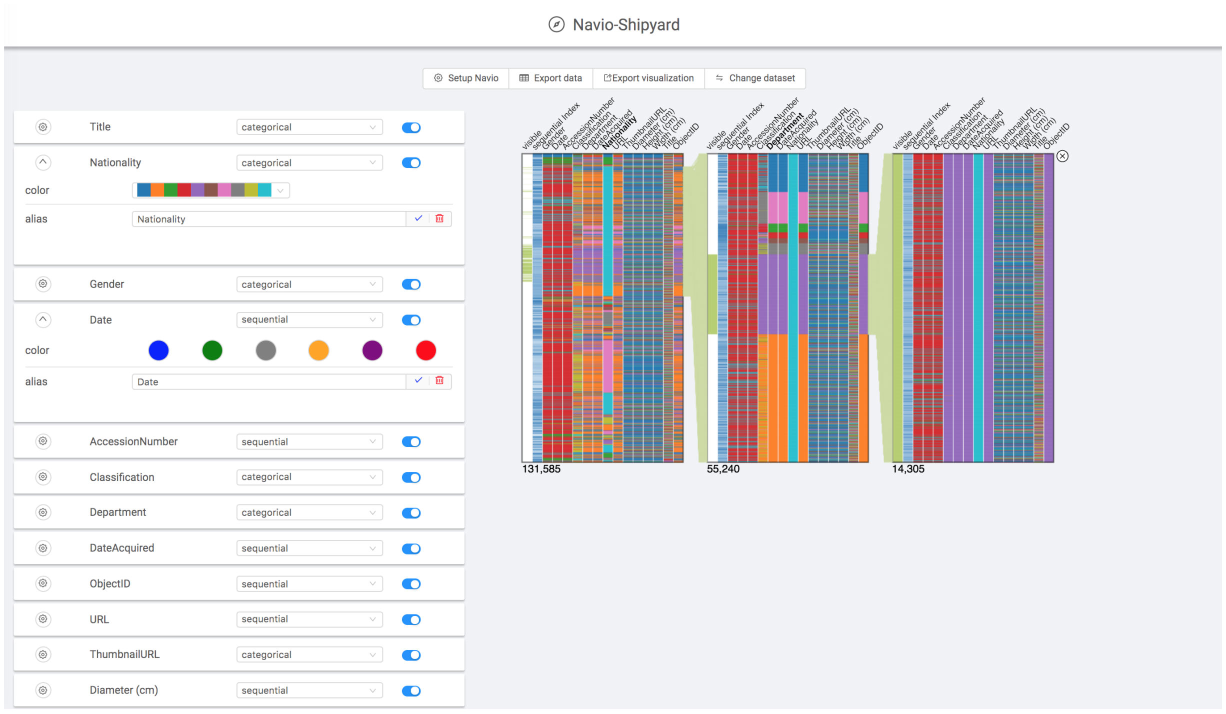
Shipyard: an application to ease the exploration, navigation and summarization of data

Juan Guillermo Murillo-Castillo[*]
Universidad de los Andes

John A Guerra-Gomez[†]
Universidad de los Andes
UC Berkeley



Title	categorical	
Nationality	categorical	
color		
alias	Nationality	
Gender	categorical	
Date	sequential	
color		
alias	Date	
AccessionNumber	sequential	
Classification	categorical	
Department	categorical	
DateAcquired	sequential	
ObjectID	sequential	
URL	sequential	
ThumbnailURL	categorical	
Diameter (cm)	sequential	

Figure 1: Shipyard's main interface after uploading MoMA collection [7]: the dataset was filtered by Nationality and Department dimensions. On the left side is displayed a configuration panel that allows the user to order, hide, show, set up an alias, select the color and the type for each of the dimensions in the dataset. Below this panel the user is able to visualize a table with all the filtered data as rows and the attributes as columns.

Abstract

Have you ever wondered how to explore, navigate and summarize large datasets in a faster and easier way? Data scientists could use different approaches to accomplish that like using programming languages to create a visualization or load the dataset into a visualization tool like Tableau, initially presented as Polaris [11], or Voyager [13]. The problem is that the tasks of explore, navigate and summarize large multidimensional datasets with a programming language or a visualization tool require visualization design expertise, time and knowledge in programming. To deal with this we present Shipyard: an open source web application that eases the tasks stated above. This web application works in the client-side and enables the user to upload their own data and explore, navigate, summarize, filter through nested queries, export the filtered data and export an interactive visualization supported by Navio [1]. Navio is a side visualization widget that eases the exploration, navigation and summarization of data. Shipyard has been evaluated with an usability study and a data scientist who used it to explore his own multidimensional data. The results of the evaluation showed that even though Shipyard behaves slow with more than 400MB of data and the navigation becomes difficult with 100+ dimensions it is a useful visualization tool that eases and facilitates the process of exploration, navigation and summarization of large datasets minimizing the visualization design expertise and time required.

Index Terms: User Interfaces—Visualization—Graphical user interfaces (GUI)—Human-centered computingVisualization systems and tools;

[*]e-mail: jg.murillo10@uniandes.edu.co
[†]e-mail: ja.guerrag@uniandes.edu.co

1 Introduction

Exploration, navigation and summarization of large datasets are not trivial tasks. These tasks are usually achieved within a visualization tool or visualizing by yourself programming in languages like Python, JavaScript and R . Between this two alternatives one could use a visualization tool decreasing the time required in the process or visualize the data with a programming language acquiring a custom visualization but increasing the time and knowledge required. In this paper, we are going to focus on the visualization tool side. Actually, there are different tools to address this and we are going to review three similar specialized solutions: Voyager [13], InfoZoom, initially presented as FOCUS [10], and Tableau, previously known as Polaris [11].

Firstly, Voyager [13] is a web application that allows the user to upload a dataset within a text plain file or choose one of the preloaded datasets. Under the hood Voyager uses CompassQL [12], a recommendation engine for visualizations. Its advantages are that the tool runs in the browser, suggests visualizations and allows users to upload different file types. On the other hand, its disadvantages are that because it is a web application running in the browser the size of data is limited, you need to setup everything in the interface to view something and the user can not export the embedded visualization.

Secondly, Infozoom [10] is a desktop application that has three different views. One of them is a view that shows the dimensions of your dataset as rows and the values of the dimensions as columns. Its advantages are that users can make nested queries and view the distribution of the data and its disadvantages are that requires some training and, at least in the on line version (demo), it does not allow the users to upload their own data.

Thirdly, Tableau [11] is a desktop and web application. Its advantages are that the user is able to connect directly to a database or upload a file and its disadvantages are that require training and visualization design expertise.

In conclusion, these three specialized tools are useful for the tasks but we found that only InfoZoom [10] allows the user to visualize an overview of the data right after the data is loaded. Given the advantages and disadvantages described above, we developed Shipyard, an open source web application. Its advantages are that after the user uploads the data an overview of the data is available and the user can make nested queries through intuitive interactions. Also, the user is able to filter and export the data and the visualization. On the contrary, falls in the same limitations as Voyager [13] related with the data size. Shipyards has different features like:

- Automatically detects the type of all the attributes of the data and allows the users to change it later if any of the attributes was classified wrong.
- Overview and filter through nested queries the dataset using Navio [1].
- Set up features of Navio [1] like the color, the order, the visibility and a custom alias for the dimensions.
- Visualize all filtered data into a table.
- Export the data filtered into a comma-separated values file.
- Export the stand alone interactive visualization widget.

On this paper we contribute:

- Shipyard: a visualization tool that allows the users to upload their own data to explore, navigate and summarize it. Besides, the users are able to export an skeleton code for the interactive visualization itself using Navio [1]. Available at `https://john-guerra.github.io/shipyard/`
- The results of the experiments: the usability study and the expert review.

2 Related Work

Nowadays, there are two approaches to the exploration, summarization and visualization of large datasets. The first approach is related to visualization tools like Voyager [13], InfoZoom [10] and Tableau (Polaris [11]). The second one is related with the visualization itself. In this approach are solutions like the Navio [1], Pixel Bar Chart [6], Value Bars [3], Parallels Coordinates [4] and Generalized Spirals. In this paper we take special attention to the first approach because it is the type of solution we are proposing, a visualization tool.

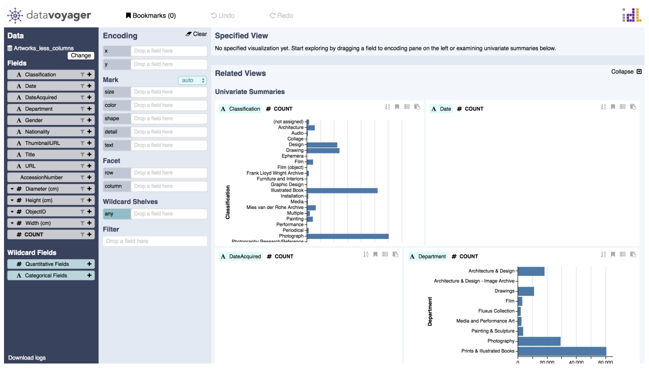


Figure 2: Voyager's main interface after uploading MoMA Collection.

On the first approach, there are three different visualization tools. First, Voyager [13] that uses CompassQL [12], a recommendation engine which enumerates, clusters and ranks visualizations according to both: data properties and perceptual principles. Second, InfoZoom [5] which is the more similar tool to our solution but with some differences that we will mention briefly. Third, Tableau [11] that is more complex to use, commercial and it is limited to relational data. Also, this visualization tool requires a previous knowledge in visualization design and the user have to setup which dimensions display and choose the visualization for those dimensions. Next, we are going deep into each of the tools.

2.1 Voyager

Voyager [13] is a visualization tool system developed by Vega and available at `https://vega.github.io/voyager/`. Its main interface is shown in figure 2 after uploading MoMA Collection, the first view is a set of recommended visualizations. This company also developed tools like Polestar (`https://vega.github.io/polestar/`), the predecessor of Voyager, and Lyra [8] (`https://idl.cs.washington.edu/projects/lyra/app/`): an interactive environment that enables custom visualization design. Polestar is no longer maintained by Vega but Voyager [13] supports all of its functionalities. According to their documentation and user interfaces the main difference between Voyager and Polestar is that Voyager supports CompassQL, the visualization recommendation engine, and two additional features: ”Facet” and ”Wildcard Shelves”. On the other hand, Lyra is based on a data pipeline, focused on designers and although it is not declared as deprecated yet, the last activity in its repository was two years ago. Main interface of voyager allows the users to upload their own data or a preloaded dataset. The users are able to upload just comma-separated value local files or JSON, CSV and TSV files from a URL. After that, Voyager uses CompassQL to fetch the recommended visualizations according to the data and on the left panel the user is able change the dataset, drag and drop different dimensions to encode the data into the two axis available and drag and drop dimensions in the mark zone, where the user could encode the data into different marks such as size, color, shape, detail and text. Even though there are some visualizations recommended by CompassQL, in order to use Voyager is required visualization design expertise because the visualizations recommended are not strictly useful to explore, navigate and summarize all your dataset. For instance, if one loads the MoMA [7] collection dataset, most of the recommendations are the different dimensions in one axis and the count on the other one. That shows a distribution of the values and depending on the number of null or invalid values in the dataset those visualizations provide the users an useful summarization of the data or not. In figure x we can visualize that, because the number of null or invalid values in the gender attribute are high, the visualization is not useful.

2.2 InfoZoom

Infozoom, initially presented as FOCUS [10], is a desktop and web application to analyze data. This tool is useful for summarizing and navigating data because compress all the data in the screen and allows the user to zoom in into a desired zone. Unfortunately, the on-line version just let the user visualize a preloaded dataset. This tool provide the user an useful overview of the data summarizing it in all the screen. In addition to the overview view, InfoZoom also provides the compressed and table view. The users are able to click into one or more values of one dimension and zoom in. If the dimension is not categorical, the users are able to drag and drop the values of interest in the overview view or apply a filter in the desired dimension clicking on the filter button to increase the precision. Moreover, InfoZoom allows the users to make nested queries and generate a pie chart visualization with the distribution of the different dimensions.

2.3 Tableau

Tableau, initially presented as Polaris [11], is one of the most popular visualization tool. With this tool the users are able to upload a file or connect directly to a database. Tableau does not limit the users to visualize pie charts and bars, but also tree-maps, scatter plots, maps and so on. Besides, the users define their own metrics and create dashboards with different visualizations, text and styles. The main problem with Tableau is that if you do not have a quick set up. When a dataset is loaded the user is faced with a blank canvas where is required visualization design expertise.

On the other approach, related with the visualization itself, we have Navio [1], Pixel Bar Chart [6], Value Bars [3], Parallels Coordinates [4] and Generalized Spirals. The visualizations were compared in the table 1 with different characteristics: the capacity of provide a useful overview of the whole dataset, the capacity of show many attributes' distribution overviews at the same time in the same view, the ability to sort to identify attributes of interest, the capacity of handling multi-dimensional data, the capacity of avoid overlapping, enable exploration, maintain context of filtered items, chain filtering and limit of dimensions. Each one of them have its own advantages and disadvantages. Because of the advantages represented according to the chart, the visualization tool will be developed over the Navio [1] widget.

3 SHIPYARD

Shipyard is a web application built with ReactJS and Redux that allows the user to upload, process, explore, summarize and visualize a dataset using Navio without writing a single line of code. In order to use the application, the user must upload a dataset; the dataset file has to be semicolon-separated values, comma-separated values or tab-separated values. After that, the Navio widget, its configuration panel and an useful reactive table with a sample of the data will be displayed as shown on figure **??**. In the configuration panel, the user is able to hide or show attributes and change their type (categorical or ordinal). Shipyard is not just an configurable interface for the Navio because the user can export the filtered data into a comma-separated values file and the embedded visualization. Due to the architecture of Shipyard, that is just front-end and is handling all the data in the client side, this application has data size limitations and in short term we expect to develop a scalable back-end that implements elastic search to overcome this limitation. Shipyard is also an open source project and is available at `https://john-guerra.github.io/shipyard/`.

3.1 Functionalities

Shipyard allows the users to navigate, explore and summarize their data into a faster and easier way. The first thing the a user needs to do in order to use Shipyard is upload a dataset. Currently, Shipyard

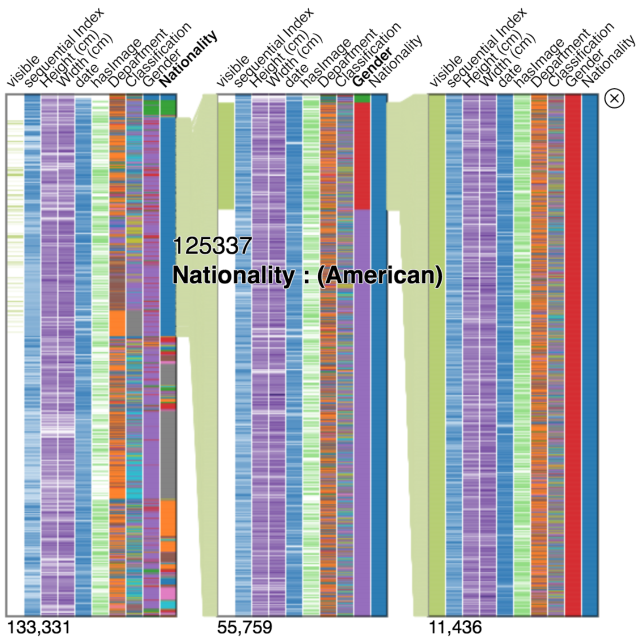


Figure 3: MoMA Collection explored with Navio. Filtering by American Nationality and Female Gender.

supports three different file types: comma-separated values, tab-separated values and txt. The data is loaded in the client side, hence the data, even if it is sensitive, has no risk of being leaked. After that, the first view is not an empty canvas as in Tableu [11] or Voyager [13]. This view is supported by Navio: a side visualization widget. In addition to this functionality, Shipyard provides a side panel where the users are able to configure the visualization. In the visualization panel the users are able to sort different dimensions and filter through nested queries a selected value or a selection. Furthermore, the data that the users filter are displayed in a table and the users have the possibility to export it into a comma-separated value file. Apart from export the data, Shipyard allows the users to export the visualization.

3.1.1 Upload data

Actually, Shipyard supports CSV, TSV and TXT file types. Under the hood, Shipyards uses Vega [9] and D3 [2] to read the files. Owing to the fact that Shipyard is just a client side web application, the size of the input file is limited by the capacity of the browser and the computer. For the load test was used a Mac Book Pro Mid 2015 with 16 GB 1600 MHz DDR3 of memory and processor 2.2 GHz Intel Core i7 running MacOS 10.13 and Google Chrome 66. Under this restrictions Shipyard loaded without crashing approximately 400MB of data. It should be mentioned that with the limit size most of the queries become slower.

3.1.2 Navigate, explore and summarize

For this functionality, Shipyard uses Navio [1]: a side visualization widget developed by Guerra-Gómez, initially to explore networks but extended to multidimensional data. This widget displays the dataset as shown in figure 3, where the dimensions are visualized as columns and each pixel in the y axis represents a sample of the data. This widget have three different ways in which the users interact with the dataset. First, the users can hover over the widget to get detailed information, the value of the record in that column. Second, users are able to sort the dataset by their attributes clicking on the name of the dimension. Third, the users can click and drag anywhere in the bar to

Characteristic \ Visualization	Pixel Bar Chart	Value Bars	Par. Coords	Gen. Spirals	Navio
Provides a useful overview of the whole dataset	X	X		X	X
Many attributes' distribution overviews		X	X		X
Ability to sort to identify attributes of interest	X		X		X
Handle Multidimensional data			X	X	X
Avoids overlaps	X	X		X	X
Enables exploration		X	X		X
Maintains context of filtered items			X		X
Nested filtering			X		X
Limit of dimensions		X			

Table 1: State of the art comparison

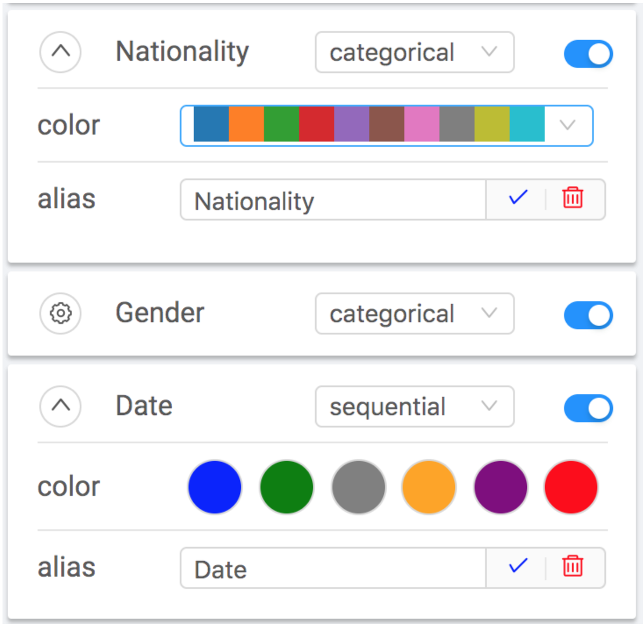

Nationality — categorical
color
alias: Nationality
Gender — categorical
Date — sequential
color
alias: Date

Figure 4: Shipyard's sidebar with three attributes. The arrow button allows the user to expand the configuration and show the color and alias options. Next to the name the user can click and change the type of the attribute and in the toggle button the user is able to show or hide the attribute.

filter by a specific value or a range of values. In this last interaction a new instance of the widget is generated displaying the filtered data. Using these three interactions the users can accomplish the tasks mentioned earlier: the navigation, exploration and summarization of data.

3.1.3 Preview of the data

As the user filters their data through complex and nested queries a table in the bottom of the application is shown with the results of the filters. This feature allows the user to inspect all the data in detail. That is required, because within the widget is displayed just a sample of 600 or less records and the user may require review all the data filtered from a custom query. This table uses pagination and no matter the number of dimensions, all of them are displayed.

3.1.4 Set up visualization

This feature is one of the more useful of Shipyard. Due to this functionality users do not require visualization design expertise or programming knowledge to use Navio, hence they do not require it to navigate, explore and summarize their own data. On the left side panel are displayed all the attributes of the data as shown in figure 4. When the data is loaded, Shipyard automatically detects the type of the data but due to null or invalid values the detection could be wrong. In this case, Shipyard provides an intuitive way to reassign the type of an attribute. In addition, each of the attributes can be drag and drop to reorder the way the attributes are being displayed in the widget. This is useful because most of the times, there are attributes that are related or make more sense to stand side by side with other attributes. For instance, in the MoMA [7] collection, users found useful to drag the attribute width and drop it next to the height attribute. Moreover, users can hide or show any of the attributes, that is useful if the users are interested in some specific attributes. Finally, users can change the color that is mapping the attribute in the widget and assign an alias to make the title of the attribute more understandable.

3.1.5 Export data

Another important feature in Shipyard is the option to export the filtered data. Initially this data is all the dataset and as the user filters the data it is updated. When the user clicks the export data button, a comma-separated values file is generated and downloaded in the computer that is running Shipyard. The file can be used to used it later with Shipyard or whatever the user wants to.

3.1.6 Export visualization

Finally, the export visualization feature enables the users to use the set up widget within a desired web page or use the stand alone widget. The exported version supports all the interactions of Navio [1]. This functionality downloads two files: one named data.csv with all the filtered data and an index.html file with the required script in order to make the side visualization widget work. With this feature, users are able to embed the visualization widget, although it may require some basic knowledge in HTML but not necessarily programming.

4 Evaluation

4.1 Methodology

The methodology used to evaluate Shipyard is divided in two. The first one is an usability study with two users. The second one is an exploratory study with a data scientist exploring his own dataset.

4.2 Usability study: Dataset

In the usability study a set of tasks were given to the user. The tasks were designed to evaluate the main features of Shipyard. Those tasks were:

- Upload a given dataset. In this case were used the MoMA Collection dataset within a comma-separated values format file.
- Assign an alias to an specific attribute.
- Change the color of one attribute.
- Drag the width attribute and drop to the side of the height attribute.

- Filter the datasets given three different queries.
- Export the filtered data of each of the queries.
- Export the visualization of each of the queries.
- Run the stand alone visualization.

For each of the tasks we measured the time and took notes of the users' feedback.

4.3 Exploratory study

In the exploratory study the user explored his own data. A dataset with information of taxpayers Colombian companies based in Bogot D.C. The number of attributes of the dataset were about one hundred (100). This made the use of the tool more complex and the fact that the file size was in the limit (400MB) implied that the queries were slow. Although, the data scientist found three important insights with the tool.

4.4 Results

The results were useful in order to understand the limitations and the future work of Shipyard. On the one hand the usability study evidenced that although the implementation of the drag and drop was based within usability guidelines, for one of the users was not so obvious that the attribute could be dragged and dropped on the left side panel. Also, the users found useful the tooltips in each of the main buttons that explained the functionality of all of them. Moreover, the users at the beginning took more time to understand how the widget work. But once they trained they feel more comfortable using Shipyard than voyager.

On the other hand, the exploratory study revealed some disadvantages of Shipyard. The complexity increases if the number of attributes were more than one hundred. Also, with the file size almost in the limit (400MB), the response of the application was very slow, about x seconds per query. Although, the data scientist found useful the application because provides an excellent overview of the data and the nested queries.

5 Limitations

Most of the limitations found in Shipyard are related with the browser and the fact that is just a client side application. Shipyard does not integrate a robust back-end with services which limits the amount of data that can be loaded. Other limitation is that actually, Shipyard just read three file types and users would find useful that they could navigate, explore and summarize a broad file types or even a custom one defining the separator. Moreover, users suggest that would be really useful to connect Shipyard directly to a database, saving time exporting the data into a readable file for Shipyard. Also, the export file functionality should provide different file types and not just CSV file type.

6 Future work

In order to reduce the limitations of Shipyard, we are working on the server-side of the application. That maybe increases the loading time but will decrease significantly the time executing a query. This back-end leverage the load in the front-end and it is planned to use elastic search. Another features, that as stated in the previous sections would be useful for the users, are modify the export functionality in order to allow the user several file type or custom file types. In addition, connecting directly to a database would be a nice to have to the users.

7 Conclusion

We presented Shipyard, an open source web application that reduces the time and the knowledge required in programming and visualization design to explore, navigate and summarize large multi-dimensional datasets. We evaluated Shipyard with an usability and exploratory study. From those studies we conclude that even though Shipyard has some limitations related with the size of data supported and the response time with a dataset near the limit of 400MB is an useful tool to find insights. Also, as shown in the studies this application eases the navigation, exploration and summarization of data and makes the process of exploring data faster and easier.

References

[1] J. alexis Guerra-Gómez, J. G. Murillo-Castillo, J. C. Ortiz-Román, and T. Hernández. Navio: a visualization widget for summarizing, exploring and navigating large multivariate datasets. 2018.

[2] M. Bostock, V. Ogievetsky, and J. Heer. D 3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 2011. doi: 10.1109/TVCG.2011.185

[3] R. Chimera. Value Bars: an information visualization and navigation tool for multi-attribute listings and tables. 000:0–1, 1998.

[4] A. Inselberg and B. Dimsdale. Parallel Coordinates: A Tool for Visualizing Multi-dimensional Geometry. In *Proceedings of the 1st Conference on Visualization '90*, VIS '90, pp. 361–378. IEEE Computer Society Press, Los Alamitos, CA, USA, 1990.

[5] D. A. Keim. Designing pixel-oriented visualization techniques: theory and applications. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):59–78, 2000. doi: 10.1109/2945.841121

[6] D. A. Keim, M. C. Hao, U. Dayal, and M. Hsu. Pixel bar charts: a visualization technique for very large multi-attribute data sets. doi: 10.1057/palgrave/ivs/9500003

[7] Museum of Modern Art. The MoMA Collection, 2015.

[8] A. Satyanarayan and J. Heer. Lyra: An interactive visualization design environment. *Computer Graphics Forum*, 2014. doi: 10.1111/cgf.12391

[9] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-Lite: A Grammar of Interactive Graphics. *IEEE Transactions on Visualization and Computer Graphics*, 2017. doi: 10.1109/TVCG.2016.2599030

[10] M. Spenke, C. Beilken, and T. Berlage. FOCUS: The interactive table for product comparison and selection. *Proceedings of the 9th annual ACM symposium on User interface software and technology - UIST '96*, pp. 41–50, 1996. doi: 10.1145/237091.237097

[11] C. Stolte, D. Tang, and P. Hanrahan. Polaris: a system for query, analysis, and visualization of\nmultidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–14, 2002. doi: 10.1109/2945.981851

[12] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Towards a general-purpose query language for visualization recommendation. *Proceedings of the Workshop on Human-In-the-Loop Data Analytics - HILDA '16*, pp. 1–6, 2016. doi: 10.1145/2939502.2939506

[13] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Voyager: Exploratory Analysis via Faceted Browsing of Visualization Recommendations. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):649–658, 2016. doi: 10.1109/TVCG.2015.2467191