chart templates and the user's data, we design a scheme named Vega-Link, which decouples data from visualization specifications and involves auxiliary primitives for extensive modifications (e.g., dashboard design). Vega-Link and Vega/Vega-Lite supplement each other. Figure 2 (V5) is an extension editor powered by Vega-Link that displays the structure of Vega-Link. The primitives of Vega-Link can be divided into three categories: **(1)** Basic options: these options are associated with the basic properties of the chart, including *author*, *name*, *description*, and *theme*, where *name* is the unique identifier of the chart, and the supported themes[33] are the same with Vega/Vega-Lite (e.g., Excel, vox, dark, and google charts); **(2)** Data options: they focus on how to link the chart template with the user's data, as well as refreshing the data sources, including *Data*, *Update*, and *Refresh*; **(3)** Position options: they relate to the position information of charts, which are helpful for view transformation, including *ChartPosition*, *ChartSize*, *PageSize*, and *Margin*. More details will be discussed, along with the explanation of GALVIS's design logic, in the following sections.

3.2 Example as Template

Some works abstract existing programmatic languages to create chart templates, which is time-consuming for producing a new template and may increase the learning curve. Instead, GALVIS fully retains the characteristics of Vega/Vega-Lite, and the design of Vega-Link allows easy templatization of existing charts. Benefiting from the efficient templatization mechanism, GALVIS can rapidly integrate a large number of templates. By default, GALVIS supports 267 chart templates, as shown in Figure 2 (V2), which derive from the Vega[31] and Vega-Lite[32] example galleries as well as their extensions during our visualization practice. The templates are

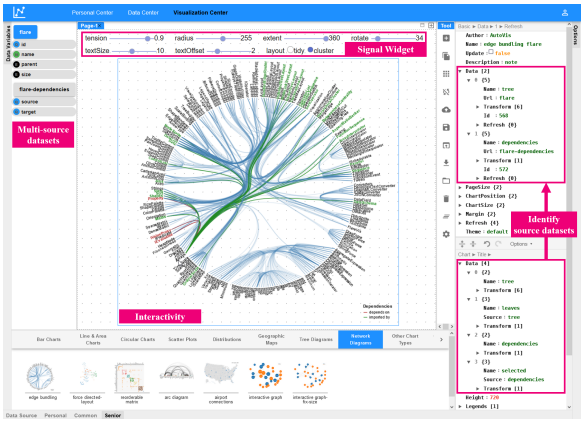


Figure 4: GALVIS can deal with multi-source datasets and retain the interactivity ability of Vega/Vega-Lite.

displayed with a hierarchical classification for easy reference. First, all templates are divided into three categories: common, senior, and personal (Table 1). Next, the common and senior templates are aligned to the same directories in Vega/Vega-Lite examples, which is convenient for users to find what they expect. The number of templates integrated in GALVIS by default is larger than existing template-based tools, such as RAWGraphs[14] (29), Ivy[15] (40), Excel (58), and Flourish[6] (150). The user's customized charts can also be directly reused as templates in GALVIS.

3.3 Data Decoupling

For instantiation of templates, **the first challenge is how to bridge the gap between the chart template and the user's data, especially multi-source datasets.** In the data model of Vega and Vega-Lite, they generally load data through the *values* and *url* primitives. However, *values* is only appropriate for inline data, and *url* does not work for our database. To templatize existing visualization examples and allow easy visualization authoring with the user's data, we believe that data should be decoupled from the specification. To this end, we redefine a *data* primitive in Vega-Link, which retains the characteristics of Vega/Vega-Lite and adapts to our database, allowing users to import both single dataset and multi-source datasets of their own. As shown in Figure 2 (V3, V4), the bar chart analyzes a single dataset[9]. GALVIS first identifies the *data* primitive in the chart template and then makes corresponding modifications. In detail, the new *url* in Vega-Link (see Figure 2 (V5)) corresponds to the name of the user's dataset, which can be seen in Figure 2 (V1). Users can easily switch the dataset by directly changing the *url*, and the data variables in the template editor will be updated automatically. The *name* and *id* of the dataset are automatically generated to reference the dataset in the template editor (see Figure 2 (V4)). For multi-source datasets, Figure 4 visualizes the flare dataset[4] and its dependency data[5] with an edge bundling chart to find dependencies between classes. In the original Vega-powered visualization template, there are four data sources as shown in the template editor, where *tree* and *dependencies* are source datasets, while *leaves* and *selected* are derived through "transform" operations. In such cases, GALVIS can automatically identify the source data and present them in the extension editor, thus avoiding duplicate data loading.

Users can also import external data from the database or through URL in GALVIS. In that case, the remote data source may update at any time (especially time series data). **So refreshing data in the visualization is another important task**. Benefiting from the decoupling of data from the specification, data refreshing can be conveniently performed at the data level. We design a *refresh* option in Vega-Link, which supports automatic data refreshing based on various configurations. First, users can choose to refresh the entire dashboard or a specific visualization or dataset. Second, there are two kinds of refresh patterns. One is to refresh at a fixed time, and the other is to refresh after a fixed time interval. Third, GALVIS supports different refresh types: increase, difference, and full.

3.4 Visual Mapping

After loading datasets, **an important problem is how to map data variables into the visual channels properly and automatically**. For this problem, we design a rule-based approach, when a template is dragged by the user and dropped on the assembly area (Figure 2 (V3)), the algorithm works. There is a consensus that data type is a vital characteristic for visualization creation, and numerous systems have developed design rules based on data types[2, 11, 16, 29, 30]. Inspired by these work, we recursively search the template specification for the data "field" primitive with its corresponding data "type". Then, data variables with the same type will be adopted to replace the original data "field", as well as the title of axis. Variables that have already been used will no longer be used to avoid duplication. If there are no variables of the same type as required by the template, the system will compromise to select a variable of a similar type. For instance, if the template requires a nominal variable, but there is no corresponding variable in the dataset, the system will continue to look for ordinal ones, followed by quantitative, and finally temporal.

3.5 View Transformation

Users can create multiple visualizations on the assembly area, as shown in Figure 2 (V3). **An important problem is how to allow users to design their multiple views freely and conveniently**. GALVIS provides a What You See Is What You Get (WYSIWYG) multi-view design experience by allowing users to interactively drag the template and move the visualizations for view transformations. The *ChartPosition* and *ChartSize* defined in Vega-Link will be updated synchronously. GALVIS also presents *PageSize* (width and height) for reference and allows for fine-grained control of the aforementioned options. The charts can also be placed on top of each other, enabling more expressive designs.

3.6 Human Interaction

Human interactions with the interface feed back into the pipeline.

Interactive Display: Interactivity is an essential characteristic of visual analysis[20]. Vega and Vega-Lite are designed to enable rapid specification of interactive charts, and their galleries incorporate numerous interactive displays. However, interactivity has received relatively less attention in current Vega/Vega-Lite-powered authoring tools. **An important issue we need to address is to retain the interactivity ability of Vega/Vega-Lite, allowing users to explore the chart interactively**. We generally divide

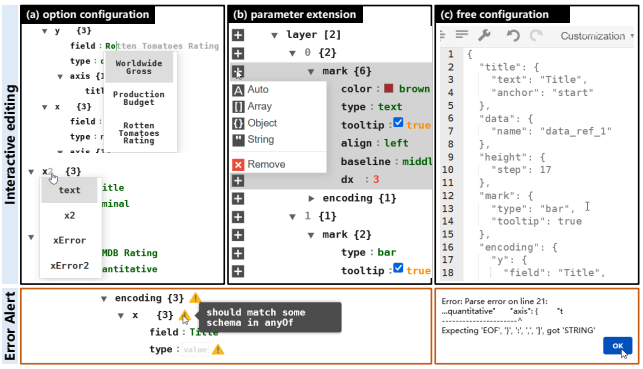


Figure 5: Template editor with different freedom degrees.

interactive technologies into two classes: one is "tooltip", When the mouse hovers on a particular data point of the chart, the details will be displayed. The system will add the *tooltip* primitive to the chart and set it to true if the template does not include it. The other is "select", *params* in Vega-Lite and *signals* in Vega are dynamic variables that drive interactive updates. They can be complex selections that map user input (e.g., mouse click, brush, drag, and zooming) to data queries, which are used for data exploration and designing interactive multi-view displays. As shown in Figure 4, they can also optionally be bound to input widgets (e.g., slider, drop-down menus, and checkboxes). Benefiting from the design logic of the system, GALVIS can retain this feature intact.

Interactive Editing Tool: Though users can conveniently create simple visualizations through drag-and-drop interactions, post-generation modifications still cannot be ignored. **A vital problem is how to enable users to modify the generated charts flexibly**. To this end, we design an interactive editing tool that consists of two parts: a template editor (see Figure 2 (V4)) for modifying visualization components and a Vega-Link-powered extension editor (see Figure 2 (V5)) for configuring auxiliary properties. Considering users with varying-level visualization skills, three different degrees of freedom are provided (Figure 5). The option configuration mode allows users to perform option-level editing rapidly, which is novice-friendly. The autocompletion and warning (⚠) mechanism can help users effectively complete input. The parameter extension mode extends the option configuration mode with the support to add and remove parameters. Users can add parameters in the three forms by clicking the addbox (⊞) on the left. The free configuration mode allows users to freely modify and even rewrite the text, which can be used to import open-source specifications as personal templates.

4 CONCLUSION

In this paper, we describe how GALVIS powers declarative programming with example galleries and enables an efficient and extensible visualization authoring pipeline. Our target users are declarative programming developers. We believe that GALVIS can help facilitate their coding process. In addition, novices can also find an appropriate way of visualization creation in GALVIS (e.g., using templates). GALVIS can even be used as an effective tool for learning Vega and Vega-Lite. In the future, we will explore more intelligent functionalities (e.g., recommendation[3, 19, 37] and natural language interface[12, 13, 28]).

REFERENCES

[1] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D3: Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 2301–2309.

[2] Çağatay Demiralp, Peter J. Haas, Srinivasan Parthasarathy, and Tejaswini Pedapati. 2017. Foresight: Recommending visual insights. *Proceedings of the VLDB Endowment* 10, 12 (aug 2017), 1937–1940.

[3] Rui Ding, Shi Han, Yong Xu, Haidong Zhang, and Dongmei Zhang. 2019. Quickinsights: Quick and automatic discovery of insights from multi-dimensional data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD'19*. ACM, Amsterdam, The Netherlands, 317–332.

[4] Flare dataset. https://vega.github.io/editor/data/flare.json. 2022.08.

[5] Flare-dependencies. https://vega.github.io/editor/data/flare-dependencies.json. 2022.08.

[6] Flourish. https://flourish.studio. 2022.08.

[7] Scarlett R. Herring, Chia Chen Chang, Jesse Krantzler, and Brian P. Bailey. 2009. Getting inspired!: Understanding how and why examples are used in creative design practice. In *Proceedings of CHI Conference on Human Factors in Computing Systems, CHI'19*. ACM, 87–96.

[8] Hollywood Most Profitable Stories dataset. https://www.kaggle.com/brendan45774/hollywood-most-profitable-stories. 2022.08.

[9] IMDB movies datasets. https://vega.github.io/editor/data/movies.json. 2022.08.

[10] 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 CHI Conference on Human Factors in Computing Systems, CHI'18*. ACM, 1–13.

[11] Yuyu Luo, Xuedi Qin, Nan Tang, and Guoliang Li. 2018. Deepeye: towards automatic data visualization. In *Proceedings of the 34th IEEE International Conference on Data Engineering, ICDE'18*. IEEE, Paris, France, 101–112.

[12] Yuyu Luo, Nan Tang, Guoliang Li, Chengliang Chai, Wenbo Li, and Xuedi Qin. 2021. Synthesizing Natural Language to Visualization (NL2VIS) Benchmarks from NL2SQL Benchmarks. In *Proc. 2021 Int. Conf. Manag. Data, SIGMOD'21*. ACM, Virtual Event, 1235–1247.

[13] Yuyu Luo, Nan Tang, Guoliang Li, Jiawei Tang, Chengliang Chai, and Xuedi Qin. 2022. Natural Language to Visualization by Neural Machine Translation. *IEEE Transactions on Visualization and Computer Graphics* 28, 1 (jan 2022), 217–226.

[14] Michele Mauri, Tommaso Elli, Giorgio Caviglia, Giorgio Uboldi, and Matteo Azzi. 2017. RAWGraphs: A visualisation platform to create open outputs. In *Proceedings of the 12th Biannual Conference on Italian SIGCHI Chapter, CHItaly '17*. ACM, 1–5.

[15] Andrew M McNutt and Ravi Chugh. 2021. Integrated Visualization Editing via Parameterized Declarative Templates. In *Proceedings of CHI Conference on Human Factors in Computing Systems, CHI'21*. ACM, 1–14. arXiv:2101.07902

[16] Dominik Moritz, Chenglong Wang, Greg L. Nelson, Halden Lin, Adam M. Smith, Bill Howe, and Jeffrey Heer. 2019. Formalizing Visualization Design Knowledge as Constraints: Actionable and Extensible Models in Draco. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (2019), 438–448.

[17] National data of China. https://data.stats.gov.cn/login.htm. 2022.08.

[18] Deokgun Park, Steven M. Drucker, Roland Fernandez, and Niklas Elmqvist. 2018. Atom: A Grammar for Unit Visualizations. *IEEE Transactions on Visualization and Computer Graphics* 24, 12 (2018), 3032–3043.

[19] Xin Qian, Ryan A Rossi, Fan Du, Sungchul Kim, Eunyee Koh, Sana Malik, Tak Yeon Lee, and Joel Chan. 2021. Learning to Recommend Visualizations from Data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, KDD'21*. ACM, Virtual Event, 1359–1369.

[20] Xuedi Qin, Yuyu Luo, Nan Tang, and Guoliang Li. 2020. Making data visualization more efficient and effective: a survey. *The VLDB Journal* 29, 1 (2020), 93–117.

[21] Donghao Ren, Tobias Höllerer, and Xiaoru Yuan. 2014. IVisDesigner: Expressive interactive design of information visualizations. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (2014), 2092–2101.

[22] Donghao Ren, Bongshin Lee, and Matthew Brehmer. 2019. Charticulator: Interactive Construction of Bespoke Chart Layouts. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (2019), 789–799.

[23] Arvind Satyanarayan and Jeffrey Heer. 2014. Lyra: An interactive visualization design environment. *Computer Graphics Forum* 33, 3 (2014), 351–360.

[24] 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* 26, 1 (2019), 1–11. arXiv:1907.13568

[25] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2017. Vega-Lite: A Grammar of Interactive Graphics. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2017), 341–350.

[26] Arvind Satyanarayan, Ryan Russell, Jane Hoffswell, and Jeffrey Heer. 2016. Reactive Vega: A Streaming Dataflow Architecture for Declarative Interactive Visualization. *IEEE Transactions on Visualization and Computer Graphics* 22, 1 (2016), 659–668.

[27] Arvind Satyanarayan, Kanit Wongsuphasawat, and Jeffrey Heer. 2014. Declarative interaction design for data visualization. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology, UIST'14*. ACM, 669–678.

[28] Leixian Shen, Enya Shen, Yuyu Luo, Xiaocong Yang, Xuming Hu, Xiongshuai Zhang, Zhiwei Tai, and Jianmin Wang. 2022. Towards Natural Language Interfaces for Data Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics* (sep 2022), 1–20.

[29] Leixian Shen, Enya Shen, Zhiwei Tai, Yiran Song, and Jianmin Wang. 2021. TaskVis: Task-oriented Visualization Recommendation. In *Processings of Eurographics Conference on Visualization, EuroVis'21 Short Paper*. Eurographics.

[30] Arjun Srinivasan, Steven M. Drucker, Alex Endert, and John Stasko. 2019. Augmenting visualizations with interactive data facts to facilitate interpretation and communication. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (2019), 672–681.

[31] Vega Examples. https://vega.github.io/vega/examples/. 2022.08.

[32] Vega-Lite Examples. https://vega.github.io/vega-lite/examples/. 2022.08.

[33] Vega/Vega-Lite themes. https://github.com/vega/vega-themes. 2022.08.

[34] Chenglong Wang, Yu Feng, Rastislav Bodik, Isil Dillig, Alvin Cheung, and Amy J. Ko. 2021. Falx: Synthesis-Powered Visualization Authoring. In *Proceedings of CHI Conference on Human Factors in Computing Systems, CHI'21*. ACM, 1–15.

[35] Hadley Wickham. 2010. A Layered Grammar of Graphics. *Journal of Computational and Graphical Statistics* 19, 1 (2010), 3–28.

[36] Kanit Wongsuphasawat, Zening Qu, Dominik Moritz, Riley Chang, Felix Ouk, Anushka Anand, Jock MacKinlay, Bill Howe, and Jeffrey Heer. 2017. Voyager 2: Augmenting visual analysis with partial view specifications. In *Proceedings of CHI Conference on Human Factors in Computing Systems, CHI'17*. ACM, 2648–2659.

[37] Mengyu Zhou, Qingtao Li, Xinyi He, Yuejiang Li, Yibo Liu, Wei Ji, Shi Han, Yining Chen, Daxin Jiang, and Dongmei Zhang. 2021. Table2Charts: Recommending Charts by Learning Shared Table Representations. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD'21*. ACM, Virtual Event, 2389–2399.

[38] Jonathan Zong, Dhiraj Barnwal, Rupayan Neogy, and Arvind Satyanarayan. 2021. Lyra 2: Designing interactive visualizations by demonstration. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2021), 304–314.