# Histogram Design

Pavares Charoenchaipiyakul
01468869

Milán Kolki
01301925

Lachezar Valkov
01308067

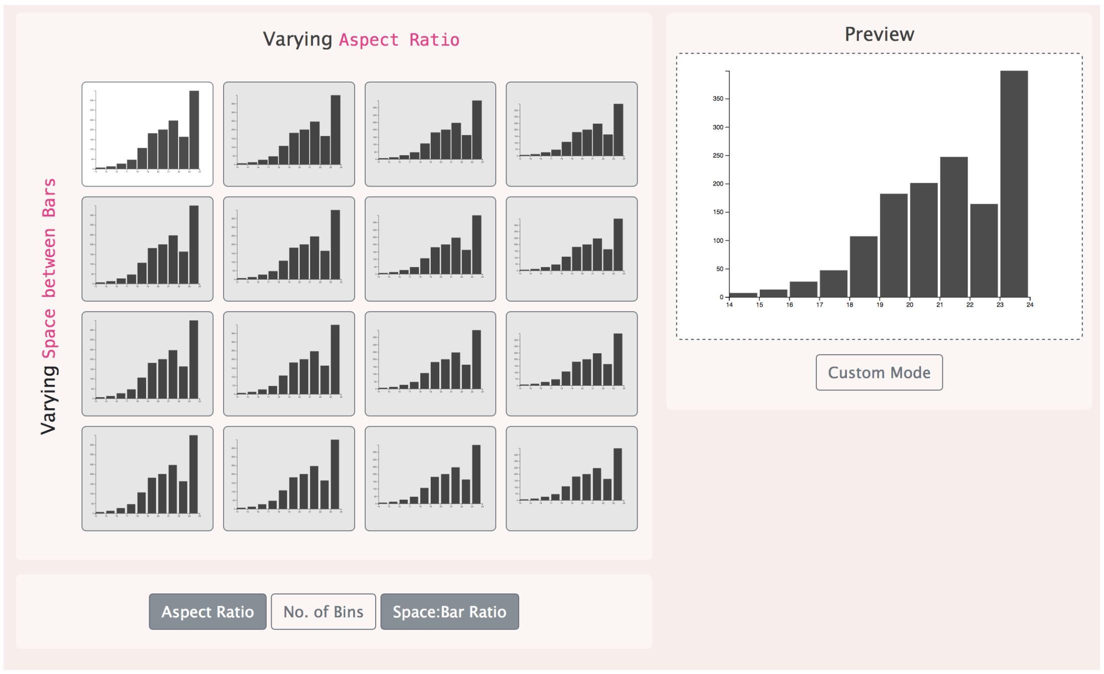


Figure 1: The final version of our tool.

**Abstract**

Histograms are often used as a method to gain a quick overview of a dataset. There are several parameters that come into play when creating a histogram, and it's often a problem to select the best combination for our purposes. We propose a tool that provides the user with a quick overview of these parameters, allowing them to select the best histogram for their own purposes. We present a design study of this tool, the progression of our prototypes, arriving to the final version. The implementattion details are also discussed. We describe a typical usage scenario, complete with a persona. We discuss the strengths and the weaknesses of our tool, concluding with the lessons we learned from this project.

**Index Terms:** Human-centered computing—Visualization—Visualization systems and tools—; Human-centered computing—Visualization—Visualization design and evaluation methods

## 1 Introduction

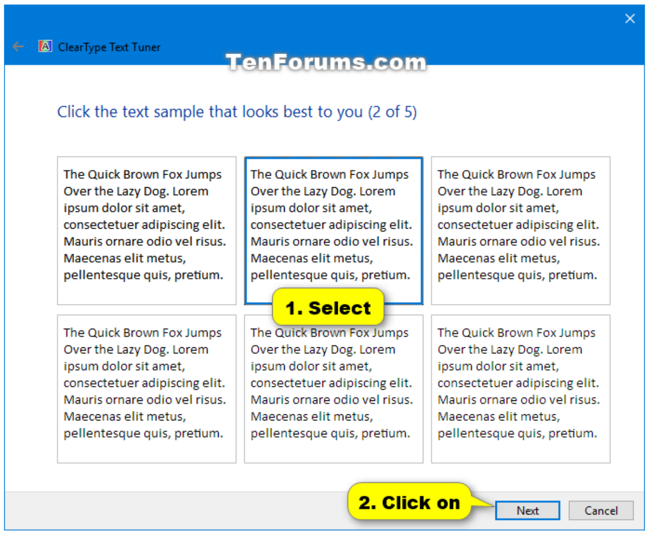
This paper was created for the University of Vienna's *VU Visualisation and Visual Data Analysis* class, taught by Torsten Möller, Thomas Torsney-Weir and Michael Sedlmair, in the winter semester of 2017-18. We are Group 8.

## 2 Motivation

According to the project description [2], "histograms are often used as the first method to gain a quick overview over the statistical distribution of a collection of values, such as the pixel intensities in an image". There are various parameters to consider, such as *bin width*, *aspect ratio*, *visual encoding*, etc. All these parameters have an influence on how the histogram is perceived. We decided to do a design study project on an application that helps the user to create the best histogram for their own purposes, where "best" is entirely dependent on the user's perspective. Taking this into consideration, we hope it will be used to create a good representation of the data, but if someone wants to intentionally create a skewed representation, that is also possible.

## 3 Related Work

Looking through the old projects listed on the course website, no one has chosen this project topic before us. There are also no commercial tools available for histogram design as far as we know, all there is are some formulas to generate an optimal bin width with. We looked for other inspiration, and so our high-fidelity prototype for Milestone 3 was inspired by a Windows utility (Fig. 2) for selecting the best font rendering for your screen. The literature provided in the project description was unfortunately very advanced and had almost nothing to do with visualization (since the project was originally supposed to be a bachelor's thesis and was adapted for this class). Fortunately, for Milestone 3 we received some additional literature which helped with understanding what the essence of this class is and the mindset we had to adapt for it. The Design Galleries paper [3], describing an automated system aiding the user in selecting perceptually different graphics and images, has influenced our final version significantly.


ClearType Text Tuner

TenForums.com

Click the text sample that looks best to you (2 of 5)

The Quick Brown Fox Jumps Over the Lazy Dog. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Mauris ornare odio vel risus. Maecenas elit metus, pellentesque quis, pretium.

1. Select

2. Click on

Next

Cancel

Figure 2: The ClearType utility to select the best font rendering on Windows. [1]

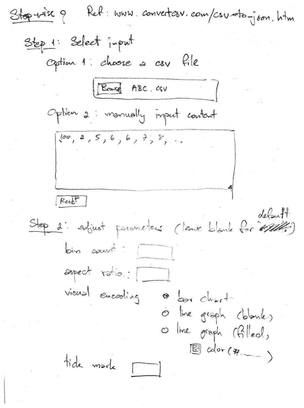


Figure 3: Pavares' low-fidelity prototype.

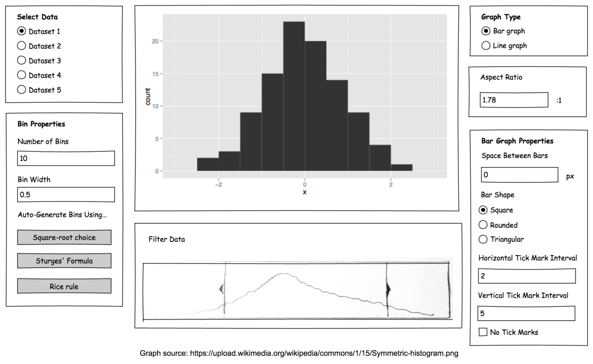


Figure 4: Milán's low-fidelity prototype that we selected for implementation.

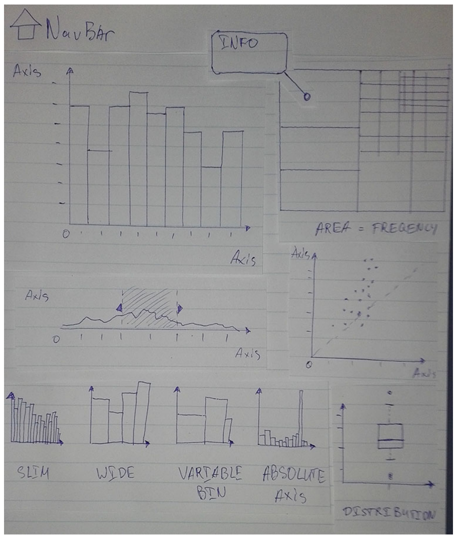


Figure 5: Lachezar's low-fidelity prototype.

## 4 APPROACH

In the beginning, we considered an evaluation project on the visual perception of histograms. At the time, we thought we had to follow the project description fairly closely, so we thought of approaching this by creating a tool that let us create all kinds of histograms, and then running a user study on the perception of different histograms. We discussed this idea with Tom, and he encouraged us to do a design study instead, arguing that a study like this would take too much effort. So we ended up scrapping the idea of an evaluation project, and started working on a design study for a histogram design tool.

Each of us created a low-fidelity prototype for an interface to create histograms with. Pavares' prototype (Fig. 3) was basically a wizard allowing the user to set the properties for a histogram sequentially, Lachezar's prototype (Fig. 5) was a complicated design with floating panes and additional graphs to help understand the data better, and Milán's was an interface where the user could input the parameters and see the histogram in the middle change right away based on these parameters. With the help of the feedback we received for Milestone 2, we chose Milán's prototype that lets the user tweak the details of the histogram on a single screen (Fig. 4), but we added sliders and scented widgets to make it a bit more intuitive.

To incorporate further visualization techniques, and to allow the user to get an idea of their decisions, we designed an interface we dubbed "ClearType", which was inspired by the Windows utility mentioned earlier. This helped the user select the properties for the histogram visually, through a logical sequence of the parameters, seeing how the histogram would change with every step, starting with selecting the visual encoding (bar/line chart), selecting the bin width, the aspect ratio, and finally, the bar width(for bar graphs only. (Fig. 6) Then, at the end, the user could reach a screen where they could tweak the parameters some more, using sliders, scented widgets and tooltips. (Fig. 7) From a visualization perspective, however, this proved to be too linear. Prof. Möller said at the Milestone 3 presentation that he imagined this *"sea of histograms"* that he would be able to swim through, seeing all the possible combinations of parameters at any given moment.

Unfortunately, we fell short of this goal. This comment about "swimming" got stuck with us and we imagined a system that would control the histogram with motion only. This wouldn't be entirely infeasible. We imagined a system, controlled with the WASD keys and the arrow keys that would allow the user to "morph" a histogram according to their will, being able to change the histogram type with simple, intuitive motions. Drawbacks of this approach include the user having to sort of memorize which key changes what, which

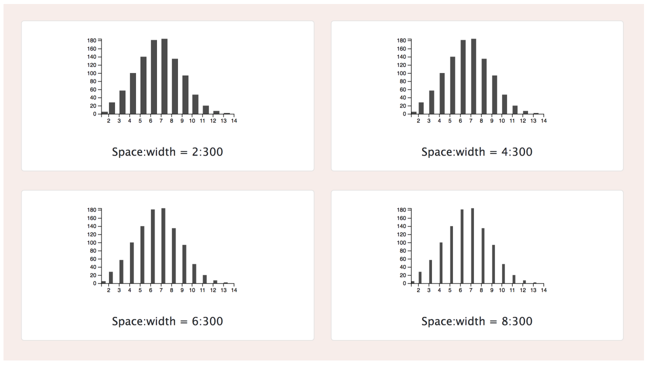


Figure 6: The step to select bar width in our high-fidelity prototype.

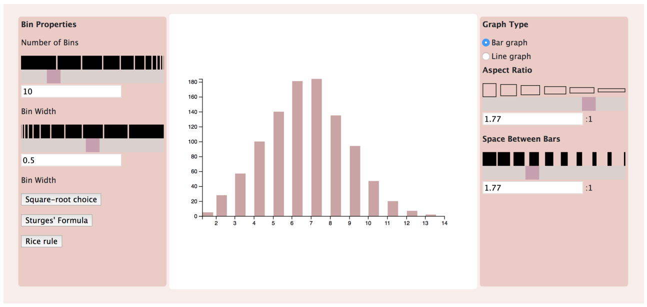


Figure 7: The screen to tweak the histogram in our high-fidelity prototype.

would take a bit of time for the process to be intuitive. Because (as far as we know) no one has to create histograms for a living, this is bit of an "overkill" approach. Considering this and other implementation challenges such as morphing the histogram smoothly, we decided to do something simpler.

We wanted to make the effect of changing multiple parameters visible, so we made a two-dimensional, four-by-four grid. The initial idea was to display 16 pre-defined histograms first(after uploading the dataset), allowing the user to select one they want to start with. However, Pavares argued that the user knows already if they'd like a bar chart or a line graph, so we should get that choice out of the way first and present them with other options only afterwards. So by clicking to select the graph type, the user reaches the main view, with the grid on the left and a detailed view of the histogram on the right. Under the grid, the user can toggle the parameters the grid is currently displaying. Above and to the left of the grid, the parameter corresponding to the respective axis is indicated.

We retained the "tweak view" from Milestone 3, which provides a way to tweak the parameters more accurately. It's possible to access this via the 'Custom Mode' button under the detailed view.

## 5 Implementation

- Language: Node.js with EMCA Script 2015 (ES6) [4]
- Main Development Tools: d3, d3-node, Express, jQuery, csv-parse, handlebars, formidable
- HTML Framework: Bootstrap v4

The idea of the implementation is quite simple. We let the user upload the dataset (as .CSV format) to the server, then the workflow is separated into 2 parts; the server-side and the client-side, whose duties are different.

### 5.1 Server Side

The server side of the application goes around *Node.JS*. As stated above, we use Node.JS in form of ECMA Script 2015, which allows the usage of node modules and class concept, however, the process overall is in Restless API architectural pattern. To dive deeper in further information, it is easier to clarify that the server side has 2 main functionalities; routing and static rendering.

Routing is simply to program and create the route paths of the site. It is analogous to the map that shows which URLs lead to which pages. This includes page flow that determines how the actions in the web application are performed and in which order. To implement this, we use *Express* module. The module is an open-source framework under MIT license that allows web-server services, including pathing and routing configurations. This is also how we develop the GET and POST actions for our application in each page to suit our needs.

The second functionality is the static rendering of the histograms. We can render the SVG elements statically with the help of *d3-node* module. The module functions similar to how the *d3.js* does. Hence, the implementation of this part resembles the implementation of the normal d3 in client-side JavaScript, except that there is no use of JavaScript DOM in d3-node module. We use the module to create strings of SVG tag identifying the histograms in question. By sending these strings to the frontend, with the help of handlebars, the SVG Tags in question are rendered to the outside world.

### 5.2 Client Side

The client side, on the other hand, doesnt involve such thing as routing. It is purely about the frontend, about what and how it will show on the screen. We use Bootstrap v4 to ease up the work and provide tidy, systematic view. Bootstraps grid is used in this implementation.

Another main section for the client-side implementation is the dynamic rendering of the histograms. The reason is that the custom mode grants the users the ability to modify their histograms in real time. Hence, it is reasonable to use (client-side) *d3.js* for such dynamic task, as it uses clients JavaScripts to manipulate the outlook without messing with GET and POST methods. The script takes variables, graphically set with HTML form elements, as parameters for real-time rendition.

The process can be realized in this stage because client-side JavaScript is able to operate the Document Object Model (DOM) elements on the page. With this, the script learns the values of each parameter on the page without submitting to the server, therefore, it can generate a histogram based on those values and assign to the specified ¡div¿. In this process, we also make use of the onclick and onchange options. Thus, in the end, the histogram here should be ready for fine adjustment by the users.

### 5.3 Challenges

We struggled with scaling down the svg images for the 4x4 view. Sometimes the bar width ended up being too small and the bars just disappeared. Fortunately we were able to solve this problem. This solution has brought another problem with itself, the app could now only run in Firefox and Microsoft Edge. This problem was also fixed.

Our third team member, Lachezar Valkov, was tasked with implementing the filter (focus-and context, see later) in the Custom Mode (see prototype in Fig. 4). He failed to do this before the deadline.

## 6 Results

### 6.1 Scenario of use

For our usage scenario we envisioned a persona:

- **Name:** Melanie Kastner

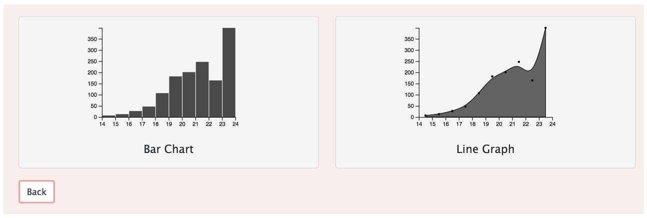


Figure 8: The screen to select the chart type in the final version.

- **Residence:** Vienna
- **Age:** 29
- **Profession:** Journalist
- **Motto:** If all is under control, you're not going fast enough.
- **Experiences:**
  - Lives with her fiancé
  - Is rather overloaded with tasks at work
  - Has little to no free time left to spare
  - Work is her highest priority
  - Desires to accomplish work goals faster, in order to be able to better manage the time spent with family
- **Scenario:** Uses the Histogram Exploration Tool to gain deeper insight into collected data, be able to process it with greater efficiency and ultimately finish her articles faster.
- **Goal:** Find the most impactful histogram representation to use in an article about youth smoking

We imagined a scenario of a journalist writing about youth smoking. She'd like to have a histogram in her article, to show some statistics. She opens up our tool to upload her dataset, 'youth-smoking.csv'. She wants a bar chart in her article, so she selects that. (Fig. 8) She'd like a narrow histogram with wide bars so that the slight differences between the bars are more visible. She selects 'Aspect Ratio' and 'Space:Bar Ratio' at the bottom, so that she has an overview of how these two parameters interact. (Fig. 9) Finally, she selects the histogram that fits her needs best. (Fig. 1) Here she has an opportunity to go to 'Custom Mode' which is the histogram-tweaking interface we designed for Milestone 3 (Fig. 7).

## 6.2 Performance

### 6.2.1 Technical Performance

To discuss about technical performance, we mainly consider the speed of execution and the stability of the application. First of all, we would like to begin with the stability issue. Stability of this application, as it is now, is fairly decent in the programming point of view as we used defensive programming to some degree to maintain and make sure that the application runs smoothly through. The uni-directional routing of the website is designed to control the flow of data not only so that the application is simple to use, but also that the we gain all the data needed for histograms rendering without failure. However, to some extent, the application cannot maintain its composure, for example, if a too large scale of data (which surpasses language or browser capacity) is loaded into the program. Moreover, because the current version of the application is still considered a (high-fidelity) prototype, it handles most of the error by throwing exceptions. This means that the program still uses the Garbage in, nothing out or Garbage in, error state out concept,

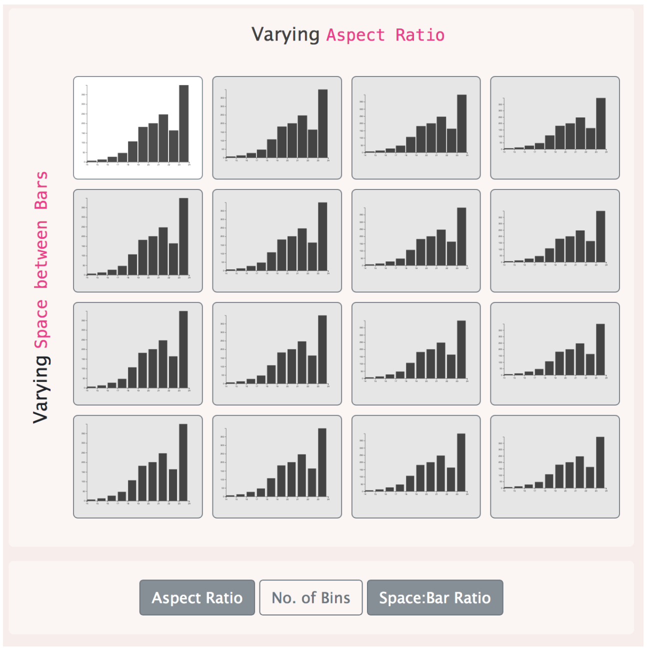


Figure 9: The screen to select the best histogram from a combination of two parameters in the final version.

which compromises its robustness (but the robustness should be more concerned in the deployed version).

Second point is about the speed of execution. In this aspect, we do not intend to measure and show how fast the program could render one histogram. Doing so would bring nothing as there are a lot of variables to concern and this is not the focus of our project. Nevertheless, we still need to mention that using *Node.JS* helps the processing and rendering time to be faster and more consistent. Let us consider the choices; *d3.js* with pure *JavaScript* and *Node.JS*. With *d3.js*, the application would be a normal client-side web application which uses clients resources and the performance would be based on both hardware of the client and software of the client. This is because *pure JavaScript* is executed by the software such as web browser, hence the performance is limited, implying that some of the users would take the problem that their hardware and software suffer their experiences using the application. *Node.JS*, on the other hand, does most calculations in the server-side and not relying on the web browser. This guarantees speed of execution in some certain degree that users with lower computer specification also receive decent services.

Not only the speed, but the memory and networking issue is also concerned and optimized by the choice of using Node. Pre-rendering allows offloading data processing. Unlike using pure JavaScript, Node allows us to put the input file into the server and directly acquire it when needed. This promises faster reaching speed and saves memory used in communication between server and hosts. In normal d3.js, there are issue of large network overhead when the program deals with large tables and the data is sent back and forth in order to produce results. The issue is solved simply by direct loading from the server itself as stated above. In conclusion, the application seems to have a decent performance up to some points but still leave a room for improvements.

### 6.2.2 Usage Performance

Usage performance or the quality of the program is somewhat ambiguous. It is the quality of the application seen from the users point of view. The application is still young and under development but it is usable for tasks, and it shows potential in the use case we came up with, therefore, the first and foremost quality contained in the program is that it runs and works fine under normal circumstances. Another usage quality worth mentioning is that the program is not difficult to learn (by ones self and without instruction manual), as it is partially step-wise and intuitive. The users are free to choose the plain and simple ways by using presets, or to encounter the sea of histograms (with some aid so that the users do not suffocate from too much histograms at once), or to fine-tune the histogram themselves making a custom-made histogram just from their tasks.

## 7 Discussion

### 7.1 Strengths and weaknesses

Our tool is intuitive, easy and quick to use, and has a light backend. It provides the user with the advantage of being able to see the effect of different parameters at once. Unfortunately, we only have two dimensions, so this is kind of limited. It would theoretically be possible to use three dimensions, or a hierarchy (like in the Design Galleries paper [3]) of different parameters. It is also limited to a handful of parameters, although that doesn't diminish the merits of our design. There's also the challenge of how it handles different kinds of data, because the presets aren't suitable for datasets with too many or too few values.

#### 7.1.1 Possible improvements

There are a couple of things that we couldn't implement in the timeframe of this class. The first is a different graph, right under the detailed histogram view (visible in our low-fidelity prototype, Fig. 4) which could be used to restrict the data and filter low and high values from the histogram. This would constitute an example of focus-and-context. This was supposed to be done by Lachezar, but he didn't contact us before the deadline, so this functionality is missing. Another improvement could be to upload multiple datasets and change between them "on the fly", showing how a certain set of parameters affects different datasets.

The set of available parameters to change could also be expanded, for example, the tick marks, the bar shapes [5], etc. However, this could cause some confusion with the toggle buttons, so maybe some icons are necessary to make it more intuitive.

### 7.2 Lessons learned

The most important lesson we learned from this class was that it's the idea that matters. No matter how nicely implemented a tool is, if the theory behind it isn't solid, it's useless. One shouldn't underestimate the effort needed for this class, however, you still have to have the skills to implement your tool, that's just not the point. It's also very important to plan ahead and carefully weigh what you can achieve within the short timeframe granted to you. Overall, what you get from this class is directly proportional to the effort you put in it.

Somehow we didn't understand in the beginning what the goal of this class is, and that it was possible to do whatever we wanted, as long as it had to do something with visualization. Unfortunately this meant that we thought we needed to follow the project description to the letter, which was a fundamental mistake. If I (Milán) had to choose a project again, I'd make sure to choose something that lets me be more creative. Also, our team assembled quite late, and we had to make a decision very quickly, and so we chose the project which had the most exact description of what was to be done. We couldn't have made a bigger mistake.

## 8 Task Separation

- **Charoenchaipiyakul:** Design, backend, frontend, *Implementation* and *Performance* sections of the report
- **Kolki:** Design, frontend(Custom Mode), rest of the report
- **Valkov:** Persona

## References

[1] Cleartype utility. `https://www.tenforums.com/attachments/tutorials/127344d1490910327-turn-off-cleartype-windows-10-a-cleartype-6.png`. Accessed: 2018-01-21.

[2] B. Fröhler. Histogram design. project description. `https://docs.google.com/document/d/1TWCuuauLR3Ok41cl1AGY0MQh_R9trTsGyMR5SNCS-Ok/edit?usp%3Dsharing`. Accessed: 2018-01-21.

[3] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, et al. Design galleries: A general approach to setting parameters for computer graphics and animation. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pp. 389–400. ACM Press/Addison-Wesley Publishing Co., 1997.

[4] R. Sanchez. Comparing node.js vs php performance. `http://www.hostingadvice.com/blog/comparing-node-js-vs-php-performance/`. Accessed: 2018-01-21.

[5] D. Skau, L. Harrison, and R. Kosara. An evaluation of the impact of visual embellishments in bar charts. In *Computer Graphics Forum*, vol. 34, pp. 221–230. Wiley Online Library, 2015.