ChartMark: A Structured Grammar for Chart Annotation

Yiyu Chen * HKUST(GZ)
Yifan Wu HKUST(GZ)
Shuyu Shen HKUST(GZ)
Yupeng Xie HKUST(GZ)
Leixian Shen HKUST
Hui Xiong HKUST(GZ)
Yuyu Luo† HKUST(GZ)

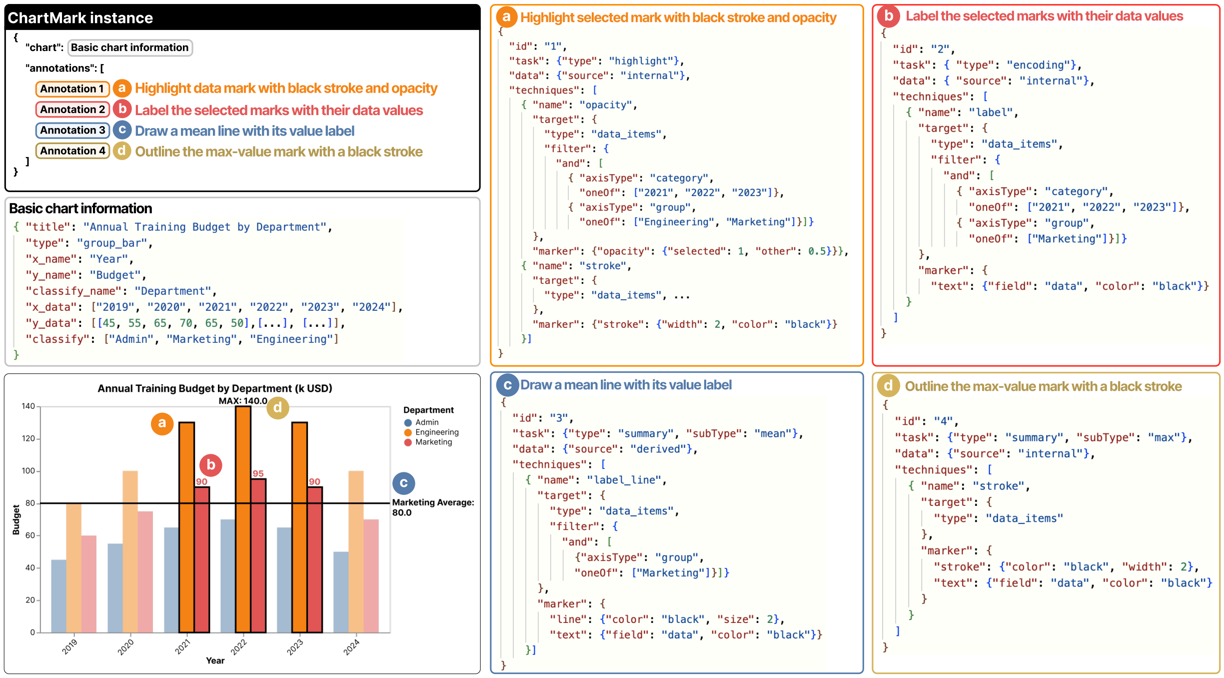


Figure 1: An annotated chart and its corresponding ChartMark grammar representation. Left-top: The overall ChartMark instance showing the annotations list structure. Left-center: Basic chart information defining the group bar chart properties. Left-bottom: The rendered chart displaying annual training budget by department with four annotations: highlighting data marks with stroke and opacity (a), labeling marks with data values (b), drawing a mean line with value label (c), and outlining the max-value mark (d). Center and Right: The detailed JSON specification for each annotation (a-d). For more examples and details, see this link.

Abstract

Chart annotations enhance visualization accessibility but suffer from fragmented, non-standardized representations that limit cross-platform reuse. We propose ChartMark, a structured grammar that separates annotation semantics from visualization implementations. ChartMark features a hierarchical framework mapping onto annotation dimensions (e.g., task, chart context), supporting both abstract intents and precise visual details. Our toolkit demonstrates converting ChartMark specifications into Vega-Lite visualizations, highlighting its flexibility, expressiveness, and practical applicability.

Index Terms: Chart Annotation, Grammar Language-agnostic

1 Introduction

Annotations are crucial elements in data visualizations [37, 38, 16], directing viewer attention, emphasizing patterns, and providing contextual information [41, 39, 4]. Despite their importance, current annotation approaches lack standardization and are often tightly coupled to specific visualization libraries [21, 33, 34], making them difficult to reuse across platforms.

*e-mail: ychen081@connect.hkust-gz.edu.cn
†e-mail: yuyuluo@hkust-gz.edu.cn (corresponding author)

Previous research has explored annotation spaces along dimensions [17, 18, 31, 9] such as data references, annotation tasks, visual forms, and implementation techniques. However, these classifications are typically incomplete or inconsistently defined, capturing only portions of the annotation landscape. Consequently, it remains challenging to create annotation systems that are both semantically rich and versatile enough for diverse applications.

To address these limitations, we present ChartMark, a structured grammar for visual annotations. ChartMark provides a comprehensive and extensible framework that explicitly decouples the semantic representation of annotations from their specific chart implementations. Unlike existing visualization grammars (e.g., Vega-Lite [21]) and annotation tools (e.g., ChartAccent [19] and Charticulator [20]), which typically couple annotation specifications tightly to particular visualization environments, ChartMark introduces an independent abstraction layer. This design allows annotations to be consistently reused across multiple visualization platforms without losing their intended semantic meanings.

Our contributions include: (1) a language-agnostic annotation representation framework structured into meaningful semantic layers, (2) the ChartMark grammar for formalizing chart annotations, and (3) a toolkit implementation (`ChartMark`) demonstrating practical application by transforming ChartMark specifications into Vega-Lite visualizations. These provide foundations for creating more systematic, reusable annotation systems across visualization platforms.

2 Related Work

Annotation in Visualization Grammar. Visualization languages and libraries [34, 5, 32, 33, 40, 42, 26, 28, 27] provide a systematic approach to defining visualizations. Some of them support annotation-like effects to varying extents. Languages based on the Grammar of Graphics model [36], such as D3 [6], Vega [22], and Vega-Lite [21], do not provide independent rendering modules for annotations and do not distinguish between graphics used for annotations and those used for basic chart elements. Annotation effects can be simulated by layering primitive marks like text and lines on top of existing visual layers. However, this design leads that the annotation structure is tightly coupled with the overall rendering logic. In contrast, languages/libraries like ECharts [15], matplotlib [13], and ggplot2 [35] offer dedicated APIs for annotations. But these APIs provide limited annotation rendering capabilities and lack support for specifying rich, expressive annotations or managing multiple annotations. Despite their differences, annotation mechanisms in both types of systems are heavily dependent on the internal design of their respective grammars. This structural dependency greatly limits the portability and reuse potential of annotations between different visualization systems, forming a technical barrier.

Annotation Tools and Platforms. The aforementioned coupling problem also exists in visualization tools that focus on annotation functions. As a pioneer in interactive chart annotation, ChartAccent [19] provides a rich variety of annotation technologies, but its implementation is still tightly bound to a specific platform. In the field of data-driven narrative systems [3, 10], DataClips [3], GeoTime Stories [10], and Timeline Storyteller [7] all integrate annotation modes that are critical to their narrative functions, but these annotation mechanisms are only designed for specific platforms and application scenarios, making it difficult to apply in a wider environment. Similarly, although the annotation functions of commercial visualization tools such as Tableau [2] and Adobe Illustrator [1] are powerful, their proprietary implementations for their own products also lead to the non-portability of annotations. The common limitation of these tools is that they focus on operational functions rather than abstract annotation semantics. Although they can help users create effective annotations, they do not provide standardized methods for representing, exchanging, or reusing these annotations in different visualization environments.

3 Goals & Requirements

A dedicated grammar for chart annotation can address the limitations of existing approaches. This grammar should include the following design considerations:

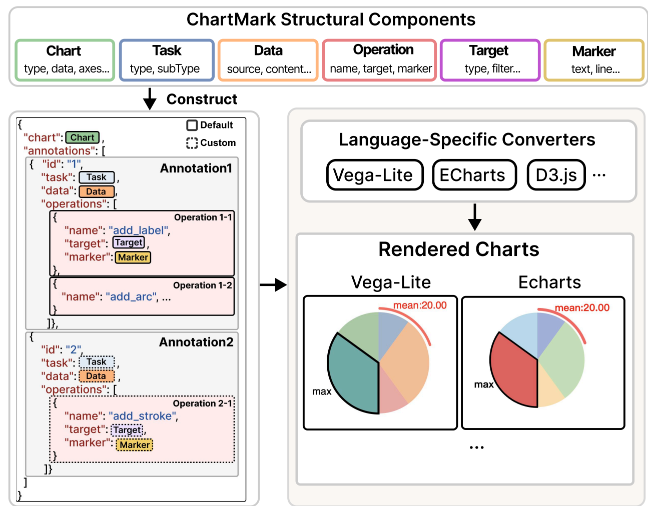
DC1: Comprehensive and Extensible Semantics. The annotation design spaces for specific domains (such as finance [11] or time series charts [12]) are limited in expression; while the existing design space solutions [9, 18] for general annotations lack scalability and cannot evolve with new scenarios or new visualization styles. So our annotation representation should cover all possible annotation scenarios in the current design space while remaining flexible enough to accommodate future extensions.

DC2: Language Independence. The annotation methods integrated in existing visualization languages [13, 21, 15, 35, 6, 22] are not designed specifically for general annotations. Their expression and usage are strictly constrained by the underlying syntax and rendering model. To address this limitation, our annotation grammar should focus on the semantic information of annotations without dependency on any specific implementation methods and rendering languages so that it can achieve broader applicability and cross-platform reuse.

DC3: Modular Architecture. Chart annotations are semantically additions to the base chart and should be represented as modules independent of the base chart. Each specific annotation should also be considered an independent unit that can be flexibly added or deleted without affecting the original chart structure.

Table 1: Dimensions and Components of Chart Annotations

Dimension	Component	Description
Data	`Data`	The data entities or aggregates that an annotation references.
Task or Goal	`Task`	The analytic or communicative goal motivating the annotation (e.g., highlight, compare, explain).
Embellishment or Marker	`Marker`	Added or modified visual elements used to convey the annotation.
Operational Target	`Target`	The specific marks or regions that the annotation acts upon.
Operation or Technique	`Operation`	The concrete methods or tactics (color change, overlay, etc.) employed to realize the annotation.
Chart Context	`Chart`	Chart features (typically chart type) that influence annotation display and rendering.



(a) JSON Format Framework (b) Converting & Rendering

Figure 2: Overview of the ChartMark grammar pipeline. It illustrates how annotations are represented hierarchically through JSON specifications (a), and (b) demonstrates the transformation of these abstract specifications into platform-specific visualizations using converters for Vega-Lite, ECharts, and D3.js.

DC4: Multi-level Semantic Representation. The expressions of annotations are usually multi-semantic. For example, they include the high-level intention of "why do it" (*e.g.,* "Highlight 'A' "), and can also describe the specific operation of "how to do it" (*e.g.,* "Set 'A' red and fade other marks"). The grammar needs to include these different semantic levels to ensure its expressive completeness.

DC5: Atomic Element Design. Basic actions like "Set Color", "Adjust Opacity" and "Add Lines" often reused between different annotation types. To reduce redundancy and enable complex representations, the grammar can view these basic actions or parameter settings as atomic elements that can be reused and combined to create richer annotation expressions.

4 ChartMark Grammar: Design and Formalization

Given the design considerations mentioned above, we propose the ChartMark grammar. Fig. 2 shows the pipeline of ChartMark from its design to rendering. ChartMark consists of five features: (1) Comprehensive integration of annotation classification methods with extensibility through custom nodes (**DC1**); (2) separation of semantic information from implementation, allowing conversion to different visualization languages through converters (**DC2**); (3) structural separation of annotations from base charts and modular arrangement of annotations (**DC3**); (4) hierarchical JSON organization to express multiple semantic levels (**DC4**); and (5) atomic components that serve as building blocks for composite annotations (**DC5**).

The ChartMark specification codes in this paper are simplified for explanation purposes and may omit some details for brevity. Please see the homepage of CharMark (chartmark.github.io) for more examples and details.

4.1 Component and Framework

By summarizing the studies on the chart annotation design space [17, 18, 31, 30, 14, 9, 19, 12, 8], we categorize annotations into multiple dimensions (see Tab. 1). In the ChartMark grammar, we design a series of basic components (see Fig. 2) that directly map to these annotation classification dimensions. Every annotated chart represented in ChartMark must contain and specify these components. This structured approach ensures comprehensive semantic expression while maintaining consistent representation across visualization platforms.

Each classification dimension in Tab. 1 contains different specific classification values, representing various semantic features of annotations across dimensions, which will be discussed in Sec. 4.2.

These classification dimensions exhibit both parallel and hierarchical relationships. To effectively represent these relationships, we structure the corresponding components in a hierarchical JSON format (see Fig. 2-a). This JSON structure naturally expresses the logical connections between classification dimensions through its parallel and nested organization:

(1) The outer layer contains the Chart and Annotation list, separating the base chart from the annotation representation, making annotations modular and pluggable.

(2) The middle layer structure contains Task, Data, and Operation lists, representing the data and operation sets used by an annotation to achieve a specific task.

(3) The inner layer structure defines the specific operation Targets and added Marker for each Operation, detailing the specifics of operation execution.

4.2 Annotation Grammar Definition

Using the representation framework as a foundation, we define the formal grammar of ChartMark. A unit ChartMark specification, denoted as *annotatedChart*, consists of a base chart and one or more annotation elements:

$$annotatedChart := (chart, annotations) \tag{1}$$

4.2.1 Chart Entity

The chart component represents the base chart and its properties:

$$chart := (title, type, x\_data, y\_data, x\_name, y\_name, \ldots) \tag{2}$$

where *type* indicates the chart type (e.g., bar, line, scatter), $*\_name$ parameters define the semantic labels for axes, and $*\_data$ parameters store the actual data values along each axis.

4.2.2 Annotation Elements

Annotations are composed of a collection of annotation units, each with a specific purpose:

$$annotations := [annotation_1, annotation_2, \ldots, annotation_n] \tag{3}$$

$$annotation := (id, task, data, operations) \tag{4}$$

where *id* provides a unique identifier for each annotation, enabling referencing and composition of annotations.

Task Component. The task component captures the purpose and intent of the annotation [14, 9]:

$$task := (type, subType?) \tag{5}$$

```
{ ...
  "annotations": [{
1     "id": "1",
      "task": {"type": "trend"},
      "data": {"source": "derived"},
      "operations": [{
        "name": "linear_regression",
        "target": {"type": "data_items"},
        "marker": {
          "line": {"size": 2, "dashed": true}
        }
      }]
  },{
2     "id": "2",
      "task": {"type": "description", "subType": "local_note"},
      "data": { ... },
      "operations": [{
        "target": {"type": "annotation", "prior": "1"}, ...}
      ]
  },{
3     "id": "3",
      "task": {"type": "description", "subType": "local_note"},
      "data": {
        "source": "external",
        "values": [
          {"type": "image", "url": "https://www.svgrepo.com/show/396235/cross-mark.svg"},
          {"type": "text", "content": "Students with Failing Grades"}]
      },
      "operations": [{
        "name": "in_plot",
        "target": {
          "type": "data_items",
          "filter": {"and": [{"axisType": "y_quantity", "lt": 60}]}
        },
        "marker": {
          "img": {"field": "data", "dataIndex": 1, "width": 10, "height": 10},
          "text": {"field": "data", "dataIndex": 2, "color": "black"}
        }
      }]}]
}
```

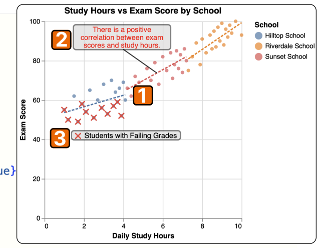


Figure 3: ChartMark specification for a scatter plot with three annotations: (1) dashed trend lines using linear regression across school groups, (2) an explanatory text note attached to the trend line highlighting the positive correlation, and (3) cross-mark symbols on failing scores (<60) with a legend explaining their meaning.

```
{ ...
  "annotations": [{
1     "id": "1",
      "task": {"type": "reference", "subType": "extra_area"}, "data": {"source": "external"},
      "operations": [{
        "name": "bounding_box",
        "target": {
          "type": "coordinate",
          "xyCoordinate": {
            "x": {"year": 2024, "month": "jul", "date": 15},
            "x1": {"year": 2024, "month": "oct", "date": 15},
            "y": 240,
            "y1": 280
          }},
        "marker": {...}
      }]
  },{
2     "id": "2",
      "task": {"type": "reference", "subType": "extra_range"},
      ...
  },{
3     "id": "3",
      "task": {"type": "reference", "subType": "extra_line"},
      ...
  },{
4     "id": "4",
      "task": {"type": "reference", "subType": "grid_line"},
      "data": {"source": "external"},
      "operations": [{
        "name": "grid_line",
        "target": {
          "type": "chart_element",
          "yAxis": {"tickCount": 6}
        }}]}
  ...]
}
```

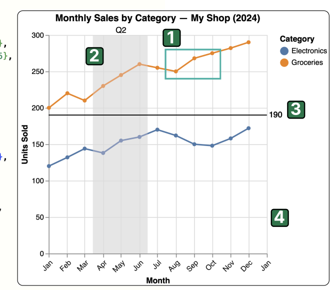


Figure 4: ChartMark specification for a group line chart with four reference-type annotations: (1) a bounding box highlighting a specific time-value region (Jun-Oct 2024, 240-280 units), (2) a shaded background marking Q2, (3) a horizontal reference line at 190 units, and (4) customized y-axis grid lines with 6 intervals.

$$type := reference \mid highlight \mid description \mid summary \mid trend \mid encoding \tag{6}$$

The optional *subType* parameter is contingent on the primary task type and provides further specification. For example, when *type* is "summary", *subType* can be "max" or "min", which means the annotation is intended to highlight the location or elements representing maximum or minimum values in the chart.

Data Component. The data component specifies the information needed by the annotation [18, 12, 8]:

$$data := (source, values) \tag{7}$$

$$source := external \mid derived \mid internal \mid none \tag{8}$$

The *source* parameter indicates the origin of data, where *external* represents information imported from outside the chart context, such as descriptive text and images (3 in Fig. 3), as well as display configuration parameters such as tickcount settings in gridlines (4 in Fig. 4). *derived* indicates computed values based on chart data, such asstatistical aggregates like mean (Fig. 1-c). *internal* refers to direct references to existing data items, used when annotations need to select data-related marks in the chart, such as in highlight annotations (Fig. 1-a). *none* is used for annotations that do not require specific data references.

Values represent a collection of multiple data item units, organized as an array. Each individual value unit is a tuple containing a type and either content or url:

$$values := [value_1, value_2, \ldots, value_n] \tag{9}$$

$$value := (type, content \mid url) \tag{10}$$

3 in Fig. 3 shows two ways to specify data values: using a url to reference external images or using the content field to embed text directly.

Operation Components. Operations are a collection of specific operating behaviors. Each operation represents a distinct annotation action that defines how visual elements are transformed [9]:

$$operations := [operation_1, operation_2, \ldots, operation_n] \tag{11}$$

$$operation := (name, target, marker) \tag{12}$$

Target Component. *target* field specifies what elements the operation acts upon, which can be summarized as four types below [18, 19, 9]:

$$target := (type, \ldots) \tag{13}$$

$$type := data\_items \mid coordinate \mid chart\_element \mid annotation \tag{14}$$

Different target types require different additional parameters:

data_items: uses filter parameters to select specific data points based on logical expressions (3 in Fig. 3). *coordinate*: defines regions, intervals, points, or lines within the coordinate system using spatial parameters (1 in Fig. 4). *chart_element*: references existing chart components for modification through component identifiers (4 in Fig. 4). *annotation*: references existing annotations through an ID parameter to enable composition of multiple annotations (1 and 2 in Fig. 3).

Marker Component. *marker* field indicates that added or modified visual elements in a operation. It serves as a collection of display-element options, represented as an optional parameter list, where each parameter specifies a single, fine-grained display-element setting [19]:

$$marker := (line?, text?, \ldots) \tag{15}$$

$$line := (size?, dashed?, \ldots) \tag{16}$$

$$text := \ldots \tag{17}$$

For example, when drawing a trend line, the detailed display parameters like size or dashed for line can be set. (1 in Fig. 3)

5 ChartMark Usage Demonstration

To demonstrate the practical capabilities of the proposed ChartMark grammar, we developed a Python package named `ChartMark.py`, available as open source at github.com/HKUSTDial/ChartMark. This toolkit facilitates converting ChartMark specifications into Vega-Lite visualizations [21] and provides users with an accessible interface to implement structured chart annotations seamlessly.

Figure 5: ChartMark in Jupyter notebook

5.1 Using ChartMark in Jupyter Notebook

Fig. 5 illustrates a complete example of using `ChartMark.py` within a Jupyter notebook. First, the users load a JSON-formatted ChartMark instance (see Fig. 1) using `load_json()`. Next, they generate a Vega-Lite specification for the base chart (without annotations) via `render_original_chart()`. To render this base chart in the notebook (upper half of Fig. 5), they pass the generated spec to `display_vegalite()`. Then, the user applies annotations by calling `render_annotations()`, which produces a fully annotated Vega-Lite specification based on the ChartMark grammar. Finally, they again use `display_vegalite()` to display the annotated chart (lower half of Fig. 5), showcasing various annotation types (e.g., highlighting, labeling, mean lines) that enhance the interpretability of the original chart.

In addition, the package includes utility methods such as `get_supported_chart_types()`, enabling users to quickly identify currently supported chart types.

5.2 Extension and Reuse

Benefiting from a modular component-based architecture, ChartMark supports the rapid construction of custom annotations by reusing modular components. As an example, consider the creation of a custom annotation operation that adds stroke for specific slices in a pie chart (Annotation2 in Fig. 2-a). In `ChartMark.py`, creating such a custom annotation generally follows three steps:

1. Decide the constructive components. Select the necessary components that make up the new annotation node. While new Marker and Operation components need to be defined to represent the custom operation's semantics and visual form, the Task, Data, and Target components can be reused directly due to their consistent roles with those in predefined annotation types.

2. Register into the ChartMark AST. Add the node representing the new annotation type to the appropriate position in the abstract syntax tree (AST), and define parsing rules to interpret the content represented by this node.

3. Implement transformation logic. Define how each operation within the annotation is mapped to concrete annotation statements in the target visualization language (e.g., Vega-Lite [21]), ensuring correct rendering.

6 Conclusion and Future Work

We introduced ChartMark, a structured grammar designed to integrate fragmented annotation methods. ChartMark's hierarchical design can take into account both high-level annotation intent and low-level visual operations, and its definition is independent of specific visualization libraries. We verify its portability and powerful expressiveness based on Vega-Lite's implementation. In future work, we plan to support dynamic annotation, explore new interactive methods that allow users to input in natural language and automatically generate annotations based on semantic intent [25, 24], and expand compatibility with more visualization libraries to enhance the accessibility of visualization and data storytelling capabilities [29, 23].

References

[1] Adobe illustrator. https://www.adobe.com/products/illustrator, 2025. 2

[2] Tableau. https://www.tableau.com, 2025. 2

[3] F. Amini, N. H. Riche, B. Lee, A. Monroy-Hernandez, and P. Irani. Authoring data-driven videos with dataclips. *IEEE transactions on visualization and computer graphics*, 23(1):501–510, 2016. 2

[4] S. Bateman, R. L. Mandryk, C. Gutwin, A. Genest, D. McDine, and C. Brooks. Useful junk? the effects of visual embellishment on comprehension and memorability of charts. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 2573–2582, 2010. 1

[5] M. Bostock, V. Ogievetsky, and J. Heer. D3: Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011. 2

[6] M. Bostock, V. Ogievetsky, and J. Heer. D³ data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011. doi: 10.1109/TVCG.2011.185 2

[7] M. Brehmer, B. Lee, N. H. Riche, D. Tittsworth, K. Lytvynets, D. Edge, and C. White. Timeline storyteller. In *Proceedings of the Computation+ Journalism Symposium, Miami, FL, USA*, vol. 6, 2019. 2

[8] C. Bryan, K.-L. Ma, and J. Woodring. Temporal summary images: An approach to narrative visualization via interactive annotation generation and placement. *IEEE transactions on visualization and computer graphics*, 23(1):511–520, 2016. 3

[9] Q. Chen, Z. Liu, C. Wang, X. Lan, Y. Chen, S. Chen, and N. Cao. Vizbelle: A design space of embellishments for data visualization. *arXiv preprint arXiv:2209.03642*, 2022. 1, 2, 3, 4

[10] R. Eccles, T. Kapler, R. Harper, and W. Wright. Stories in geotime. *Information Visualization*, 7(1):3–17, 2008. 2

[11] J. Hao, M. Yang, Q. Shi, Y. Jiang, G. Zhang, and W. Zeng. Finflier: Automating graphical overlays for financial visualizations with knowledge-grounding large language model. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–17, 2024. 2

[12] J. Hullman, N. Diakopoulos, and E. Adar. Contextifier: automatic generation of annotated stock visualizations. In *Proceedings of the SIGCHI Conference on human factors in computing systems*, pp. 2707–2716, 2013. 2, 3

[13] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. 2

[14] N. Kong and M. Agrawala. Graphical overlays: Using layered elements to aid chart reading. *IEEE transactions on visualization and computer graphics*, 18(12):2631–2638, 2012. 3

[15] C. Li, J. Zhao, B. Lee, Q. Yuan, and X. Luo. Echarts: A declarative framework for rapid construction of web-based visualization. *Visual Informatics*, 2(1):1–12, 2018. 2

[16] Y. Luo, X. Qin, N. Tang, and G. Li. Deepeye: Towards automatic data visualization. In *ICDE*, pp. 101–112. IEEE Computer Society, 2018. 1

[17] D. Moritz, C. Wang, G. L. Nelson, H. Lin, A. M. Smith, B. Howe, and J. Heer. Formalizing visualization design knowledge as constraints: Actionable and extensible models in draco. *IEEE transactions on visualization and computer graphics*, 25(1):438–448, 2018. 1, 3

[18] M. D. Rahman, G. J. Quadri, B. Doppalapudi, D. A. Szafir, and P. Rosen. A qualitative analysis of common practices in annotations: A taxonomy and design space. *IEEE Transactions on Visualization and Computer Graphics*, pp. 360 – 370, 2024. 1, 2, 3, 4

[19] D. Ren, M. Brehmer, B. Lee, T. Höllerer, and E. K. Choe. Chartaccent: Annotation for data-driven storytelling. In *2017 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 230–239. Ieee, 2017. 1, 2, 3, 4

[20] D. Ren, B. Lee, and M. Brehmer. Charticulator: Interactive construction of bespoke chart layouts. *IEEE transactions on visualization and computer graphics*, 25(1):789–799, 2018. 1

[21] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-Lite: A Grammar of Interactive Graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):341–350, 2017. 1, 2, 4

[22] A. Satyanarayan, K. Wongsuphasawat, and J. Heer. Declarative interaction design for data visualization. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST)*, pp. 669–678. ACM, 2014. doi: 10.1145/2642918.2647360 2

[23] L. Shen, H. Li, Y. Wang, and H. Qu. Reflecting on Design Paradigms of Animated Data Video Tools. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pp. 1–21. ACM, 2025. 4

[24] L. Shen, H. Li, Y. Wang, X. Xie, and H. Qu. Prompting Generative AI with Interaction-Augmented Instructions. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems, CHI EA '25*, pp. 1–9. ACM, 2025. 4

[25] L. Shen, E. Shen, Y. Luo, X. Yang, X. Hu, X. Zhang, Z. Tai, and J. Wang. Towards Natural Language Interfaces for Data Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics*, 29(6):3121–3144, 2023. 4

[26] L. Shen, E. Shen, Z. Tai, Y. Song, and J. Wang. TaskVis: Task-oriented Visualization Recommendation. In *Proceedings of the 23th Eurographics Conference on Visualization, EuroVis'21*, pp. 91–95. Eurographics, 2021. 2

[27] L. Shen, E. Shen, Z. Tai, Y. Wang, Y. Luo, and J. Wang. GALVIS: Visualization Construction through Example-Powered Declarative Programming. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, CIKM'22*, pp. 4975–4979. ACM, 2022. 2

[28] L. Shen, E. Shen, Z. Tai, Y. Xu, J. Dong, and J. Wang. Visual Data Analysis with Task-Based Recommendations. *Data Science and Engineering*, 7(4):354–369, 2022. 2

[29] L. Shen, Z. Tai, E. Shen, and J. Wang. Graph Exploration With Embedding-Guided Layouts. *IEEE Transactions on Visualization and Computer Graphics*, 30(7):3693–3708, 2024. 4

[30] S. Shen, S. Lu, L. Shen, Z. Sheng, N. Tang, and Y. Luo. Ask humans or ai? exploring their roles in visualization troubleshooting. *arXiv preprint arXiv:2412.07673*, 2024. 3

[31] Y. Shi, B. Li, Y. Luo, L. Chen, and N. Tang. Augmenting realistic charts with virtual overlays. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pp. 1–23, 2025. 1, 3

[32] T. Siddiqui, A. Kim, J. Lee, K. Karahalios, and A. Parameswaran. Effortless data exploration with zenvisage: An expressive and interactive visual analytics system. *Proceedings of the VLDB Endowment*, 10(4):457–468, 2016. 2

[33] C. Stolte, D. Tang, and P. Hanrahan. Polaris: a system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):52–65, 2002. 1, 2

[34] H. Wickham. A Layered Grammar of Graphics. *Journal of Computational and Graphical Statistics*, 19(1):3–28, 2010. 1, 2

[35] H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer International Publishing, 2 ed., 2016. 2

[36] L. Wilkinson. The grammar of graphics, in 'handbook of computational statistics', 2012. 2

[37] Y. Wu, L. Yan, L. Shen, Y. Mei, J. Wang, and Y. Luo. Chartcards: A chart-metadata generation framework for multi-task chart understanding. *arXiv preprint arXiv:2505.15046*, 2025. 1

[38] Y. Wu, L. Yan, L. Shen, Y. Wang, N. Tang, and Y. Luo. ChartInsights: Evaluating Multimodal Large Language Models for Low-Level Chart Question Answering. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 12174–12200. ACL, Stroudsburg, PA, USA, 2024. 1

[39] Y. Wu, L. Yan, Y. Zhu, Y. Mei, J. Wang, N. Tang, and Y. Luo. Boosting text-to-chart retrieval through training with synthesized semantic insights. *arXiv preprint arXiv:2505.10043*, 2025. 1

[40] Y. Xie, Y. Luo, G. Li, and N. Tang. Haichart: Human and ai paired visualization system. *Proceedings of the VLDB Endowment*, 17(11):3178–3191, 2024. 2

[41] X. Yang, Y. Wu, Y. Zhu, N. Tang, and Y. Luo. Askchart: Universal chart understanding through textual enhancement. *arXiv preprint arXiv:2412.19146*, 2024. 1

[42] Y. Yu, L. Shen, F. Long, H. Qu, and H. Chen. PyGWalker: On-the-fly Assistant for Exploratory Visual Data Analysis. In *Proceedings of IEEE Visualization and Visual Analytics, IEEE VIS'24*, pp. 1–5, 2024. 2