Tabulator: Automated Table Generator

Pontus Orraryd*
Oskar Ankarberg†

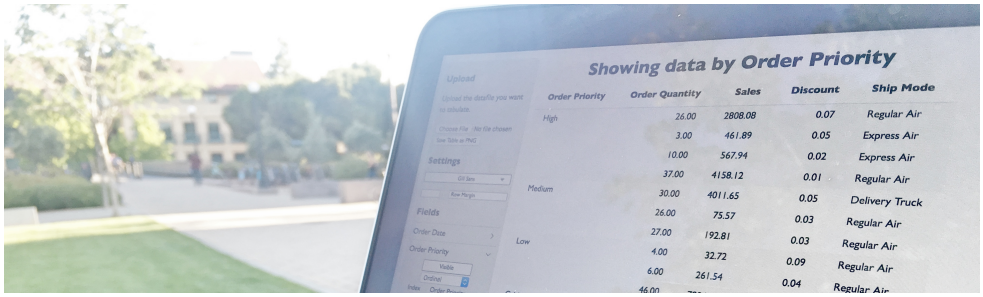

Figure 1: *Tabulator 1.0*

Abstract

Being able to use tables and visualizations to show data is a powerful way to quickly understand trends and implications of the data. However, most users do not want to spend time or effort in creating good visualizations. They may also not be aware of what specifically makes a table easy to read and interpret. While systems for generating good graphical visualizations exist, there are not many tools that create well designed tables. Most users resort to the standard generators found in program such as Microsoft Excel, which frankly, are not very good. While they can be customized to create pretty decent tables, the problem is that users may not know what characteristics a well designed table should have.

Here we present Tabulator, a web app to quickly create and download a well-designed table for use in papers and similar. The table generated upholds known principles in visualization and perception and is very easy to use. All the user needs is a data file containing the data they want to tabulate. After creation, some optional customization of the table is possible to allow users to specialize the table for their needs. With Tabulator, we hope to fill the gap between suboptimal table generators shipped with popular program and complex visualization tools not really suited for people not used to programming or advanced computer systems.

Keywords: data visualization, automated graph generation, table design, table generation

*orraryd@stanford.edu
†oankarbe@stanford.edu

1 Introduction

Tables may be one of the oldest ways to present data. Almost everybody who have ever written a paper or scientific report have most likely attempted to use one to show data of some kind. Using a table is perfect when we need to show exact numerical values[Tufte 1990]. However, using the usual tools to create a table in software like Excel is usually cumbersome, and the resulting table is often perceptually very bad. It does not help how these tools also present the user with hundreds of options - each more unnecessary than the next and may cause the user to actually make the table worse.

Order Date	Order Priority	Order Quantity	Sales	Discount	Ship Mode
13/10/10	Low	6	261.54	0.04	Regular Air
20/02/12	Not Specified	2	6.93	0.01	Regular Air
15/07/11	High	26	2808.08	0.07	Regular Air
15/07/11	High	24	1761.4	0.09	Delivery Truck
15/07/11	High	23	160.2335	0.04	Regular Air
15/07/11	High	15	140.56	0.04	Regular Air

Figure 2: *Automated table created in Numbers*

Consider the table made in Figure 2 in Numbers for Mac. The grid lines attract a lot of focus, almost imprisoning the data in their cells. The cells are spaced too close together vertically and the very dark backgrounds on the headers make the text hard to read. While there are most likely options hidden somewhere that might address these issues, most users will not spend their time and effort on this, and would most likely just use this default table. By doing so they do themselves, and others, a disfavor. A table designed like this is very suboptimal for displaying data [Few 2004].

While solutions for creating good graphical visualizations automatically have started to appear, there still has not been much work done to improve automatic creation of tables. There are tools that can create good tables, but only if the user have the necessary skills to find the right parameters. In reality, there are lots of users who use tables that do not possess these skills. Users who may not be very experienced with complex software and definitely does not know any programming.

We have created Tabulator, a web app to quickly create and download a well-designed table for use in papers and similar documents. The table is designed based on well-known principles in visualization and perception.

2 Related Work

2.1 Visual Perception

To know what makes an effective visualization, we need to know how human perception works. Cleveland & McGills studies on graphical perception is probably considered the most important contribution concerning this [Cleveland and McGill 1984]. They conducted studies and found what type of data encodings humans can decode most accurately. For example, for quantities, we can decode position and size very accurately but color hue and volumes less effectively.

2.2 Tables and Table Design

A lot of research have been done on what makes a good table. Tufte said that a good table should be free from visual clutter (”chartjunk”) [Tufte 1990]. Tschichold said that ”(rules) should be used only when they are absolutely necessary”. [Tschichold 1967]

Stephen Fews book describes multiple ways to improve table design.[Few 2004] He identifies multiple design principles that should be used when designing a table. Similarly to Tufte, Few puts a lot of emphasis on white space and how it is vital in making it easy to quickly scan the data. He goes into detail on different design choices and will serve as our main reference in our implemented system. Some examples on things Few talks about is margins, text formatting, fill color and grouping of tables.

Another important thing Stephen Few identifies is what different type of tables there are. A table can be Quantitative-to-Categorical or Quantitative-to-Quantitative. To put it simply, the first type serves as a look up of individual values, and the second type serves as a comparison between values. Tables can also be grouped into two main categories of *how* they display data. A table that is *unidirectional* have categorical items laid out in one direction, while a table that is *bidirectional* have categorical items laid out in two. Usually, the benefit of using a *bidirectional* table is that you can fit more data in fewer cells.

Finally, when should we use tables over graphs? Both Tufte and Few notes that the main reason to use tables above all else is that they are easy to look up individual values in. They also work well when comparing two specific values and provide a level of detail that graphs do not. We should however, not use tables for massive data sets (the table simply gets too large), or when we are trying to show trends and overall patterns of a data set. In those cases, some kind of graph visualization will be more effective.

2.3 Excel and Numbers for Mac

The well known spreadsheet software Excel originally called VisiCalc [1978] and became Microsoft Excel 1985 is a tool for visualizing data in graphs and spreadsheets. The software has a rich set of features but is rather complicated to start using without any background reading of its functionality. Figure 3 is an example of a quickly generated table from a CSV file in Excel without any input from the user. This table does not make use of any *whitespace* and column headers are not styled anything.This table would not reach a high score in [Tufte 2001]

Apples counterpart Numbers in Figure 2 admittedly creates better tables, but as mentioned earlier it also shows certain problems. This tool is more intuitive and more user friendly for first time users to create tables than Excel. The table comes out visually appealing and a lot of users will probably use this table to present the data. Instead, the user is tricked into believing that the generated table is efficiently displayed. But in fact, lots of redundant ink is displayed in the column header section as both fill color and text color is used.

Order Date	Order Priorit	Order Quant	Sales	Discount	Ship Mode
13/10/10	Low	6	261,54	0,04	Regular Air
20/02/12	Not Specifiec	2	6,93	0,01	Regular Air
15/07/11	High	26	2808,08	0,07	Regular Air
15/07/11	High	24	1761,4	0,09	Delivery Truck
15/07/11	High	23	160,2335	0,04	Regular Air
15/07/11	High	15	140,56	0,04	Regular Air

Figure 3: *Automated table created in Excel*

2.4 Web Applications and Frameworks

There are quite a few table generators found online. While they usually possess the ability to generate a table from a data file, they are not very well designed initially. Many function more like editors and the generated table require a lot of manual designing before it can be considered to be finished.

https://datatables.net is a framework for designing tables on the web. The framework does not aim to provide the most correct scientific visualization related to the provided data, it is rather a tool for developers to generate tables that look good, and not always display the data in an effective way according to visual perception. It is also specifically designed for use on the web and not for publications, which is what Tabulator is focusing on.

Tablesgenerator.com is another web application that can generate different tables for use in Latex, the web or in Markdown. The user is also able to style the table with multiple options after generating the table. Figure 4 show the table when it was generated automatically. The table have a black grid that steal focus from the data and have not styled the header columns any differently, making it easy to confuse it with data. This tool is also not meant to be automatic but require lots of tweaking before the table is finished.

Order Date	Order Priority	Order Quantity	Sales	Discount	Ship Mode
13/10/10	Low	6	261.54	0.04	Regular Air
20/02/12	Not Specified	2	6.93	0.01	Regular Air
15/07/11	High	26	2808.08	0.07	Regular Air
15/07/11	High	24	1761.4	0.09	Delivery Truck
15/07/11	High	23	160.2335	0.04	Regular Air
15/07/11	High	15	140.56	0.04	Regular Air

Figure 4: *Automated table created in Tablesgenerator.com*

A different kind of table generator is Bertifier [Perin et al.]. It is based upon Bertins matrix analysis method[Bertin 1981] and is an interactive tool for exploring tabular data. The tool allows you to reorder the data, add graphs and much more. It is more intended for exploratory data analysis than for creating effective graphics. The interactive features have been a major inspiration in how our interactivity has been designed.

3 Methods

3.1 Design Principles

Before implementing our system. We attempted to identify multiple design principles that our system would try to uphold.

- **User Friendly and Fast**: Our system should be easy and quick to use. Our target audience is people that do not feel comfortable using advanced tools and do not know any programming. However, almost everybody can find the need to use a table. We want our system to be usable even for these people. We want to make sure that our initial table have a good design even before any kind of modification is made by the user.
- **Visual perception over pretty design**: The tables our system creates are meant to be easy to read and interpret. While this usually also means it tends to look good, it is not our primary focus to create fancy tables. We will rather base of design choices on visual perception.
- **Minimize users impact on design**: Our system will not give the users too much choice regarding the design of the table. Too much choice can cause confusion and also lead to worse visualizations in the hands of inexperienced users.

These principles served as general guidelines for every design choice we made and we repeatedly asked ourselves if we still upheld these rules.

3.2 System Overview

Our system is split into three distinct parts.

1. The first part of the system reads the data from a data file a process it. It identifies certain properties of the data such as type of data. We call this part the *data extractor*.
2. Once the data is extracted and certain metadata on the data has been identified, we pass this data to the *table drawer*. Since the focus of Tabulator is on table design this is our biggest part of the system. The table drawer does not do any data processing but simply creates a table to present the data in. This separation allows us to create a more modular system. For instance, our table drawer could be used in any context where you want to create a table of `json` data (along with some metadata). It also makes it easy to separate operations on the data from the parts that are responsible for the design of the table.
3. Finally, we have the *table renderer*, which is simply responsible for generating an image file of the generated table.

3.3 Data Extractor

Our system takes as input a csv data file. To parse the file we use an external library called Papaparse[1]. A current limitation of our system is that we expect the first line of the csv file to contain headers. While this is often the case, a csv file without headers is still a "valid" csv file and we would parse these files incorrectly.

Once the files has been uploaded, we attempt to identify the data type of the different columns. We currently use a brute-force approach where we interpret something as numerical data if we can interpret all its rows as a number. If we are not able to do this, we consider the data type to be nominal. Due to this brute-force approach, we allow users to change the inferred data type after the creation of the table. For example, sometimes numbers are meant to be representing ids rather than a quantity and should therefore be represented as nominal data, but our system would interpret it as a numerical data type.

We also let the user define ordinal data types, as our system does not automatically detect these. The user then have to order the different values in a list which will allow our system to sort in that order (rather than alphabetically). This is done by using the specified ordinal order array as a sort function.

The data extractor also attempts to identify cases where the table is bidirectional, i.e when the first column is not data but categories. We do this by checking two things:

- The column only contains unique values.
- The column is not numerical.

These two simple requirements turned out to be an effective way to identify a potential bidirectional table.[2]

A small but significant feature is that our system also reformat header titles that may be written in camel case or snake case. These formats are very common but do not look very pretty for presentation purposes. Our system properly adds spaces and capitalization as you would expect from a header title, and improves the impression of the table created.

3.4 Table Design & the Table Drawer

The design of the table is determined by design principles from [Tufte 1990] [Few 2004]. Both Few and Tufte are well respected within the data visualization community and have written numerous books on the subject. We have attempted to translate their, sometime vague, ideas into practical implementations. These ideas will be discussed below.

One of the most important concepts in table design, according to Few, is the concept of *white space*. When data is clamped to tightly together, it can be hard to read. Using a lot of white space between data is a very effective way to make it easier for the reader to separate data from different rows.

The idea of *whitespace* have been very central for how we choose to design our tables. Both rows and columns always have a reasonable amount of space between them and by doing that we improve readability of the table a lot.

However, as a table get more rows, we also need to consider the amount of space the full table will occupy. The margin between rows will therefore decrease as the amount of data gets larger. To determine an appropriate amount of white space between rows, we use linear interpolation between our minimum and maximum amount of white space. The minimum is set to be about 20% of the row height. The maximum is set to be 100% of the row height. This range is mentioned by Few as being the extremes of reasonable margins. Making the margin lower than 20% and the rows will be very hard to separate. Having over 100% margin on the other hand, and our data density start being too low.

$$m(t) = (1-t)m_{max} + tm_{min} \tag{1}$$

[1] http://papaparse.com/

[2] While our system does identify a bidirectional table, it is currently not styled in any special way.

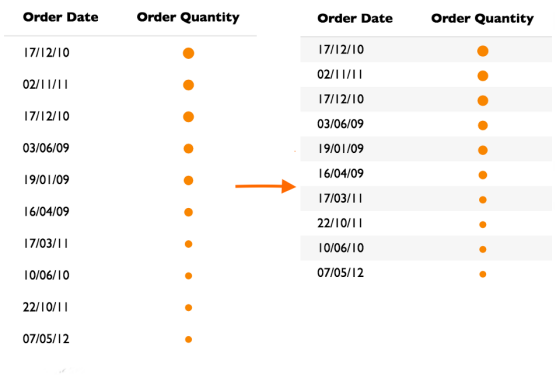

Figure 5: *Fill color appears when the margins get smaller.*

The parameter t is set by dividing the amount of rows with a reference size.

$$t = \frac{n_{rows}}{n_{ref}} \tag{2}$$

The reference value determines how fast the margins get smaller, and its value determine the limit when we reach the minimum value. We chose to set this to $n_{ref} = 50$. This effect is demonstrated in figure 5.

A very common element in tables is grid lines and rules used to separate rows and columns from each other. In a lot of cases, these gridlines are just *chartjunk* [Tufte 1990]. There are less intrusive ways to make it easier for the reader to separate the different data cells. Our system primarily uses *white space* for this purpose. When the margins get smaller however, we need to help the reader separate the rows in other ways. To accomplish this we introduce a light fill color that increasingly gets more apparent the smaller the rows get. We do this by using a similar interpolation as with row margins. Our table generator does not, under any circumstance, use grids to separate data cells.

When the user chooses to sort a nominal or ordinal value, we will only list each value once. This is done by aggregating the data and comparing the previous cell value with current cell value for the sorted column. We then use fill color to help users separate the different groups of data. This feature makes reading the data more efficient, as the user does not have to rescan the table for information that they already know [Few 2004].

To help readability when sorting this way. There will always be light fill color separating the groups. While fill color should be used sparingly, we felt it did more good than harm in this case as it really helps showing which rows belong to which category.

Text Formatting and typography

The text representation is formatted in a way to make it as perceptive as possible to read the data. Text is aligned to the left and numbers are aligned to the right [Few 2004], both together with their corresponding column header. Text in Tabulator is never presented in any other orientation than horizontal [Few 2004]. Numbers have always the same amount of decimals and digits are spaced equally to make decimal points align perfectly. This helps readability and makes it easier to compare numbers to each other.

The default font for Tabulator is Gill Sans, a Sans-Serif font which is legible and a favorite from Tufte [Tufte 2001]. The font for the generated table can thereafter be changed to 5 different fonts legible for tables, Helvetica, Arial, Verdana, Times New Roman, Palatino

Payment Type	Name	City
Amex	Heidi	Eindhoven
	Kelly	Reston
	Ritz	Pittsfield
	Sylvia	Pittsfield
	Nicole	Houston
Diners	Sheila	Brooklyn
	Janis	Ballynora
	Hani	Salt Lake City
	Bryan Kerrene	New York
	barbara	Hyderabad
	Leanne	Julianstown
	Stacy	New York
Mastercard	Federica e Andrea	Astoria
	Sean	Shavano Park

Figure 6: *Example of nominal sorting*

[Few 2004]. This gives the user some choice over what font to use, but all fonts are well suited for being used in a table.

User Interface

The main focus of this project has not been to create an efficient and visually appealing user interface. Although some focus has been towards the user interaction. We want to minimize the time the user spends on our web application. Figure 7 shows the user interface when interacting with the ordinal sort function. This sort functionality is optimized by introducing drag and drop by rows. When the user drops the row, the table immediately updates and sorts by the requested ordinal order.

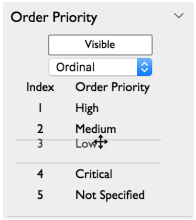

Figure 7: *View of the user interface ordinal sort*

3.5 Interactive Features

Quantities as shapes

Our web application lets the user represent quantitative data as circles. It detects if a column is represented by numbers and adds the functionality to those columns. Our reasoning for adding this feature is that sometimes the actual numerical value of a column may not be that important, but the relationship between the values are. When that is the case, we can more effectively infer relationships from shapes rather than from text.

We scale the circles by area. Research in perception has determined that we tend to underestimate area and so we have compensated this according to James Flannerys model [Flannery 1971]. We use the following scaling:

$$S = 0.98A^{0.87}$$

It is worth noting that this method has been criticized by Tufte (among others) for distorting the data. Indeed if one were to measure the areas of the circles, they would get an incorrect value. However, when comparing the two methods we made the choice that Flannerys model made sense in this case.

Reordering of columns

The user might want to change the order of which the columns should be represented. Therefore we added the functionality to reorder columns by dragging and dropping a column between desired columns. This is done by cloning the table into a temporary table on drag and then reordering the data on drop. Since tables does its best job of presenting data in a limited size [Tufte 2001], the user also has the ability to hide undesired columns.

Sorting

The user can sort each column by clicking on each column header. This will instantly change the order and the displayed table header seen in Figure 9. The sorting is done by aggregating the data and comparing cell value by value. As previously mentioned, when sorting nominal or ordinal data the table will not repeat the same value multiple times as demonstrated in Figure 6.

3.6 System Output

When the user is satisfied with the table, he/she can generate an image file of the table. The output of the system is currently a png file generated in Javascript. HTML5 lets us define a `<canvas>` element that will be rendered as a png file.

4 Results

Our system is implemented as a web application written in Javascript. To process the data into tables we have used a common visualization library called D3 [Bostock]. D3 allows us to bind data to specific DOM elements, which is very handy in any type of visualization.

Our system generates a table from an uploaded csv data file. The user then has some options concerning the design of the table. Our system minimizes the user mistakes while still providing custom styling. Some of these style options include choosing font, changing order of columns, change data type and sorting. Once the user is satisfied with the table, he/she can download a png image of the table for use.

A complete view of our application can be seen in figure 8. This shows both the user interface and a generated table which has also been modified. In this specific screenshot, the user has taken advantage of the ordinal data type by choosing what order to sort the data.

Figure 9 show how the same table would look initially after being uploaded. Since we want our system to generate good tables without any user interaction, this is also an important part of our system.

5 Discussion

While table design may seem simple at first glance, there are lots of things to think about. One thing we realized quickly was that there are no perfect design. Almost every design choice involved a trade

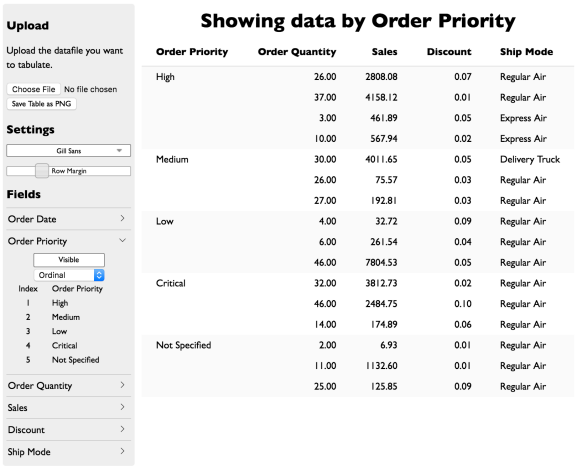

Figure 8: *Full view of our system*

Showing data by Order Date

Order Date	Order Priority	Order Quantity	Sales	Discount	Ship Mode
13/10/10	Low	6.00	261.54	0.04	Regular Air
20/02/12	Not Specified	2.00	6.93	0.01	Regular Air
15/07/11	High	26.00	2808.08	0.07	Regular Air
02/11/11	Critical	46.00	2484.75	0.10	Regular Air
17/03/11	Critical	32.00	3812.73	0.02	Regular Air
17/12/10	Low	46.00	7804.53	0.05	Regular Air
16/04/09	High	37.00	4158.12	0.01	Regular Air
28/01/10	Medium	26.00	75.57	0.03	Regular Air
18/11/12	Low	4.00	32.72	0.09	Regular Air
07/05/12	High	3.00	461.89	0.05	Express Air
10/06/10	Medium	27.00	192.81	0.03	Regular Air
10/06/10	Medium	30.00	4011.65	0.05	Delivery Truck
30/04/12	Not Specified	11.00	1132.60	0.01	Regular Air
20/10/11	Not Specified	25.00	125.85	0.09	Regular Air
11/09/11	High	10.00	567.94	0.02	Express Air
07/08/10	Critical	14.00	174.89	0.06	Regular Air

Figure 9: *The initial table generated*

off of some kind. For instance, it would make sense perceptually to have bigger tables have larger row margin than smaller tables. After all it becomes harder to separate the rows if there are more of them. However, we chose to do the very opposite, and make the margins smaller instead. Our reasoning were that users often have a practical limit in the form of space and so a larger table have to be clamped together to accommodate.

Finding just the right amount of customization for a system like this is tricky. If you give a user too much freedom they may unknowingly reduce the readability of their created table when they customize it. On the other hand, if you give no freedom at all and just produce a completely static table, users may find it hard to use these tables in their specific context. This is something we are still trying to balance. Our current view is that the user is barely allowed to change anything that could potentially make the table worse perceptually, but has some choice of the tables visual look and what to display.

Ordinal data turned out to be a tricky case, and for a long time we considered to supporting it at all. As we had no way to automatically detect the ordering of this data we knew we would have to implement some kind of interface to allow the user to specify the order. This felt like it went against one of our design principles to keep the system very simple. Nevertheless we eventually decided that being able to sort by ordinal data is very valuable so we did choose to implement it. We tried to keep the functionality as simple as possible with a drag-and-drop interface which we feel worked out pretty good.

One of our main design principles is to always let visual perception be more important than ”pretty” design. While those who have knowledge in the field of data visualization or perception can see the value in this, a user of the type we are targeting may instead find the tables bland or even boring. It can certainly be problematic that the users we are trying to target may not appreciate the design of the tables and it is definitely important to realize that not all users will be able to ”see” the value of some of our design choices.

While our system concerns the automatic creation of tables, it is important to realize that a table is definitely not the tool for all kinds of data presentation. First and foremost, tables are limited by *size*. A table should be used for small data sets (or aggregations of larger data sets) and never for displaying a large amount of data. Our system will not deny any size of data, which could potentially lead users to create massive tables that are too big to be comprehensible. Some way to warn the user that they may try to show too much data in a table could be appropriate to partially solve this.

6 Future Work

There are multiple ways this system could be extended and improved. Most improvement involve automating certain processes that is currently not automatic. The next step in our system would be including more graphical components as ways to show data. Small bar charts, spark lines or other type of charts can effectively combined with a table to show data effectively and is something we would like to support. We need to careful however in how features like this will make the system more complicated and may scare users away.

We would also like to automate more types of formatting. Dates for instance, which we currently do not try to identify, also have some special formatting you can apply to improve their readability. We did not implement this because of how many date formats there are. Without knowing beforehand anything about how the user may input dates, it is hard to robustly identify them.

In the future we would like the user to select between a set of output formats e.g, svg,png and xlsx. This will although have to be done server side as a browsers cannot generate vector graphics. Vector graphics would be a better choice for tables than a static image file. Overall improvements are needed to the system that renders the final image to create images of better quality.

Another extension would be to support more complex types of tables. The tables created by Tabulator always create one rectangular table. Sometimes, it can be more beneficial to group certain subsets of the data into separate tables, shown side by side.

One major extension (and way beyond the scope of this project) would be the inclusion of some kind of word processing to infer meaning and context automatically. This would enable automatic detection of specific orders of ordinal data, hierarchical patterns (like country, state and county) and in turn design the table with this knowledge in mind. Currently the user have to enter the order of ordinal data manually, which can be tedious. Automating this would be a big step in truly automating the table generation process.

7 Conclusion

We have created Tabulator, an automatic table generator designed to quickly generate well designed tables that use ideas from visual perception to make it as easy as possible to read data from the table. We specifically target people who are not very experienced or comfortable with more advanced software to create visualizations. We hope that our tool can be used by everyone to quickly create a table that is usable in papers or similar.

References

BERTIN, J. 1981. *Graphics and Graphic Information Processing*. Walter De Gruyter Inc.

BOSTOCK, M. D3. `https://d3js.org/`.

CLEVELAND, W. S., AND MCGILL, R. 1984. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American statistical association 79*, 387, 531–554.

FEW, S. 2004. *Show Me the Numbers*. Analytics Press.

FLANNERY, J. 1971. The relative effectiveness of some common graduated point symbols in the presentation of ouantitative data. *The Canadian Cartographer 8*, 2.

JOCK D. MACKINLAY, P. H., AND STOLTE, C. 2007. Show me: Automatic presentation for visual analysis. *IEEE Trans Visualization Comp Graphics 13*, 6, 1137–1144.

MACKINLAY, J. 1986. *Automating the Design of Graphical Presentations of Relational Information*. PhD thesis, Stanford University.

PAT HANRAHAN, CHRIS STOLTE, D. T. 2002. Polaris: A system for query, analysis and visualization of multi-dimensional relational databases. *IEEE Trans on Visualization Comp Graphics 8*, 1.

PERIN, C., DRAGICEVIC, P., AND FEKETE, J.-D. Bertifier. `http://www.bertifier.com/`.

TSCHICHOLD, J. 1967. *Asymmetric Typography*. Reinhold Publishing.

TUFTE, E. R. 1990. *Envisioning Information*. Graphics Press LLC.

TUFTE, E. R. 2001. *The Visual Display of Quantitative Information*. Graphics Press LLC.