RidgeBuilder: Interactive Authoring of Expressive Ridgeline Plots

Shuhan Liu
State Key Lab of CAD&CG,
Zhejiang University
Hangzhou, China
shliu@zju.edu.cn

Yangtian Liu
State Key Lab of CAD&CG,
Zhejiang University
Hangzhou, China
yt-liu@zju.edu.cn

Junxin Li
State Key Lab of CAD&CG,
Zhejiang University
Hangzhou, Zhejiang, China
College of Information Sciences and Technology,
Hangzhou Normal University
Hangzhou, Zhejiang, China
lijunxin@stu.hznu.edu.cn

Yanwei Huang
State Key Lab of CAD&CG,
Zhejiang University
Hangzhou, China
huangyw@zju.edu.cn

Yue Shangguan
University of Texas at Austin
Austin, Texas, United States
sgyersula@utexas.edu

Zikun Deng
School of Software Engineering
South China University of Technology
Guangzhou, China
zkdeng@scut.edu.cn

Di Weng*
School of Software Technology,
Zhejiang University
Ningbo, China
dweng@zju.edu.cn

Yingcai Wu
State Key Lab of CAD&CG,
Zhejiang University
Hangzhou, China
ycwu@zju.edu.cn

Abstract

Ridgeline plots are frequently employed to visualize the evolution or distributions of multiple series with a pile of overlapping line, area, or bar charts, highlighting the peak patterns. While traditionally viewed as small multiple visualizations, their ridge-like patterns have increasingly attracted graphic designers to create appealing customized ridgeline plots. However, many tools only support creating basic ridgeline plots and overlook their diverse layouts and styles. This paper introduces a comprehensive design space for ridgeline plots, focusing on their varied layouts and expressive styles. We present RidgeBuilder, an intuitive tool for creating expressive ridgeline plots with customizable layouts and styles. In particular, we summarize three goals for refining the layout of ridgeline plots and propose an optimization method. We assess RidgeBuilder's usability and usefulness through a reproduction study and evaluate the layout optimization algorithm through anonymized questionnaires. The effectiveness is demonstrated with a gallery of ridgeline plots created by RidgeBuilder.

*Di Weng is the corresponding author.

CHI '25, April 26-May 1, 2025, Yokohama, Japan

ACM ISBN 979-8-4007-1394-1/25/04
https://doi.org/10.1145/3706598.3714209

CCS Concepts

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

Keywords

Ridgeline plot, visualization authoring, creative design support

ACM Reference Format:
Shuhan Liu, Yangtian Liu, Junxin Li, Yanwei Huang, Yue Shangguan, Zikun Deng, Di Weng, and Yingcai Wu. 2025. RidgeBuilder: Interactive Authoring of Expressive Ridgeline Plots. In *CHI Conference on Human Factors in Computing Systems (CHI '25), April 26-May 1, 2025, Yokohama, Japan.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3706598.3714209

1 Introduction

Ridgeline plots are a type of data visualization that comprises a pile of line, area, or bar charts stacked closely on top of each other to form a distinctive pattern similar to mountain ridges. Initially created in the 1970s to display pulsar radio signals[13], such graphics have achieved broad cultural resonance as their stacked waveform-like patterns formed pulses that communicate strong emotions [12]. Despite having emerged for quite a long time, ridgeline plots were only widely recognized as a data visualization form. They became popular in the visualization community in 2017 after Henrik Lindberg shared his famous artwork [28], and Claus O. Wilke created an R package called ggridges [67]. The plot was initially named "joyplot" in April 2017 [8] but later renamed to ridgeline plots due to inappropriate connotations [66].

Nowadays, ridgeline plots have gained widespread popularity and serve both scientific and artistic purposes. The ggridges package

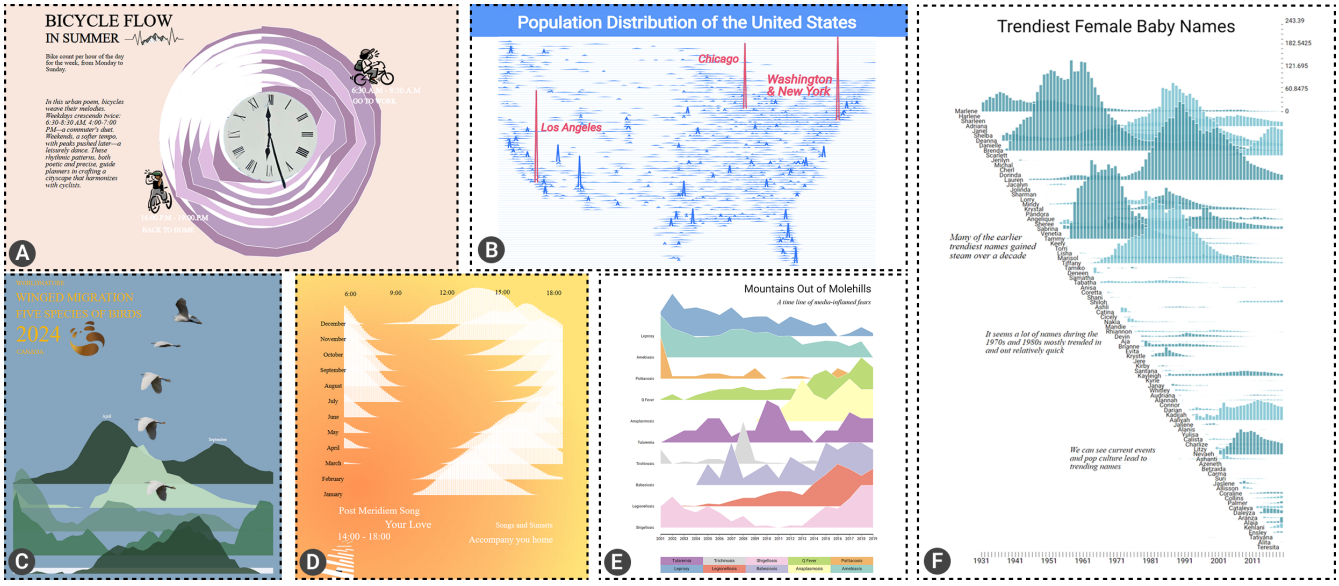

Figure 1: Six ridgeline plots recreated with RidgeBuilder. These six cases demonstrate RidgeBuilder's expressiveness in adjusting layout, styles, data mappings, and embellishments to create informative and visually appealing ridgeline plots. (A) displays Washington, D.C. station bicycle flows during the summer months with a cyclic layout revealing the peaks of rush hours. (B) visualizes the distribution of the U.S. population with a line-based multi-layer ridgeline plots (adapted from [11]). (C) shows Google Trend of bird migration in Canada with diverse embellishments (using data from [56]). (D) visualizes a person's music listening trend over a year (adapted from [37]). (E) employs RidgeBuilder's recommended layout and uses color hue mapping to display a timeline of media fear of infectious diseases (adapted from [32]). (F) visualizes the trendiest female baby names with an unaligned x-axis (adapted from [69]).

emerges as the most popular visualization library for creating ridgeline plots, which has been downloaded over 60,000 times in August 2024 and cited in hundreds of scientific publications. The aesthetics of the ridgeline plots also attract many designers to extend the plots and create appealing visualization artworks [11, 28, 32, 58, 69] as shown in Fig. 2. These artworks showcase the expressive power of ridgeline plots as a scalable visualization form for presenting the evolution or distributions of multiple variables and are well-liked by the community.

However, authoring a compelling ridgeline plot remains a challenging task. The layout of a ridgeline plot, such as the degree of overlap and the order of ridges, has a strong influence on its effectiveness. For example, if the scale parameter, which dictates the extent of overlap, is not finely tuned to the represented data, key patterns, such as temporally consecutive peaks, may become obscured, leading to potential misinterpretation of the data. Furthermore, creating diverse artistic effects in the ridgeline plots, such as interleaved colors and progressively indented axes as shown in Fig. 2, requires labor-intensive efforts and a solid proficiency with graphic design software like Adobe Illustrator [2].

The existing approaches that support the authoring of ridgeline plots can be divided into two categories: code-based and interactive approaches. The code-based approaches include ggridges [67], Vega-Lite [48], D3.js [6], and Observable Plot [40]. While these approaches offer considerable flexibility, they come with a significant

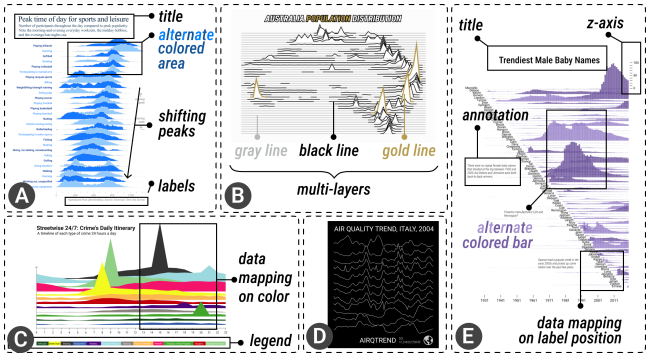

Figure 2: This figure displays five visualization artworks based on ridgeline plots, reproduced from existing artworks [11, 28, 32, 58, 69], using the datasets [10, 22, 50, 57, 61] Many visual channels encode diverse information. The color, position, order, mark type, labels, and backgrounds expressively convey visual patterns like geological distributions, trends, and pulses.

learning curve and necessitate a certain level of programming expertise for effective customization. The interactive approaches like PowerViz [42] and Tableau [46] simplify the creation of traditional ridgeline plots by offering user-friendly interfaces; however, they are limited in terms of the complexity of layout and the variety of

embellishment options they provide. Consequently, designers often engage in meticulous efforts to customize the layout and style of ridgeline plots in other sophisticated design software. For example, a user may need to adjust the order of ridges one by one to achieve a desired shifting-peak pattern (Fig. 2A).

The limitations of the current approaches motivate us to develop RidgeBuilder, an interactive authoring tool that facilitates the easy creation and customization of diverse and expressive ridgeline plots. Based on a comprehensive survey of the existing ridgeline plots from diverse sources, we first delineate a design space for ridgeline plots that comprises five dimensions: element, layout, style, embellishment, and composition. Subsequently, we distilled four design goals through a series of interviews, which have been instrumental in shaping RidgeBuilder. The proposed tool offers not only an intuitive approach to diverse layout creation but also a suite of style customization options and templates, empowering users to craft sophisticated ridgeline plots tailored to their needs effortlessly.

The effectiveness of RidgeBuilder is evaluated through a recreation study, where the participants were asked to recreate two ridgeline plot designs with RidgeBuilder, and an algorithm evaluation study, where the performance of RidgeBuilder's layout optimization algorithm was benchmarked against that of established baselines through the use of anonymized questionnaires. A gallery of the expressive ridgeline plots created by RidgeBuilder is also curated and featured in Fig. 1 and on the project page[1]. To our knowledge, this study is one of the first efforts to thoroughly investigate the design space and authoring aspects of diverse ridgeline plots. Our objective is to spark greater interest within the information visualization community regarding this visualization form and to encourage visualization practitioners to delve into its expansive potential applications.

2 Related Work

This section reviews research on ridgeline plots from the perspectives of design aesthetics and authoring approaches.

2.1 Ridgeline Plot Visualization

We discuss ridgeline plots in the visualization community to outline the development of the design aesthetics. To fairly investigate the development of ridgeline plots in the visualization academic community, we conduct a systematic survey. We list keywords, including *ridgeline plot/chart*, *ridge plot/chart*, *joy plot/chart*, *stacked area chart/plot*, and *stacked line chart/plot*. Then, we perform a strict search in prominent visualization journals and conference proceedings, such as *IEEE TVCG*, *CGF*, *JoV*, *PacificVis*, and *SIGCHI*. Existing studies on ridgeline plots can be divided into two categories: application and theory.

In terms of application, some visual analytics systems employ ridgeline plots and ridgeline-like visualization. CriPAV [20] and Facetto [23] are the only two application studies in the decade that explicitly mentioned the ridgeline plot or joy plot. Many other studies mainly use ridgeline-like plots, including faceted area charts (e.g. Screenit [15], PlanningVis [51]) and faceted bar charts (e.g. ECoalVis [29], LiveRetro [68]). These applications avoid the problem of overlapping neighbor ridges to ensure no visual occlusion, but it comes at the expense of scalability and the loss of distinctive "ridgeline" features. Additionally, the horizon chart is a type of plot that may be mistaken for a ridgeline plot due to its similar appearance [18]. However, the horizon chart has fixed bands with no overlap between them, whereas ridgeline plots feature overlaps between different ridges.

In terms of theory, there is a scarcity of studies that delve into the design space, weigh the advantages and disadvantages, or consider the design considerations of ridgeline plots. Only Moritz et al. [36] discussed the limitation of the ridgeline plot when visualizing hundreds of time series. Besides, Lekschas et al. [25] regarded the ridgeline plot as small multiples and generally surveyed the scope and considerations. Furthermore, several general-purpose tools have explored common layouts and styles for visualization charts, including Charticulator's [44] examination of layout and Data Illustrator's discussion on partition and repeats [31]. Nonetheless, these broad explorations have yet to provide a comprehensive definition of the design space and the application of diverse styles within the context of ridgeline plots.

In conclusion, despite the active usage of ridgeline plots in the fields of infographic design and other non-visualization domains, the visualization academic community has paid little attention to this type of plot. Thus, this paper comprehensively explores the design space and supports authoring expressive ridgeline plots.

2.2 Visualization Authoring Approaches

In this section, we explore a range of visualization authoring tools, encompassing both general-purpose tools and specialized tools crafted for specific authoring purposes.

For the **general-purpose visualization authoring tools**, we extend the categories introduced in prior surveys [21, 33] and divide these tools into five types: *code-*, *configuration-*, *demonstration-*, *example-*, and *Natural-Language-Instruction (NLI)*-based approaches.

Code-based approaches that support the creation of ridgeline plots mainly include ggridges [67], D3.js [6], and Vega-Lite [48, 59]. Additionally, some studies use ridgeline plots as a basic distribution visualization chart to explore the syntax of conditional probability visualization [43]. However, these tools rely on a specific programming language and have a particular syntax, resulting in a higher learning curve compared to no-code methods. Moreover, creating expressive ridgeline plots using code-based methods requires a high level of user expertise.

Configuration-based tools allow users to map data onto different visual channels through shelf configurations. In the decade, many configuration-based methods have emerged, such as Data Illustrator [31], Charticulator [44], and Lyra [47]. These methods are widely popular due to their no-code approach. Among these methods, Tableau [46], PowerViz [42], and EnrichVisBox [64] support creating ridgeline plots, but users still need to polish their work using other creative design tools (e.g. Adobe Illustrator [2]) if they desire a customized layout instead of a basic template one.

To further reduce the mental and interaction burden on users, many new visualization creation interactions have been proposed in recent years. *Demonstration-based* methods (e.g. Lyra2 [72]) allow users to drag data onto visualization channels, enabling them to see what they get during the creation process. *Example-based methods*

[1]https://github.com/RidgeBuilder/RidgeBuilder

(e.g. Falx [62]) allow users to provide a pair of input data and output visualizations to generate the target visualization automatically. With the rise of large language models, NLI-based methods (e.g. ChartGPT [55], Data Formulator [63]) further reduce the burden of creating visualizations. However, these emerging intuitive visualization creation interaction tools only consider simple and common visualizations regarding feasibility and generality so visualizations with complex layouts like ridgeline plots are ignored.

Specific visualization authoring studies can be divided into three aspects according to the creative intent of visualization: *effectiveness-*, *efficiency-*, and *expressiveness-oriented*.

Effectiveness-oriented studies focus on how to create unbiased visualization. These works discuss how to create the correct visualization chart from multiple aspects such as aesthetics, visualization theory, and cognitive science. For example, for the storyline, Tanashi et al [52]. proposed three principles for optimizing the layout. GeoLinter [24] helps create correct map charts. *Efficiency-oriented studies* focus on how to help users create specific visualizations more quickly, mainly through improving the optimizing algorithms (e.g. Storyflow [30]). *Expressiveness-oriented studies* focus on creating interesting and expressive visualizations. Many methods (e.g. Plotthread [53], iStoryline [54]) are designed for a specific visualization form - storyline. Some are designed for specific types of data, like time-varying data [41, 49] and high-dimensional data [70, 71].

Through extensive research, we are inspired by existing works. We learn the difficulties and opportunities of ridgeline plot authoring in terms of expressiveness, effectiveness, and efficiency. As this visualization technique is relatively unexplored in the academic community, we have chosen to focus on its expressiveness in this paper. Our goal is to not only contribute to the understanding of ridgeline plots but also to generate interest and encourage more research on this intriguing chart.

3 Ridgeline Plot Design

In this section, we aim to summarize a design space for ridgeline plots that facilitates expressive creation by conducting a content survey of existing ridgeline plots collected from diverse sources.

3.1 Survey Process

The survey of ridgeline plots included two stages: the collecting stage and the labeling stage. Three co-authors, with more than two years of work experience in visualization design and implementation, were involved in the survey process.

In the collecting stage, the three co-authors used search engines (e.g. Google Search, Google Scholar, and Google Image) to collect diverse existing ridgeline plots. The search constraints include the time range (2017-2024) and strictly match keywords (*ridgeline plot/chart*, *ridge plot/chart*, or *joy plot/chart*). This list is different from the one we list in section 2 because we found the keywords *stacked area chart/plot* and *stacked line chart/plot* ineffective and removed them. Few studies with the two keywords are relevant to ridgeline plots. We delegated the task to three co-authors responsible for collecting ridgeline plots from a specific time range: 2017-2019, 2020-2021, and 2022-2024, respectively. After the collection, three co-authors collaboratively merged and deleted duplicated ridgeline plots, as two duplicated ridgeline plots may be from different sources but look the same.

In the labeling stage, as the three co-authors gained some insights during the collecting stage, we constructed a basic category through brainstorming and divided it roughly into three dimensions: *elements (axes, ridges)*, *layouts (overlap, order)*, and *styles (color, opacity)*. The merged collection of ridgeline plots is divided equally into three groups. Each co-author was responsible for two groups to decrease ambiguity and bias via cross-validation. Finally, the three co-authors discussed the exceptional cases and extended the space. We also discussed the divergent cases to reach an agreement.

3.2 Design Space

We collected a dataset of 182 ridgeline plots (273 in total, removed 91 duplicated records). We summarized a design space based on the dataset. The design space consists of five dimensions: element (*ridge, axis, optional elements*), layout (*transform, layer, order, overlap*), style (*color, opacity, border*), embellishment (*annotation, background, decoration*), and composition. The overview of the design space is shown in Fig. 3. Complete details and the dataset can be found on our homepage[2] and the folder "design scope" in the supplementary materials. In this section, we detail the design space dimensions, indicating the type of each dimension and the corresponding number of instances. e.g. "area (170)" signifies that there are 170 ridgeline plots in the dataset utilizing "area" as the ridge.

3.2.1 Element. ***Elements*** refer to the visual components that make up the ridgeline plot. A ridgeline plot typically includes three basic visual elements: ridge, axis, and optional elements.

Ridge. This is the main visual elements of ridgeline plots. The ridge refers to the basic shape that reveals value changes for each variable. Its basic shapes come in four types: area (170), bar (3), line (8), and step (1). Line- and step-based ridges have better scalability than area- or bar-based ones because they require less space. Still, the area and bar ridges are more expressive because the filled color can reveal more information.

Axis. A ridgeline plot includes x-, y-, and z-axes. The x-axis is always laid out horizontally and is shared by all ridges, typically encoding time (30), ratio (136), longitude (5), level (1), age (3), and score (7). The y-axis usually encodes discrete data, whether ordinal (16) or nominal (166). Finally, the z-axis typically encodes continuous ratio value (166) and discrete count value (16).

Optional elements. This includes the mode line and the reference line. The mode line is a vertical line that marks a special value of each ridge. It usually marks the median value and the extreme peak value, revealing a summary statistic to compare the relative positions of different ridges. The reference line, on the other hand, is a vertical line that usually represents a threshold for all ridges.

3.2.2 Layout. We discuss ***Layouts*** of the ridgeline plot from a macro level (the entire plot) to a micro level (an individual ridge).

[2] https://ridgebuilder.github.io/RidgeBuilder/

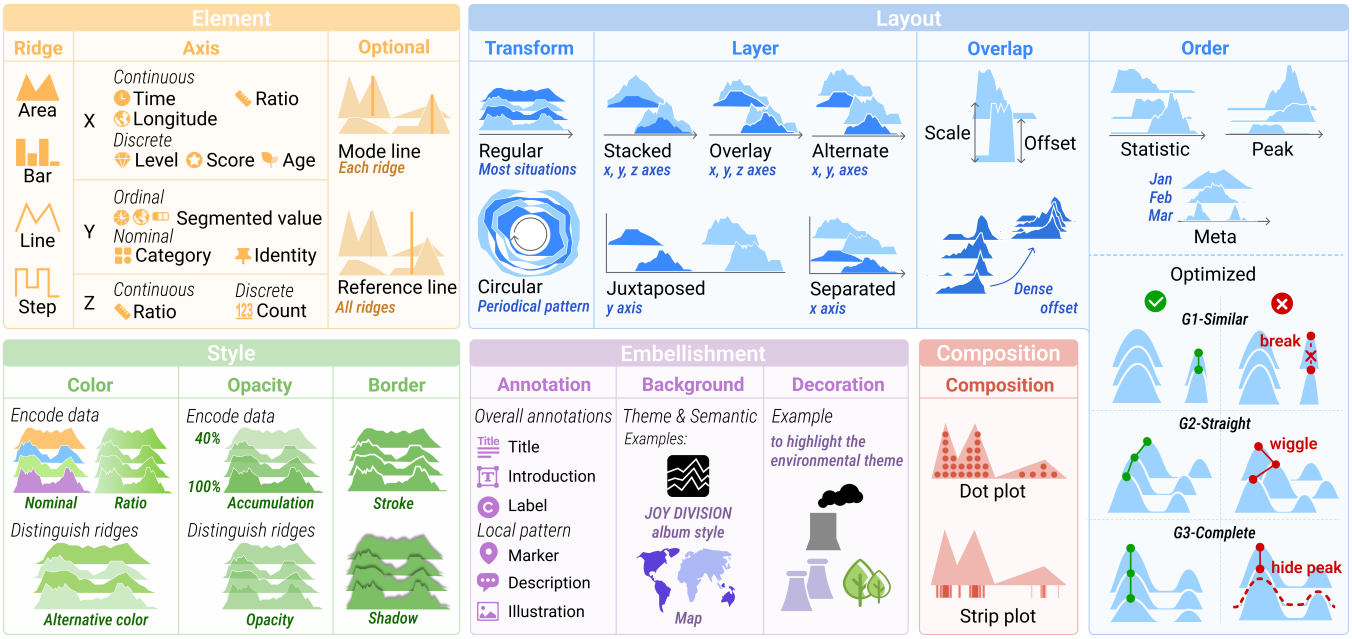

Figure 3: This figure displays the design space of ridgeline plots. The design space consists of five dimensions: element, layout, style, embellishment, and composition. We discuss categories of each dimension and extensively summarize its contents. This figure also includes several examples.

Transform. The transformation of the ridgeline plot describes the overall shape of the plot. Most plots follow the left-to-right regular layout (182), aligning the ridges so that they are evenly spaced. The regular layout is suitable for most cases and is used to compare the distribution of data across different variables. We extensively add the circular layout inspired by silhouette graphs [3]. The circular layout is used when the data being plotted exhibits a periodic pattern, such as when the x-axis represents a periodical timeline.

Layer. A ridgeline plot can consist of multiple layers, each of which contains a group of ridges. For example, one layer represents a controlled group, while the other layer represents an experimental group. Five main layer layouts can be used in a ridgeline plot. Overlay (17): This layout is used when different groups record similar semantic values, and the layers share the same x, y, and z-axis. The overlay layout is typically used when the goal is to compare the distribution of data across different groups. Stacked (0): This is used when layers share the same x and y-axis but are stacked on the z-axis. We add this layout because it is useful when the goal is to show the total distribution of data across different groups while also highlighting the differences between them. Alternated (2): This is used when different groups record different semantic values, and the layers share the same x and y-axis. Juxtaposed (9): This is used when layers only share the same y-axis. It is similar to the alternate layout but is typically used when there are few similar ridge-like patterns between two layers to avoid pattern hidden. Separated (8): This is used when layers record different semantic values and only share the x-axis.

Order. Different orders of ridges convey different visual patterns. Typical sort methods include three types. The statistic-based (30) sorts ridges of specific statistical value, such as the maximum and the mean value. The statistic value can reveal the distribution of peaks. For example, if a peak appears early, then the mean value of that ridge will be small. Thus, this method is usually used to highlight a peak-shifting pattern. The meta-based (51) method requires that the y-axis must be ordered. For example, the y-axis maps to twelve months. This method can convey the pattern varying along with the y-axis. The peak-based method (9) is usually adopted when the domain ranges of different ridges vary significantly. If all ridges have the same scale on the z-axis, the ridge with a larger domain range will require more space. Thus, the ridge with a larger domain range is laid at the top to avoid hiding other ridges. The order of ridges can also be optimized to highlight "ridgeline" pattern. The "ridgeline" pattern, characterized by several continuous, shifting peaks, effectively reveals the evolution of each ridge (see Fig. 2A). To pinpoint key factors influencing the pattern, we analyzed our collected ridgeline plots, particularly those with salient ridgelines. We define "salient ridgelines" as lines with discernible trends, identifiable through Gestalt theory [65], offering users analytical insights such as evolution. In the dataset, 72 ridgeline plots meet the criteria. We summarized their common characteristics from diverse aspects, including relations, shapes, and layouts of the ridges, and finally identified three main design goals. These goals are illustrated in Fig. 3-layer-order with examples.

G1. ***Neighboring ridges should have similar peak patterns.*** Examples with this characteristic constitute 69/72. We find that placing similar peaks nearby can form a continuous

ridgeline and reduce visual disruptions, which aids in identifying patterns and trends in the data.

G2. ***There should be fewer wiggles in ridgelines.*** Examples with this characteristic constitute 46/72. We find that a straight ridgeline more accurately represents how peaks shift than a ridgeline with many wiggles, which can obscure important details.

G3. ***The layout should avoid hiding peaks.*** Examples with this characteristic constitute 66/72. In ridgeline plots, peaks reveal the main distribution of the data, and we find that hiding peaks can lead to a loss of important information about the distribution.

Overlap. The overlap degree of a ridgeline plot is determined by two parameters: the scale of each ridge and the offset between each pair of neighbor ridges. If the ridge scale is fixed, decreasing the ridge offset increases overlap and vice versa. A smaller overlap between ridges indicates that there are fewer ridges and less hidden data, but the ridgeline is less obvious, while dense ridges being hidden more have an obvious ridgeline.

3.2.3 Styles. ***Styles*** of visual elements are diverse. There are many stylish channels, including but not limited to color, stroke, opacity, and shadow. We mainly discuss styles that have specific usage in ridgeline plots.

Color. Ridges can be assigned different color hues based on various schemes. Sometimes, the color hues of ridges encode data (89). The encoding principle follows existing studies [38]. Additionally, filling ridges with different color hues can also differentiate them (59). For instance, using alternating color hues can help distinguish between neighboring ridges.

Opacity. The opacity of ridges is often used to reduce overlap and to help distinguish neighboring ridges (69). It can also be an effective visual tool for encoding ordinal data (5) in certain situations. For example, suppose the y-axis represents the progress of a project (10%, 20%, . . . , 100%). In cases where a user wants to emphasize the accumulation pattern along the y-axis, opacity can be used to encode the percentage of completion.

Border. Stroke (158) is commonly used in area- and bar-based ridgeline plots but is seldom used in line- and step-based ones. We add the shadow effect as it also seems like a "border" and can help differentiate neighboring ridges.

3.2.4 Embellishments. Expressive ridgeline plots always include diverse ***Embellishments***. Typical embellishments can be categorized into three types based on different intents: annotation, background, and decoration.

Annotation. Annotations vary greatly depending on the context of the visualization. It includes elements for overall visualization and local patterns. By using annotations effectively, designers can enhance the viewer's understanding of ridgeline plots.

Background. The background of ridgeline plots can be designed to match the subject matter in a specific thematic style, such as the classic black and white style used in Joy Division album style (see Fig. 2D). Additionally, the background can also be used to convey a specific semantic meaning, such as using a map to represent a geographic location (see Fig. 2B). Users employ pure-colored backgrounds or image backgrounds based on their design intents.

Decoration. Some cases include decorations that serve an aesthetic purpose. These decorations convey specific semantic information, further highlighting the theme.

3.2.5 Composition. During the survey, we found several cases that compose the ridgeline plot with other types of visualizations, particularly in scientific publications where the discrete sample values are marked as dots or stripes and placed on the plots. Since this study mainly focuses on the ridgeline plot itself, a comprehensive analysis of this particular aspect of the design space will be left for future research endeavors.

3.3 Reflections on Design Space

This section discusses our reflections on the design space of ridgeline plots. Although the design space of ridgeline plots overall follows the existing visualization framework, we made some extension attempts focusing on ridgeline plots' specific characteristics and also gained insights from the user's point of view.

In ***Elements*** scope, ridgeline plots are used to be treated as faceted x-y area plots, but we clarify that it would be better to describe the ridgeline plot using three axes: x, y, and z (ridge). On the one hand, three axes can better align with users' cognition. We found even authoring a basic ridgeline plot using general visualization grammars is challenging because users must learn advanced layout functions and align them with the faceted way of ridgeline plots, for example, users must know "bounds" in Vega [60] and know "nested layout" in Charticulator [44]. On the other hand, three axes can preserve the flexibility layouts of ridgeline plots. We found that if a grammar only supports two axes, it will fail to create ridgeline plots with alternate layers, such as Vega-Lite [48]. Although in some grammars (e.g., Observable Plot [39]) facet channels can also be used as encoding channels, users still must exert more efforts (e.g., redefine the scalers and domains) because facet options are typically designed for non-overlapping plots. This is also not directly mapped with users' cognition.

In ***Layouts*** scope, the proposed design space offers a more direct approach to describing expressive ridgeline plots. Within this space, new types of ridgeline plots can be generated, such as cyclic ridgeline plots and layer-stacked ridgeline plots. Furthermore, this space enables users to highlight different ridgeline visual patterns. For instance, setting the optimized G2-straight in the order dimension to 0 yields a "zig-zag ridgeline," while setting it to 1 produces a "straight ridgeline." By leveraging the options within this design space, users can easily specify intricate layouts and associated visual patterns.

In ***Styles*** scope, we highlight several distinctive styles associated with ridgeline plots. As ridgeline plots aggregate patterns of ridge overlap, they have evolved various style combinations to mitigate the adverse effects of overlap. For instance, techniques such as alternating colors and cumulative opacity can be employed. These style combinations also indicate the visual patterns inherent in ridgeline plots.

Overall, this design space enables users to describe expressive ridgeline plots more explicitly than existing ones. Users can create

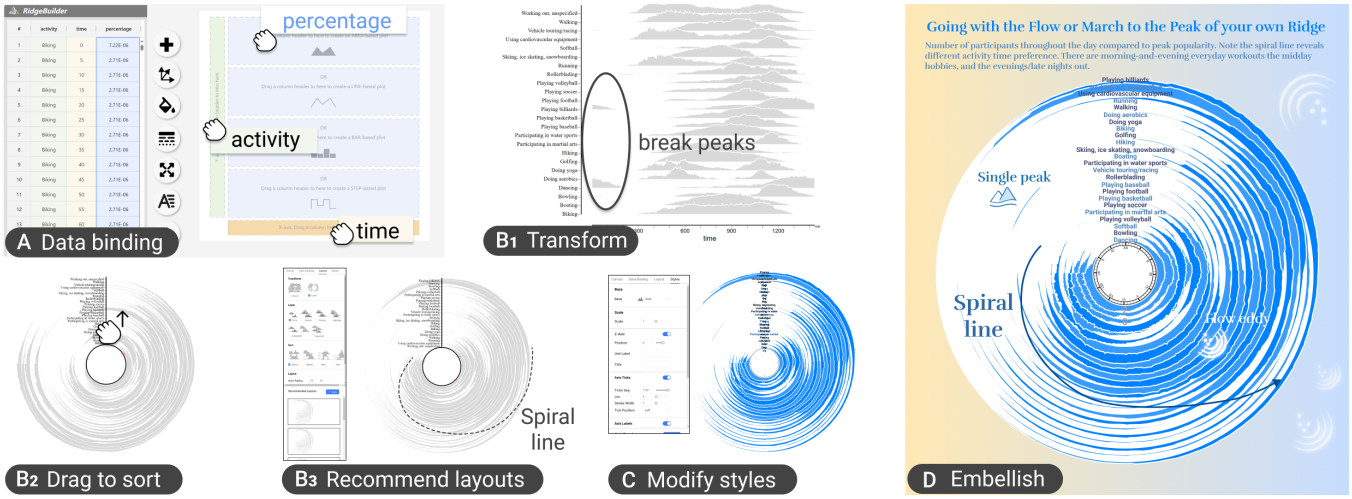

Figure 4: This usage scenario illustrates how a data analyst named Amy replicates a classical ridgeline plot artwork created by Lindberg [28] and customizes it for further exploring data patterns. Her workflows include binding data, adjusting layouts, modifying styles, and finally embellishing.

previously unattainable ridgeline plots and find it easier to describe a variety of existing ridgeline plots. This design space also helps us to understand more about the overlapping features of ridgeline plots. Furthermore, this design space allows users to highlight desired visual patterns, such as "zig-zag" ridgelines.

Based on these specific features of the ridgeline plot, we implement a ridgeline plot data model. Its declaration has three aspects of advances on the existing grammar: (1) we use three axes, x, y, and z, instead of facet; (2) we support diverse layout options, including the layer ways, the sort methods, and the direct specification of overlap dimension; (3) we allow users to specify special styles (e.g. alternative color) directly. Based on this data model, further declarative grammar can be built in the future to create ridgeline plots intuitively or even visualization plots that have layout-like properties like ridgeline plots. We also attached the specification of the data model as a proof of concept to Appendix 3.

4 Design Context and Goals

This section describes the usage scenario and goals that motivated RidgeBuilder's design. On the one hand, we aim to demonstrate the potential workflows and design choices that can be made by presenting a specific scenario. On the other hand, we conduct one-on-one interviews with potential users to identify the key design goals and objectives that would help ensure the tool meets their requirements.

4.1 Usage Scenario

This scenario is based on a dataset processed from the American Time Use Survey [57], which records the time used on sports and leisure activities of Americans. The processed dataset includes three columns: activity, percentage, and time. Imagine a data analyst, Amy, who saw the classical ridgeline plot created by Lindberg [28] on Twitter. Amy was impressed by how the ridgeline plot conveyed the temporal distribution of American activity preferences. She was intrigued by the shifting peaks formed by the plot, which seemed to create a "ridgeline" pattern. Therefore, Amy wanted to recreate a similar plot and customize it to explore the data further.

Binding data on the ridgeline plot. After uploading the dataset, Amy finds a template that guides her on how to bind the data. Since she wants to present the changes in activity over time, Amy drags the column "time" to the x-axis placeholder and "activity" to the y-axis placeholder. Next, she needs to select the basic shape of the ridges. Amy drags the real number column "percentage" onto the placeholder corresponding to the area-based ridgeline plots as she thinks the fill color can make the ridgeline plot more expressive. An initial grey ridgeline plot with many ridges appears on the screen.

Adjusting the layout to highlight the ridgeline. Amy finds two separated peaks corresponding to a 24-hour timeline divided at midnight, so she transforms the layout into a cyclic one to make the peaks appear head-to-tail. However, she is not satisfied with the current order of ridges as ridges for playing billiards and dancing have a distinct distribution. Amy then drags them to the top, but it still does not meet her requirements. So, she browses through recommended-ordered ridgeline plot layouts in the side panels and selects one that best matches her vision. The layout she chose features a spiral line of peaks (see Fig. 4B_3), signifying that the peak times for different sports activities vary across the day's timeline. Amy is satisfied now because she thinks the spiral line looks interesting.

Modifying styles. Amy finds the style of Lindberg's work [28] is in the template list, so she immediately applies it. Then, Amy adjusted the axes. She centers the Y-axis labels and removes tick lines to avoid overlap. Since she plans to use a clock image to show the corresponding time, Amy also removes the X-axis labels.

Embellishing the ridgeline plot. Amy finally embellishes the work. Now this circular visualization, which includes numerous ridges, resembles an eddy. It inspires Amy's creativity so she names the image "Going with the Flow or March to the 'Peak' of your

own 'Ridge"'. Amy hopes to emphasize the theme by adding an image, in which the left part with the sandy desert corresponds to the nighttime and the right part with the eddy water corresponds to the daytime. Then, she uses some shapes and texts to further highlight the shifting-peak pattern. Amy completes the authoring and exports it as an image.

4.2 Design Goals

We conduct one-on-one informal interviews[3] with eight participants (4 F, 4 M) to further explore specific design goals. Six experts have more than two years of experience in authoring visualizations, while two participants are novices. The interviews consist of two parts and are completed in 60 minutes. First, interviewees should imagine they are recreating four ridgeline plots and describe the workflow. Interviewees are allowed to use any familiar tools. They should also follow the think-aloud protocol [19]. The interviewees can stop and switch to the next ridgeline plot when they think they have completely expressed the authoring process of one plot. Second, interviewees should point out the most difficult part of the process and tell us the desired functions. We summarize four design goals from the interview.

DG1. Support direct data binding to create and edit ridgeline plots. This design goal includes two aspects. First, our interviewees prefer a template-based binding approach for direct element binding, particularly for the x, y, and z axes. Such binding simplifies plot creation by avoiding complex facet or element expansion interactions found in general authoring tools. Second, other interviewees emphasized that the system should provide direct specification of special data binding channels for ridgeline plots, such as cumulative opacity and special axis label position.

DG2. Support diverse ridgeline plot layouts. Interviewees stressed the importance of accommodating a variety of layout methods to cater to different ridge patterns, an area where existing tools fall short. For instance, some have tried adjusting ridge layer positions, e.g. order and offset, to emphasize specific distribution patterns. Others expressed a desire for cyclic layouts to represent periodicity. Moreover, interviewees sought guidance on optimized layouts, noting the challenges in achieving the desired peak-shifting visual effects with current authoring tools.

DG3. Support reducing the visual clutter caused by overlap using styles. Ridgeline plots capitalize on overlapping ridges to represent multiple variable distributions within a confined space, highlighting aggregated visual patterns. This overlap, while beneficial for density, can lead to visual occlusion. To mitigate this, ridgeline plots employ specialized styles, such as alternative colors, borders, and shadows, which are typically adjusted individually in general tools. Interviewees indicated a need for streamlined interaction within the system. Furthermore, ridgeline plots feature various popular style templates, such as the Joy Division album style and geoline style. The system should offer template recommendations to help new users learn about ridgeline plots.

DG4. Support customizable embellishments. To boost ridgeline plot expressiveness, users should be able to add embellishments like text labels to emphasize data points or enhance visual appeal. The system must support intuitive interactions, such as drag-and-drop, for both adding and editing embellishments.

[3]This interview has been approved by State Key Lab of CAD&CG, Zhejiang University.

5 RidgeBuilder

This section presents RidgeBuilder, an interactive web-based system designed to facilitate the authoring of ridgeline plots. RidgeBuilder offers four main functions: binding data, adjusting layouts, modifying styles, and adding embellishments, each corresponding to one design goal (**DG1-4**).

5.1 Implementation

RidgeBuilder consists of four key modules: the interface, an encoder, a layouter, and a parser. Each ridgeline plot is represented as a data model object, which is modified and updated as users interact with the system. The workflow of RidgeBuilder after a user interacts on the system is: (1) the encoder module updates the current data model object based on the user's actions; (2) the layouter generates alternative layouts and expands the data model object modified in the first step; (3) the parser module then interprets these data model objects generated in the first and second steps; (4) the interface displays the updated ridgeline plot in the main view with recommended alternative plots in the side panel. RidgeBuilder was implemented using React-Redux-TypeScript [1].

5.2 Data Binding

RidgeBuilder provides a data panel and two types of interactions to support data binding (**DG1**). The table-based data panel allows users to import and preview data before binding. For flexible data binding on general visual channels, such as color, users can map data onto the visual element by drag-and-drop interactions. For the necessary visual elements, such as the axes and ridges, RidgeBuilder provides data-binding templates to simplify the process and embeds simple principles to guide users in appropriate mapping.

Data panel. Before binding data to the visualization, users need to import a processed tidy data file in the .csv format into the data panel. The data panel displays the imported data as a table, with each row representing a data point and each column representing a variable. The column headers display the variable names. Users can drag the column headers to map them to different visual channels in the ridgeline plot.

Drag-and-drop data binding (Fig. 5A) This is designed to fulfill general data-binding requirements. RidgeBuilder follows existing studies (e.g. Lyra2 [72]) to support direct drag-and-drop data mapping. We summarize the available visual encoding channels into six categories: *position*, *fill*, *stroke*, *size*, *text*, and *shape*. A toolbar lists these channels with different icons. Users can select an icon from the toolbar (channel), a binding direction (x, y, z), and then drag a column onto a specific element to map a data column onto a visual channel.

Template-based data binding (Fig. 4A). This is designed for initial data binding on ***Elements***. RidgeBuilder displays a template with three placeholders corresponding to the x, y, and z channels to guide users in generating initial ridgeline plots. When a user drags a column header from the data panel, placeholders that can accommodate the column are highlighted. Once a placeholder is mapped to a column, the placeholder and the column turn the

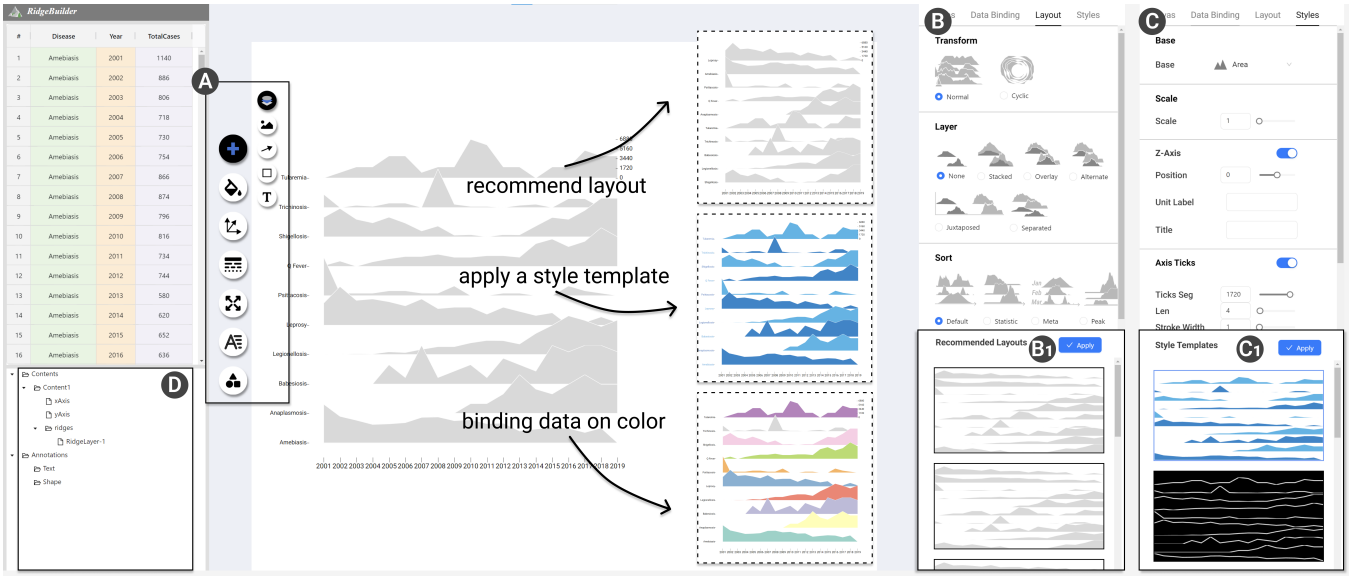

Figure 5: This shows the user interface of RidgeBuilder. (A) The toolbar lists icons for general data mapping. (B) displays predefined layout options and recommended layouts. (C) displays adjustable style options and available style templates. (D) is the layer tree of current added visual elements.

same color to remind users of the mapping. RidgeBuilder predefines rules to determine whether a data column can be mapped to a placeholder's corresponding visual channel, which follows the design space of ***Elements***.

5.3 Adjusting Layout

RidgeBuilder allows users to use predefined layouts and incorporates an optimization algorithm to help users find optimal layouts to address their layout needs (**DG2**).

Predefined layouts (Fig. 5B). Users can switch the side panel to obtain an overview of optional predefined layouts. We summarize different templates for diverse ***Layouts***. In addition to the normal transform layout, RidgeBuilder supports the cyclic layout by introducing polar coordinates. For layers, RidgeBuilder supports overlay, stacked, alternated, juxtaposed, and separated layers. In terms of order, RidgeBuilder provides statistic-, meta-, and peak-based approaches. The statistic-based method computes a representative x-value, like the mean value of each ridge, and sorts these ridges. The meta-based method leverages predefined rules (e.g. month) to obtain a possible order of the metadata. The peak-based method computes the maximum z-value of each ridge and sorts them in descending order to avoid hidden ridges. Furthermore, RidgeBuilder enables users to drag each ridge's label, allowing for a more flexible layout by adjusting their order. Besides these layout dimensions, users can move sliders in the panel to adjust parameters, such as the offset between ridges.

Optimized layout (Fig. 5B$_1$). We propose an optimal algorithm for recommending layouts that highlight the "ridgeline" pattern.

During our analysis of the dataset, we observed that due to the potential presence of multiple peaks, it is sometimes necessary to make trade-offs among the three goals (**G1, G2, G3**) summarized in the design space. As shown in Fig. 6A, a ridgeline plot has several ridges, while each ridge has many peaks. Given that a single ridge can possess multiple peaks, it is possible for a ridgeline plot to display several ridgelines simultaneously. Therefore, when configuring the layout, it is essential to consider the interplay among these multiple ridgelines. Different ridgelines within the same plot may have distinct optimal arrangements (Fig. 6A$_1$ and A$_2$), so we must balance various design goals. Inspired by studies on visualization layout trade-offs [4, 52], we formulate the process of arranging ridges as an optimization problem to address these three goals.

To simplify the narration, we illustrate necessary notations and the following three equations using an example in Fig. 6B. For **G1**, RidgeBuilder detects peaks P_i for each ridge r_i. Each peak $p_{ik} \in P_i$ is recorded as a tuple (x_{ik}, z_{ik}), where x is the horizontal position of the peak and z is the peak height. The distance between two peak p_{ik} and p_{jt} is defined by their horizontal distance $|x_{ik} - x_{jt}|$. We evaluate the similarity between two ridges by the average peak distance $d_1(r_i, r_j)$.

$$d_1(r_i, r_j) = \frac{1}{|P_i||P_j|} \sum_{k}^{P_i} \sum_{t}^{P_j} |x_{ik} - x_{jt}| \tag{1}$$

, where $|P_i|$ and $|P_j|$ indicate the peak counts of ridges r_i and r_j.

For **G2**, we find that if edges linking peaks keep in the same direction, there will be fewer wiggles. Thus, we modify d_1 to differentiate between peaks linked in the same direction and those not. We select left-to-right as the optimization direction to simplify the problem (if we select the opposite direction, change signs in the equation). Specifically, for each peak of ridge r_i, we only consider the peaks of ridge r_j that are to the left of it and make a pair. We record the sum count of paired peaks as Z_{ij}. The refined distance

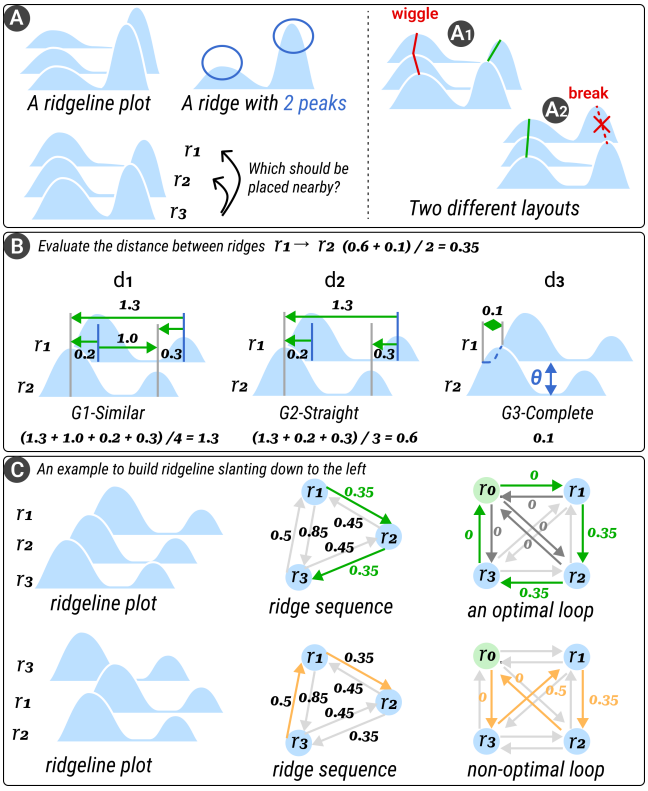

Figure 6: This figure illustrates the optimization goals and equations of ridgeline plots. (A) This illustrates the need to balance multiple design goals when the data presents multiple peaks in the layout. (B) We illustrate three equations for optimizing the layout of ridgeline plots using an example. We evaluate the similarity between ridges by computing the average peak distance between them. We only consider distances in one direction to avoid wiggles. The hidden distance is the horizontal length that a ridge is hidden by another ridge in front of it. (C) We display how to formulate the layout optimization problem as a Traveling Salesman Problem (TSP) using an example ridgeline plot with three ridges. The cost to place r_1 nearby and under r_2 is 0.35. We can transfer the ridge sequence order problem into a shortest loop search problem by introducing a virtual node with a distance of 0. It also illustrates optimal loop against non-optimal loop.

$d_2(r_i, r_j)$ is:

$$d_2(r_i, r_j) = \frac{1}{|Z_{ij}|} \sum_{k}^{P_i} \sum_{x_{ik} >= x_{jt}}^{p_{jt} \in P_j} x_{ik} - x_{jt} \tag{2}$$

For **G3**, we evaluate the hidden possibility between two ridges. Given an offset θ, we calculate the horizontal range of the ridge r_i that will be hidden by r_j for all data points.

$$d_3(r_i, r_j) = \frac{1}{M} \sum count(z_i < z_j + \theta) \tag{3}$$

, where M means the length of each ridge. Thus, the distance between two ridges is $d(r_i, r_j) = d_2(r_i, r_j) + d_3(r_i, r_j)$.

An optimized layout of a ridgeline plot aims to find an order of ridges that minimizes the sum of distances between neighboring ridges (given an offset θ), thus realize trade-off between three goals **(G1, G2, G3)** and highlight the "ridgeline" pattern. We denote the ordered i−th ridge by r_{y_i}. The optimization function is then:

$$\min_{y_1, y_2, \ldots, y_n} \sum_{i=1}^{n-1} d(r_{y_i}, r_{y_{i+1}}) \tag{4}$$

, where n is the total number of ridges. This formulation introduces a novel approach to the field by framing the layout problem as an optimization challenge.

The subsequent step involves developing an approach to determine the optimal result of Equation 4. This entails finding a permutation of integers from 1 to n (representing the order of ridges) that minimizes Equation 4. A direct method to achieve this is through exhaustive permutation enumeration, which yields the optimal solution. However, this approach becomes impractical for real-time optimization as n exceeds 10 due to the computational complexity. Given that a ridgeline plot can have anywhere from just a few to nearly 100 ridges, making exhaustive search impractical for larger datasets. An alternative is to seek an approximate solution, like using heuristic search [17] or genetic algorithms [35], which can efficiently provide a locally optimal solution.

Upon further investigation, we discovered that our problem could be mapped to the well-established Traveling Salesman Problem (TSP). The TSP is typically framed as finding the shortest possible route that visits each node in a graph exactly once and returns to the origin node [5]. Equation 4 describes a special case of TSP, known as Computer Wiring Problem [16], which does not consider going back to the starting point. Specifically, the primary distinction between the TSP and our problem is that TSP mandates a loop path starting and ending at the same node, whereas our problem involves a sequence from r_{y_1} to r_{y_n}. To bridge this gap, we can follow a typical transition method [5] by introducing a virtual node r_{y_0} connected to every other node with zero distance, transforming the problem into finding a loop path $r_{y_0} \rightarrow r_{y_1} \rightarrow \ldots \rightarrow r_{y_n} \rightarrow r_{y_0}$, like the example in Fig. 6C. Thus, our problem has been effectively reformulated into an equivalent TSP, allowing us to utilize mature solvers (e.g. DFJ algorithm [14]) to find the optimal solution in real-time, even for node scales near 100.

5.4 Modifying Styles

RidgeBuilder follows the widely used shelf configuration to support users in modifying styles and making the overall plot rich and colorful. Additionally, RidgeBuilder recommends templates based on ***Styles*** to provide users with reference and inspiration (**DG3**). RidgeBuilder features a layer tree and a style menu.

Style menu (Fig. 5C) and layer tree (Fig. 5D). We mainly follow shelf configuration methods in existing tools [48]. To modify the styles of a visual element, users can select the element in the canvas or in the layer tree. Adjustable styles of visual elements in the style menu are the same as those summarized in the design space section (Sec.3.2).

Style templates (Fig. 5C$_1$). We summarize several typical style templates from our collected dataset of ridgeline plots. For example, the JOY DIVISION album style has black background with white lines, while the Lindberg [28] style refers to the alternative color hue. Each template is saved as a data model object without metadata. Since many styling templates involve part of the embellishments, such as the background, we also include background in templates.

5.5 Embellishing

To address embellishing needs (**DG4**), RidgeBuilder allows importing additional elements and drawing shapes. According to the summarized ***embellishments*** ✨, RidgeBuilder provides three forms of embellishments: *text*, *shape*, and *image*. Users can create text and freeform shapes by clicking the toolbar. Images (.png, .jpg) are also allowed to be added to the canvas. In addition, if a user wants to further polish the ridgeline plot, RidgeBuilder allows the generated plot to be exported as an SVG file or an image file.

6 Evaluation

In this section, we evaluate[4] the proposed system, RidgeBuilder, and the layout algorithm qualitatively and quantitatively.

6.1 Reproduction Study

The main goal of this reproduction study [45] is to evaluate the usability and the learnability of RidgeBuilder as well as collect feedback.

6.1.1 Method. **Participants and datasets.** We recruited 12 participants (6 F, 6 M) from a diverse range of academic backgrounds, including computer science (4), software engineering (1), industrial design (1), digital media (2), geographic information system (1), math (1), and education (2). We prepared two ridgeline plot infographics, one displays a timeline of media-inflamed fears about diseases [32] (Similar to Fig. 2C), the other displays the trendiest female baby names [69] (Similar to Fig. 2E). Then, we collected two datasets for the reproduction, one records popular baby names [50] in the U.S., the other records the cases for infectious diseases [9].

Procedure and experiment settings. We conducted the study one-on-one with participants on a host computer with dual screens, a keyboard, and a mouse running Windows 10. RidgeBuilder was running on Google Chrome. The study consisted of four stages. (1) Tutorial [10 min]: we provided a tutorial on ridgeline plots and how to use RidgeBuilder. (2) Warm-up [10 min]: participants explored RidgeBuilder freely using a tutorial dataset [57], with the option to ask the first author for help if encountering problems. (3) Reproduction [40 min]: participants were shown one infographic at a time and given 1 minute of introduction on the theme and content, without direct mention of the data encoding interaction. For each infographic, participants had 5 minutes to think and 15 minutes to replicate using RidgeBuilder. Two infographics were presented one after the other on the side screen. Participants were not allowed to search the tutorial or ask for help during this stage. Although participants were not required to finish the reproduction as quickly as possible, if a participant could not finish an infographic in 15 minutes, this stage for that particular infographic would end. Participants were asked to follow a think-aloud protocol [19] to provide insight into their thought process while using RidgeBuilder. After reproduction, if there was still enough time, participants were asked to explore designs freely. (4) Semi-structured interview [10 min]: participants should fill in the System Usability Scale (SUS) questionaire [7] with a 5-point Likert scale. We collected their feedback on usability, learnability, and improvements. After the study, each participant was compensated $14.

Study record and analysis. On the one hand, we processed the SUS questionnaire results and analyzed them quantitatively. On the other hand, we used one online meeting software to record the screen recording and used the transcription and translation functions to record the whole process. We made thematic coding and qualitative analysis of the records and the interviews.

6.1.2 Quantitative Analysis of Results. The average SUS score of RidgeBuilder is 88.75 given by participants (detail see Fig. 7), while 84.1 is the bar score of A^+ based on the existing survey done by Jeff Sauro [27]. Most of our participants expressed a strong willingness to use RidgeBuilder in the future and commented that RidgeBuilder is interesting and useful. Many of our participants praised the fluency interactions and consistency designs. Following the study of SUS's factor structure [26], we also report the detailed score compositions, where the usability score is 89.32, and the learnability score is 86.46. The overall result shows that RidgeBuilder has excellent usability and good learnability.

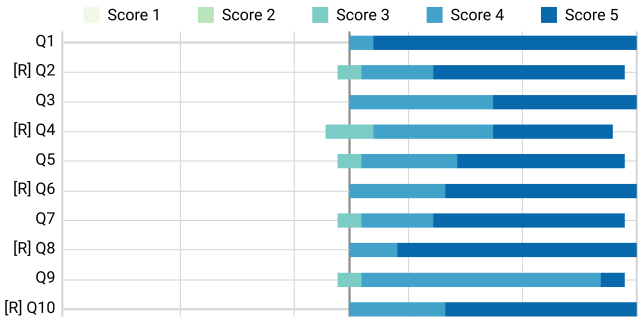

Figure 7: This figure displays the system usability evaluation results using a 5-scale SUS questionnaire [7]. The score of problems with a negative description (Q2, Q4, Q6, Q8, Q10) is presented inverted for better visualization.

6.1.3 Qualitative Analysis of Results. Most participants complete reproduction in the given time. We thematically code think-aloud records and interview records from six aspects. The details about the thematic coding can be found in Appendix 1D.2.

The usability and learnability of RidgeBuilder. We gathered feedback from participants on the pros and cons of RidgeBuilder, as well as the rationale behind their SUS scores. A majority of the participants (10/12) provided explicit positive feedback on RidgeBuilder's usability. They described it as *"useful"* and *"intuitive"*.

[4]The user study and questionnaires in this section have all been approved by State Key Lab of CAD&CG, Zhejiang University.

Participants particularly appreciated RidgeBuilder's straightforward method for creating ridgeline plots. As P2 noted, *"The initial ridgeline plot generation is quite intuitive and easy to implement. I just need to perform a simple drag and drop."* Regarding learnability, many participants (7/12) explicitly stated that RidgeBuilder is easy to learn. They offered various reasons for this. Some (P3, P4, and P9) mentioned their familiarity with the shelf-configuration style design. P7 commended the overall well-integrated design of RidgeBuilder, while P5 explained, *"Like building blocks, I can easily construct an initial ridgeline plot based on template binding."* However, a few participants also highlighted potential learning challenges, such as understanding the concept of ridgeline plots (P8) and grasping the visualization data binding (P10).

Flexible and diverse ridgeline plot configurations. Many participants (8/10) expressed positive sentiments regarding RidgeBuilder's support for creating expressive ridgeline plots. They particularly appreciated the tool's provision of unique stylistic options, such as alternative color schemes (P7 and P12), varied ridge bases (P1 and P6), and adjustable scale of ridges (P1). Additionally, several participants (P4, P8, and P12) expressed interest in the diverse layout options available. P12, who experimented with the cyclic layout, highlighted its potential for visualizing periodic patterns. P8 praised the predefined order options, stating, *"This [RidgeBuilder] offers a range of reasonable alternatives that can be directly experimented with."* These diverse style and layout options have been instrumental in helping users craft expressive ridgeline plots. P3 summed up the collective sentiment by saying, *"The style features are well integrated, allowing me to use this tool to create a visually appealing image."*

Optimized layout recommendations. Almost all participants (10/12) gave positive comments on the layout recommendations. They found the layout recommendations *"useful"*, *"beneficial"*, and *"effective"*. Many (6/12) explicitly mentioned that the recommended layout can clearly display the trend and the overall distribution of peaks. As P9 said, *"the layout recommendation is very useful because it can directly generate a layout that allows me to see the pattern immediately."* Other participants emphasized different advantages. P6 noted that *"the recommended layout was much more aesthetically pleasing than other layouts"*. P4 and P12 praised the usability of the layout recommendations. P12 said, *"just click and I gain exactly what I want"*.

Styling templates recommendations. As for style recommendations, many of our participants (7/12) found the style templates to be *"appealing"* and *"interesting"*. P5 was particularly fond of the classical JOV DIVISION style, noting, *"The black-style template, in particular, looks quite appealing."* Moreover, several participants praised the usability of the style templates. Both P4 and P10 considered this feature to be practical, as it facilitates the easy discovery of desired styles. P2 mentioned that this could help save time, which is a valuable asset in the design process. In addition, P8 commented that the style templates can serve as a source of inspiration, igniting his creative spark.

Comparing with existing authoring tools. During interviews, several users discussed RidgeBuilder in comparison to existing visualization authoring tools. Some users noted that RidgeBuilder is more user-friendly and easier to learn for creating ridgeline plots than code-based methods. P8 remarked, *"D3 requires more coding, which is more time-consuming and laborious compared to this interactive tool. Moreover, this system requires less effort to implement customized styles of ridgeline plots."* P1 expressed a preference for using RidgeBuilder in the future, stating, *"I've used d3 for this kind of chart before; adjusting parameters is quite troublesome. Although D3 is more flexible, it requires more coding and is harder to learn."* Additionally, some users mentioned interactive tools. P12, while using RidgeBuilder, commented on its similarity to drag-and-drop tools like Data Illustrator [31] but noted that RidgeBuilder offers more customized styles and layouts specifically for ridgeline plots, which reduces the further embellishment effort. P9 shared a similar view, saying, *"I've seen this chart in Tableau Public. Though such general tools are more flexible, RidgeBuilder's recommendations are uniquely designed for ridgeline plots, helping me generate patterned layouts that general tools cannot achieve."*

Reflecting on improvements. Our participants also proposed some valuable suggestions on how to improve RidgeBuilder. P1 and P9 suggested adding support for saving and sharing customizable style templates. P4 also suggested adding hints of interactions and toolbars. We improved RidgeBuilder based on these suggestions.

6.2 Algorithm Evaluation

6.2.1 Method. **Dataset.** To build a dataset of ridgeline plots with shifting-peak trends, we summarized the variables as follows: N represents the ridge number, M represents the number of data points for each ridge, and P represents the number of ridgelines in a ridgeline plot. We enumerated the pair of (N, M) from (10, 200) to (100, 2000), randomly selected two P, one for sparse peaks ($P <= 4$), and one for dense peaks ($P > 4$), and generated 20 test cases in total. We generate the dataset through the following steps: (1) We enumerated the pair of (N,M). (2) For a given pair of (N, M), we randomly selected P; (3) For each ridgeline $p_i \in P$, we set the slope as 1 and randomly selected the range of the ridge index $[start, end]$ and the intercept b; (4) We computed the cross points between ridgelines and ridges and recorded them as positions of peaks (X); (5) For each peak $x_i \in X$, we adopted a normal distribution function to generate data points, randomly selecting the scale σ and the height θ. We then combined and normalized the generated data points; (6) For each pair of (N,M), we repeated steps 2-5 twice: once with sparse peaks ($P <= 4$), and once with dense peaks ($P > 4$). The generated dataset can be found in RidgeLayouter repository[5].

Procedure and experiment settings. We conducted an evaluation of our layout algorithm's effectiveness and time efficiency by implementing four baseline algorithms (*statistic-based mean, statistic-based max, peak-based, layout randomly*) and three adjusted algorithms designed for the ablation study. To evaluate the impact of the individual constraints and objectives, we compared our full algorithm (**G1**+**G2**+**G3**) with variations of it that omitted one or more of the constraints/objectives. We were unable to construct **G2**+**G3** as **G2** is a directional constraint that must be with **G1**. Thus, we evaluated **G1** by comparing **G1**+**G3** with **G3**. To test the effectiveness, we conducted an online questionnaire-based user study. For each test case, we ran all eight algorithms and merged the same results generated by different algorithms. Then, we ask users to select the best two results for each test case. The order of

[5]https://github.com/RidgeBuilder/RidgeLayouter

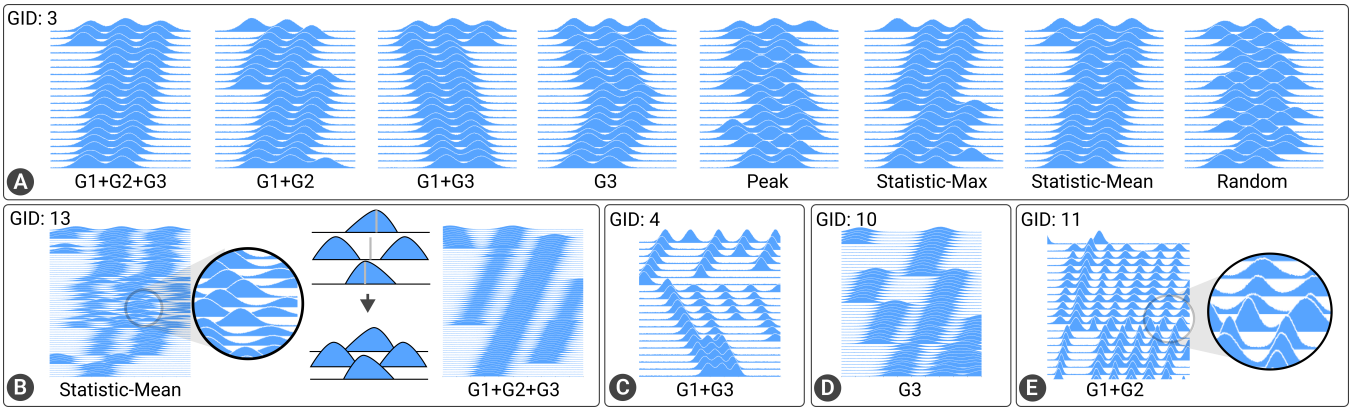

Figure 8: This figure presents the outcomes of various layout algorithms, with GID denoting the index of the dataset employed. (A) It displays the outcomes from eight distinct algorithms. (B) It elucidates the S-Mean algorithm's subpar performance: when multiple ridges possess similar mean values but differ in the number of peaks (one odd, one even), this can result in non-adjacent placement of similar ridges. (C) This indicates that some users favor a zig-zag layout. (D) It reveals that, at times, disregarding G1 and G2 can lead to the creation of intriguing stepped ridgelines. (E) This demonstrates that, in certain scenarios, omitting G3 can better maintain the continuity of the ridgeline.

options was randomized in the questionnaire. The questionnaire was distributed on an online crowdsourcing website and social media platform for university publications and communication. Each participant was paid $1 and could only submit the questionnaire once. The time efficiency testing was conducted on a computer with an Intel(R) Core(TM) i7-5820K processor and Windows 10 operating system. For each test case, we ran the test ten times and computed the average time cost.

Data collection and analysis. We implemented a rule that removes responses from participants who select the random layout more than twice to prevent participants from filling out questionnaires in bad faith. For valid questionnaires, we computed a score for each algorithm. If an option was selected, the corresponding algorithm received 1 point. However, if an option was merged from n images, meaning that n algorithms generated the same layout, all of these algorithms received $1/n$ point. Finally, we computed the score ratio between selected paired algorithms to evaluate the effectiveness of **G1**, **G2**, and **G3**.

6.2.2 Results. We finally collected 45 valid responses (46 in total, with one participant selecting the random layout 4 times). In Table 1, we can see our layout algorithm (G1+G2+G3) gains the highest score (77.33). We compute the score ratio between paired groups and prove that adding **G1**, **G2**, and **G3** can improve the performance, with corresponding ratio 221.10%, 192.53%, and 194.41%.

In Fig. 8A, we present an example group of the outcomes from various layout algorithms. We observe the following: A1 exhibits continuous, unidirectional ridgelines that are the most compact, without any occlusion; A2 lacks the desired compactness; A3 includes wiggles that disrupt the ridgeline; A4 not only has wiggles but also fails to place similar ridges in close proximity. The remaining layout results each present distinct issues: A5 obscures the ridgelines to the point where they are nearly imperceptible, A6 is dominated by the maximum peak in each ridge, highlighting the longest ridgeline but neglecting the arrangement of other peaks, and A7, influenced by the distribution of mean values, exhibits minor fluctuations. Furthermore, by integrating the score results with the layout images generated by algorithms, we gleaned several insights. First, our user experiments revealed that S-Max outperforms S-Mean. As depicted in Fig. 8B, we pinpointed the reason: while both are heavily influenced by a single statistical value, ridges with similar means that have peaks of differing odd and even numbers can lead to interlaced peaks, which disrupt the continuity of the ridgeline. In contrast, S-Max ensures that at least one max-peak ridgeline remains intact. Second, we discovered some different layouts that are also favored by several users. By disregarding G2's directional constraints, we can achieve a zig-zag layout (Fig. 8C); by ignoring both similarity and directional constraints, we can create an intriguing stepped ridgeline (Fig. 8D); and by neglecting G3, allowing for hiding some peaks can sometimes result in a more continuous ridgeline (Fig. 8E). For the complete set of 20 example groups in the datasets, please refer to the folder "layout results" in the supplementary materials.

Table 1: Scores of different layout algorithms. Our four algorithms all use different combinations of optimization goals. Among them, G1+G2, G1+G3, and G3, are designed for ablation study. Peak-based, statistic-max-based, and statistic-mean-based algorithms are baseline methods. The random one is used to control the quality of survey answers.

Methods	Ours			
	G1+G2+G3	G1+G2	G1+G3	G3
Score	**77.33**	39.78	40.17	18.17
Methods	Peak	S-Max	S-Mean	Random
Score	3.33	15.67	4.33	1.22

For all test cases, the average time cost of RidgeBuilder is 5.05s, while the statistic-max method has the best time efficiency with an average time cost 0.0123s. We argue that RidgeBuilder can respond in real-time (<1s) for common cases ($N <= 40$) and keep users' attention (<10s) in most cases ($N <= 80$) according to the existing surveys [34]. Detailed results can be found in Appendix 2B.

6.3 Gallery

Six use cases are demonstrated in Fig. 1 to showcase the expressiveness of the ridgeline plots created by RidgeBuilder. Fig. 1A adopts a cyclic layout revealing a periodical pattern with two peaks, each means a rush hour using bicycles. Fig. 1B displays multi-layered ridgeline plots with a map-based background. Fig. 1C employs a classical alternative-color style and tells a story about how migratory birds transport in Canada. Fig. 1D uses the meta-based order and reveals the distribution of a users' music listening time. Fig. 1E shows the trend of infectious diseases by encoding categories with different colors. Fig. 1F is bar-based and encodes temporal data on positions of y-axis labels.

7 Discussion

In this section, we discuss the implications, limitations, analysis in relation to other tools, and future work of this study.

7.1 Implications

We mainly discuss the implications and contributions of RidgeBuilder from four aspects.

Theory. In this paper, we survey existing ridgeline plots and systematically summarize a design space. In addition to well-known dimensions including visual elements and embellishments, we identify special layouts and styles for ridgeline plots. The design space can help visualization practitioners understand ridgeline plots better. Moreover, we hope this study will inspire future visualization researchers to explore more use of ridgeline plots.

Technique. To our knowledge, this study is among the first to formalize the layout optimization problem of ridgeline plots. Existing methods mainly focus on the semantic orders of ridges without highlighting the distinct visual pattern "ridgeline". Specifically, we summarize three optimization goals for highlighting salient ridgelines based on content analysis from the analysis perspective. Furthermore, we formulate it as an optimal problem, design an algorithm, and evaluate its effectiveness through a user study. We not only prove that our algorithm performs better than other baselines but also validate the effectiveness of the optimization goals.

Application. We implement a web-based tool called RidgeBuilder to support the interactive authoring of expressive ridgeline plots. The results obtained from the reproduction study demonstrate that RidgeBuilder has the potential to empower non-experts to create effective and expressive ridgeline plots quickly and easily.

Generalizability. Although this paper focuses on the study of ridgeline plots, the ideas discussed can be generalized to other areas of visualization. First, our approach to ridgeline plots, which inherently involves overlap, has yielded insights that are pertinent to other small multiples that leverage overlap to enhance the information-ink ratio and accentuate aggregated visual patterns. Our strategies, which include considering design space elements, style, and layout in the context of overlap, can guide the development of such small multiples. Second, our methodology for layout optimization—assessing perspectives, framing goals as optimization problems, and modeling them as TSP instances—could be generalized to other visualizations, such as storylines, for layout optimization.

7.2 Limitations

Several limitations should be discussed.

First, RidgeBuilder does not currently support composing other visualizations on ridgeline plots because our investigation of the design space found that the technique is mainly used to report data samples for distribution in academic scenarios but is less commonly used in artworks. It is possible that adding other visualizations to already expressive ridgeline plots may be counterproductive. We acknowledge that this issue requires further investigation and plan to address it in future work.

Second, we acknowledge that the RidgeBuilder layout algorithm may not respond in real time when the number of ridges is large. However, we argue that this is not a major issue because having too many ridges can cause cognitive burden and visualization scalability issues, making ridgeline plots less practical. Based on our survey, the number of ridges varies from a few to nearly 100; the current algorithm is capable of responding in real-time. Nevertheless, we plan to improve the time efficiency of the algorithm in the future by exploring modern optimal methods and approximate TSP algorithms, such as genetic algorithms or heuristic search.

Third, our current evaluation with RidgeBuilder faces limitations. On the one hand, most of the baseline tools at our disposal do not accommodate variations in ridgeline plot layouts, with many necessitating manual adjustments to the ridge order. This makes it challenging to design study tasks that are equitable. On the other hand, although code-based tools provide ample flexibility to perform the same tasks as RidgeBuilder, it would be an unfair comparison to assess them solely on usability and learnability when pitted against interactive tools. To address these limitations, we make RidgeBuilder open-source. This move will enable us to gather user feedback in the future and continuously refine its design and implementation. Furthermore, we analyzed RidgeBuilder alongside other authoring tools and discussed RidgeBuilder's underlying framework alongside some grammar, identifying both shared and distinct features.

Fourth, we currently model the layout optimization problem of ridgelines based on the relationships between adjacent ridges. However, in very dense ridgeline scenarios, overlap is not limited to adjacent ridges; instead, multiple consecutive ridges can overlap. Given the computational complexity of modeling, specifically, assuming the number of variables is n, the number of combinations considering k variables is P_n^k. Considering the real-time requirements of interactive tools, we opted to model only adjacent ridges.

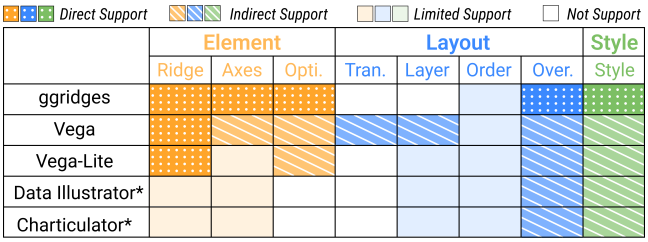

Figure 9: This displays the commonalities and differences between the underlying framework of RidgeBuilder and candidate grammars or frameworks. These candidates include three grammars ggirdges [67], Vega [60], Vega-Lite [48] and two underlying frameworks Data Illustrator [31] and Charticulator [44]. We use the mark * to identify the framework.

7.3 Analysis of RidgeBuilder in Relation to Other Grammars and Authoring Tools

We explored insights from the perspectives of grammar and authoring tools, discussing potential candidates that could offer a basis for understanding RidgeBuilder relative to other tools.

7.3.1 Analyze RidgeBuilder alongside existing visualization grammars. We list five candidates: ggridges [67], Vega [60] (the underlying framework of Lyra [47]), Vega-Lite [48], the underlying framework of Data Illustrator [31], and the underlying framework of Charticulator [44]. We classify the support for ridgeline plot design space in candidates across four categories:

- Direct Support: The direct configuration of this grammar covers all options for the selected design dimension that RidgeBuilder supports.
- Indirect Support: This grammar can cover all options for the selected design dimension, but it is achieved through indirect and complex configuration combinations, demanding high user expertise.
- Limited Support: This grammar covers a subspace for the design dimension.
- Not Support: This grammar's configurations don't support for editing the design dimension.

We provide a brief summary of the comparison results in Fig. 9, where each cell indicates the level of support for one dimension of each candidate. Detailed introduction with screenshots can be found in Appendix 4. In the discussion, we elaborate on the details of the "Indirect Support" and "Limited Support" cells from the functionality perspective and then discuss limitations and advantages.

In ***Elements*** scope, we discuss three dimensions. (1) Ridge: Data Illustrator and Charticulator currently lack support for "step" shaped ridgeline plots, which are crucial for effectively visualizing continuous changes in levels over time. Tools with this feature have better support for discrete data and can broaden the use of ridgeline plots in various contexts. (2) Axes: There are two handling approaches. One method involves the explicit representation of three independent axes-x, y, and z (ridge) as seen in tools like ggridges and RidgeBuilder. The alternative approach views the ridgeline plot as a foundational chart with x and y axes while performing layout variations, such as facet rows or repetition, as utilized in Vega-Lite and Data Illustrator. Vega also operates with three axes, but it achieves the z-axis by setting the y-axis as a band and nesting an inner y-axis within it. Charticulator introduces the z-axis through the height of the rectangle for bar-based ridgeline plots and uses layout changes like facet rows for line-based plots. Our implementation of various ridgeline plots indicates that tools supporting three axes facilitate relatively easy creation, as such structure aligns more naturally with user cognition. (3) Optional elements: Vega and Vega-Lite achieve this via rules.

In ***Layouts*** scope, we discuss four dimensions. (1) Transform: Vega supports the cyclic transform layout via the polar axis. Tools lacking cyclic transformation support cannot effectively create ridgeline plots for periodic patterns, limiting their expressiveness. (2) Layer: Vega allows the creation of multiple layers using scales with different ranges. Vega-Lite supports the stacked layer and the overlay layer through the stack option. Data Illustrator only supports the stacked bar-based layers. Charticulator can achieve the overlay bar-based layer by linking grouped bars. Different layers can visualize the inner ridgeline among groups and the outer ridgeline among variables so supporting diverse layers can enhance the expressiveness of ridgeline plots. (3) Order: Listed candidates allow users to sort ridges by descending or ascending the order of the statistic value. The limited support for ordering severely restricts the expressiveness of ridgeline plots, as shown in 6.2.2. Additionally, manually determining the optimal order for multiple variables is highly time-consuming and labor-intensive. RidgeBuilder outperforms other tools in addressing this challenge. (4) Overlap: Candidates except ggridges require users to configure "row gap" as negative numbers to set overlap. Besides, users must set the "bound" as "flush" in Vega-Lite.

In ***Styles*** scope, since color, opacity, and border dimensions share similar features, we discuss them together. RidgeBuilder and ggridges offer direct declarations for special styles, such as alternative colors, whereas other tools require manual specification for each element. Thus, RidgeBuilder and ggridges enable users to create special styles more efficiently than others.

Overall, RidgeBuilder's underlying data model offers us insights in two key areas. On one hand, it provides users with an authoring data model that is more aligned with the nature of ridgeline plots, while also supporting expressive and flexible visualizations. Our data model is designed to be closer to user cognition, with direct declaration options for elements such as the three axes, layers and order in layout, and special styles. As shown in the Sec 6.1.3, our design can further lower the learning curve for users. On the other hand, we explored the possibilities of declaring layout-optimized options. RidgeBuilder includes features that allow users to specify their layout preferences, facilitating the achievement of the desired layout. For example, by setting the G2 ratio to 0, users can create a zig-zag ridgeline plot. This feature significantly enhances RidgeBuilder's flexibility and user-friendliness, making it particularly beneficial for users who need customized visualization layouts.

7.3.2 Analyze RidgeBuilder alongside existing visualization tools. We list six candidates: Data Illustrator [31], Charticulator [44], Lyra [47], Lyra2 [72], Tableau [46], PowerViz [42].

DG1: Lyra, Lyra2, PowerViz, and Tableau provide direct interaction for creating basic ridgeline plots but are limited in support

specific channel bindings, for example, the axis labels' position. Data Illustrator utilizes an indirect approach to ridgeline plot creation with repeat and partition. Charitulator enables the direct creation of ridgeline plots with bar-based ridges; however, it requires users to invest extra effort in layout adjustments when transitioning to line-based ridges.

DG2: All candidates exhibit limitations in supporting the diverse layouts characteristic of ridgeline plots. Notably, the ridgline plots derived from RidgeBuilder's design space, such as cyclic ridgeline plots and those featuring multiple ridges and layers, are challenging to edit using these tools. Furthermore, RidgeBuilder offers guidance on optimizing layouts to highlight visual patterns, assisting users in effectively achieving the desired layout for ridgeline plots.

DG3: While various tools offer the ability to adjust the style of ridgeline plots, many exhibit limited support for specific styles. For instance, the popular alternate color pattern often requires manual adjustments for each ridge in most tools. Similarly, applying cumulative transparency from top to bottom typically necessitates individual modifications in other tools. In contrast, RidgeBuilder facilitates easier interaction for modifying these styles, allowing users to make adjustments more efficiently.

DG4: Data Illustrator and Tableau allow users to add embellishments while the other four do not.

7.4 Future Work

We plan to improve RidgeBuilder from three aspects in the future.

Integrating human-model-collaborative designs. In the reproduction study interviews, many participants shared their suggestions on the recommendation functions of RidgeBuilder, mainly focusing on the human-model collaborative design possibility of ridgeline plots in future work. Our participants proposed potential requirements for dynamic layout optimization goals. P5 commented that the aesthetic consideration of layout varies under different situations, so it is important to consider recommendations based on interactions. P12 explicitly suggested adding user interactions as the context in the layout recommendation. Also, P5 and P7 suggested we extend fixed-style templates to data-driven ones. Furthermore, P7 mentioned the recommendation for data binding will significantly reduce the mental burden of users. P10 wanted us to provide more guidance on the embellishments, that is, help her find possible insights and add annotations. Considering all these suggestions, we plan to propose a context-aware human-model-collaborative design framework for ridgeline plots. Specifically, we will further survey the human factors, such as *which visual patterns users are interested in* and *what are the other layout criteria users are concerned about* in designing ridgeline plots. Then, we will build a framework to clearly describe the design context of ridgeline plots and enhance the recommendations.

Enhancing direct manipulation on plots. While RidgeBuilder currently supports drag-and-drop data mapping and allows sorting of ridges through dragging, some interactions still rely on shelf-configuration options, which require users to search for the correct settings. To enhance the intuitiveness of RidgeBuilder in future iterations, we plan to invest engineering resources to enhance direct manipulation features within RidgeBuilder. Specifically, for layout adjustments, users will be able to drag to modify the offset and dynamically resize plots and marks. Regarding styles, RidgeBuilder should incorporate a style-brushing feature and customizable hotkeys. For advanced manipulation, RidgeBuilder could enable users to sketch desired patterns by drawing a ridgeline trend, which the system would then use to optimize the layout accordingly.

Improving RidgeBuilder's connection with existing authoring systems. To make RidgeBuilder more accessible and reach a broader user base, we are looking into improving its compatibility. For visualization tools that currently do not support all ridgeline plot authoring features, we plan to identify compatible features like providing a layout optimizer. For design tools, we now support exporting SVG file and we also intend to offer RidgeBuilder as a plugin for existing design authoring systems like Figma, allowing users to integrate our tool seamlessly into their workflows.

8 Conclusion

In this paper, we investigate ridgeline plots from the design space, authoring method, and layout criteria. We conduct a comprehensive survey of existing ridgeline plots and propose a design space with five dimensions. We develop RidgeBuilder, an interactive authoring tool, to help users create expressive ridgeline plots easily with the automatic layout optimization algorithm based on three identified goals. We evaluate RidgeBuilder through a reproduction study and several use cases and validate the effectiveness of the layout optimization goals and algorithm through a user study. Our findings demonstrate that RidgeBuilder is effective in supporting users in creating visually appealing ridgeline plots. In the future, we plan to enhance RidgeBuilder with intelligent models and a context-aware human-model-collaborative design framework.

Acknowledgments

This research was supported by NSFC (U22A2032, 62402421, 62402184). We extend our sincere gratitude to the anonymous reviewers for their insightful feedback and constructive suggestions and all participants in our studies.

References

[1] Dan Abramov. 2015. React Redux. Retrieved September 5, 2024 from https://react-redux.js.org
[2] Adobe Inc. 2019. *Adobe Illustrator.* https://adobe.com/products/illustrator
[3] Wolfgang Aigner, Silvia Miksch, Heidrun Schumann, and Christian Tominski. 2023. *Visualization of Time-Oriented Data.* https://doi.org/10.1007/978-1-4471-7527-8
[4] Michael Behrisch, Benjamin Bach, Nathalie Henry Riche, Tobias Schreck, and Jean-Daniel Fekete. 2016. Matrix Reordering Methods for Table and Network Visualization. *Computer Graphics Forum* 35, 3 (2016), 693–716. https://doi.org/10.1111/cgf.12935
[5] Mandell Bellmore and George L Nemhauser. 1968. The Traveling Salesman Problem: A Survey. *Operations Research* 16, 3 (1968), 538–558. https://doi.org/10.1287/opre.16.3.538
[6] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D^3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 2301–2309. https://doi.org/10.1109/TVCG.2011.185
[7] John Brooke. 1996. SUS: A Quick and Dirty Usability Scale. *Usability Evaluation in Industry* (1996), 189–194. https://doi.org/10.1201/9781498710411-35
[8] Jenny Bryan. 2017. I hereby propose that we call these "joy plots" #rstats. Twitter. Retrieved September 5, 2024 from https://twitter.com/JennyBryan/status/856674638981550080
[9] California Department of Public Health, Center for Infectious Diseases, Infectious Diseases Branch, Surveillance and Statistics Section. 2024. Infectious Diseases by Disease, County, Year, and Sex. Retrieved September 5, 2024 from https://data.chhs.ca.gov/dataset/infectious-disease

[10] Center for International Earth Science Information Network - CIESIN - Columbia University. 2016. Gridded Population of the World, Population Count. NASA Socioeconomic Data and Applications Center. Retrieved September 5, 2024 from https://doi.org/10.7927/H4X63JVC

[11] James Cheshire. 2013. Population Lines: How and Why I Created It. Retrieved September 5, 2024 from https://jcheshire.com/featured-maps/population-lines-how-and-why-i-created-it/

[12] Jen Christiansen. 2015. Pop culture pulsar: Origin story of Joy Division's Unknown Pleasures album cover. Scientific American. Retrieved September 5, 2024 from https://www.scientificamerican.com/blog/sa-visual/pop-culture-pulsar-origin-story-of-joy-division-s-unknown-pleasures-album-cover-video/

[13] Harold Dumont Craft Jr. 1970. *Radio Oprofiles and Dispersion Measures of Twelve Pulsars*. Ph. D. Dissertation. Cornell University. https://pubsmisc.informs.org/doi/10.1287/opre.16.3.538

[14] George Dantzig, Ray Fulkerson, and Selmer Johnson. 1954. Solution of a Large-Scale Traveling-Salesman Problem. *Journal of the Operations Research Society of America* 2, 4 (1954), 393–410. https://doi.org/10.1287/opre.2.4.393

[15] Kasper Dinkla, Hendrik Strobelt, Bryan Genest, Stephan Reiling, Mark Borowsky, and Hanspeter Pfister. 2017. Screenit: Visual Analysis of Cellular Screens. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2017), 591–600. https://doi.org/10.1109/TVCG.2016.2598587

[16] E. L. Lawler, Jan Karel Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys. 1985. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley.

[17] Stefan Edelkamp and Stefan Schrödl. 2011. *Heuristic search: theory and applications*. Elsevier.

[18] Stephen Few. 2008. Time on the Horizon. *Visual Business Intelligence Newsletter* (jun 2008). http://www.perceptualedge.com/articles/visual_business_intelligence/time_on_the_horizon.pdf

[19] Marsha E. Fonteyn, Benjamin Kuipers, and Susan J. Grobe. 1993. A Description of Think Aloud Method and Protocol Analysis. *Qualitative Health Research* 3, 4 (1993), 430–441. https://doi.org/10.1177/104973239300300403

[20] Germain García-Zanabria, Marcos M. Raimundo, Jorge Poco, Marcelo Batista Nery, Cláudio T. Silva, Sergio Adorno, and Luis Gustavo Nonato. 2022. CriPAV: Street-Level Crime Patterns Analysis and Visualization. *IEEE Transactions on Visualization and Computer Graphics* 28, 12 (2022), 4000–4015. https://doi.org/10.1109/TVCG.2021.3111146

[21] Lars Grammel, Chris Bennett, Melanie Tory, and Margaret-Anne Storey. 2013. A Survey of Visualization Construction User Interfaces. In *Eurographics Conference on Visualization (Short Papers)*. 19–23. https://doi.org/10.2312/PE.EuroVisShort.EuroVisShort2013.019-023

[22] KCPD Information Technology. 2021. Retrieved September 5, 2024 from https://data.kcmo.org/Crime/KCPD-Crime-Data-2020/vsgj-uufz/about_data

[23] Robert Krueger, Johanna Beyer, Won-Dong Jang, Nam Wook Kim, Artem Sokolov, Peter K. Sorger, and Hanspeter Pfister. 2020. Facetto: Combining Unsupervised and Supervised Learning for Hierarchical Phenotype Analysis in Multi-Channel Image Data. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2020), 227–237. https://doi.org/10.1109/TVCG.2019.2934547

[24] Fan Lei, Arlen Fan, Alan M. MacEachren, and Ross Maciejewski. 2024. GeoLinter: A Linting Framework for Choropleth Maps. *IEEE Transactions on Visualization and Computer Graphics* 30, 2 (2024), 1592–1607. https://doi.org/10.1109/TVCG.2023.3322372

[25] Fritz Lekschas, Xinyi Zhou, Wei Chen, Nils Gehlenborg, Benjamin Bach, and Hanspeter Pfister. 2021. A Generic Framework and Library for Exploration of Small Multiples through Interactive Piling. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2021), 358–368. https://doi.org/10.1109/TVCG.2020.3028948

[26] James R. Lewis and Jeff Sauro. 2009. The Factor Structure of the System Usability Scale. In *Proceedings of the 1st International Conference on Human Centered Design*. 94–103. https://doi.org/10.1007/978-3-642-02806-9_12

[27] James R. Lewis and Jeff Sauro. 2018. Item Benchmarks for the System Usability Scale. *Journal of Usability Studies* 13, 3 (may 2018), 158–167. https://doi.org/10.1007/978-3-642-02806-9_12

[28] Henrik Lindberg. 2017. Peak time for sports and leisure #dataviz. Twitter. Retrieved September 5, 2024 from https://twitter.com/hnrklndbrg/status/883675698300420098

[29] Shuhan Liu, Di Weng, Yuan Tian, Zikun Deng, Haoran Xu, Xiangyu Zhu, Honglei Yin, Xianyuan Zhan, and Yingcai Wu. 2023. ECoalVis: Visual Analysis of Control Strategies in Coal-fired Power Plants. *IEEE Transactions on Visualization and Computer Graphics* 29, 1 (2023), 1091–1101. https://doi.org/10.1109/TVCG.2022.3209430

[30] Shixia Liu, Yingcai Wu, Enxun Wei, Mengchen Liu, and Yang Liu. 2013. StoryFlow: Tracking the Evolution of Stories. *IEEE Transactions on Visualization and Computer Graphics* 19, 12 (2013), 2436–2445. https://doi.org/10.1109/TVCG.2013.196

[31] 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 CHI Conference on Human Factors in Computing Systems*. 1–13. https://doi.org/10.1145/3173574.3173697

[32] David McCandless and Fabio Bergamaschi. 2018. Mountains Out of Molehills. Information is Beautiful. Retrieved September 5, 2024 from https://informationisbeautiful.net/visualizations/mountains-out-of-molehills/

[33] Honghui Mei, Yuxin Ma, Yating Wei, and Wei Chen. 2018. The Design Space of Construction Tools for Information Visualization: A Survey. *Journal of Visual Languages and Computing* 44 (2018), 120–132. https://doi.org/10.1016/j.jvlc.2017.10.001

[34] Robert B. Miller. 1968. Response Time in Man-Computer Conversational Transactions. In *Proceedings of the Joint Computer Conference*. 267–277. https://doi.org/10.1145/1476589.1476628

[35] Seyedali Mirjalili. 2019. *Genetic Algorithm*. Springer International Publishing, Cham, 43–55. https://doi.org/10.1007/978-3-319-93025-1_4

[36] Dominik Moritz and Danyel Fisher. 2018. Visualizing a Million Time Series with the Density Line Chart. (2018). https://doi.org/10.48550/arXiv.1808.06019 arXiv:arXiv:1808.06019

[37] NetEase Cloud Music. 2023. 2023 Annual Listening Report. Retrieved September 5, 2024 from https://huaban.com/pins/5874746563

[38] L. Nowell, R. Schulman, and D. Hix. 2002. Graphical Encoding for Information Visualization: An Empirical Study. In *Proceedings of the IEEE Symposium on Information Visualization*. 43–50. https://doi.org/10.1109/INFVIS.2002.1173146

[39] Inc. Observable. 2020. Observable Plot. Retrieved February 5, 2025 from https://observablehq.com/plot/

[40] Observable, Inc. 2024. Observable Plot. Retrieved September 5, 2024 from https://observablehq.com/plot/

[41] Anna Offenwanger, Matthew Brehmer, Fanny Chevalier, and Theophanis Tsandilas. 2024. TimeSplines: Sketch-Based Authoring of Flexible and Idiosyncratic Timelines. *IEEE Transactions on Visualization and Computer Graphics* 30, 1 (2024), 34–44. https://doi.org/10.1109/TVCG.2023.3326520

[42] PowerViz.ai. 2024. Ridgeline Plot - PowerViz. Retrieved September 5, 2024 from https://powerviz.ai/ridgeline-plot

[43] Xiaoying Pu and Matthew Kay. 2020. A Probabilistic Grammar of Graphics. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–13. https://doi.org/10.1145/3313831.3376466

[44] 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. https://doi.org/10.1109/TVCG.2018.2865158

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

[46] Salesforce, Inc. 2024. Business Intelligence and Analytics Software | Tableau. Retrieved September 5, 2024 from https://www.tableau.com/

[47] Arvind Satyanarayan and Jeffrey Heer. 2014. Lyra: An Interactive Visualization Design Environment. *Computer Graphics Forum* 33, 3 (2014), 351–360. https://doi.org/10.1111/cgf.12391

[48] 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. https://doi.org/10.1109/TVCG.2016.2599030

[49] David Schroeder and Daniel F. Keefe. 2016. Visualization-by-Sketching: An Artist's Interface for Creating Multivariate Time-Varying Data Visualizations. *IEEE Transactions on Visualization and Computer Graphics* 22, 1 (2016), 877–885. https://doi.org/10.1109/TVCG.2015.2467153

[50] Social Security. 2023. Popular Baby Names. Retrieved September 5, 2024 from https://www.ssa.gov/oact/babynames/limits.html

[51] Dong Sun, Renfei Huang, Yuanzhe Chen, Yong Wang, Jia Zeng, Mingxuan Yuan, Ting-Chuen Pong, and Huamin Qu. 2020. PlanningVis: A Visual Analytics Approach to Production Planning in Smart Factories. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2020), 579–589. https://doi.org/10.1109/TVCG.2019.2934275

[52] Yuzuru Tanahashi and Kwan-Liu Ma. 2012. Design Considerations for Optimizing Storyline Visualizations. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (2012), 2679–2688. https://doi.org/10.1109/TVCG.2012.212

[53] Tan Tang, Renzhong Li, Xinke Wu, Shuhan Liu, Johannes Knittel, Steffen Koch, Thomas Ertl, Lingyun Yu, Peiran Ren, and Yingcai Wu. 2021. PlotThread: Creating Expressive Storyline Visualizations using Reinforcement Learning. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2021), 294–303. https://doi.org/10.1109/TVCG.2020.3030467

[54] Tan Tang, Sadia Rubab, Jiewen Lai, Weiwei Cui, Lingyun Yu, and Yingcai Wu. 2019. iStoryline: Effective Convergence to Hand-drawn Storylines. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (2019), 769–778. https://doi.org/10.1109/TVCG.2018.2864899

[55] Yuan Tian, Weiwei Cui, Dazhen Deng, Xinjing Yi, Yurun Yang, Haidong Zhang, and Yingcai Wu. 2024. ChartGPT: Leveraging LLMs to Generate Charts from Abstract Natural Language. *IEEE Transactions on Visualization and Computer Graphics* (2024), 1–15. https://doi.org/10.1109/TVCG.2024.3368621

[56] Google Trends. 2024. Birds Migration in Canada. Retrieved September 5, 2024 from "https://trends.google.com/trends/explore?cat=66&date=2023-01-01%202023-12-31&geo=CA&q=cormorant,cuckoo,mallard,American%20Robin,Canada%20Goose"
[57] U.S. Bureau of Labor Statistics. 2023. American Time Use Survey. Retrieved September 5, 2024 from https://www.bls.gov/tus/home.htm
[58] USGS Vizlab. 2023. Rio grande river, New Mexico. Information is Beautiful. Retrieved September 5, 2024 from https://www.informationisbeautifulawards.com/showcase/6592-ridgelines-plot
[59] Jacob VanderPlas, Brian Granger, Jeffrey Heer, Dominik Moritz, Kanit Wongsuphasawat, Arvind Satyanarayan, Eitan Lees, Ilia Timofeev, Ben Welsh, and Scott Sievert. 2018. Altair: Interactive Statistical Visualizations for Python. *Journal of Open Source Software* 3, 32 (2018), 1057. https://doi.org/10.21105/joss.01057
[60] vega [n. d.]. Vega – A Visualization Grammar. https://vega.github.io/vega/
[61] Saverio Vito. 2008. Air Quality. UCI Machine Learning Repository. https://doi.org/10.24432/C59K5F
[62] Chenglong Wang, Yu Feng, Rastislav Bodik, Isil Dillig, Alvin Cheung, and Amy J Ko. 2021. Falx: Synthesis-Powered Visualization Authoring. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–15. https://doi.org/10.1145/3411764.3445249
[63] Chenglong Wang, John Thompson, and Bongshin Lee. 2024. Data Formulator: AI-Powered Concept-Driven Visualization Authoring. *IEEE Transactions on Visualization and Computer Graphics* 30, 1 (2024), 1128–1138. https://doi.org/10.1109/TVCG.2023.3326585
[64] Shisheng Wang, Yi Zhong, Jingqiu Cheng, and Hao Yang. 2021. EnrichVisBox: A Versatile and Powerful Web Toolbox for Visualizing Complex Functional Enrichment Results of Omics Data. *Journal of Computational Biology* 28, 9 (2021), 922–930. https://doi.org/10.1089/cmb.2020.0564
[65] Max Wertheimer. 1938. Gestalt theory. (1938).
[66] Claus O. Wilke. 2017. Goodbye joyplots. Retrieved September 5, 2024 from https://clauswilke.com/blog/2017/09/15/goodbye-joyplots/
[67] Claus O. Wilke. 2024. ggridges package - R Documentation. Retrieved September 5, 2024 from https://www.rdocumentation.org/packages/ggridges
[68] Yuchen Wu, Yuansong Xu, Shenghan Gao, Xingbo Wang, Wenkai Song, Zhiheng Nie, Xiaomeng Fan, and Quan Li. 2024. LiveRetro: Visual Analytics for Strategic Retrospect in Livestream E-Commerce. *IEEE Transactions on Visualization and Computer Graphics* 30, 1 (2024), 1117–1127. https://doi.org/10.1109/TVCG.2023.3326911
[69] Nathan Yau. 2022. Trendiest Baby Name Every Year Since 1930, in the U.S. Flowing Data. Retrieved September 5, 2024 from https://flowingdata.com/2022/02/17/trendiest-baby-name-every-year-since-1930-in-the-u-s/
[70] Lu Ying, Xinhuan Shu, Dazhen Deng, Yuchen Yang, Tan Tang, Lingyun Yu, and Yingcai Wu. 2023. MetaGlyph: Automatic Generation of Metaphoric Glyph-based Visualization. *IEEE Transactions on Visualization and Computer Graphics* 29, 1 (2023), 331–341. https://doi.org/10.1109/TVCG.2022.3209447
[71] Lu Ying, Tan Tang, Yuzhe Luo, Lvkeshen Shen, Xiao Xie, Lingyun Yu, and Yingcai Wu. 2022. GlyphCreator: Towards Example-based Automatic Generation of Circular Glyphs. *IEEE Transactions on Visualization and Computer Graphics* 28, 1 (2022), 400–410. https://doi.org/10.1109/TVCG.2021.3114877
[72] 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. https://doi.org/10.1109/TVCG.2020.3030367