Towards A General-Purpose Query Language for Visualization Recommendation

Kanit Wongsuphasawat
University of Washington
kanitw@uw.edu

Dominik Moritz
University of Washington
domoritz@uw.edu

Anushka Anand
Tableau Research
aanand@tableau.com

Jock Mackinlay
Tableau Research
jmackinlay@tableau.com

Bill Howe
University of Washington
billhowe@uw.edu

Jeffrey Heer
University of Washington
jheer@uw.edu

ABSTRACT

Creating effective visualizations requires domain familiarity as well as design and analysis expertise, and may impose a tedious specification process. To address these difficulties, many visualization tools complement manual specification with recommendations. However, designing interfaces, ranking metrics, and scalable recommender systems remain important research challenges. In this paper, we propose a common framework for facilitating the development of visualization recommender systems in the form of a specification language for querying over the space of visualizations. We present the preliminary design of CompassQL, which defines (1) a partial specification that describes enumeration constraints, and (2) methods for choosing, ranking, and grouping recommended visualizations. To demonstrate the expressivity of the language, we describe existing recommender systems in terms of CompassQL queries. Finally, we discuss the prospective benefits of a common language for future visualization recommender systems.

Keywords

Visualization Tools, Visualization Recommendation, Mixed-initiative Systems

1. INTRODUCTION

Visualization is an essential tool for data analysts to explore and reason about data. However, to create a visualization, analysts typically have to specify queries and visual encodings manually, either via a programming library such as ggplot [22, 23] or via a graphical interface such as Tableau [17]. While manual specification provides flexibility for creating a variety of charts, it can be tedious and impede comprehensive coverage of the data in exploratory analysis.

Moreover, creating visualizations that effectively achieve analysis or presentation goals requires visual analysis expertise and knowledge of the data domain. In practice, users might not have complete specifications of the ideal visualizations in mind; as a result, they have to iteratively create and refine visualizations to search for satisfactory results [9]. Due to the tedium of the specification process and the size of the search space, users may explore only a fraction of the space, choose an inferior presentation, or even overlook critical insights in the data [25].

To address these difficulties, some tools complement manual specification with recommendations, fostering mixed-initiative interactions [10]. Users can indicate their intent in the form of partial specifications, and the system in turn completes the unspecified parts to produce recommendations, reducing the burden of providing complete and correct specifications. For example, a user of Voyager [25] can select a set of data fields, and the system in turn suggests visual encodings ranked according to perceptual principles. To help users explore large or unfamiliar data, some systems [16, 21] use summary statistics to suggest potentially interesting fields or relevant subsets of the data.

These different types of recommendations are useful for specific tasks. To develop more effective visualization recommender systems, important research challenges include designing appropriate interfaces, ranking metrics, and scalable data processing engines. To assist these goals and provide a shared foundation for research, we propose a unified framework in the form of a query language for expressing desired recommendations.

In this paper, we first discuss different types of recommendations and common patterns in the recommendation process, including phases of enumerating, choosing, and ranking candidate visualizations. We then present the preliminary design of CompassQL, a query language that defines a set of visualization recommendations through a set of enumeration constraints combined with methods for choosing, ranking, and grouping visualizations. We demonstrate the expressivity of the language design by describing existing visualization recommendation systems in terms of CompassQL queries. Finally, we discuss the potential benefits of the query language for the development of future visualization recommendation tools.

2. BACKGROUND

2.1 Visualization Specification Languages

Specification languages are fundamental to visualization creation, either as programming interfaces [1, 22] or as un-

HILDA '16 San Francisco, CA USA

DOI: 10.1145/1235

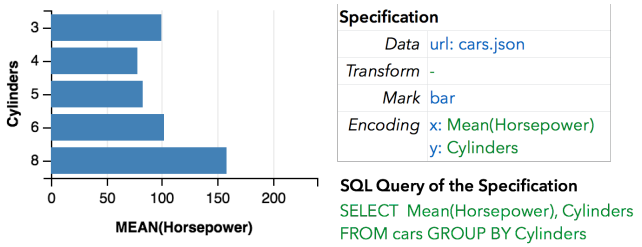


Figure 1: A bar chart showing aggregate data, its Vega-Lite specification, and a SQL query that defines an aggregation equivalent to the specification. The specification defines data source, mark type, and mappings between encoding channels (`x` and `y`) and (potentially transformed) data fields.

derlying representations for graphical user interfaces [17, 25]. Chart typologies, or templates commonly found in spreadsheet and dashboard tools, provide the simplest interfaces, but have limited expressivity: users can only create a chart if the tool provide a template. Conversely, with low-level graphic APIs like OpenGL one can express any number of visualizations, but at the cost of significant software engineering effort. Grammar-based languages offer primitives such as marks and scales for composing visualizations. Low-level grammars such as D3 [7] and Vega [15] support a broad range of custom visualization designs, but are verbose and require explicit definitions of all primitive components. Higher-level grammars such as ggplot2 [22], Vega-Lite [25] and Tableau's VizQL [17] are more concise and convenient for enumeration. These specifications can omit details such as scale and axis definitions. A rule system is used to resolve appropriate defaults for these components.

2.2 Vega-Lite

In this paper, we represent visualizations using the Vega-Lite grammar. Vega-Lite specifications describe visualizations as mappings from data to properties of graphical marks such as points or bars. Fig. 1 shows a schematic Vega-Lite specification for a bar chart. The `data` property describes a tabular data source (*e.g.,* via url), which is a set of records with named fields. Based on the `mark` type and a set of `encoding` mappings between visual channels and data fields, Vega-Lite automatically generates necessary visualization components including scales, axes, and legends. Vega-Lite uses a rule-based approach to determine the default properties of these components. To facilitate analysis, Vega-Lite supports data transformation such as aggregation, binning, filtering, and sorting. In addition, it also supports faceting a single plot into trellis plots [5] with `row` and `column` channels.

With Vega-Lite, users can express a variety of common, useful plots of both raw and aggregate data. Examples include bar charts, histograms, dot plots, scatter plots, line graphs, and area graphs. The online documentation [3] describes the complete syntax and semantics of Vega-Lite.

3. VISUALIZATION RECOMMENDATION

We now discuss different types of visualization recommenders, and a general pattern for the recommendation process.

3.1 Types of Recommender Systems

Fig. 2 displays a matrix that categorizes visualization tools

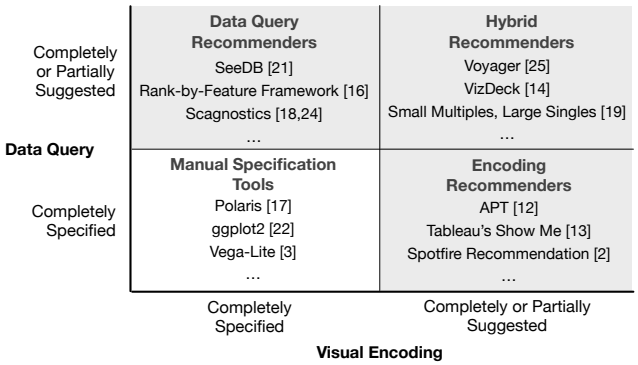


Figure 2: Matrix of visualization tools grouped by the type of recommendations.

based on whether the system suggests *what* data to visualize (data query recommendations) or *how* to visualize it (visual encoding recommendations).

Encoding recommenders such as APT [12], Tableau's Show Me [13], and Spotfire Recommendations [2] suggest effective graphical representations for provided data fields, assisting users with less design expertise to quickly create visualizations. Nevertheless, users still have to manually specify data queries to explore different data.

Conversely, *data query* recommenders [16, 21] and related techniques [4, 6, 24] suggest variations of query parameters (such as different projections) for a fixed visual encoding template. These systems can facilitate exploration of large data sets, but their utility is limited by the use of fixed visualization forms.

To support a broader set of analysis tasks, a few projects have developed *hybrid recommenders* [14, 19, 20, 25], which suggest both visual encodings and data queries. While hybrid recommendation is still nascent, initial user studies show promise for this approach [25]. Going forward, important challenges include formulating and evaluating useful ranking metrics, and designing interfaces that appropriately balance user control and automatic recommendation.

3.2 Recommendation Process

Visualization recommender systems typically follow a similar process. First, a recommender system searches for a set of visualizations that match the user's intent. To perform this search, the system enumerates different data query and/or visual encoding parameters to produce specifications that satisfy a set of constraints. Given some satisfiable candidates, the system then orders recommendations based on a utility function. To avoid redundant recommendations, the system may group similar candidates and select a top-ranked representative from each group.

Enumeration. A system may enumerate data query parameters including projected fields, filters, aggregation, sorting, and field transformations such as binning. It may also enumerate visual encoding parameters including encoding mappings between data fields and visual channels such as x-position, y-position, or color.

To qualify for recommendation, a candidate visualization must satisfy two types of constraints. First, it should respect any *data query or encoding constraints* that partially express the user's intent. Second, it must satisfy *expressiveness constraints* [12]. For example, a visualization that encodes a quantitative field with the shape of symbols is

misleading since different shapes are unordered and therefore cannot convey the magnitude of quantitative values. Visualizations that use misleading encodings can be omitted from consideration.

Ranking. With a set of qualified candidates, the system then suggests the top item or produces an ordered list of recommendations based on various ranking functions.

Variations of encodings are typically ranked based on perceptual effectiveness metrics, applying prior works [8, 12] that rank the effectiveness of each encoding mapping based on the type of the encoded field and the encoding channel.

To rank different data queries, a recommender system might use a naive order based on the data schema [25]. This simple approach can work for datasets with small numbers of fields, but does not scale if there are many fields. In this case, the system might take a data-driven approach to compute statistics that best suit a task. For example, if the user is interested in correlation between two fields, the system might rank visualizations based on Pearson's correlation coefficient [16]. If the user looks for anomalies in the data, the system might instead use the number of outliers or other metrics that measure deviation, normality, or uniformity to rank the recommendations [11, 16, 21].

For hybrid recommendations, designing a holistic ranking metric that takes both data query and encoding parameters into account can be complicated. Existing tools apply separate encoding-based or data-based metrics to rank the recommendations. The Small Multiples, Large Singles system [19] varies only one of either the data query or the encoding parameter at a time, and thus can apply either type of metric directly. Voyager [25] groups visualizations backed by the same data query, applies an encoding-based metric to choose group representatives, and orders them using a data-based metric.

Reducing Redundancy. Many candidate visualizations may be similar and thus redundant, increasing the number of charts the user has to consider without providing additional value. For example, it is often unnecessary to suggest both horizontal and vertical bar charts of the same data fields. A recommender system might group similar charts to reduce redundancy and encourage diversity. For example, Show Me Alternatives [13] suggests only one instance each for a set of basic chart types, as shown in Fig. 3c. To support broader exploration, Voyager (Fig. 8) groups visualizations with the same data query and only presents the most perceptually effective item from each group in its main view.

4. VISUALIZATION QUERY LANGUAGE

We present the preliminary design of CompassQL, a query language that supports each of our identified phases of visualization recommendation. We plan to extend the Compass visualization recommender engine in Voyager [25] using this query language design to support a variety of visualization recommender tools.

A CompassQL query consists of a *partial specification* (which provides enumeration constraints) coupled with methods for choosing, ranking and grouping recommendations. With this query language, a recommender system may set some query parameters as a part of a template, with other parameters interactively specified by users. For illustration, in subsequent figures we highlight user provided parameters in red.

Partial Specification. We extend the Vega-Lite visualization grammar [3] with explicit *enumeration specifiers* to define properties that should be enumerated. We denote enumeration specifiers with bold, capital letters. Enumerated values are implicitly any compatible values. For example, setting the `mark` property in Fig. 3a to **M** means that the system should enumerate all possible mark types (*e.g.,* `bar`, `line`, `area`, `point`). To enumerate a set of visualizations with a varying number of fields, an encoding mapping can be optional as well. For example, a query in Fig. 7 enumerates both 1D and 2D visualizations.

Besides matching specified values, *expressiveness* constraints are also implicitly applied. For example, a recommender system should not assign quantitative fields to the `shape` channel, nor enumerate `mean` aggregation for ordinal fields. To avoid combinatorial explosion of the result set, a partial specification might contain extra constraints for enumeration specifiers. For instance, the aggregate function **A** in Fig. 8d is constrained to either `-` (no aggregate) or `mean`.

Choosing and Ordering. A query might specify the `choose` property to define how the system chooses the top recommendation, or set the `order by` property to produce a ranked list recommendations in decreasing order of scores.

Both the `choose` and `order by` properties refer to ranking methods that are either provided natively by the query engine or defined by developers as user-defined functions. We assume that commonly used ranking methods such as encoding effectiveness (`Effectiveness`) will be provided by the query engine by default.

These ranking methods take a visualization specification S and the data relation D as input. Some ranking methods such as encoding effectiveness may depend only on the specification and simple statistical properties such as cardinality. However, some data-driven metrics such as mutual information and deviation require more expensive data computation. Some ranking methods might also invoke an external module such as a learning model that improves over time.

For example, Fig. 3a shows a query for Show Me Automatic Marks [13], which attempts to suggest the most effective mark; the query `choose` a specification S that maximizes the `effectiveness` score. In Fig. 4a, the query sets `order by` property to a user-selected ranking R_1 to produce an ordered list of histograms.

Grouping. To reduce redundancy, a query can include a `group by` clause, providing a key function to induce groupings. When `group by` is provided, the `choose` property defines how the system chooses a top representative for each group; meanwhile, the `order by` property specifies how the system order the representative from each group in the list of recommendations. For example, as shown in Fig. 8d, Voyager groups visualizations by matching data queries and chooses the most perceptually effective visualization to represent a group; the chosen visualizations are then ordered by query simplicity: raw plots are shown before aggregate plots. Similar to the ranking methods, system developers may provide user-defined key functions to the query engine.

5. EXAMPLE COMPASSQL QUERIES

We now consider example queries for different types of recommendations supported by existing systems.

5.1 Encoding Recommendations

Encoding recommenders generate variations of visual encodings for a fixed, user-provided data query. Their recomm-

Figure 3: Tableau's Show Me features [13] and their CompassQL queries, which rank effectiveness of each specification S for dataset D. (a) *Automatic Marks* determines the most effective mark type for the specified data query and encoding mappings. (b) *Add To Sheet* recommends the most effective encoding mapping for a new field added to an existing visualization. (c) *Show Me Alternatives* suggests chart types for provided data fields. The interface highlights the most effective chart type with blue border.

endation queries have enumeration specifiers for mark type or visual channels in the encoding mappings. All of them rank outputs based on their encoding effectiveness.

Show Me [13] is a set of features that provides automation to facilitate the creation of visualizations in Tableau. Fig. 3a shows a query for *Automatic Marks*. Since the `mark` property is specified as enumerable, the system generates multiple candidate visualizations by varying the mark type and recommends the most effective presentation. In Fig. 3b, the user selects a field (`Origin`) to add it to the view with *Add To Sheet*. The system enumerates and ranks alternative mappings between the selected field and available encoding channels. Fig. 3c shows a query for *Show Me Alternatives*,

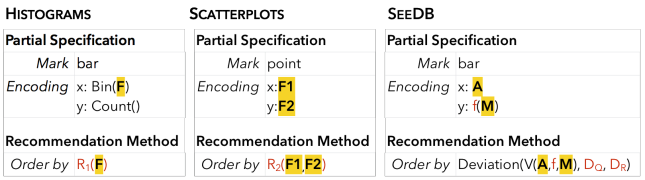


Figure 4: Data query recommendations. (a-b) Rank-by-Feature Framework [16] queries for ranking histograms and scatterplots by the selected metrics R_1 and R_2. (c) a query for ranking bar charts of aggregate views V with varying dimension A and measure M by the deviation between select target and reference data subsets (D_Q, D_R) akin to SeeDB [21].

which recommends alternative chart types for selected fields or fields of the current chart, if users do not select any fields. All channels in the encoding mappings and *marks* are specified as enumerable. The recommendations are shown as a list of compatible chart types, with the top-ranked type highlighted. If the user selects a chart type that has multiple compatible encoding mappings, Show Me recommends the top-ranked encoding.

APT [12] and **Spotfire Recommendations [2]** suggest encodings for selected fields akin to Show Me Alternatives (Fig. 3c) and thus have similar partial specifications. However, APT only recommends the top result (`Choose`: $argmax_v$ Effectiveness(S, D)). Meanwhile, Spotfire appears to suggest variations of the same chart type with different encoding mappings and thus does not `group by` chart types.

5.2 Data Query Recommendations

CompassQL queries that recommend the data to view fix the visual encoding templates and only have enumeration specifiers for data query parameters such as the data fields.

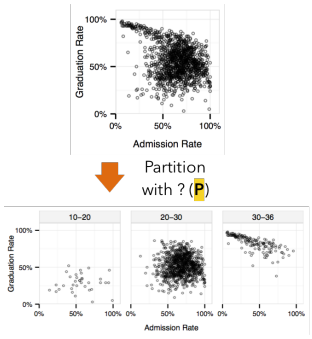
The **Rank-by-Feature Framework [16]** ranks histograms and scatterplots based on selected metrics. In the queries shown in Fig. 4 (a-b), the metrics R_1 and R_2 are functions of the enumerated fields (**F** for histograms; **F1** and **F2** for scatterplots). The framework provides many metrics such as distribution normality and distribution uniformity for histograms as well as correlation and the number of potential outliers for scatterplots. Other works such as **Scagnostics [24]** propose a set of cognostics [18], or metrics that measure the relative interestingness of different displays, for ranking scatterplots as well.

SeeDB [21] suggests bar charts of aggregate views with the highest deviation between the user provided target and reference data subsets (D_Q and D_R). Fig. 4c shows a recommendation query akin to this suggestion[1]. The system ranks each aggregate view V that computes a selected aggregate function f of field **M** grouped by field **A** based on the deviation between the data subsets, which is defined as the distance between their probability distributions:

$$Deviation(V, D_Q, D_R) := S(P[V(D_Q)], P[V(D_R)])$$

where S is a distance function (*e.g.*, Earth Mover's Distance) and $P[V(D)]$ is a probability distribution of V given data subset D.

[1] SeeDB presents recommended views as grouped bar charts that juxtapose each dimension's value of both data subsets. Our formulation enables an implementation of an equivalent display that shows small multiple bar charts of the recommended view (each multiple represents each data subset).



Partial Specification

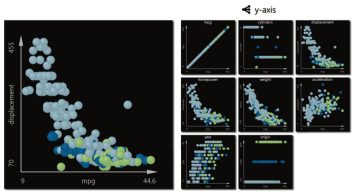
Mark	point
Encoding	x: Admission Rate y: Graduation Rate row: P

P ∈ {*Distance to an employment center, …*}

Recommendation Method

Order by	InterestingConditional-Structure(S, D)

Figure 5: Automatic Selection of Partitioning Variables for Small Multiple Displays [4] can be expressed as a CompassQL query that enumerates partition fields and ranks them based on a randomized, non-parametric permutation test that determines interesting conditional structure in the data.



Partial Specification

Mark	point
Encoding	x: Miles_per_Gallon y: F

Recommendation Method

Order by	SchemaIndex(F)

Figure 6: The Small Multiples Large Singles system [19] when a user chooses to enumerate the field on y-axis, expressed as a CompassQL query.

Partial Specification

Mark	M
Encoding	C1: A1(F1) C2: A2(F2) (Optional)

A1, A2 ∈ {*-, Mean*}

Recommendation Method

Order by	RankingModel(S, D, userVotes)

Figure 7: VizDeck [14] showing 1D and 2D visualizations ordered by a ranking model trained with user votes, expresses as a CompassQL query.

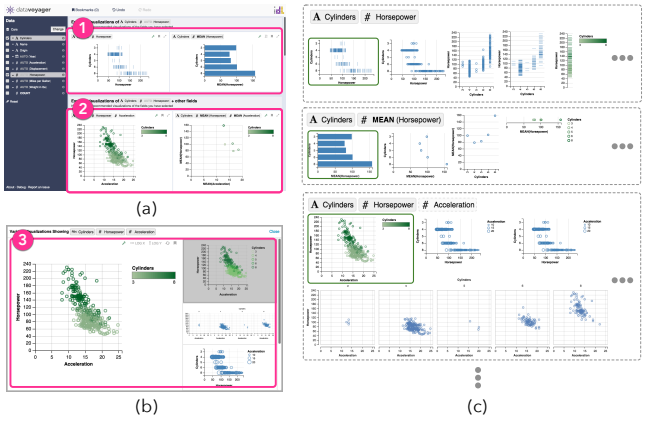
Automatic Selection of Partitioning Variables for Small Multiple Displays [4] is a method to rank partitioning fields that reveal interesting pattern in the data with a randomized, non-parametric permutation test and cognostics [18]. The query shown in Fig. 5 varies the partition field **P**, which is mapped to the `row` channel to create small multiples of scatterplots, and ranks each small multiple based on this method.

5.3 Hybrid Recommendations

Hybrid recommendation queries can have enumeration specifiers for both data query and encoding parameters.

Small Multiples, Large Singles [19] shows small multiple displays that are variants of a main display. The variants are produced by changing either a data query (filtering data, changing an axis) or a visual encoding parameter (changing size, color, or the parameters of a layout algorithm). For example, Fig. 6 shows the interface and a CompassQL query with the field **F** on the y-axis varied.

VizDeck [14], as shown in Fig. 7, displays a ranked list of 1D and 2D visualizations which the user can vote up or down. The system ranks the results using a combination of



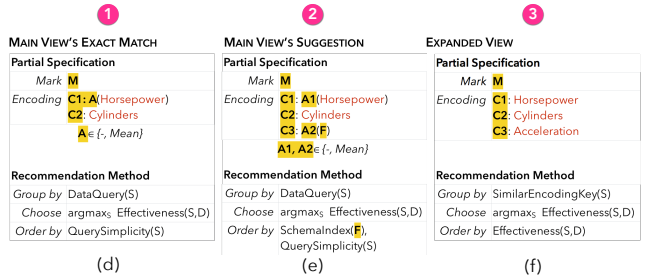


Figure 8: Voyager [25]. (a) The *Main View* shows visualizations with different data queries, promoting exploration of different fields. The *Exact Match* section contains all and only the selected fields. The *Suggestion* section contain all selected fields and one extra field to encourage further exploration. (b) The *Expanded View* displays various encodings of the same data query. (c) Groups of enumerated visualizations. The *Main View* only displays the top item from each group (highlighted with green border). (d-f) CompassQL queries for different sections.

heuristics and a model of visualization quality that learns the relationship between summary statistics of the data (e.g., entropy, coefficient of variation, kurtosis, and periodicity) and voting feedback collected by the interface.

Voyager [25] suggests visualizations based on selected fields, showing both variations of data queries and visual encodings. The *Main View* (Fig. 8a) has two sections that display visualizations for different data queries to encourage exploration of different fields. The *Exact Match* section lists visualizations of selected fields with varying aggregation levels. The *Suggestion* section adds an extra field to promote further exploration. Fig. 8d and 8e illustrate recommendation queries for the *Main View*. Both queries group results by data query and suggest the top chart from each cluster. The *Exact Match* section orders visualizations by data query simplicity, showing raw plots before aggregate plots. The *Suggestion* section orders visualizations by schema indices of the added fields, then by the same query simplicity criteria. Users can browse different encodings of a particular data query in the *Expanded View* (Fig. 8b). Fig. 8f shows its query, which is similar to encoding recommenders. It enumerates only mark and encoding mappings, and then ranks outputs by encoding effectiveness. To reduce redundancy, the *Expanded View* also groups visualizations with similar encodings and only suggests one visualization for each cluster. For example, it does not suggest another scatterplot that is a transpose of the scatterplot shown in Fig. 8b.

6. DISCUSSION & FUTURE WORK

In this paper, we present a preliminary design of CompassQL, a query language that describes parameters of the visualization recommendation process, and demonstrate example queries for different types of recommendations.

As a next step, we plan to extend Compass recommendation engine to supports CompassQL queries. Just like SQL has enabled researchers and companies to develop highly optimized but compatible engines and various higher-level applications, we hope that CompassQL will facilitate the development of compatible recommender engines as well as recommender interfaces. For example, having a clear separation of the engine and the interface may allow database researchers to focus on optimizing scalable processing while allowing HCI researchers to focus on the design of better recommender interfaces. The visualization community could assess the encoding effectiveness rankings and integrate the findings into recommender systems. Meanwhile, statisticians might formulate improved data-centric ranking methods. Overall, we hope to foster collaborative research efforts to design visualization recommender systems that help people explore data more effectively.

7. ACKNOWLEDGMENTS

We thank Alvin Chueng, and members of the Interactive Data Lab for their feedback. This work was supported in part by the Intel Big Data ISTC, DARPA XDATA, the Gordon & Betty Moore Foundation, and the University of Washington eScience Institute.

8. REFERENCES

[1] Matplotlib documentation. http://matplotlib.org/.

[2] Spotfire recommendations. http://spotfire.tibco.com/recommendations.

[3] Vega-Lite documentation. https://vega.github.io/vega-lite/docs/.

[4] A. Anand and J. Talbot. Automatic selection of partitioning variables for small multiple displays. *Visualization and Computer Graphics, IEEE Transactions on*, 22(1):669–677, 2016.

[5] R. A. Becker, W. S. Cleveland, and M.-J. Shyu. The visual design and control of trellis display. *Journal of computational and Graphical Statistics*, 5(2):123–155, 1996.

[6] E. Bertini, A. Tatu, and D. Keim. Quality metrics in high-dimensional data visualization: an overview and systematization. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2203–2212, 2011.

[7] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2301–2309, 2011.

[8] W. S. Cleveland and R. McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association*, 79(387):531–554, 1984.

[9] L. Grammel, M. Tory, and M. Storey. How information visualization novices construct visualizations. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):943–952, 2010.

[10] E. Horvitz. Principles of mixed-initiative user interfaces. In *Proc. ACM Human Factors in Computing Systems (CHI)*, pages 159–166, 1999.

[11] S. Kandel, R. Parikh, A. Paepcke, J. M. Hellerstein, and J. Heer. Profiler: Integrated statistical analysis and visualization for data quality assessment. In *Proc. Advanced Visual Interfaces (AVI)*, pages 547–554. ACM, 2012.

[12] J. Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2):110–141, 1986.

[13] J. Mackinlay, P. Hanrahan, and C. Stolte. Show me: Automatic presentation for visual analysis. *IEEE Transactions on Visualization and Computer Graphics (Proc. InfoVis)*, 13(6):1137–1144, 2007.

[14] D. B. Perry, B. Howe, A. M. Key, and C. Aragon. Vizdeck: Streamlining exploratory visual analytics of scientific data. 2013.

[15] A. Satyanarayan, R. Russell, J. Hoffswell, and J. Heer. Reactive vega: A streaming dataflow architecture for declarative interactive visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 22(1):659–668, 2016.

[16] J. Seo and B. Shneiderman. A rank-by-feature framework for interactive exploration of multidimensional data. *Information Visualization*, 4(2):96–113, 2005.

[17] 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.

[18] J. W. Tukey and P. A. Tukey. Computer graphics and explaoratory data analysis: An introduction. In *Proceedings of the Sixth Annual Conference and Exposition: Computer Graphics*, 1985.

[19] S. van den Elzen and J. J. van Wijk. Small multiples, large singles: A new approach for visual data exploration. *Computer Graphics Forum*, 32(3pt2):191–200, 2013.

[20] M. Vartak, S. Huang, T. Siddiqui, S. Madden, and A. Parameswaran. Towards visualization recommendation systems.

[21] M. Vartak, S. Madden, A. Parameswaran, and N. Polyzotis. SeeDB: Automatically generating query visualizations. *Proceedings of the VLDB Endowment*, 7(13):1581–1584, 2014.

[22] H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer, 2009.

[23] L. Wilkinson. *The Grammar of Graphics*. Springer, 2005.

[24] L. Wilkinson, A. Anand, and R. L. Grossman. Graph-theoretic scagnostics. In *INFOVIS*, volume 5, page 21, 2005.

[25] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *Visualization and Computer Graphics, IEEE Transactions on*, 22(1):649–658, 2016.