Viz-Blocks: Building Visualizations and Documents in the Browser

G. McNeill[1] and S. A. Hale[1,2]

[1]Oxford Internet Institute, University of Oxford, UK
[2]Alan Turing Institute, UK

Abstract

Viz-Blocks is a simple browser-based UI for data exploration and document creation. It incorporates the Vega-Lite grammar of graphics for standard visualizations (including multiple views and interaction) whereas 'code blocks' provide the full power of the JavaScript ecosystem for creating custom visualizations and other bespoke content. Visualizations are treated as reusable 'blocks' that are easily created, modified and compared during exploration. When preparing results for dissemination, visualizations can be customized and combined with Markdown and image blocks to produce a single or multi-page HTML document that is easily styled and exported. Viz-blocks was designed in consultation with academics, students and policy makers to bridge the gap between visualization tools and more traditional document-authoring tools. The application is aimed at a wide audience: the lightweight, hybrid UI allows all users to access the core functionality, while experienced users can take advantage of code-blocks and the option to use advanced features of Vega-Lite via JSON/YAML snippets.

CCS Concepts

• ***Human-centered computing*** $\rightarrow$ ***Visualization systems and tools;*** • ***Applied computing*** $\rightarrow$ *Document preparation;*

1. Introduction

Increasingly users with more diverse backgrounds and skill levels are creating data visualizations. This creates a need for easy-to-use applications that cater to a wide audience while providing experienced users efficient access to more advanced techniques. The range of tasks that visualization is used for must also be taken into account when designing such tools. For example, data exploration typically involves generating many plots to better understand the data [DiB90], whereas disseminating insights derived from data often requires that selected visualizations be polished and placed in a wider narrative such as an article, slide deck, report or blog.

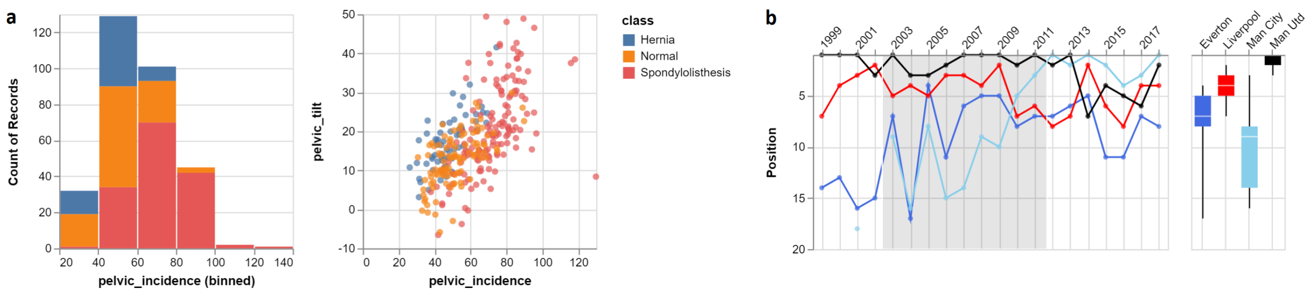
In this paper, we present Viz-Blocks: a lightweight interface for creating visualizations and documents in the browser. The core contributions of this work—each accompanied by an example use case—are as follows:

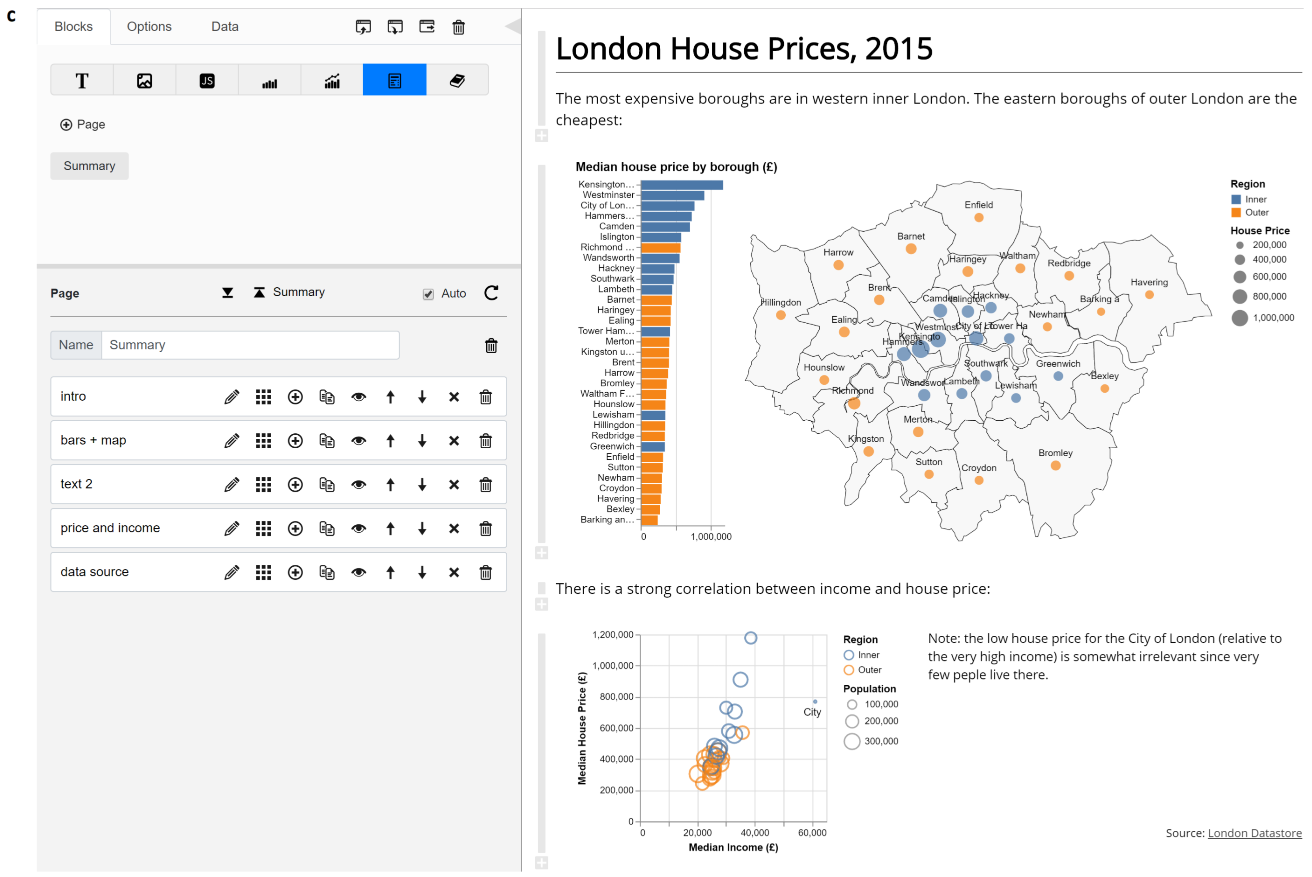
- Enable users of all levels to easily explore data by providing a simple interface to the Vega-Lite grammar of graphics [SMWH17] that treats visualizations as reusable, composable blocks.
  Use Case: A clinician with no programming experience wishes to quickly 'eyeball' patient data [Kag] to see how various biomechanical measurements relate to two common conditions. She quickly generates plots such as those in Figure 1a and related multi-view visualizations including a scatter plot matrix (not shown).
- Give advanced users the ability to efficiently construct more complex or bespoke visualizations by accessing the advanced functionality of Vega-Lite as well as free code blocks.
  Use Case: A data scientist at a news organization is asked for a visualization summarizing the performance of the big northwest clubs in the English Premier League (football) over the last twenty years (Figure 1b). He creates a multi-view Vega-Lite visualization, polishes it (including using the team colors and inverting the y axes), and adds crossfiltering interaction—the boxplots are based on the years selected by the user.
- Embed the visualization process in the wider context of document creation, allowing users to explore data, prepare visualizations for dissemination, and combine them with other document elements (text, images, mathematical expressions etc.) as required in a live preview environment.
  Use Case: A local government analyst is asked to prepare a short document explaining how London house prices relate to income and geography [Gre] (Figure 1c).

2. Related Work

Vega-Lite [SMWH17] is a high-level grammar of graphics that includes composition and interaction. Vega-Lite 'specs' are written in the popular JSON format, which makes the library an attractive target for applications [WMA*16, MWN*18, WKD19] and other libraries [VGH*18, veg, Tes]. Viz-Blocks builds on Vega-Lite by providing a lightweight UI, representing visualizations as reusable blocks and adding additional block types to allow document creation and composition. Voyager 2 [WMA*16, WQM*17],

DOI: 10.2312/evs.20191177





Name: Summary

London House Prices, 2015

The most expensive boroughs are in western inner London. The eastern boroughs of outer London are the cheapest:

There is a strong correlation between income and house price:

Note: the low house price for the City of London (relative to the very high income) is somewhat irrelevant since very few peple live there.

Figure 1: *a) Exploring clinical data. b) Summarizing football data: box-plots are based on user-selected years. c) Preparing a short report on London house prices: the sidebar and edit/add block buttons are visible when editing a document.*

also builds on Vega-Lite. It assists in targeted data exploration by generating multiple Vega-Lite charts and 'related views' based on a partial specification. While Viz-Blocks does not guide exploration, it does enable access to the full functionality of Vega-Lite in combination with other document elements whereas Voyager 2 only uses a subset of Vega-Lite's functionality and does not allow results to be customized.

The UI elements of Viz-Blocks draw on experience using visualization environments such as Tableau [STH02] and Power BI [Mic], which can be used to create interactive, multi-view visualizations without programming. These environments include a range of features and customization options, but are complex applications that can be intimidating to newcomers. Simpler template-based tools such as DataWrapper [Dat], Flourish [Kil], and InfoGram [Inf] are user-friendly and produce high quality charts. Flourish includes stories (slides); InfoGram includes a wide range of templates for slides and reports along with PowerPoint-like editing. These applications are well-suited to dissemination. However, their template-based approach limits the ease with which one can switch between or combine different representations of the data or create related multi-view visualizations (e.g., Vega-Lite's repeat operator), making them of limited use for data exploration.

In addition to the UI, Viz-Blocks incorporates approaches used by markup-based tools. Visdown [Kap16] allows users to write Vega-Lite specs in JSON/YAML [BKEdN05] inside Markdown. Visdown is an effective tool for authoring simple documents, but lacks the tools that Viz-Blocks introduces (such as a UI, code blocks and the ability to reuse and combine visualizations) to handle anything more complex. Idyll [CH18] uses Markdown for text and 'components' (included or custom) to add interactive elements. Idyll is purely markup-based and is used to create immersive 'interactive narratives' whereas Viz-Blocks is UI-based and aimed at efficient data exploration and document creation. Whereas Idyll components have their own markup language, Viz-Blocks uses plain JavaScript for bespoke content. The Litvis notebooks of Wood et al. [WKD19] combine text written in Markdown with Vega/Vega-Lite visualizations constructed in Elm [Cza12]. Their 'literate visualization' approach encourages users to justify the design process through a textual narrative aided by a 'narrative schema' that questions the design decisions. While Viz-Blocks is aimed at a more general audience, it can be seen as a lightweight alternative to Litvis: Viz-Blocks satisfies two of the three characteristics of literate visualization identified by Wood et al. and the reusable blocks approach of Viz-Blocks has some similarities to the branched design exposition of Litvis.

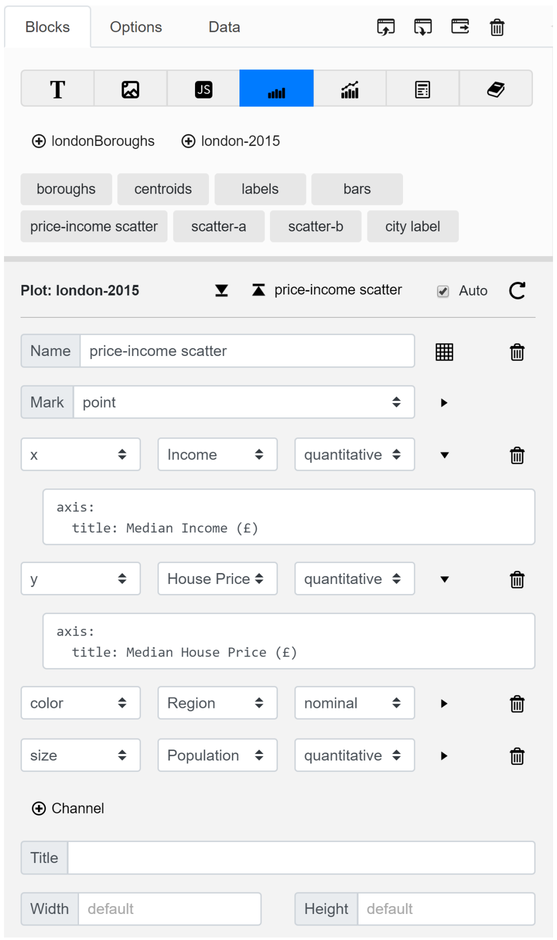
Finally, Viz-Blocks has some elements of notebook environments: 'blocks' can be thought of as cells in environments such as Observable [Bos], R Notebooks [RSt], and Jupyter [RPG*14]. The blocks can be assembled into a document just as the traditional notebook environments can create a document with Markdown and code output. (The code in notebook environments can often be collapsed/hidden leaving only the output while there is no option to display the 'code' in Viz-Blocks.) Viz-Blocks also differs from standard notebook approaches by introducing a dedicated UI for each type of cell (block). We believe that the result is an accessible, efficient platform that will appeal to both current notebook users and those who lack the programming skills to use traditional notebooks.

3. Design

Viz-Blocks was developed in an iterative manner with policy makers and computational social scientists. Our users wanted to quickly explore new datasets, document key insights informally to share with colleagues, and to 'polish' visualizations and text for inclusion in research articles, blog/social media posts, and presentation slides. We designed Viz-Blocks to be a lightweight application that caters to a wide range of users and visualization tasks. More concretely, we created the following **design objectives** specifying the features and behavior that the application needed to support:

O1 Exploration: create, modify and compare visualizations.
O2 Dissemination: customize/polish visualizations.
O3 Functionality: support as many common visualization types as possible as well as multi-view visualizations and interaction.
O4 Lightweight: easy to learn and accessible to those with little visualization experience and/or limited technical skills.
O5 Productivity: experienced users should be able to quickly and easily access the full range of functionality.
O6 Documents: create visualizations and entire documents.

We chose Vega-Lite [SMWH17] as the main visualization library for Viz-Blocks as it includes a wide range of visualization types (O3) and the sensible default options rarely need changed during



Plot: london-2015 price-income scatter Auto

Name: price-income scatter
Mark: point
x: Income, quantitative

```
axis:
  title: Median Income (£)
```

y: House Price, quantitative

```
axis:
  title: Median House Price (£)
```

color: Region, nominal
size: Population, quantitative
Title:
Width: default
Height: default

Figure 2: *Sidebar, Blocks tab → plot blocks (blue). The user can create or open plots (top). The price-income scatter plot is open in the editor (bottom, some options are not shown). The user has added YAML to overwrite the axis titles pulled from the dataset.*

exploration (O1). For dissemination (O2), visualizations are sufficiently customizable for most tasks/audiences. Users that require features not included in Vega-Lite (such as network diagrams) have the option of writing Vega [SRHH16] (the more expressive grammar that Vega-Lite is compiled to) in YAML or JSON. Furthermore, JavaScript code blocks (discussed below) enable the use of alternative plotting packages (e.g., Chart.js [cha] and Echarts [ech]) and lower-level libraries such as D3 [BOH11] and p5.js [McC]. Our first prototype was entirely UI-driven, but we struggled to keep the UI simple and intuitive (O4) while also enabling the full functionality of Vega-Lite (O3, O5). As a result, we developed a hybrid interface whereby many Vega-Lite options have dedicated UI elements, and others (such as interaction) may be specified in YAML or JSON (Figure 2).

Objectives O1 and O2 require that a user can create, combine, revisit, modify and customize a potentially large number of visualizations. Furthermore, they should be able to navigate between visualizations easily—for example, to modify an individual plot of a multi-view visualization. From a design perspective, we need to accommodate these requirements within a wider document structure (O6) without bloating the UI (O4). Our solution is to use the hierarchical 'building blocks' summarized in Figure 3. The block types are:

Text: written in Markdown and can include KaTeX [EA13] mathematical expressions and inline HTML.
Image: path/url and, optionally, width and height of image.
Code: JavaScript code automatically wrapped in a function. If this returns an HTML element, it is displayed.
Plot: a Vega-Lite 'unit'.
Composition: a Vega-Lite composition.
Page: comprised of sub-blocks that can be of any type except page and book. Sub-blocks are stacked vertically.
Book: comprised of pages. One page is shown at a time and controls to change the page are automatically displayed.

The same block can be used any number of times—e.g., as a stand-alone plot and as part of a multi-view visualization. Any changes to a block affect all blocks that use it, but a block can be easily duplicated when this behavior is not desired.

There are various technical requirements related to using Vega-Lite specs as reusable blocks. For example: restricting what compositions can contain; avoiding circular references; making the repeat operator behave as a true composition; not repeating inline datasets. To address these issues, we filter the sub-blocks made available to any given parent block and our blocks are 'pre-specs': JavaScript objects containing many standard spec properties as well as additional information that is used to create a valid Vega-Lite spec from the pre-spec whenever the visualization is rendered.

The inclusion of code blocks enables custom visualization and data analysis using any JavaScript library. Furthermore, code blocks can be used to add interaction (to their own output or other blocks) and to modify the document.

In keeping with our desire to create a lightweight, accessible application, Viz-Blocks runs in the browser (entirely client-side) and does not collect any information from the user. A Viz-Blocks *workspace* can be loaded or saved as a JSON file and any block can be exported as a stand-alone HTML file. From both Viz-Blocks and exported documents, users can export any Vega-Lite visualization as a PNG or SVG file.

4. User Interface

The Viz-Blocks user interface is comprised of a *sidebar* (Figure 1c, 2) and a *result panel* (Figure 1c). The *sidebar* has three tabs:

Data Data can be loaded from local or remote files or be inputted manually. The 'current dataset' is shown in the *result panel* along with a suggested name and default Vega-Lite types. GeoJSON/TopoJSON is automatically previewed and users can set options such as the default projection. Multiple datasets can be loaded; composition blocks may use several datasets.
Blocks New and existing blocks can be opened from the *Blocks tab*; the properties of the block are shown in the *editor* (Figure 1c, 2) and the block is rendered in the *result panel*.
Options The *Options tab* is used to set high-level options such as those discussed in Section 5.

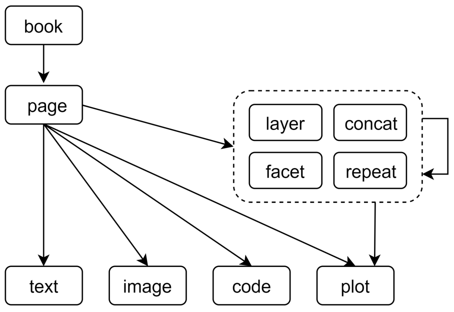


Figure 3: *Summary of block types and their relationships. The block types inside the dashed line are composite visualizations. Depending on the task, any block can be a stand-alone 'end result'.*

When a book, page or composition block is opened, its sub-blocks are shown in the *editor* (Figure 1c). From here, the user can take various actions such as to open, remove or add a sub-block, or change the order of sub-blocks. For pages and books, buttons for editing and adding sub-blocks are also shown in the *result panel*—the light gray buttons on the left in Figure 1c. These buttons are hidden when the *sidebar* is hidden to give a clear view of the document. By default, the *result panel* automatically updates when the open block is changed. The page number and scroll position of book, page and text blocks is tracked so that blocks are always opened at the expected positions.

5. Document Style

Documents have a consistent layout with sensible default styles. To achieve a consistent document appearance, image, code, plot and composition blocks share the same basic appearance and options: they are slightly narrower than text blocks and can be boxed, hidden (with a summary always visible), and can have a caption displayed above, below or on either side of the content (the scatter plot in Figure 1c has a side caption). Like text blocks, captions are written in Markdown/KaTeX.

High-level style options allow the user to make significant changes to the document's appearance. For example, there are simple UI elements to set the Vega-Lite theme, the document's maximum width, background color and text color, and various space and font settings for both text blocks and captions.

6. Discussion

Meeks [Mee18] recently noted that the lines between traditional categories of visualization software are blurring: "Notebooks are becoming more dashboard-like, dashboards are becoming more storytelling-like and in general there's a growing cross-pollination and convergence amongst media/modes." Viz-Blocks lies at the heart of this convergence, allowing a wide range of users to produce standard plots, interactive multi-view visualizations, text-based narratives and presentations with a minimalist piece of software. We are currently conducting user studies to assess how Viz-Blocks performs against our design objectives and to inform future development.

References

[BKEdN05] Ben-Kiki O., Evans C., döt Net I.: Yaml ain't markup language. *yaml. org, Tech. Rep* (2005), 23. URL: https://yaml.org/spec/cvs/spec.pdf. 2

[BOH11] Bostock M., Ogievetsky V., Heer J.: D^3 data-driven documents. *IEEE transactions on visualization and computer graphics 17*, 12 (2011), 2301–2309. 3

[Bos] Bostock M.: Observable. URL: https://beta.observablehq.com/. 3

[CH18] Conlen M., Heer J.: Idyll: A markup language for authoring and publishing interactive articles on the web. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 2018), UIST '18, ACM, pp. 977–989. URL: http://doi.acm.org/10.1145/3242587.3242600, doi:10.1145/3242587.3242600. 3

[cha] Chart.js. URL: https://www.chartjs.org/. 3

[Cza12] Czaplicki E.: Elm: A delightful language for reliable webapps, 2012. URL: https://elm-lang.org/. 3

[Dat] Datawrapper GmbH: Datawrapper. URL: https://www.datawrapper.de/. 2

[DiB90] DiBiase D.: Visualization in the earth sciences. *Earth and Mineral Sciences 59*, 2 (1990), 13–18. 1

[EA13] Eisenberg E., Alpert S.: KaTeX, 2013. URL: https://khan.github.io/KaTeX/. 4

[ech] Echarts. URL: https://ecomfe.github.io/echarts-doc/public/en/index.html. 3

[Gre] Greater London Authority: London datastore. URL: https://data.london.gov.uk/. 1

[Inf] Infogram: Infogram. URL: https://infogram.com/. 2

[Kag] Kaggle: Biomechanical features of orthopedic patients. URL: https://www.kaggle.com/uciml/biomechanical-features-of-orthopedic-patients. 1

[Kap16] Kapoor A.: Visdown, 2016. URL: https://visdown.com/. 2

[Kil] Kiln Enterprises Ltd: Flourish. URL: https://flourish.studio/. 2

[McC] McCarthy L.: p5.js. URL: https://p5js.org/. 3

[Mee18] Meeks E.: 3rd wave data visualization: Understanding the convergence of tools, audiences and modes, December 2018. URL: https://towardsdatascience.com/3rd-wave-data-visualization-824c5dc84967. 4

[Mic] Microsoft: Power BI. URL: https://powerbi.microsoft.com/en-us/. 2

[MWN*18] Moritz D., Wang C., Nelson G. L., Lin H., Smith A. M., Howe B., Heer J.: Formalizing visualization design knowledge as constraints: Actionable and extensible models in draco. *IEEE transactions on visualization and computer graphics* (2018). 1

[RPG*14] Ragan-Kelley M., Perez F., Granger B., Kluyver T., Ivanov P., Frederic J., Bussonnier M.: The Jupyter/IPython architecture: A unified view of computational research, from interactive exploration to communication and publication. *AGU Fall Meeting Abstracts* (Dec. 2014), H44D–07. 3

[RSt] RStudio: R notebooks. URL: https://rmarkdown.rstudio.com/index.html. 3

[SMWH17] Satyanarayan A., Moritz D., Wongsuphasawat K., Heer J.: Vega-lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (2017), 341–350. doi:10.1109/TVCG.2016.2599030. 1, 3

[SRHH16] Satyanarayan A., Russell R., Hoffswell J., Heer J.: Reactive vega: A streaming dataflow architecture for declarative interactive visualization. *IEEE transactions on visualization and computer graphics 22*, 1 (2016), 659–668. 3

[STH02] Stolte C., Tang D., Hanrahan P.: Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics 8*, 1 (2002), 52–65. doi:10.1109/2945.981851. 2

[Tes] Testard F.: Vegalite.jl: Julia bindings to vega-lite. URL: https://github.com/fredo-dedup/VegaLite.jl. 1

[veg] Vegas: The missing MatPlotLib for Scala + Spark. URL: https://github.com/vegas-viz/Vegas. 1

[VGH*18] VanderPlas J., Granger B. E., Heer J., Moritz D., Wongsuphasawat K., Lees E., Timofeev I., Welsh B., Sievert S.: Altair: Interactive statistical visualizations for Python. *Journal of open source software* (2018). 1

[WKD19] Wood J., Kachkaev A., Dykes J.: Design exposition with literate visualization. *IEEE Transactions on Visualization and Computer Graphics 25*, 1 (2019), 759–768. doi:10.1109/TVCG.2018.2864836. 1, 3

[WMA*16] Wongsuphasawat K., Moritz D., Anand A., Mackinlay J., Howe B., Heer J.: Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2016), 649–658. doi:10.1109/TVCG.2015.2467191. 1

[WQM*17] Wongsuphasawat K., Qu Z., Moritz D., Chang R., Ouk F., Anand A., Mackinlay J., Howe B., Heer J.: Voyager 2: Augmenting visual analysis with partial view specifications. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2017), CHI '17, ACM, pp. 2648–2659. URL: http://doi.acm.org/10.1145/3025453.3025768, doi:10.1145/3025453.3025768. 1