

InfoVis Editors

Manuel Schweiger, Reinhold Seiss, Maximilian Tauß, Stefan Tscheppe

Information Visualisation course at
Institute of Interactive Systems and Data Science (ISDS),
Graz University of Technology
A-8010 Graz, Austria

14 May 2018

Abstract

Information Visualisation is a growing field and gains popularity as data aggregation becomes easier and big data more accessible. Many people working in different areas rely on Information Visualisation tools and editors in order to present their findings and support their argumentations. However, there exist many tools with varying functionality. It is thus important to provide an overview.

This survey studies popular Information Visualisation editors and compares them according to a set of metrics. The various editors and tools have been grouped into three distinct types: Plug and Play, Programmable, and Data-Guided Drawing.

For simple Plug and Play editors we considered Raw, Datawrapper, and LiquidDiagrams. For Programmable tools we looked at Vega with Lyra and amCharts. For Data-Guided Drawing editors we tested Hyecoo, Data-Illustrator, and iVisDesigner. At the end of this survey we recommend one editor for each of the groups defined above.

© Copyright 2018 by the author(s), except as otherwise noted.

This work is placed under a Creative Commons Attribution 4.0 International (CC BY 4.0) licence.

Contents

Contents	ii
List of Figures	iv
List of Listings	v
1 Introduction	1
1.1 Types of Editors	1
1.1.1 Plug And Play	1
1.1.2 Programmable	1
1.1.3 Data-Guided Drawing	2
1.2 Metrics For Evaluating The Different Editors	2
1.2.1 Platform	2
1.2.2 Pricing And License	2
1.2.3 Customisability	2
1.2.4 Range Of Chart Types	2
1.2.5 Usability	2
1.2.6 Export Formats	3
2 "Plug'n'Play Editors"	5
2.1 RAW Graphs	5
2.1.1 About	5
2.1.2 Workflow	5
2.1.3 Extendability	9
2.1.4 Evaluation Of RAW Graphs	9
2.2 Datawrapper	10
2.2.1 About	10
2.2.2 Workflow	10
2.2.3 Evaluation Of Datawrapper	10
2.3 Liquid Diagrams	13
2.3.1 About	13
2.3.2 Workflow	13
2.3.3 Evaluation Of Liquid Diagrams	13

3	Programmable	17
3.1	Vega with Lyra	17
3.1.1	What Is Vega?	17
3.1.2	Vega Vs. Vega-Lite	18
3.1.3	Lyra	18
3.1.4	Evaluation Of Lyra	24
3.2	amCharts(Live)	25
3.2.1	About	25
3.2.2	Workflow	25
3.2.3	Evaluation Of amCharts	25
4	Data Guided Drawing	29
4.1	Hyecoo	29
4.1.1	About	30
4.1.2	Features Of Hyecoo	30
4.1.3	Workflow	30
4.1.4	Evaluation Of Hyecoo	31
4.2	Data Illustrator	33
4.2.1	About	33
4.2.2	Features Of Data Illustrator	33
4.2.3	Workflow	33
4.2.4	Evaluation Of Data Illustrator	34
4.3	iVisDesigner	37
4.3.1	About	37
4.3.2	Workflow	37
4.3.3	Evaluation Of iVisDesigner	41
5	Concluding Remarks	43
5.1	Recommendations	43
	Bibliography	45

List of Figures

2.1	RAW Graphs: Uploading Data Set	6
2.2	RAW Graphs: Choosing A Chart	7
2.3	RAW Graphs: Mapping data to specific axes	7
2.4	RAW Graphs: Customizing The Chart	8
2.5	RAW Graphs: The Finished Barchart	8
2.6	RAW Graphs: Downloading The Barchart	9
2.7	Datawrapper: Uploading The Dataset	11
2.8	Datawrapper: Checking/Modifying The Dataset	11
2.9	Datawrapper: Selecting and modifying the chart	12
2.10	Datawrapper: Export Finished Chart	12
2.11	Liquid Diagrams: Selecting the type of chart	14
2.12	Liquid Diagrams: Import and check the data-set	14
2.13	Liquid Diagrams: Modify The Chart	15
2.14	Liquid Diagrams: Export Finished Chart	15
3.1	One dot per zip code in the U.S. example visualisation	18
3.2	Lyra: Graphical User Interface	21
3.3	Lyra: Data Pipelines Panel	21
3.4	Lyra: Visualisation Preview Panel	22
3.5	Lyra: Layers and properties panel	22
3.6	Lyra: File Options Bar	23
3.7	Lyra: Napoleon’s march to Moscow by Charles Minard, 1869	23
3.8	amCharts: Select the type of chart	26
3.9	amCharts: Import and check the data-set	26
3.10	amCharts: Set category in general settings	27
3.11	amCharts: Set value field in graph section	27
3.12	amCharts: Export As Embedded HTML	28
3.13	amCharts: Export As Included JavaScript	28
4.1	Hyecoo: Flower Chart	29
4.2	Hyecoo: Empty Project	30
4.3	Hyecoo: Data Guides	31
4.4	Hyecoo: Repeating Links	32
4.5	Hyecoo: Exporting SVG	32

4.6	Data Illustrator: Overview	33
4.7	Data Illustrator: Drawing A Rectangle	34
4.8	Data Illustrator: Repeating A Rectangle	35
4.9	Data Illustrator: Splitting A Shape	35
4.10	Data Illustrator: Final Result	36
4.11	iVisDesigner: Different Components Explained	37
4.12	iVisDesigner: Copy and pasting the dataset	38
4.13	iVisDesigner: Creating The Axes	39
4.14	iVisDesigner: Creating The Bars	39
4.15	iVisDesigner: The Finished Barchart	40
4.16	iVisDesigner: Exporting The Chart	40

List of Listings

- 3.1 Vega Specification Example 19
- 3.2 Vega-Lite Specification Example 20

Chapter 1

Introduction

Information Visualisation editors are used to produce charts representing a specific set of data. Typically the more complex the data is, the more sophisticated the editor needs to be in order to output a meaningful representation of the given data. There are many visualisation editors out there suited for different tasks and charts. Depending on your use-case you might want to choose a different editor in order to get the job done.

1.1 Types of Editors

As already said, there exists a wide variety of different types of editors. Ranging from simple editors with low flexibility to complex editors with high flexibility, there is generally a tool available that meets the requirements. To make things simple, we defined three groups of editors: Plug and Play editors, Programmable editors, and Data-Guided Drawing editors. The following three sections give a brief overview of said groups.

1.1.1 Plug And Play

With *Plug and Play* editors we consider simple and easy to use editors. These editors might not be very powerful in terms of customisability and range of chart types, however they are very beginner-friendly. They produce fast outputs with many predefined chart templates ready to be used.

This type of editors is meant for people who don't want to spend a lot of time on producing charts for their data, but rather plug in their data and get the desired result within a few minutes. We evaluated the following:

- *RAW Graphs* by DensityDesign Research Lab, Calibro, and ContactLab
- *Datawrapper* by Datawrapper GmbH
- *LiquidDiagrams* by the Institute of Interactive Systems and Data Science (ISDS) at Graz University of Technology

1.1.2 Programmable

Programmable editors are more sophisticated editors that provide high customisability by allowing the user to personalise charts programmatically, either by providing the user an API or by generating code that may be accessed by the user's application. This makes them less beginner-friendly, however they are very powerful in the hands of experts, providing them with a big range of different chart types.

Also note, that since you are able to customise charts programmatically, this group of editors is very well suited for developing highly interactive visualisations. Editors that fall into the *Programmable* category and which we looked at are:

- *Vega with Lyra* by the Interactive Data Lab of the University of Washington
- *amCharts* by amCharts

1.1.3 Data-Guided Drawing

Data-Guided Drawing editors take on a different approach. Instead of letting the data define certain values in the chart (e.g. the height of a bar in a bar chart), the data functions as guides, around which the user may draw shapes that represent the respective values in the data set.

Obviously, this type of editors are not limited by predefined chart types. The editor provides the user with a set of graphics tools that may be found in any other vector graphic based editing software (e.g. Adobe Illustrator). The range of chart types is virtually unlimited. These editors are suited for people who have an eye for design and want to get very creative with their visualisations. We looked at the following three *Data-Guided Drawing* editors:

- *Hyecoo* by Nam Wook Kim (Ph.D. student at the university of Harvard)
- *Data-Illustrator* by Adobe
- *iVisDesigner* by Donghao Ren

1.2 Metrics For Evaluating The Different Editors

In order to effectively compare the various editors, we defined six metrics for evaluating the different editors.

1.2.1 Platform

Which platform the editor is running on. For example lots of the discussed editors are available as hosted web applications.

1.2.2 Pricing And License

Under which pricing or license the editor is available. Is it open-source? If not, how much do you have to pay in order to use or publish charts created by the respective editor? Is it a monthly subscription or do you have to pay only once?

1.2.3 Customisability

How customisable the charts are the editor can produce. Does the editor support custom colors, custom title, or custom labels?

1.2.4 Range Of Chart Types

How many different types of charts the editor is able to produce. Are you limited to a set of predefined chart types or can you get very creative with the editor and design your own chart types?

1.2.5 Usability

How good the usability of the editor is. Is it very beginner-friendly? Does it require a lot of time to understand the interface? Is a good documentation available describing the features of the editor?

1.2.6 Export Formats

How can the newly created chart be exported and used. Does the editor support exporting the chart as e.g. SVG/PNG/JSON? Does the editor provide custom code that may be embedded in a web application?

Chapter 2

"Plug'n'Play Editors"

The first class of editors we take a look at, are the *Plug and Play* editors. All editors in this category are very easy to use, include a self-explanatory user interface and offer a predefined range of chart types. Overall they are very beginner-friendly and one can produce charts very quickly.

Due to being streamlined and very polished from beginning to end, they cut down on customisability. You often can only choose from a limited set of chart types. Sometimes axis labels or other label like color descriptions can not be added in the editor itself and have to be added by the user in a post-processing step if needed.

We named this category *Plug and Play* editors, since you actually can plug in your data, make a few more clicks and export your chart. You do not need to be an expert, nor are any programming skills required to be able to use any of these editors.

Following are three editors which fit our criterias of *Plug and Play* editors.

2.1 RAW Graphs

As short and simple the name of this editor, as easy it is to use as well. RAW Graphs [Mauri et al. [2017]] is the first editor in the category of *Plug and Play* editors, we take a look at.

2.1.1 About

The editor, more specifically a hosted web application of the editor, can be found at their website [RAW Graphs Website [2018]]. RAW Graphs is an open source data visualization framework. Its goal is to make the visualization of complex data easy and accessible to everyone [RAW Graphs Website [2018],About]. Although it is self-hosted, RAW specifically states that any uploaded data stays on your machine and is not uploaded to any of their servers.

The framework is based on the popular JavaScript data visualization library D3 [RAW Graphs Github [2018]]. Initially it was developed by the DensityDesign Research Lab and Calibro and after a couple of years ContactLab got involved as well.

More information about RAW Graphs can be found on the webpage of its inventors.

2.1.2 Workflow

We will illustrate the workflow of RAW Graphs on a practical example. Usually RAW Graphs is used as a link in-between any spreadsheet software like Microsoft Excel and any vector graphics editor like Adobe Illustrator. This will become more clear as we will explain how to create a simple barchart, step by step.

The dataset we are going to use for the barchart example is about the medals Austria won in the Winter Olympics of the last 5 years. We will use the same dataset for further workflow examples for other editors to make it easier comparing them. In RAW Graphs the creation of a chart entails the following steps:

1. Upload the dataset and make adjustments if needed (see Figure 2.1).
2. Choose one of the predefined chart types (see Figure 2.2). Note that there always is a description of the highlighted chart type.
3. Link and map your given data to the specific axes available (see Figure 2.3). Depending on the chosen chart type this view might be different.
4. Customize the chosen chart (see Figure 2.4).
5. Double check the created chart (see Figure 2.5) and make further adjustments if necessary.
6. Download your chart (see Figure 2.6) by using one of the available options. Note that you also can directly copy the embed SVG code.

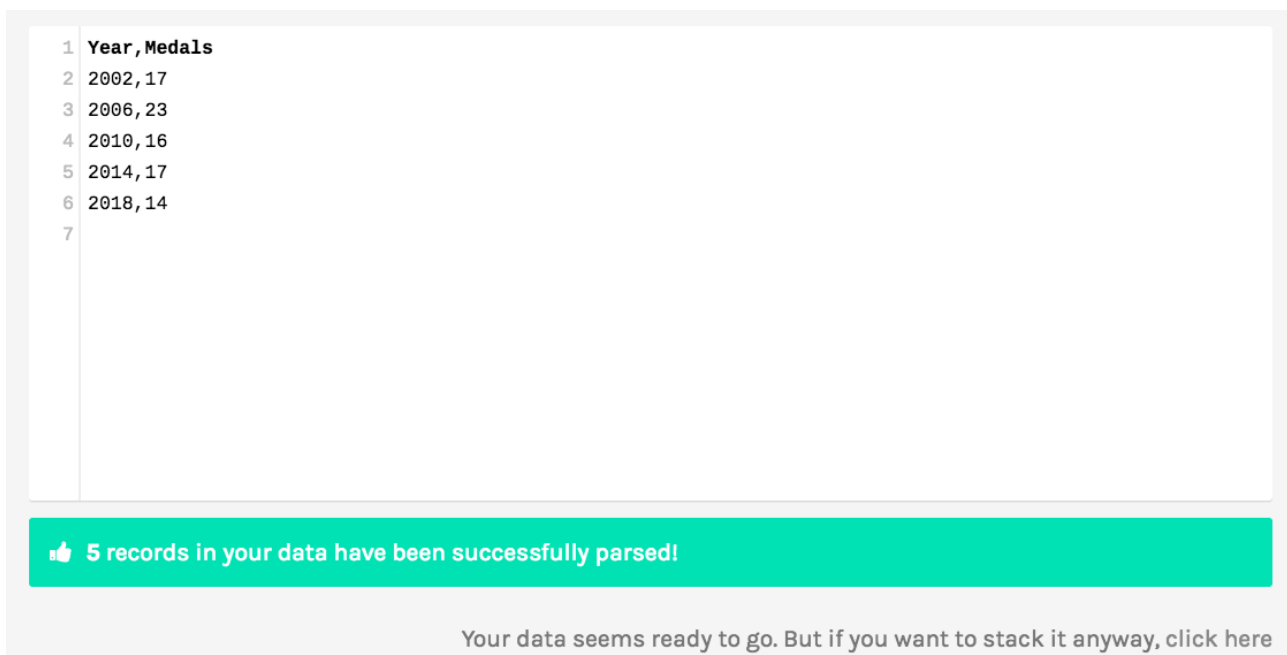
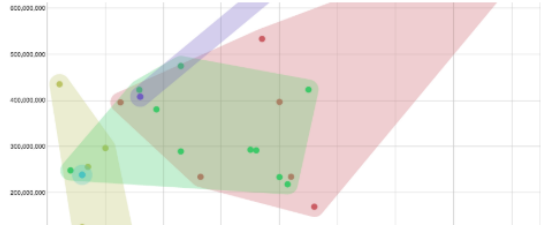


Figure 2.1: After we have uploaded our example data set.


Choose a Chart

Convex Hull
Dispersion




In mathematics, the convex hull is the smallest convex shape containing a set of points. Applied to a scatterplot, it is useful to identify points belonging to the same category.

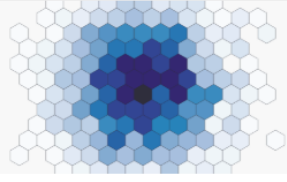
Based on <http://blocks.org/mbostock/4341699>



Convex Hull
Dispersion



Delaunay Triangulation
Dispersion



Hexagonal Binning
Dispersion

Figure 2.2: Choosing A Chart

Map your Dimensions

Year number →

Medals number →

X Axis *

Drag numbers, strings here

Year number ×

Height

Drag numbers here

Medals number ×

Figure 2.3: Mapping data to specific axes.

Customize your Visualization

Width
800

Height
600

Vertical Padding
0

Horizontal Padding
0,1

Use Same Scale

Color Scale
Ordinal (categories) ▾

Search...
null #bf6969

Figure 2.4: Customizing The Chart

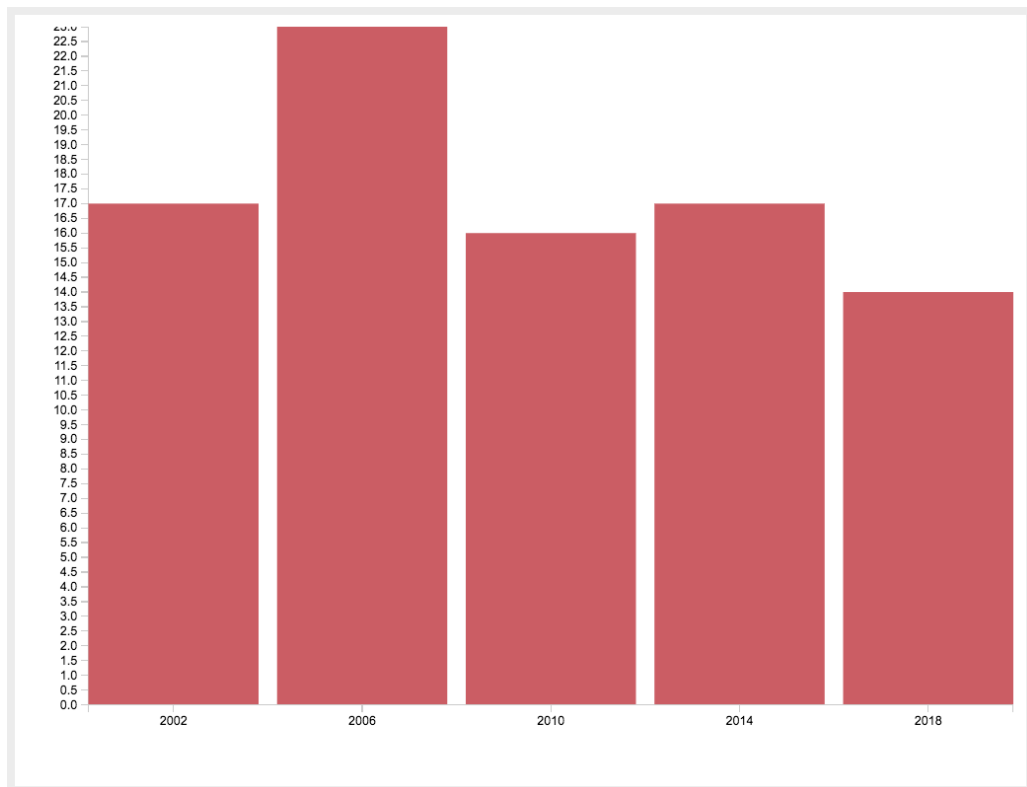


Figure 2.5: The Finished Barchart

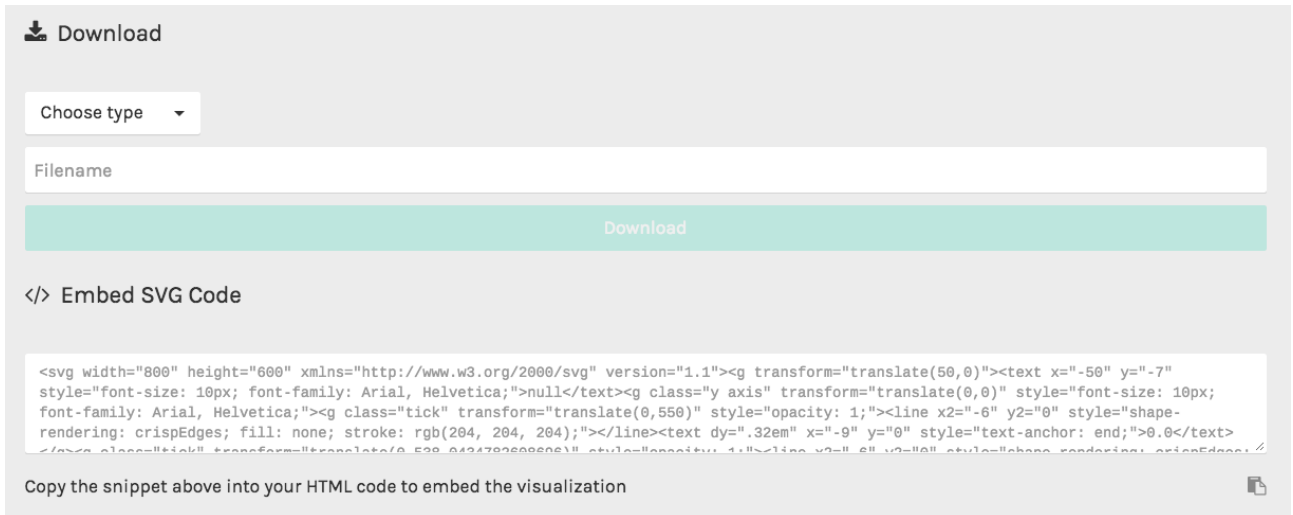


Figure 2.6: Downloading The Barchart

2.1.3 Extendability

Note that if you have the required skills you can also fork the entire project on Github [*RAW Graphs Github* [2018]] and self host RAW Graphs or create custom chart types this way.

2.1.4 Evaluation Of RAW Graphs

Here you can see our evaluation along with the metrics we defined above. RAW Graphs is a finished product and you can clearly see that the services it offers are highly polished and streamlined. Its user base is well defined and it cemented its existence into the heart of the data visualization community, which wants to do just that – visualize data. Following is our evaluation of RAW Graphs according to our metrics:

1. *Platform*: hosted web application (uploaded data stays local)
2. *Pricing and license*: free under the Apache License 2.0
3. *Customisability*: low
4. *Range of chart types*: low, but extendable if you fork the Github project
5. *Usability*: very good
6. *Export formats*: SVG / PNG / JSON

2.2 Datawrapper

Similar to RAW, and as expected in this section, Datawrapper is also a very easy to use editor and tool in which we decided to look into.

2.2.1 About

Datawrapper was designed mainly for journalists and content-creators as a easy tool to for creating simple charts and also maps. It was created by the Datawrapper GmbH and is still being enhanced and extended. It is developed as a hosted web-application and offers a free branded trial, but is mainly intended to be used with the monthly subscription.

2.2.2 Workflow

As above we will also use the example of creating a bar chart to show the workflow. This is done in four simple steps:

1. Uploading the data (see Figure 2.7)
2. Checking/editing the data (see Figure 2.8)
3. Choosing a chart-type and customize it (see Figure 2.9)
4. Exporting of the finished chart (see Figure 2.10)

2.2.3 Evaluation Of Datawrapper

In general Datawrapper is a very good tool for the task it was intended for. It is easy to use, offers good looking charts and does not burden the user with too many customizability options. The two biggest flaws of it would be the very restricted export formats (embedded HTML for free users and PNG/PDF for paying customers) and the very unreasonable pricing. Not only is it branded up to the second tier(130 \$ per month) also the enabling of PNG export only for paying customers seems to be weird. Following is our evaluation of Datawrapper according to our metrics:

1. *Platform*: Hosted web-application
2. *Pricing and license*: free(branded) - 800+\$ per month
3. *Customisability*: low (medium)
4. *Range of chart types*: low
5. *Usability*: very good
6. *Export formats*: Embedded HTML (PNG/PDF)

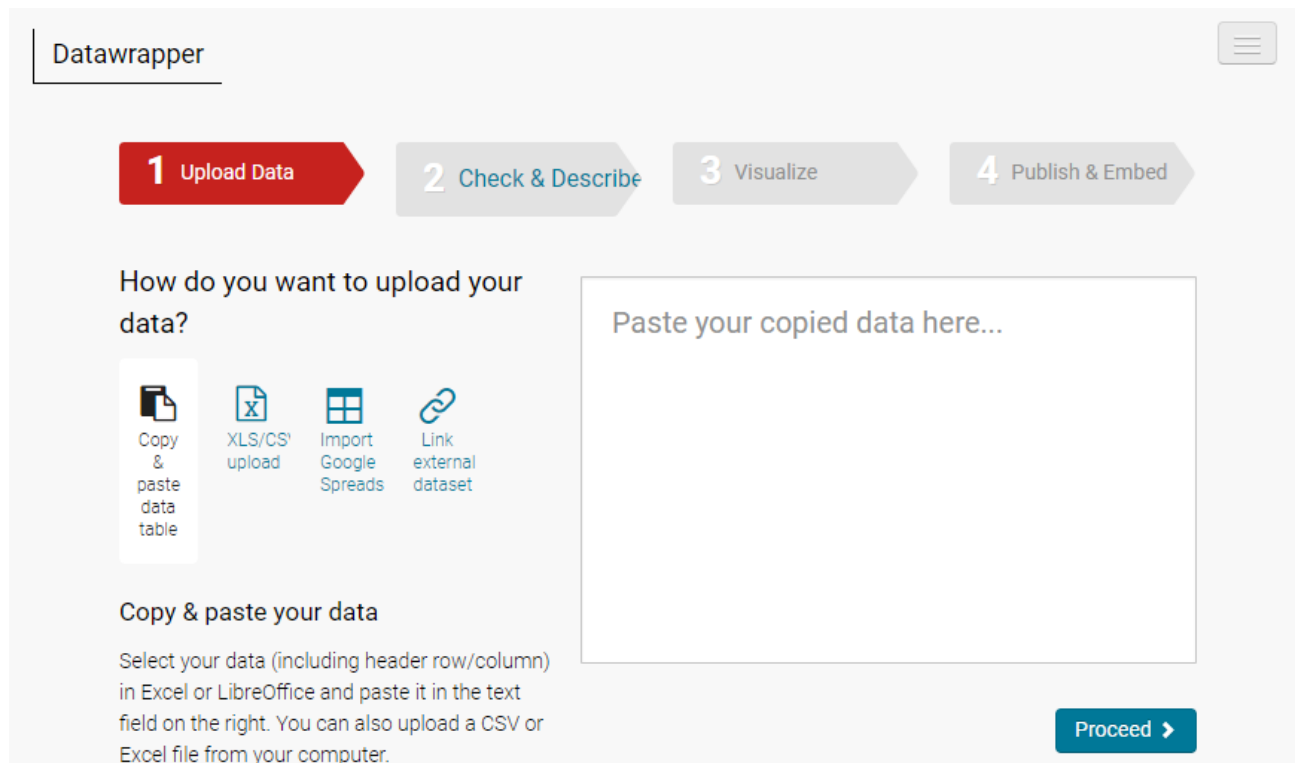


Figure 2.7: Uploading The Dataset.

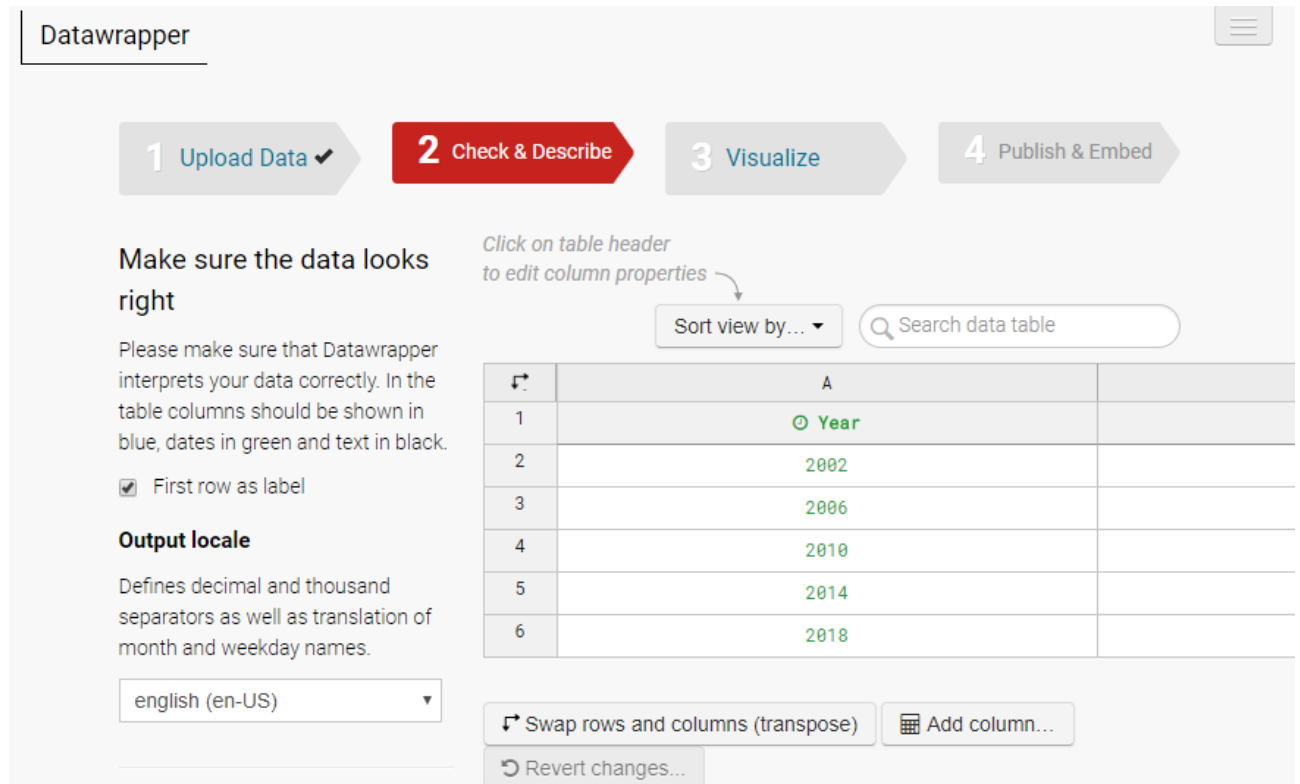


Figure 2.8: Checking/Modifying The Dataset.

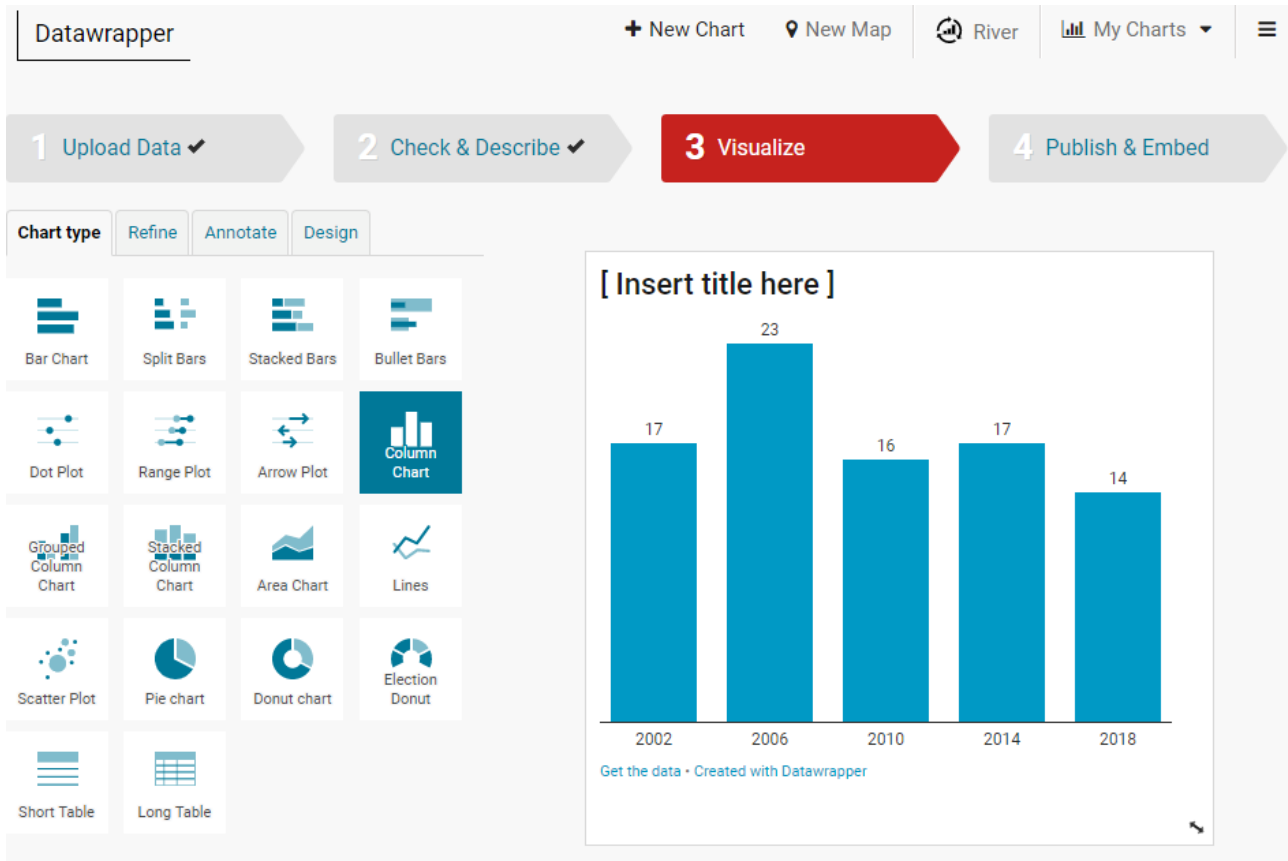


Figure 2.9: Selecting the chart-type and modifying the chart.

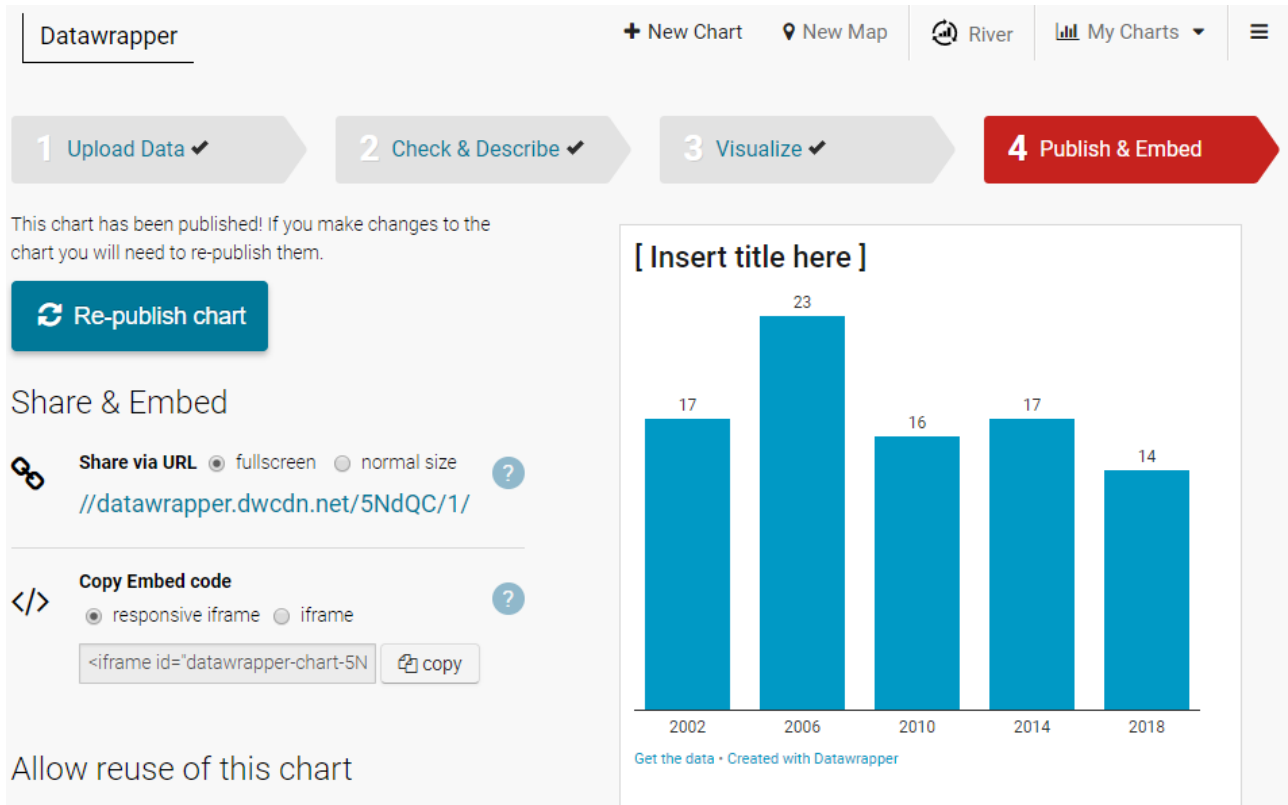


Figure 2.10: Export Finished Chart

2.3 Liquid Diagrams

A bit different to the other two editors in this category Liquid Diagrams is not a commercial product, but a university project. It also differs in the way that it is not a web-hosted application, but an standalone tool.

2.3.1 About

This editor was made by the Institute of Interactive Systems and Data Science at the Graz University of Technology. It was put into practice using action script and adobe air. It is not updated anymore as both these technologies are no longer developed.

2.3.2 Workflow

The workflow of Liquid Diagrams is very similar to that of the other two tools in this category and will also be demonstrated by the example of an bar-chart.

1. Choose the chart-type (see Figure 2.11)
2. Import and check data (see Figure 2.12)
3. Modify the chart (see Figure 2.13)
4. Exporting of the finished chart (see Figure 2.14)

2.3.3 Evaluation Of Liquid Diagrams

Liquid Diagrams is a very good compromise between RAW and Datawrapper. It offers some customisability which is dearly missed in RAW(e.g.: directly adding titels/labels) and also offers a better choice of export format then Datawrapper (for free). Its biggest flaw would be that it is based on a outdated technology and therefore it is not kept up to date. Following is an evaluation of Liquid Diagrams according to our metrics:

1. *Platform:* Windows/MacOS (Adobe AIR)
2. *Pricing and license:* free
3. *Customisability:* low
4. *Range of chart types:* low
5. *Usability:* very good
6. *Export formats:* SVG/PNG

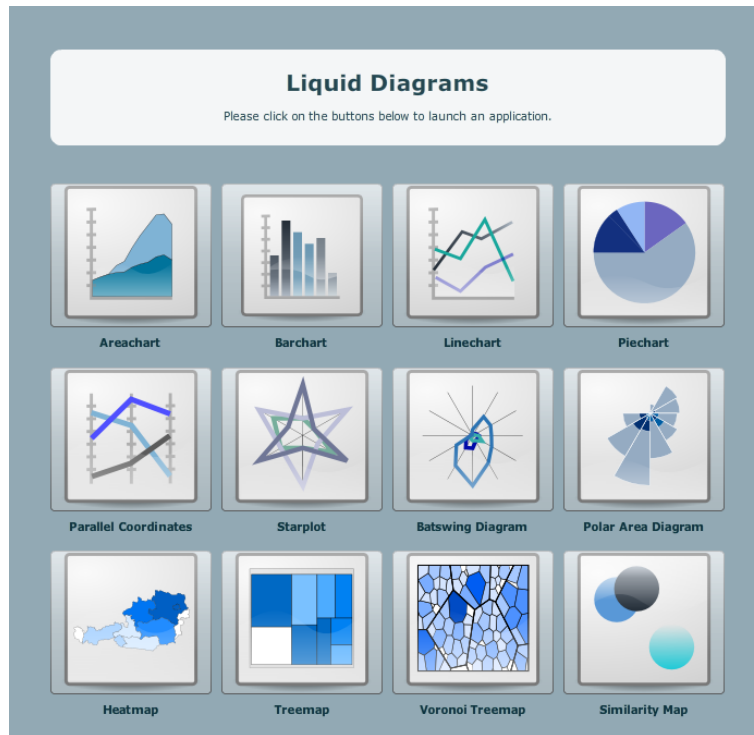


Figure 2.11: Select the type of chart.

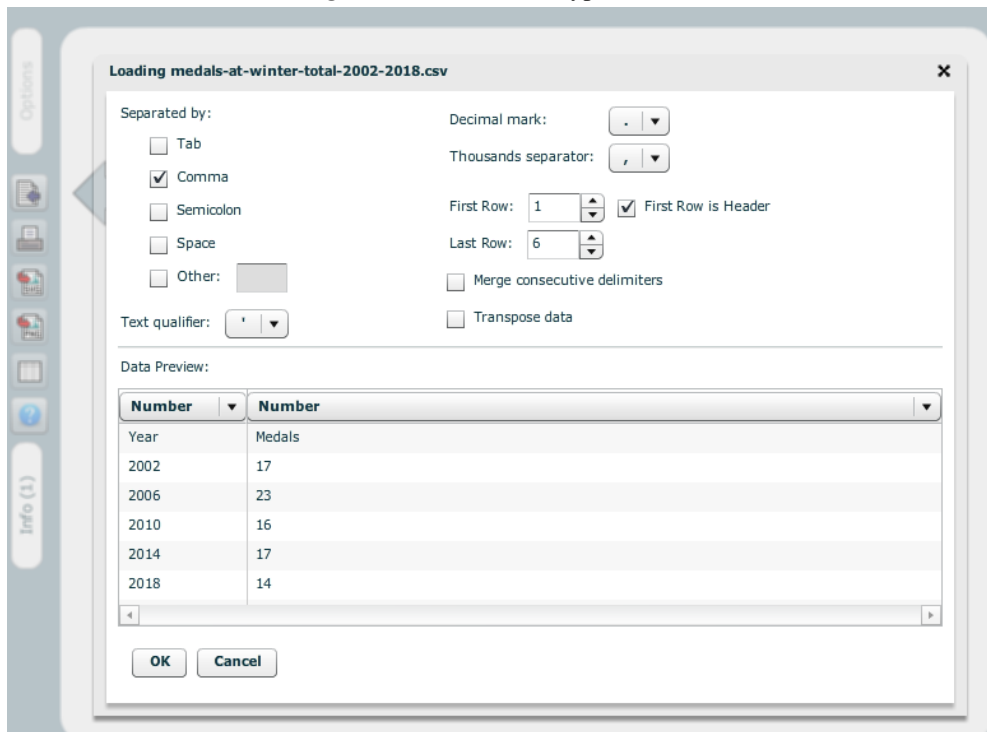


Figure 2.12: Import and check the data-set.

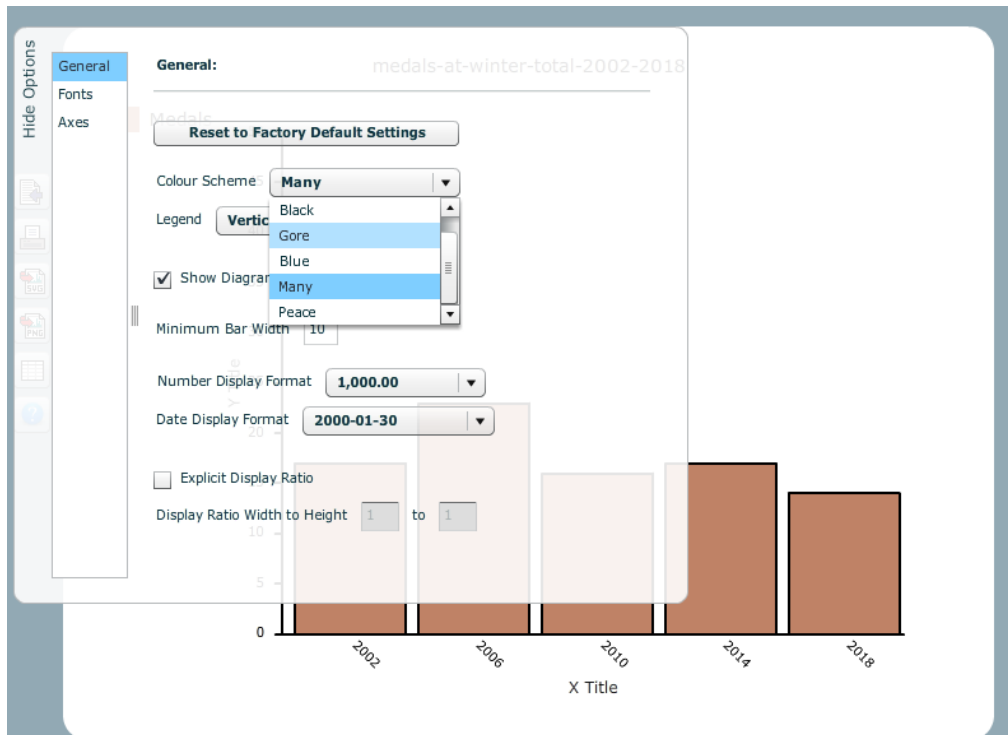


Figure 2.13: Modify The Chart

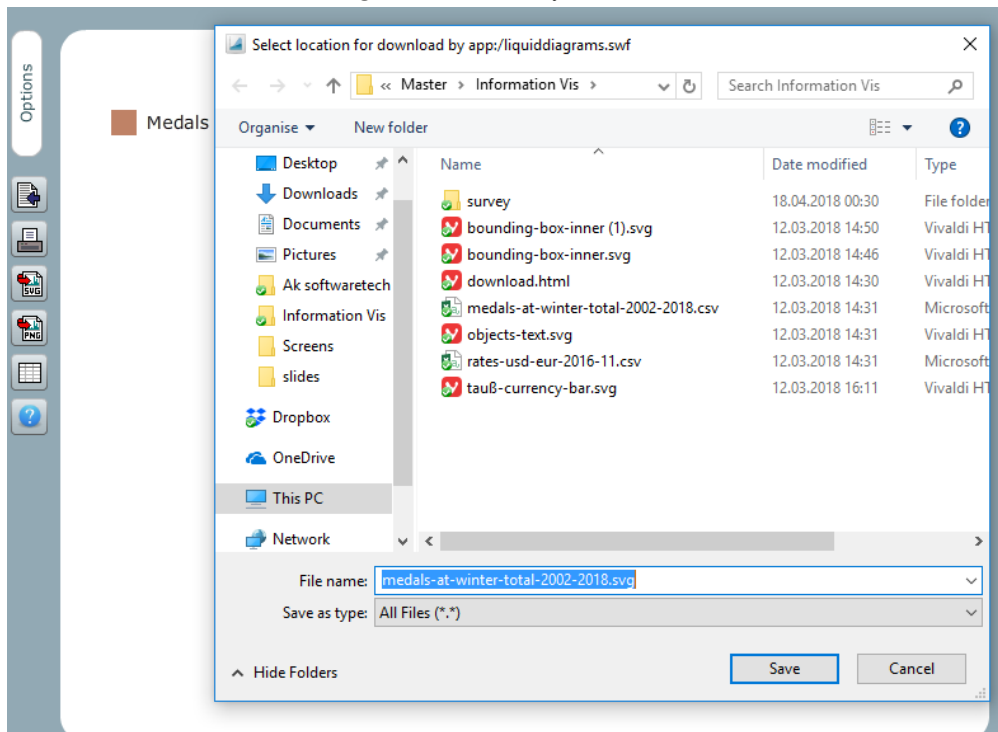


Figure 2.14: Export Finished Chart

Chapter 3

Programmable

The second type of editors we are going to discuss are *programmable* editors. These editors are more sophisticated tools that are able to produce more complex charts. They are not limited to a set of standard predefined chart types, although they can be used to output those too.

The downside to these type editors is the fact that they are generally speaking harder to master and use. They are less beginner-friendly and may take a while to get used to working with them. However, once mastered, this set of editors provide very high customisability and flexibility for creating charts.

The name for this group comes from the ability of customising charts programmatically. These editors either produce code representing charts that can be altered during run time of an application or provide an API that can be accessed in a program to produce the chart itself. This group of editors is also very well suited for developing applications with highly interactive information visualisations. In this chapter we are going to look at two tools that meet the criteria of *programmable* editors.

3.1 Vega with Lyra

The first *programmable* editor we are going to discuss is Lyra developed by the Interactive Data Lab of the University of Washington. Lyra is a hosted web application that relies on a visualisation grammar called Vega, also developed by the Interactive Data Lab of the University of Washington. Before diving into Lyra we first need to introduce Vega in order to understand what the editor is capable of and how it functions all together.

3.1.1 What Is Vega?

As mentioned above, Vega is a visualisation grammar. It specifies parameters of a chart in a declarative fashion. The specifications are written in JSON format. The generated specifications can then be used inside e.g. web applications and display powerful visualisations.

Vega is merely the language that is used to describe charts. The ecosystem built around it provides tools that let you create charts and also integrate them into other applications.

You can imagine the ecosystem consisting of two sets of applications. On the one hand you have let's call them *generator* tools that let you create Vega charts and store the output as Vega specifications in JSON format. On the other hand you have *processor* tools that take these Vega specifications as input and either visualise them or analyse the charts and the data further.

Vega specifications can get very complex. Due to this, the UW Interactive Data Lab developed a simplified grammar called Vega-Lite. Vega-Lite basically provides a layer of abstraction to Vega and can be translated to a full-blown Vega specification if required.

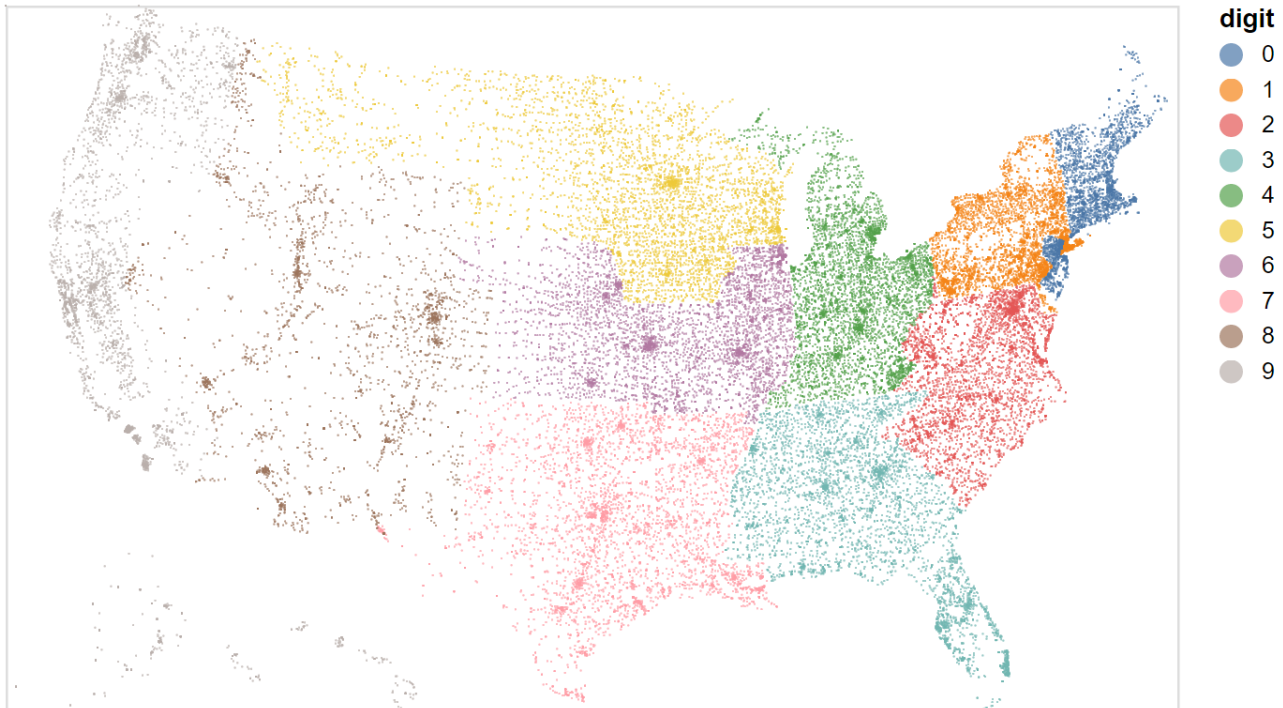


Figure 3.1: One dot per zip code in the U.S. example visualisation taken from the examples page of the official Vega-Lite website. Both, Listing 3.1 and Listing 3.2 describe the displayed visualisation.

3.1.2 Vega Vs. Vega-Lite

As already stated, Vega-Lite is basically a simplified version of Vega developed for convenience reasons since Vega in some cases can be very bloated and unpractical to work with.

In order to get a feeling for the difference between Vega and Vega-Lite, we will naively compare the number of lines of code of the respective JSON specifications. Obviously this comparison is not very professional since technically JSON can be reduced to a single line of code. However, if the file is somewhat written in readable manner, Vega stands out by its length compared to Vega-Lite.

Listings 3.1 and 3.2 show a Vega and Vega-Lite specification respectively in JSON format for the chart displayed in Figure 3.1. Comparing the length of the two specifications, one can say that the Vega specification is roughly more than three times as long as the Vega-Lite specification.

3.1.3 Lyra

Lyra is an editor built on top of Vega and Vega-Lite. It is a Vega *generator* that lets the user create complex visualisations relying on Vega through a graphical user interface and without writing any code. The created chart can then be exported as either SVG, PNG, or in form of a Vega specification in JSON format.

Lyra is open source with its source code publicly available at GitHub. However, it seems that the project has been abandoned two years ago when it was still in beta unfortunately. It is therefore almost impossible to get Lyra run locally since many dependencies are deprecated. Lyra has been hosted as a web application where it can be used. Since it is an unfinished product, there are still many bugs in the software.

```

1  {
2  "schema": "https://vega.github.io/schema/vega/v3.0.json",
3  "autosize": "pad",
4  "padding": 5,
5  "width": 500,
6  "height": 300,
7  "style": "cell",
8  "data": [
9    {
10     "name": "source_0",
11     "url": "data/zipcodes.csv",
12     "format": {
13       "type": "csv",
14       "parse": {"longitude": "number", "latitude": "number"},
15       "delimiter": ",",
16     },
17     "transform": [
18       {
19         "type": "formula",
20         "expr": "substring(datum.zip_code, 0, 1)",
21         "as": "digit"
22       },
23       {
24         "type": "geojson",
25         "fields": ["longitude", "latitude"],
26         "signal": "geojson_0"
27       },
28       {
29         "type": "geopoint",
30         "projection": "projection",
31         "fields": ["longitude", "latitude"],
32         "as": ["x", "y"]
33       }
34     ]
35   },
36 ],
37 "projections": [
38   {
39     "name": "projection",
40     "size": {"signal": "[width, height]"},
41     "fit": {"signal": "geojson_0"},
42     "type": "albersUsa"
43   }
44 ],
45 "marks": [
46   {
47     "name": "marks",
48     "type": "symbol",
49     "style": ["circle"],
50     "from": {"data": "source_0"},
51     "encode": {
52       "update": {
53         "opacity": {"value": 0.7},
54         "fill": {"scale": "color", "field": "digit"},
55         "x": {"field": "x"},
56         "y": {"field": "y"},
57         "size": {"value": 1},
58         "shape": {"value": "circle"}
59       }
60     }
61   }
62 ],
63 "scales": [
64   {
65     "name": "color",
66     "type": "ordinal",
67     "domain": {"data": "source_0", "field": "digit", "sort": true},
68     "range": "category"
69   }
70 ],
71 "legends": [
72   {
73     "fill": "color",
74     "title": "digit",
75     "encode": {
76       "symbols": {
77         "update": {"shape": {"value": "circle"}, "opacity": {"value": 0.7}}
78       }
79     }
80   }
81 ],
82 "config": {"axisY": {"minExtent": 30}}
83 }

```

Listing 3.1: Vega specification example taken from the official Vega website. It specifies the chart displayed in Figure 3.1.

```

1 {
2   "$schema": "https://vega.github.io/schema/vega-lite/v2.1.json",
3   "width": 500,
4   "height": 300,
5   "data": {
6     "url": "data/zipcodes.csv"
7   },
8   "transform": [{"calculate": "substring(datum.zip_code, 0, 1)", "as": "digit
9     "}],
9   "projection": {
10    "type": "albersUsa"
11  },
12  "mark": "circle",
13  "encoding": {
14    "longitude": {
15      "field": "longitude",
16      "type": "quantitative"
17    },
18    "latitude": {
19      "field": "latitude",
20      "type": "quantitative"
21    },
22    "size": {"value": 1},
23    "color": {"field": "digit", "type": "nominal"}
24  }
25 }

```

Listing 3.2: Vega-Lite specification example taken from the official Vega website. It specifies the chart displayed in Figure 3.1.

Figure 3.2 is a screenshot of Lyra’s graphical user interface. The panel on the left (highlighted in Figure 3.3) is used to load data and perform some transforms on the data before using the data. The panel in the middle (highlighted in Figure 3.4) is the visualisation preview panel and a bar containing tools to draw primitives (called marks) onto the canvas. The panel to the right (highlighted in Figure 3.5) holds all the elements of the visualisation along with their properties. The elements are organized in layers. The bar at the bottom of the interface is used to undo or redo changes of the visualisation and also to export and save the chart. As mentioned before, supported export formats are SVG, PNG, and Vega in JSON format.

Lyra’s graphical user interface is very intuitive. Everything works with drag and drop functionality. The user basically creates marks and binds data values to the marks’ properties by dragging and dropping the desired values from the panel to the left over to the panel to the right.

However Lyra is capable of more complex charts such as the visualisation of Napoleon’s march to Moscow seen in Figure 3.7.

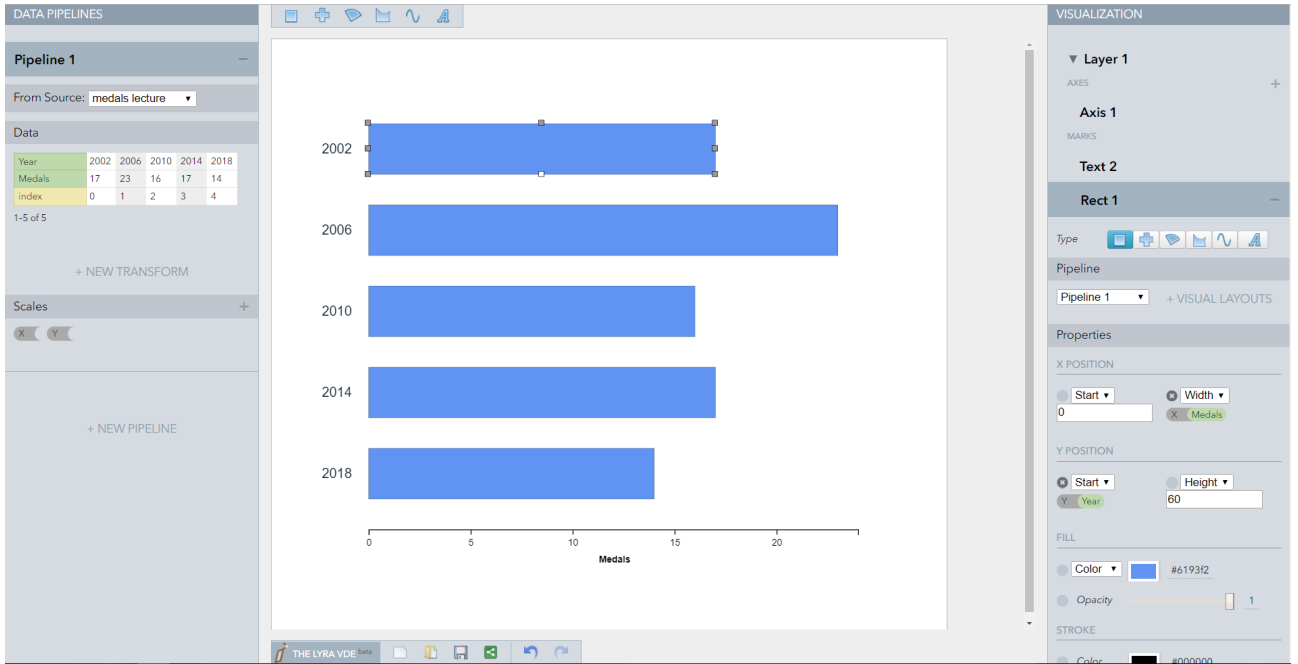


Figure 3.2: Lyra’s graphical user interface.

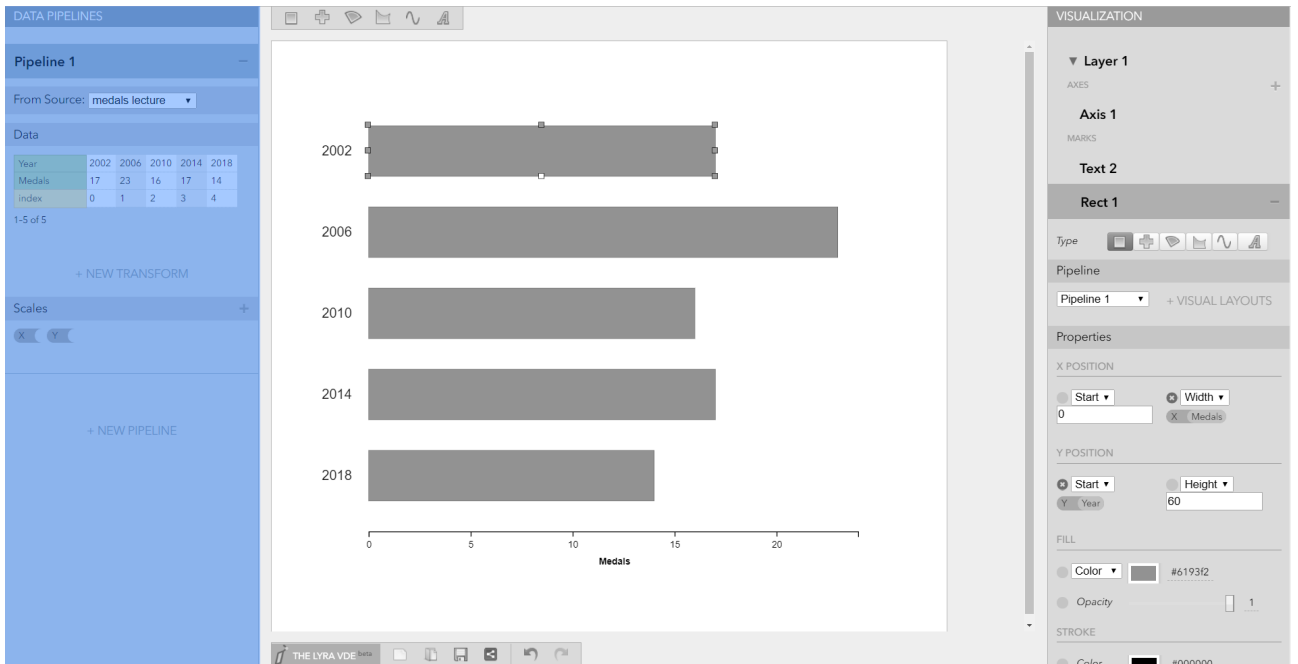


Figure 3.3: The highlighted area is Lyra’s data pipeline panel. It is used to load data and perform certain transforms on the data before being used in the visualisation.

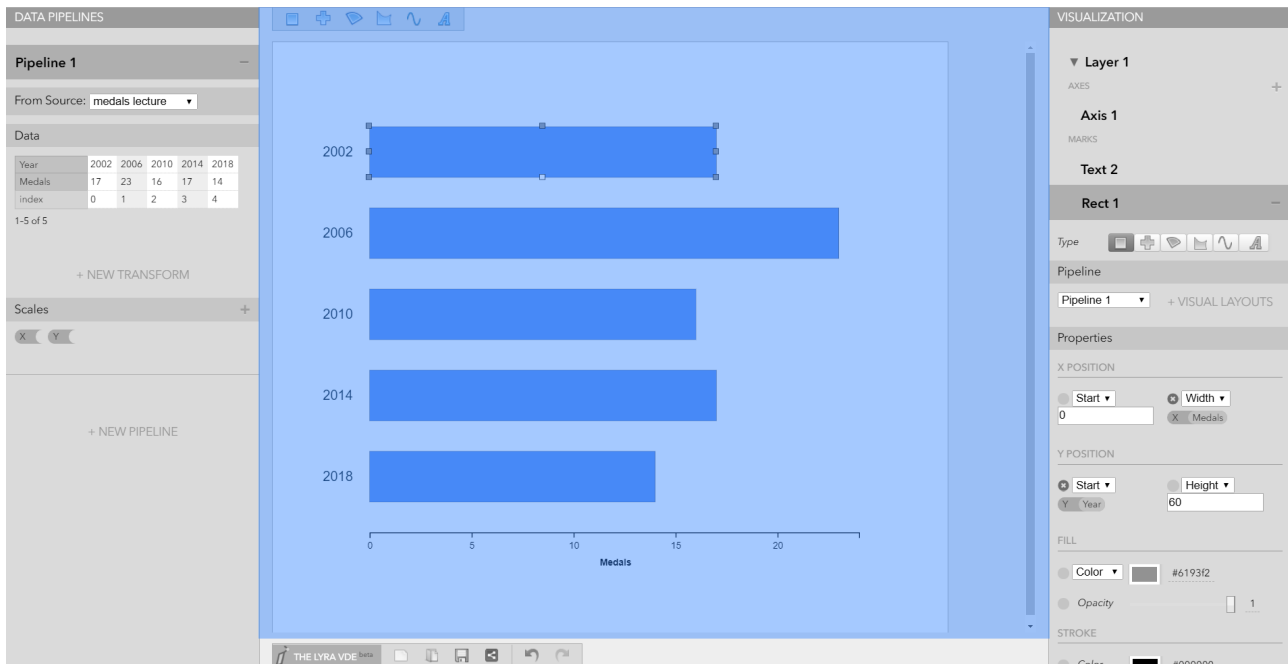


Figure 3.4: The highlighted area is Lyra's visualisation preview panel and the visualisation marks bar containing tools to draw primitives onto the canvas.

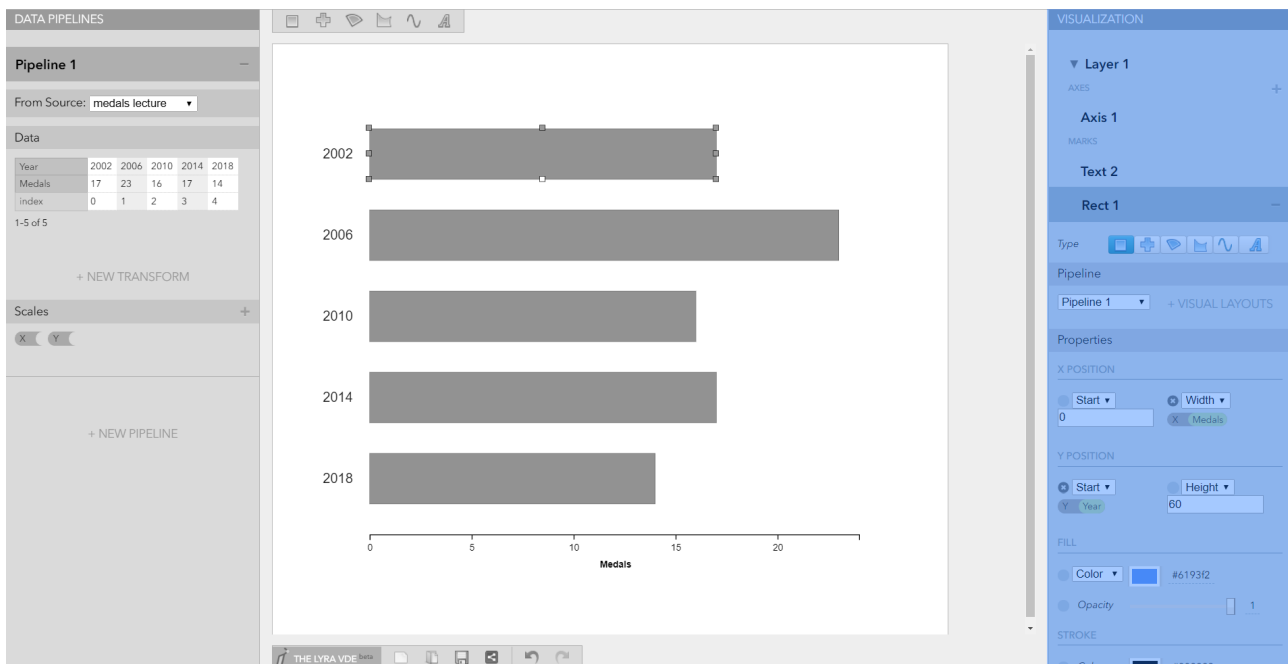


Figure 3.5: The highlighted area is Lyra's layers and properties panel. It holds all the elements and their properties. The elements are organized in layers.

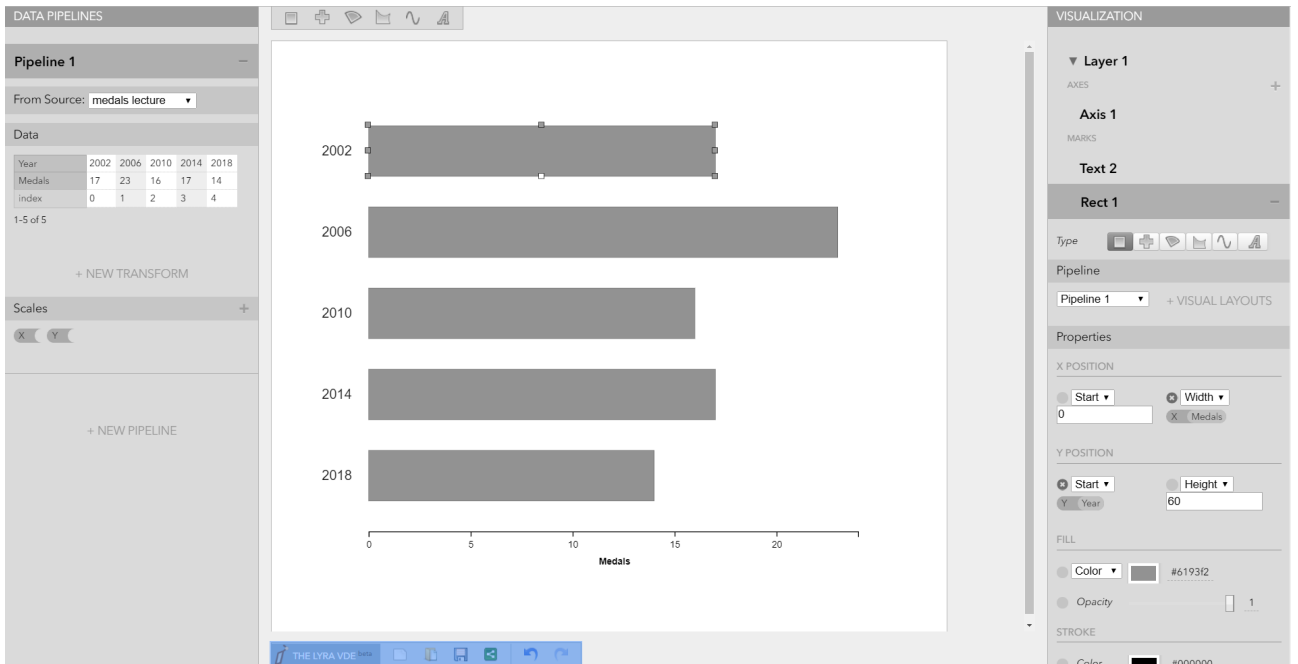


Figure 3.6: The highlighted area is Lyra’s file options bar. It contains buttons to undo or redo changes made to the visualisation, as well as to export the chart to various export formats.

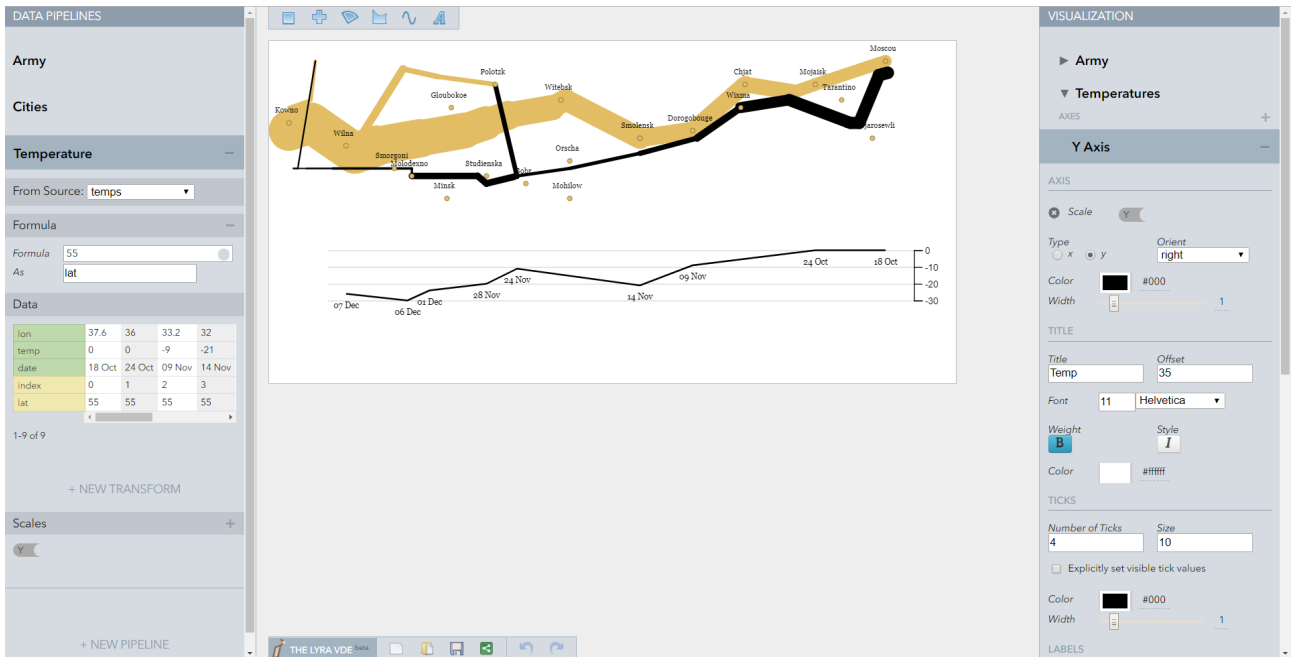


Figure 3.7: Lyra is capable of more sophisticated data visualisations such as the visualisation of Napoleon’s march to Moscow by Charles Minard, 1869.

3.1.4 Evaluation Of Lyra

Lyra is a very sophisticated editor giving the user a lot of freedom when it comes to creating custom, powerful visualisations. The obvious downside is that the project apparently has been abandoned roughly two years ago. Since then, development didn't progress, leaving an unfinished product behind that struggles with lots of bugs. According to the metrics defined above, Lyra has been rated as follows:

1. *Platform*: Lyra is a hosted web application. Since lots of the project's dependencies are deprecated, running Lyra locally is unreasonably hard. It is not worth it diving into resolving all the node modules that need to be updated or replaced in order to work properly again.
2. *Pricing and license*: Lyra is open source and its source code publicly available on GitHub.
3. *Customisability*: It provides very high customisability through its many options and properties that the user may change.
4. *Range of chart types*: The range of chart types is only limited by the user's creativity, thus unlimited.
5. *Usability*: Lyra's usability seems initially intuitive and good, however, since the editor doesn't provide proper documentation and still suffers from lots of bugs, its usability is bad. If the product would have been finished, it might be different.
6. *Export formats*: The produced chart can be exported as SVG, PNG, or Vega (JSON).

3.2 amCharts(Live)

Our section editor in this class is the live editor from amCharts. This is a web hosted application which provides a graphical user interface for the creation of charts using the amCharts JavaScript library.

3.2.1 About

This editor is developed by the company with the same name, *amCharts*. This is a very small company which developed the editor itself and also the language its using. It was founded in 2004 and is still enhancing and improving amCharts.

3.2.2 Workflow

Similar to the editors in the *Plug and Play* section of this survey we will highlight the workflow of amCharts by making a bar-chart. While this example is of course very simple and does not really show the possibilities of amCharts it enables us to make better comparisons between the different kind of editors.

1. Choose the chart-type (see Figure 3.8)
2. Import and check data (see Figure 3.9)
3. Set the category field in the general section(see Figure 3.10)
4. Set the value field in the graph section (see Figure 3.11)
5. Export the graph either as embedded HTML(see Figure 3.12) or as the JavaScript Code (see Figure 3.13)

It should be mentioned that editor also gives the option to always directly modify the JavaScript Code. But since this would blow up the scope of this survey we will not give an explanation on how to use it. This combination does give the user the possibility to quickly draft simple charts but it also makes it possible to make very interactive and complicated graphs.

3.2.3 Evaluation Of amCharts

The combination of the graphical editor and the possibility to directly write in JavaScript make amCharts to a very powerful tool for a wide range of applications. While the editor needs some getting used to it is still fairly useful for creating simple graphs or for creating a base to work on with the JavaScript directly. The pricing also seems to be very reasonable as they offer a free (branded)version for non commercial use which has the full functionality. And the most expensive license even includes the source code for even enhanced customisability. Following is our evaluation of amCharts according to our predefined metrics:

1. *Platform*: hosted web application / JavaScript
2. *Pricing and license*: free(branded) - 4500\$
3. *Customisability*: high
4. *Range of chart types*: high
5. *Usability*: medium
6. *Export formats*: JavaScript, embedded HTML

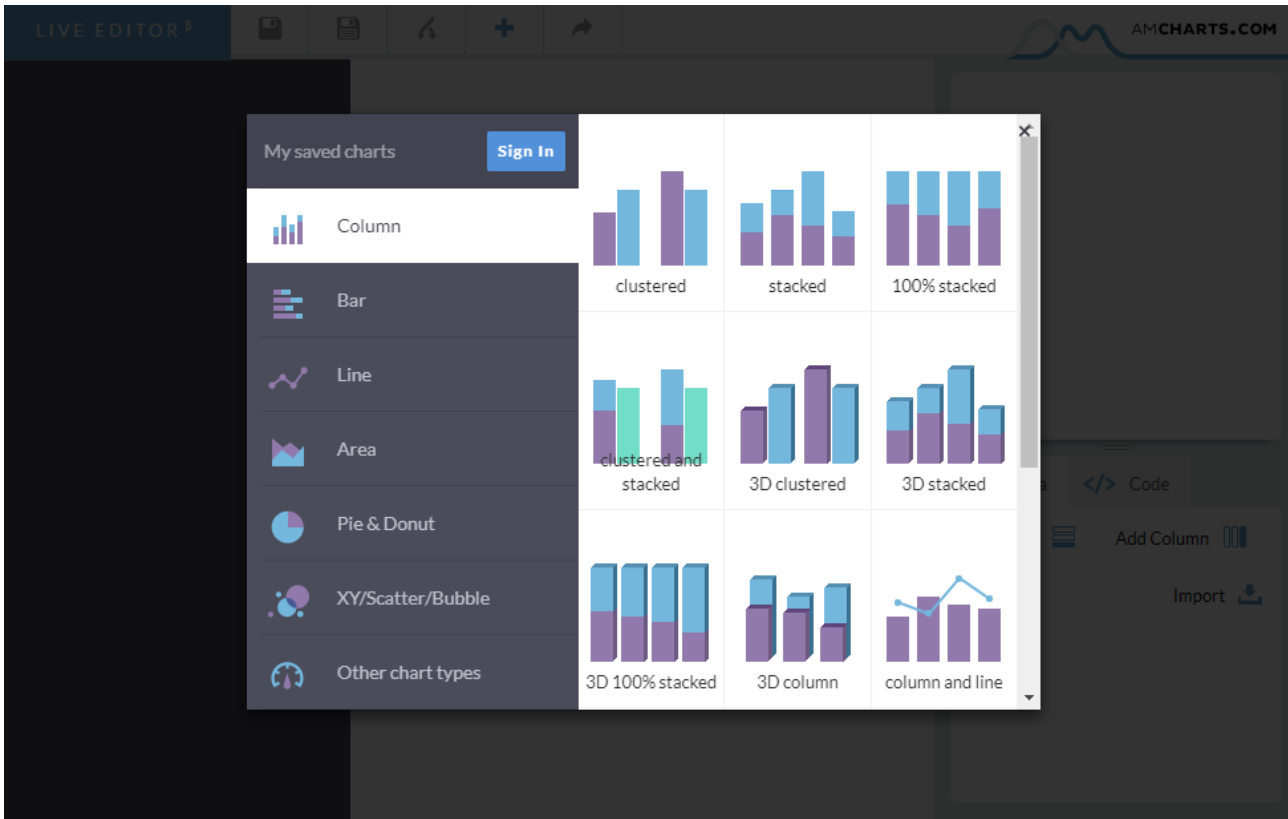


Figure 3.8: Select the type of chart.

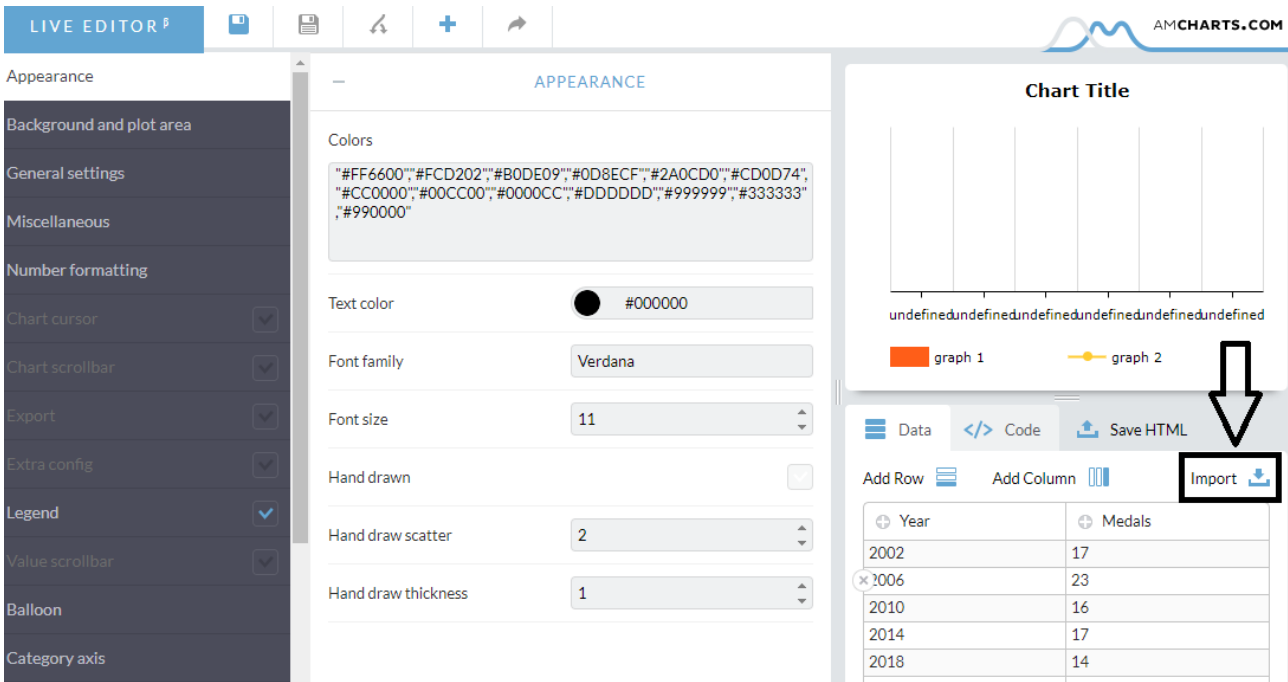


Figure 3.9: Import and check the data-set.

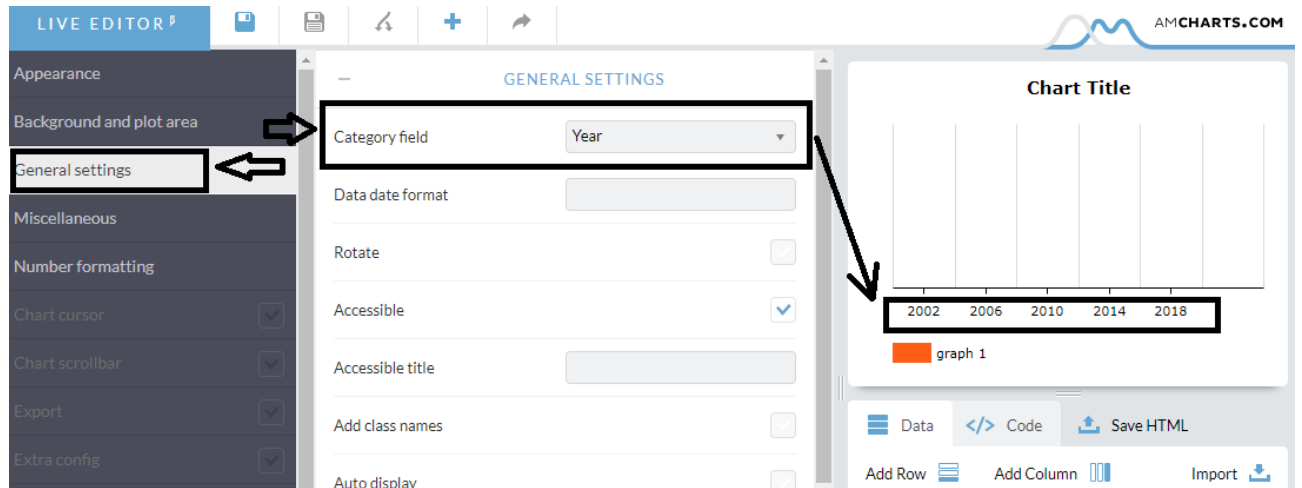


Figure 3.10: Set category in general settings.

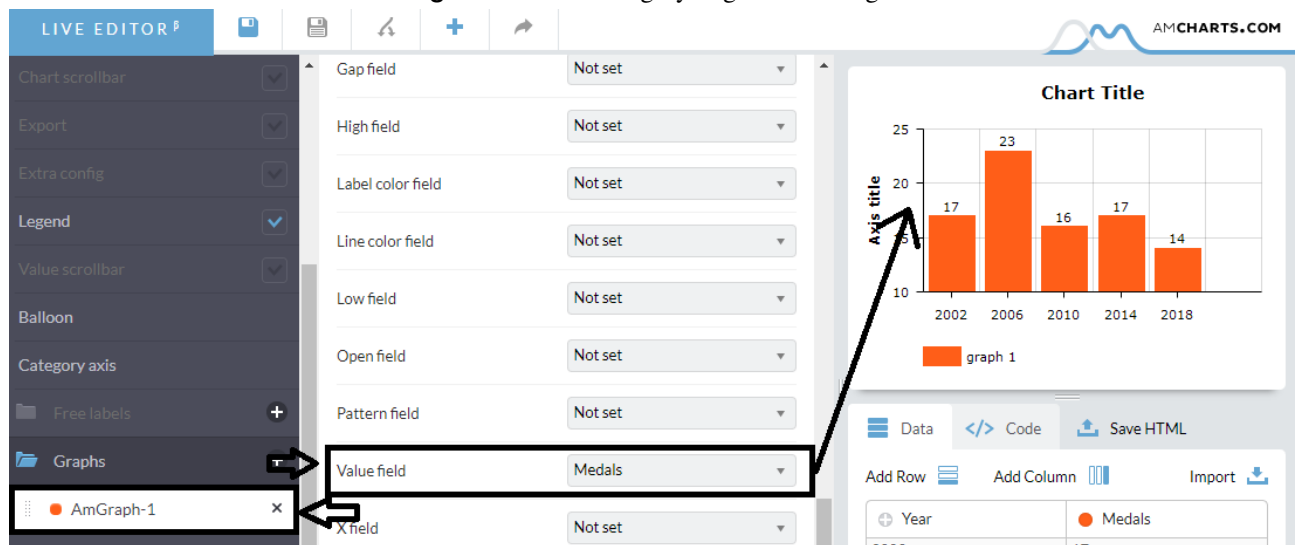


Figure 3.11: Set value field in graph section.

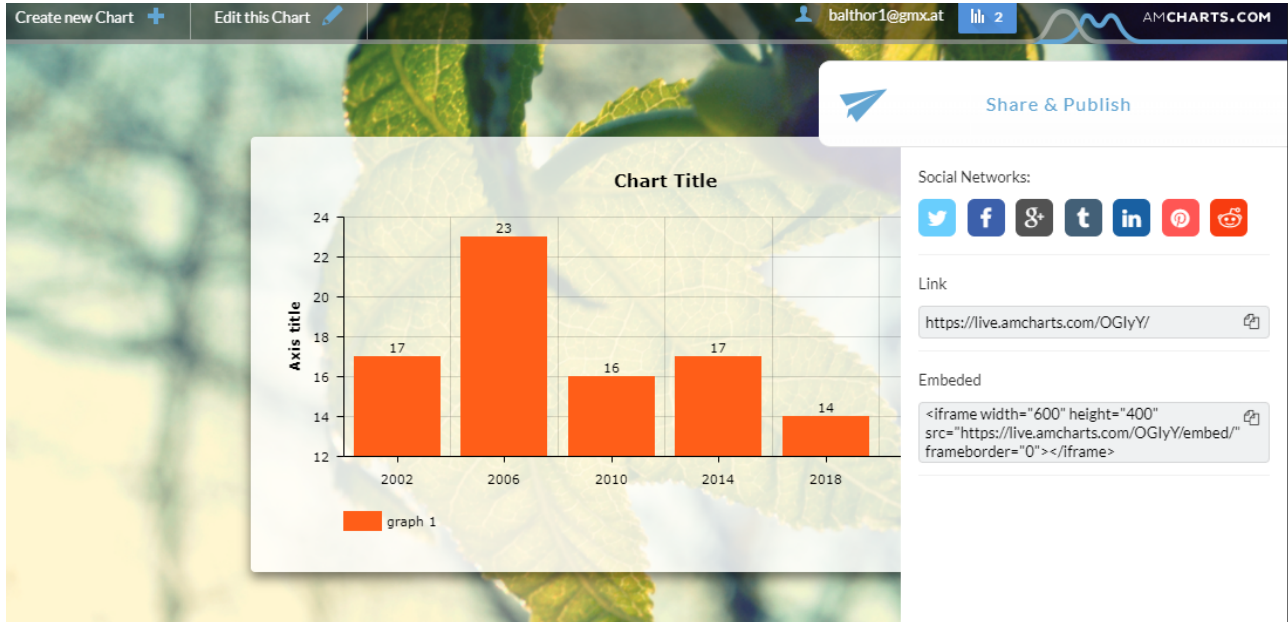


Figure 3.12: Export As Embedded HTML

```

<!DOCTYPE html>
<html>
  <head>
    <title>column and line | amCharts</title>
    <meta name="description" content="chart created using amCharts live editor" />

    <!-- amCharts javascript sources -->
    <script type="text/javascript" src="https://www.amcharts.com/lib/3/amcharts.js"></script>
    <script type="text/javascript" src="https://www.amcharts.com/lib/3/serial.js"></script>

    <!-- amCharts javascript code -->
    <script type="text/javascript">
      AmCharts.makeChart("chartdiv",
        {
          "type": "serial",
          "categoryField": "Year",
          "startDuration": 1,
          "backgroundAlpha": 0.8,
          "categoryAxis": {
            "gridPosition": "start"
          },
          "trendLines": [],

```

At the bottom of the code editor, there are two buttons: 'Copy to clipboard' and 'Save to filesystem'.

Figure 3.13: Export As Included JavaScript

Chapter 4

Data Guided Drawing

We formed the group of “Data-Guided Drawing Tools” since the next three tools are a bit different than the other tools before. One fundamental part of the last tools were predefined chart patterns. In every tool or programming framework sooner or later the user was able to choose between more or less chart types like “Bar-Chart”, “Scatter-Plot” and so on. Data-guided drawing tools don’t feature these patterns. Instead the user must create and draw the whole chart by his own. This is a completely new approach and allows the user to work more freely and more creative.

4.1 Hyecoo

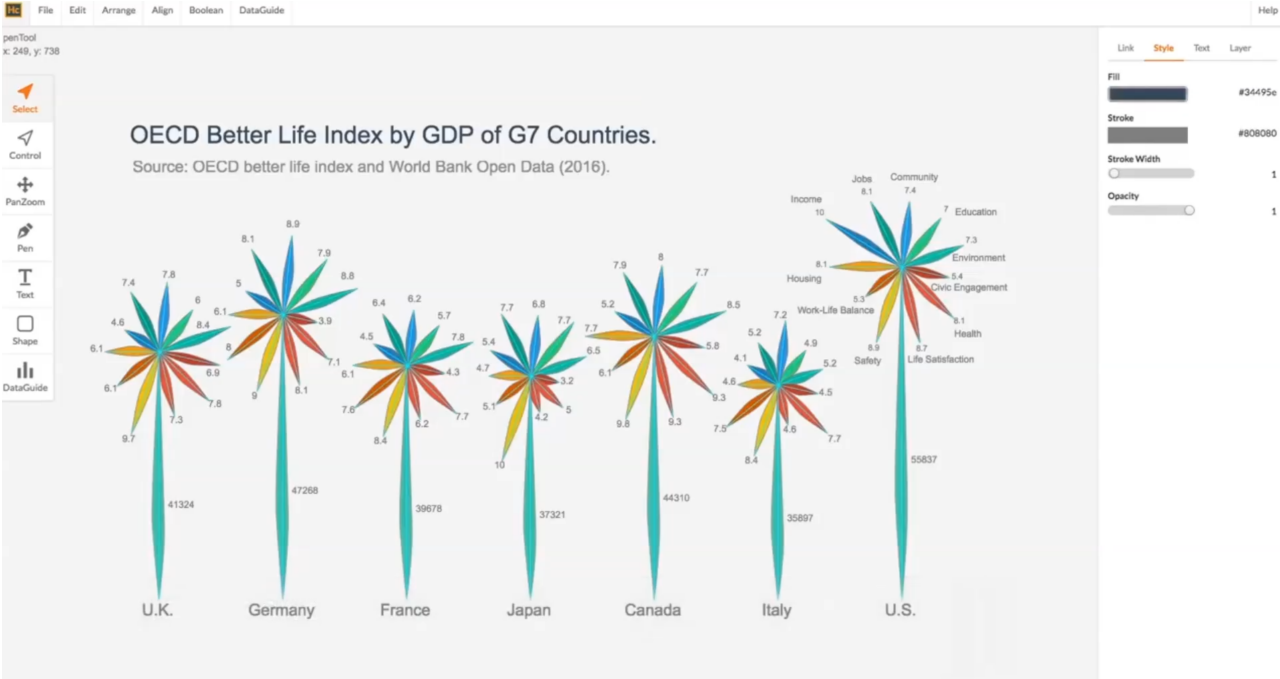


Figure 4.1: An Example of a custom chart released in Kim et al. [2017].

Hyecoo is a web-hosted application created by Nam Wook Kim, a Ph.D. Student from Harvard, who released and used it in his paper on data-driven guides [Kim et al. [2017]].

4.1.1 About

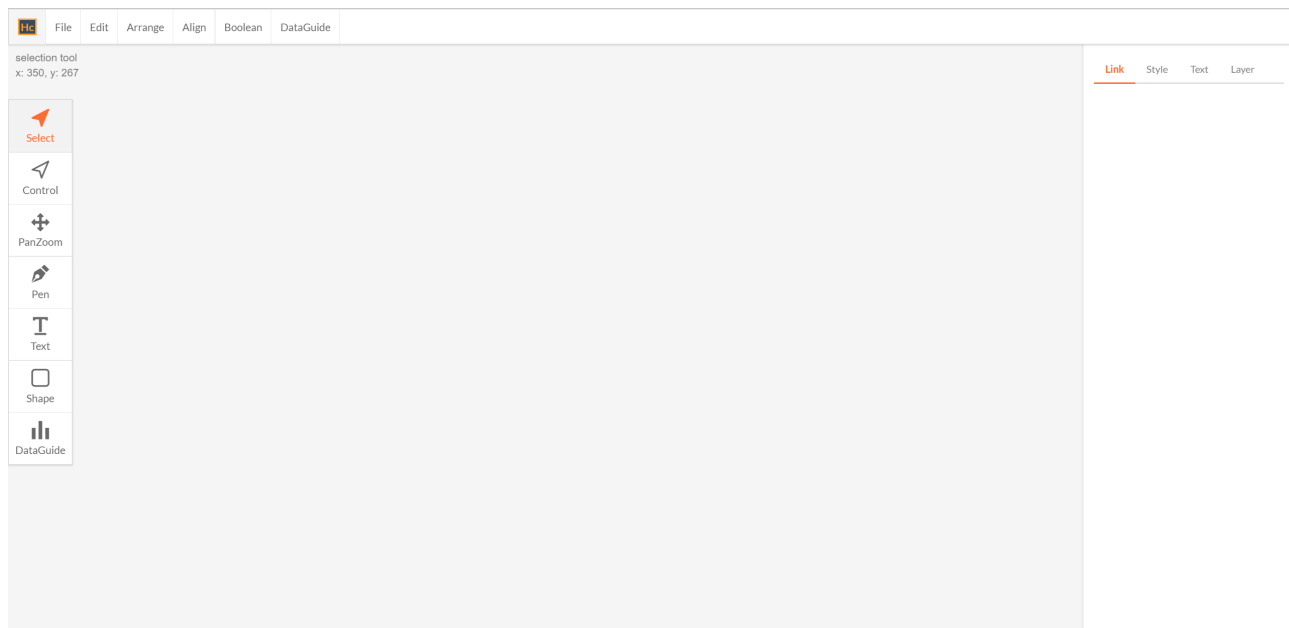


Figure 4.2: An empty Hyecoo project. Tools are located on the left side, properties of e.g. shapes are located on the right side.

Basically, Hyecoo looks and feels like a lightweight SVG drawing tool, but with the ability to insert data guides. This can be done by importing data sets, consisting out of exactly one feature. With these data guides, one can draw predefined shapes like rectangles or custom shapes, and scale them automatically according to these lines. The big advantage compared to usual SVG drawing tools like “Illustrator” is that Hyecoo can then repeat this scaling along a data line for each value in the data set.

4.1.2 Features Of Hyecoo

The structure of Hyecoo is very simple and intuitive. The tools of Hyecoo are located on the left side. They consist out of different zoom and movement tools as well as drawing tools. The user can choose between drawing shapes, e.g. rectangles, drawing paths and texts. The last and most important tool is the “DataGuide” tool, which imports a data set and draws the according data guides onto the canvas. Additional settings for the project can be adjusted on the right side. The first one of these settings are the “link settings”. After creating the data guides and after applying shapes to them, links are created between the shapes and the guides. All these links appear in this section. “Style” and “Text” are for design adjustments like colouring of the shapes and font type and size of the labels. “Layer” is for hiding different parts of the project, similar to the “Photoshop” layer panel.

4.1.3 Workflow

To show the basic workflow, we will again create a simple bar chart which illustrates the winter Olympic medals of Austria from the last 5 tournaments. The first step is the import of the date set, which can be done by clicking on the “DataGuide” tool. After inserting the data set the data guides appear on the canvas and can be moved and scaled.

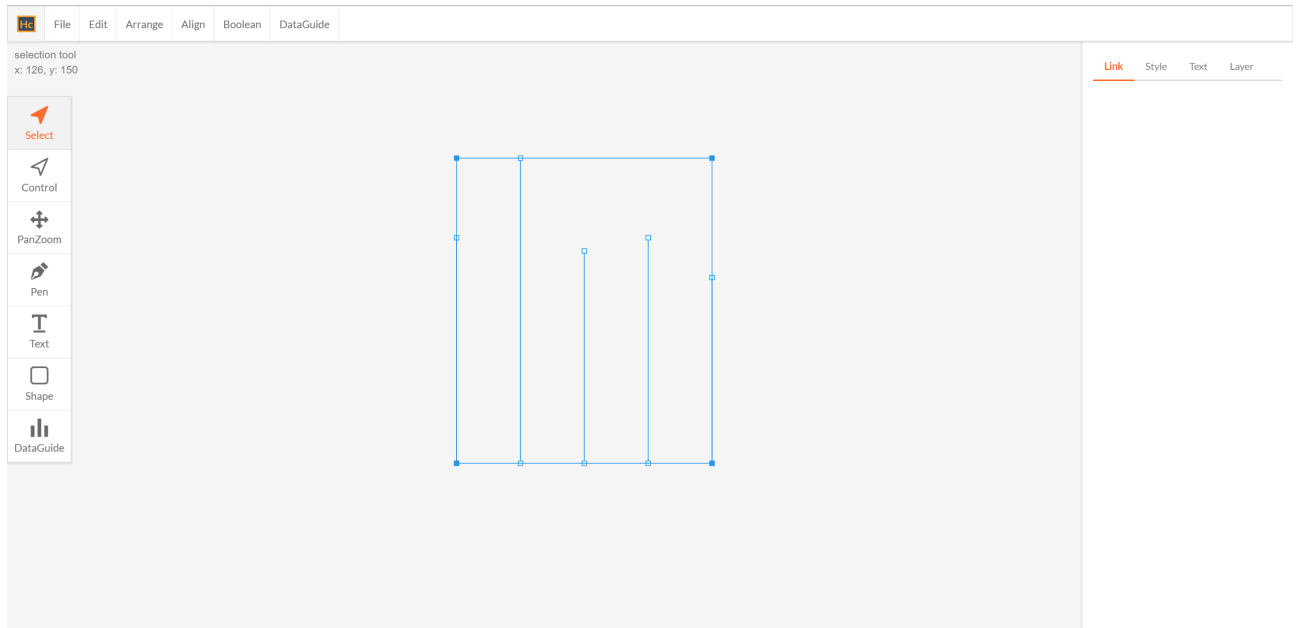


Figure 4.3: After importing the data set, the data guides get added to the canvas.

Each line of the data guides represents one year of the Austrian Olympic medal earnings. To make a bar chart we now need a shape which fits on one of the data guides. We could use the “Pen” tool to make a custom shape, or we can use the “Shape” tool to draw a rectangle.

After moving and scaling the rectangle to fit one of the data lines, a link is created between the shape and the data guide. As default, the data guide moves the shape if it changes, which is not the behaviour we want to achieve. For a bar chart we want the data line to stretch the rectangle which can be done by selecting “deforms” in the “Link” section on the right. As soon as we are done with styling we can now repeat the link by right-clicking the shape and selecting “Repeat Link”.

For each data guide a rectangle gets created. To finish the bar chart we can now add labels by right-clicking a shape and selecting “Generate Labels”. Finally, we can hide the data lines in the layers menu and export the chart to a SVG file.

4.1.4 Evaluation Of Hyecoo

Hyecoo was created by an Ph.D. student called Nam Wook Kim, during research on data-driven guides [Kim et al. [2017]]. Therefore, Hyecoo is not a finished product. It is still in the alpha version and it is not sure if the application will ever get another update. The basic idea of Hyecoo was to give the user complete freedom while visualising data and Hyecoo does a good job in that. There are basically no limitations for the user and especially the combination of a data visualisation tool and an SVG editor makes it to a very unique product. We chose a bar chart to show the basic workflow of Hyecoo and to compare it with the other tools of our survey. This is usually not the field of application where we would use Hyecoo. We would use it for creating complex and stylish infographics which could be used for websites, blogs and other cases where a unique design and a high memorability is needed. Unfortunately it lacks useful features, e.g. magnetic/sticky behaviour of the shapes, which makes this application not useful for doing serious work efficiently. The overall usability is good because of its intuitive design and clear structure, but the number of bugs in the application makes it very uncomfortable to work with. According to our metrics Hyecoo has been rated as follows:

1. *Platform:* hosted web application

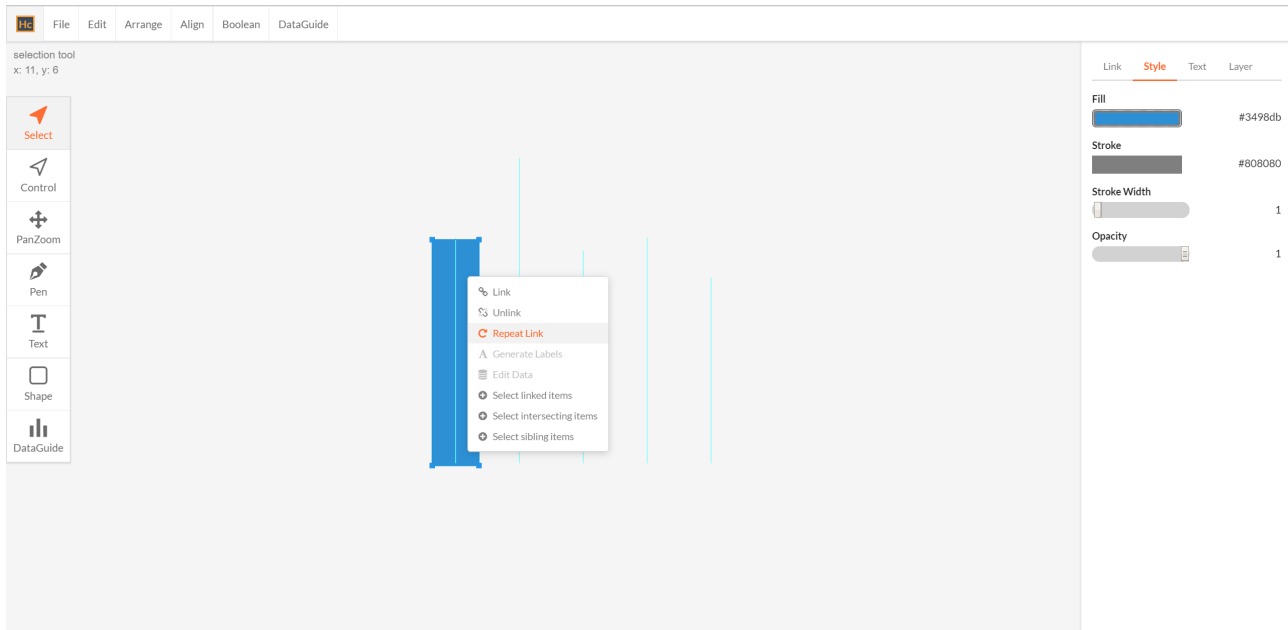


Figure 4.4: The rectangle gets moved and scaled to fit the data guide. This link gets repeated to form a bar chart.

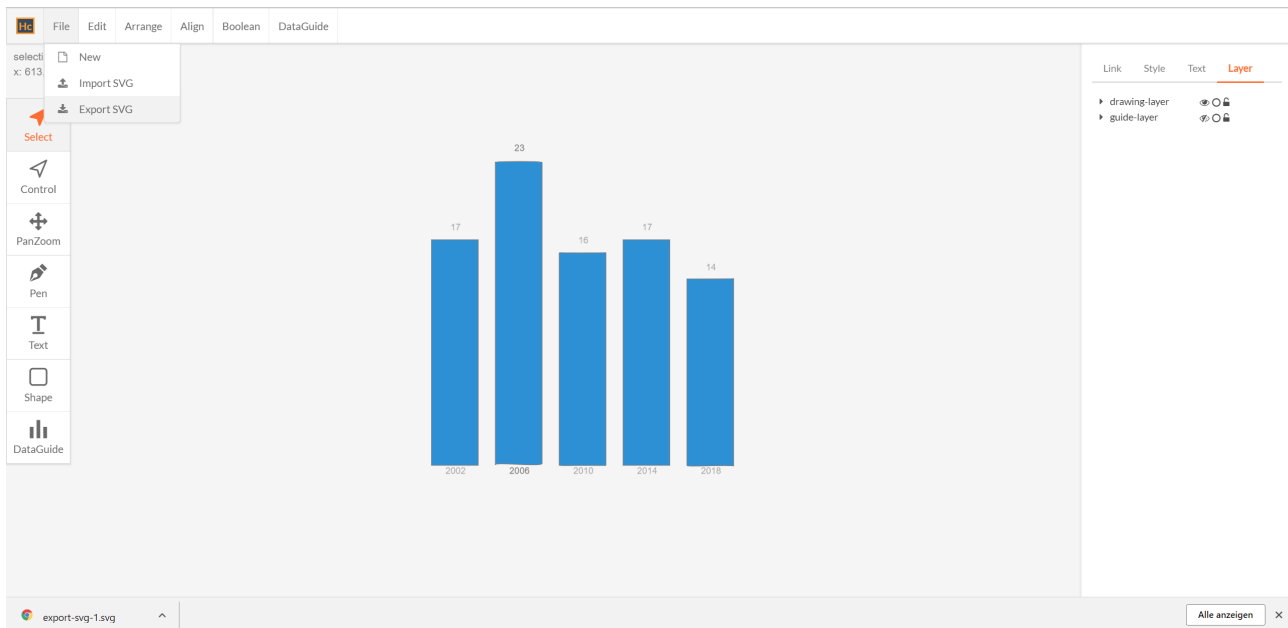


Figure 4.5: After finalizing the chart, we export it as SVG.

2. *Pricing and license:* alpha version
3. *Customisability:* very high
4. *Range of chart types:* unlimited
5. *Usability:* good but buggy
6. *Export formats:* SVG

4.2 Data Illustrator

4.2.1 About

“Data Illustrator” is another web hosted application used for data guided drawing. It is made by “Adobe” and is still in research. The first look on their homepage (<http://data-illustrator.com/>) already tells that this is going to be a big program build with a lot of resources and in a good quality. They feature a big gallery, where they show for what chart types “Data Illustrator” should be used and with a video for each example they even show exactly how to do it. The results are amazing and have an outstanding quality.

4.2.2 Features Of Data Illustrator

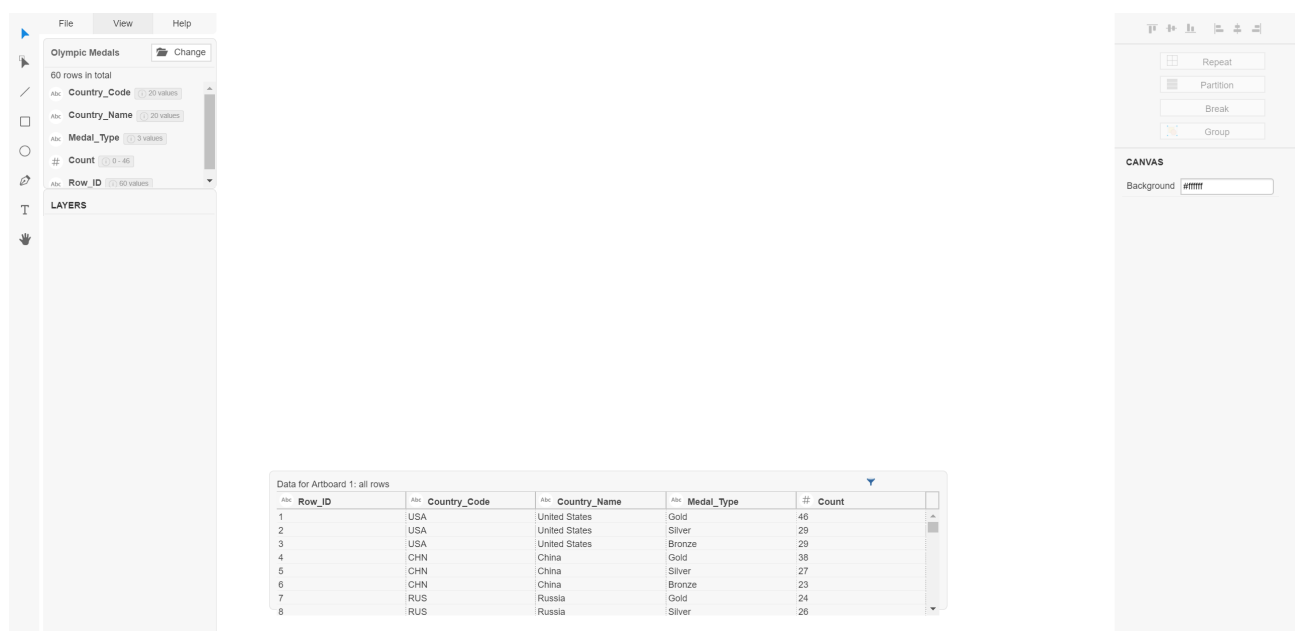


Figure 4.6: After importing the data set, it appears on the bottom of the screen as a data sheet.

“Data Illustrator” has a typical structure of interface. On the left side one can find all available tools such as select- and draw-tools including, like “Hyecoo”, a rectangle, ellipse, line and path tool. Right next to the toolbar is the “data overview”. Here one can find all the feature names of the data set. This means that “Data Illustrator” can not only handle data sets with one feature but with multiple features, which is one of the biggest advantages over other tools. Underneath this box the “Layers”-box is located. After drawing on the canvas a tree-structure of the illustration gets created and shown in this box. On the bottom of the screen one can find the complete data set formatted like an excel sheet. The properties of the currently selected shape can be seen and adjusted on the right side. Here one can do design refinements and create data bindings between design properties and features of the data set.

4.2.3 Workflow

To give a quick and simple overview we are again showing the workflow by creating a bar-chart. In this case we used a sample data-set with the amount of Olympic medals of different countries over several years. Additionally, the data set also includes the number of each medal type (gold, silver, bronze).



Figure 4.8: After clicking on the "Repeat" button, a rectangle appears for each country.

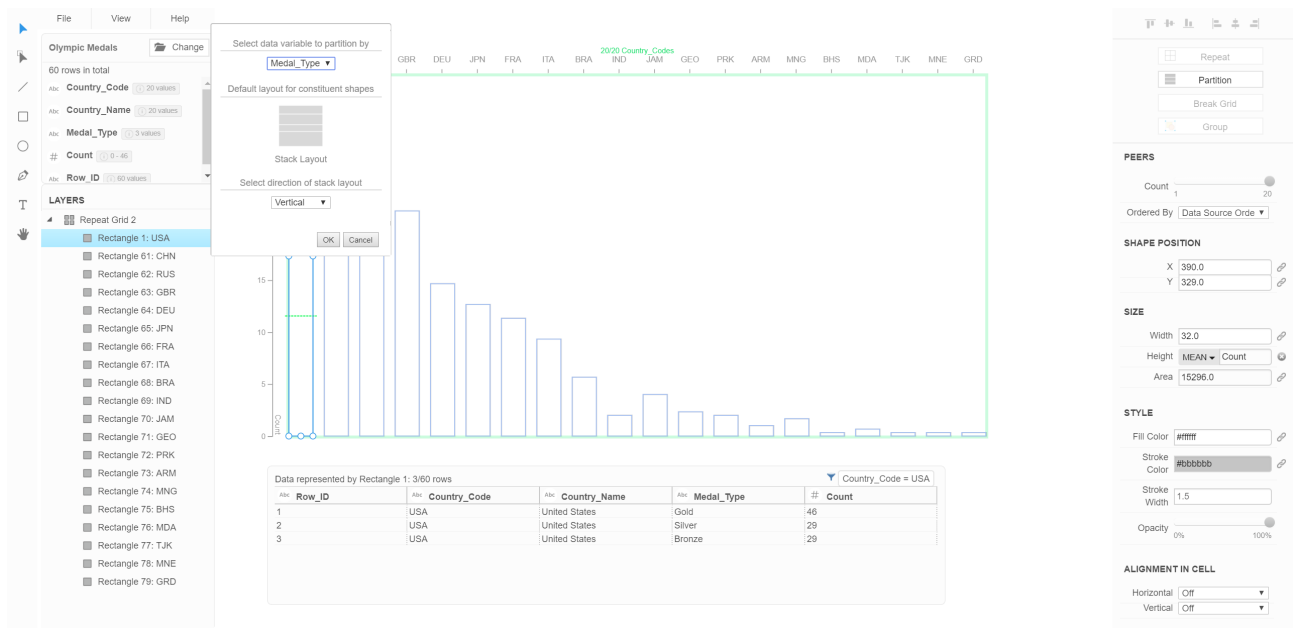


Figure 4.9: After clicking on the "Partition" button, the rectangles get split for the selected value.

the early state of the application more formats may follow. Data Illustrator has been rated as follows according to our metrics:

1. Platform: hosted web application
2. Pricing and license: still in research
3. Customisability: very high

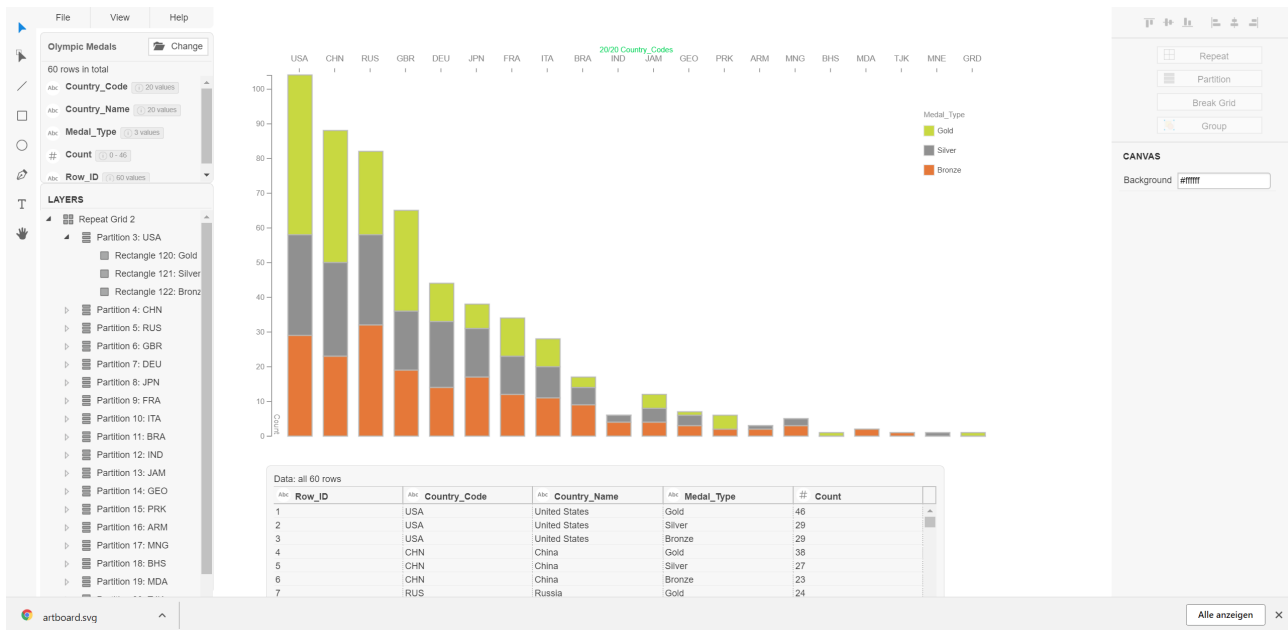


Figure 4.10: The final result of the medal data set visualized by an bar chart.

4. *Range of chart types:* unlimited

5. *Usability:* very good

6. *Export formats:* SVG

4.3 iVisDesigner

4.3.1 About

As the last editor in the category of *Data Guided Drawing* we look at iVisDesigner. The development of this editor started as an undergraduate thesis project of Donghao Ren. It is open source and can be found on Github [*iVisDesigner Github* [2018]]. As the previous two editors, iVisDesigner allows the mapping and snapping of given data to any drawings done in the editor. Its goal is to offer high customization whilst allowing users to visualize complex datasets [*iVisDesigner Website* [2018]].

Currently it is in version *0.10alpha*.

4.3.2 Workflow

Before we start on a practical example we first introduce the main windows of the editor (see Figure 4.11).

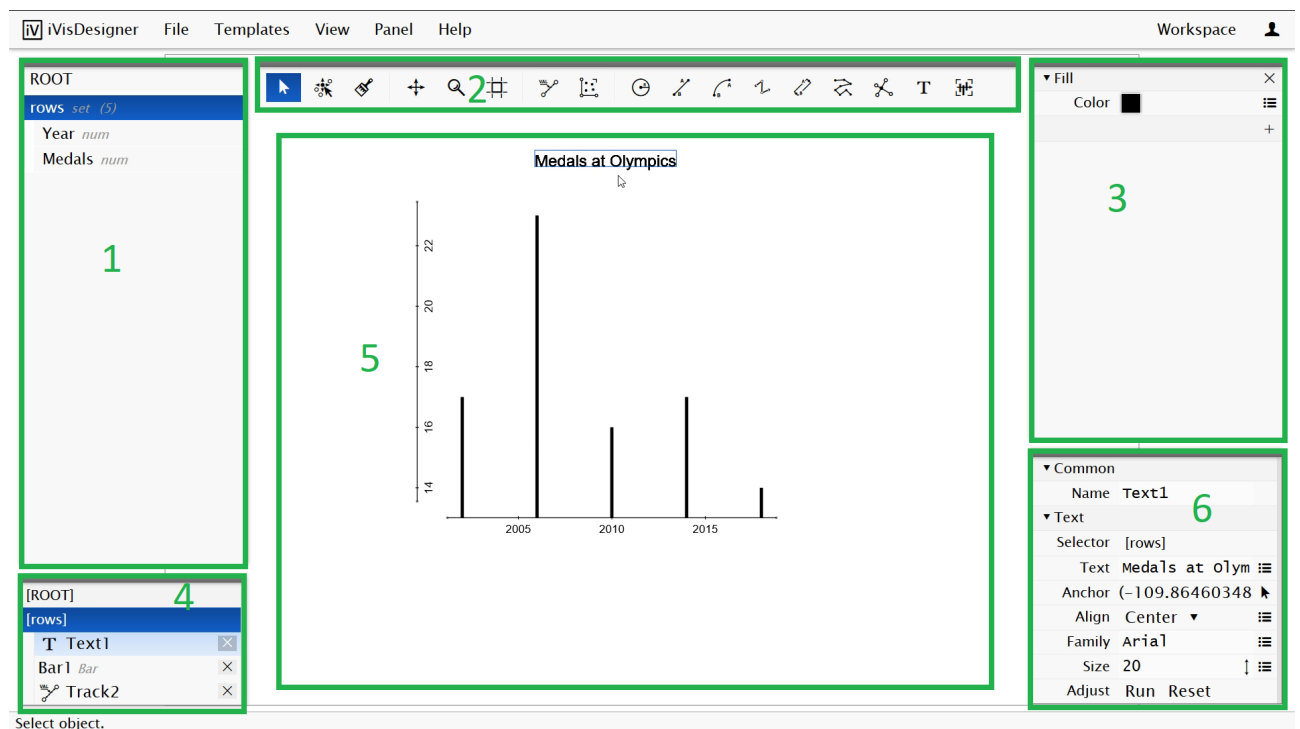


Figure 4.11: 1: The imported data, sorted by columns. 2: The tool window. 3: Fill window with color options. 4: The chart components. 5: The main drawing window. 6: The component window containing all the customisability options of the selected chart component.

We will demonstrate the workflow of iVisDesigner by creating a barchart. As done previously we are going to use the dataset of Austria’s medals at the Winter Olympics of the last 5 years. In iVisDesigner creating the barchart entails the following steps:

1. Copy and paste the dataset into iVisDesigner (see Figure 4.12).
2. Create the axes needed for a barchart (see Figure 4.13).

3. After adding a scatterplot field as a helper, select "ROWS" on the left and create the bars (see Figure 4.14). Note that you only need to do this once, as iVisDesigner recognizes that there exist 5 rows and creates 5 bars accordingly.
4. Post-process the barchart by adding labels and adjusting color, font, label placement, etc (see Figure 4.15).
5. Export the chart (see Figure 4.16).

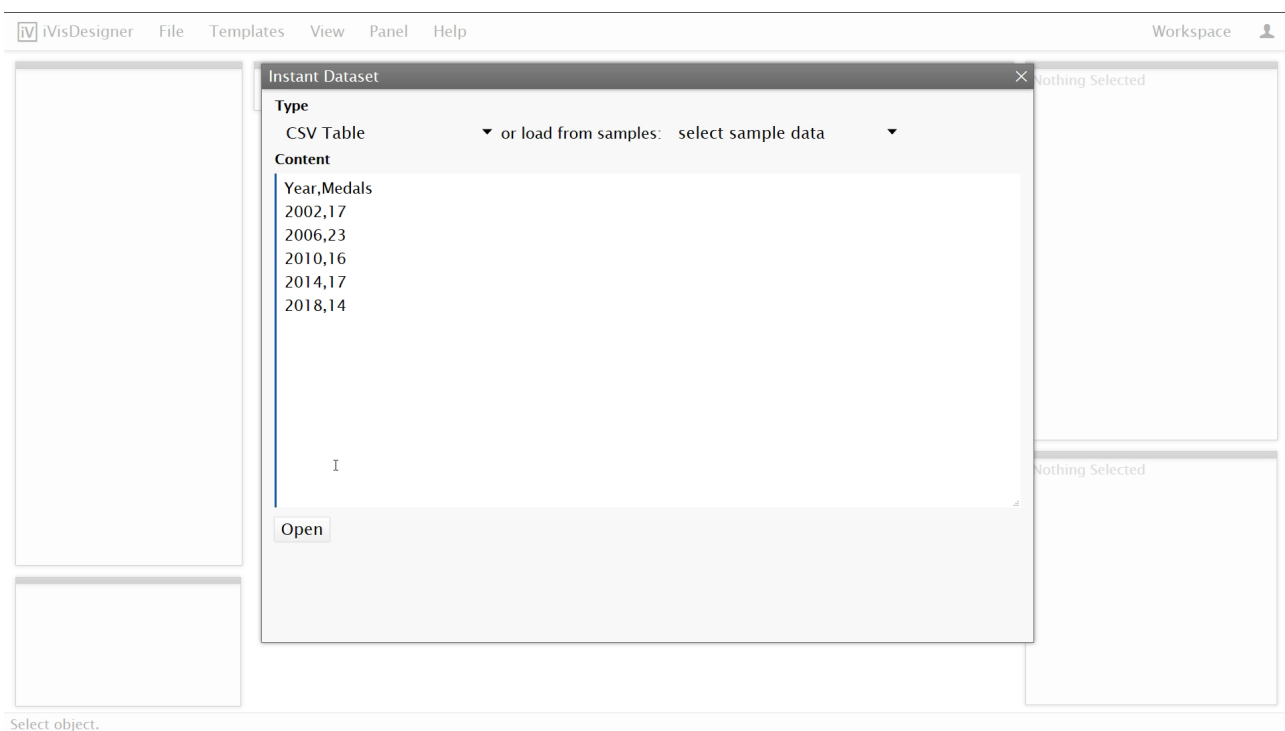


Figure 4.12: Copy and paste the dataset into iVisDesigner.

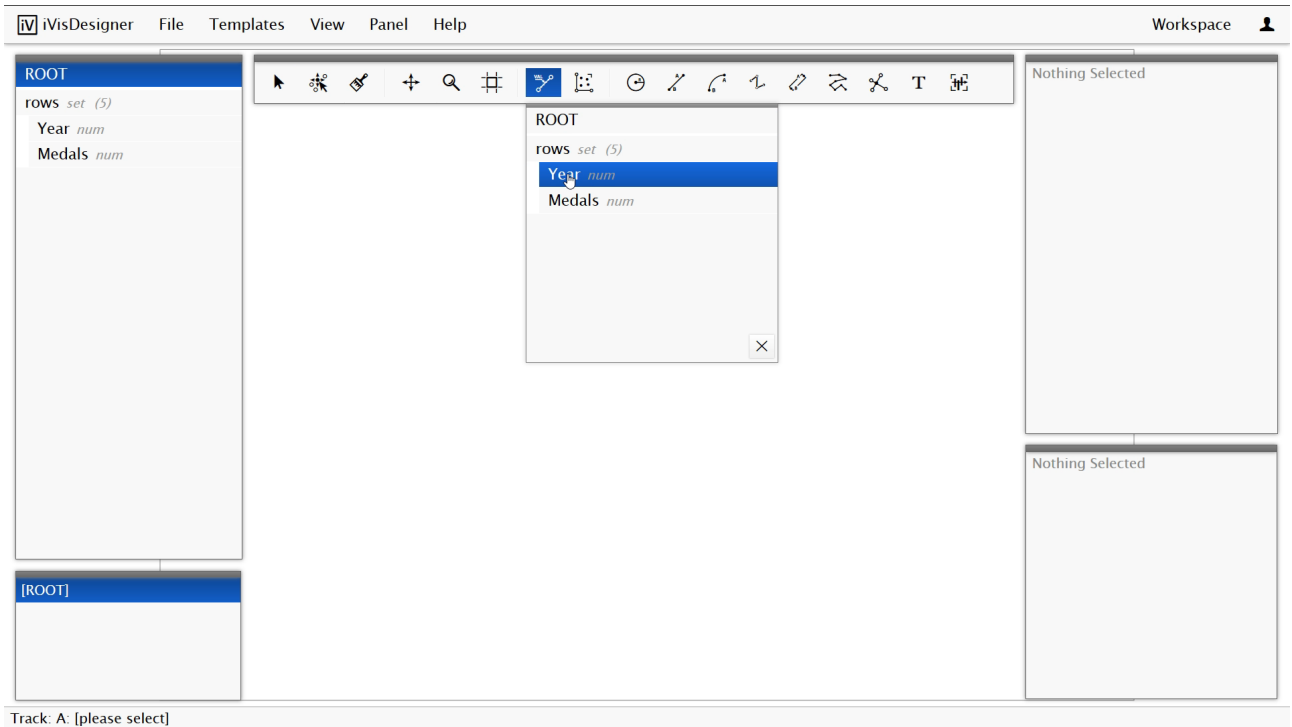


Figure 4.13: Creating The Axes

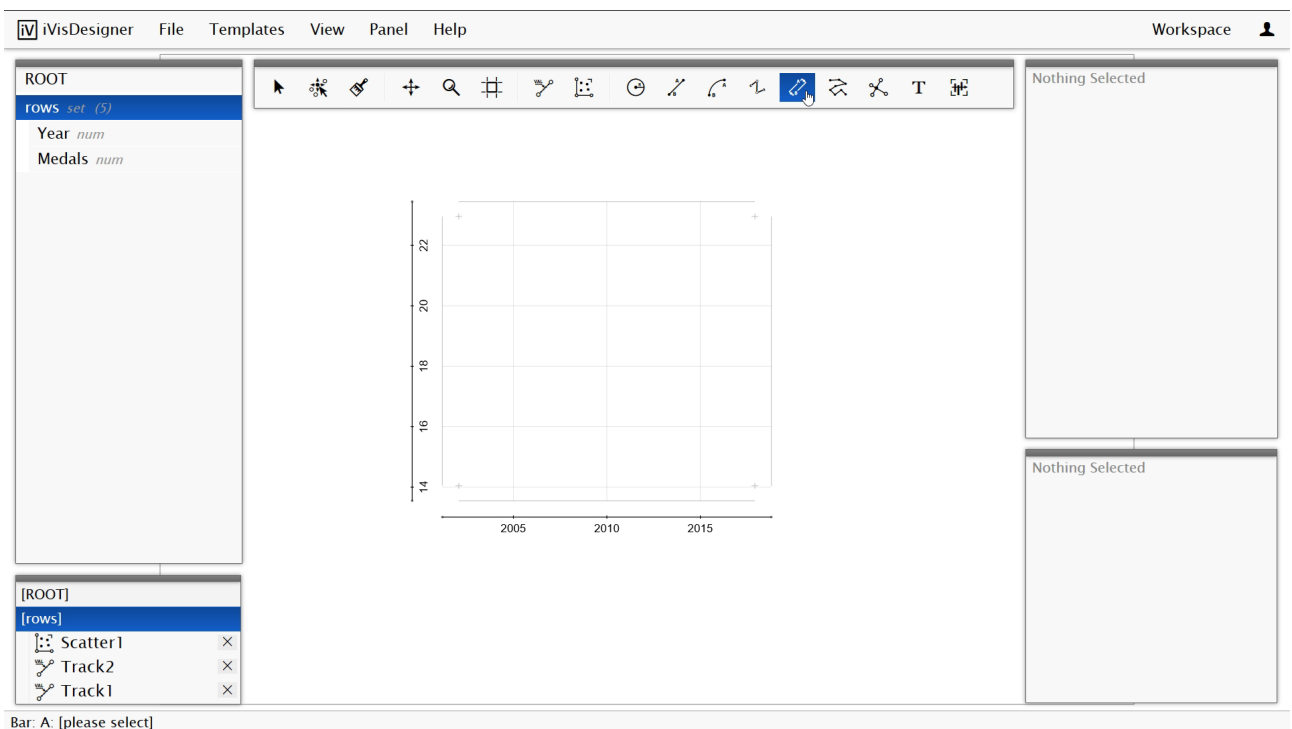
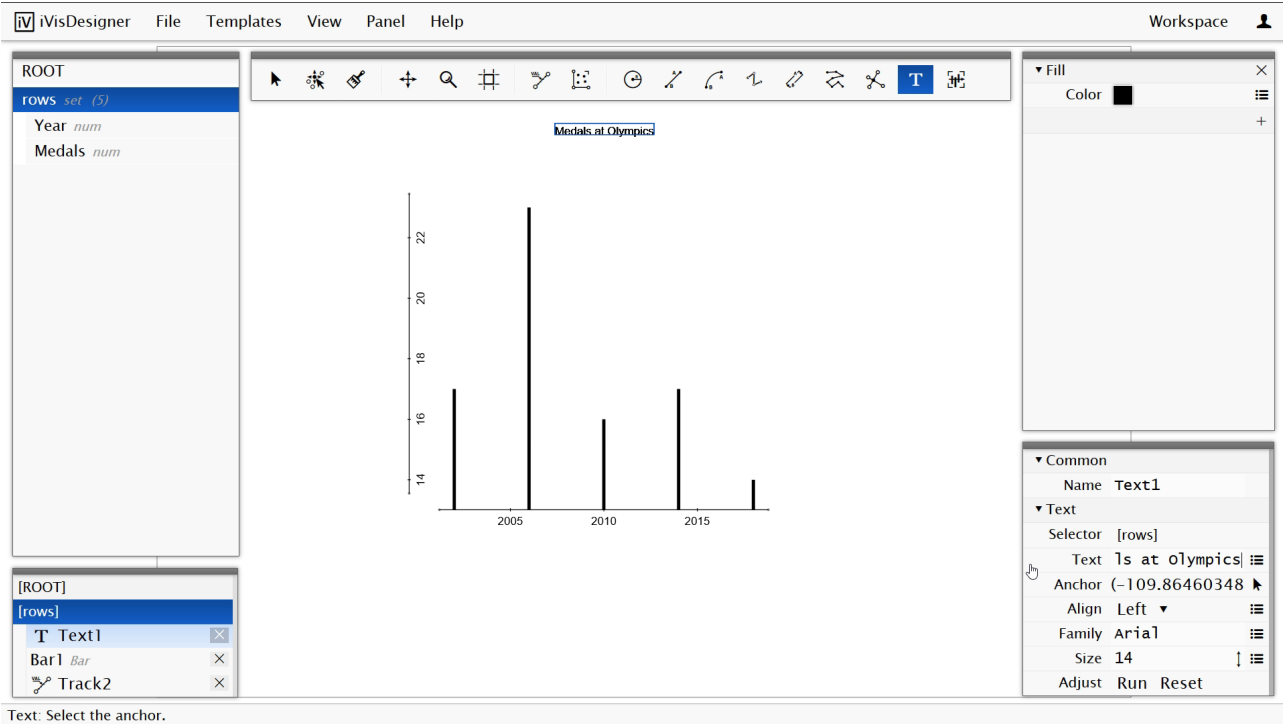
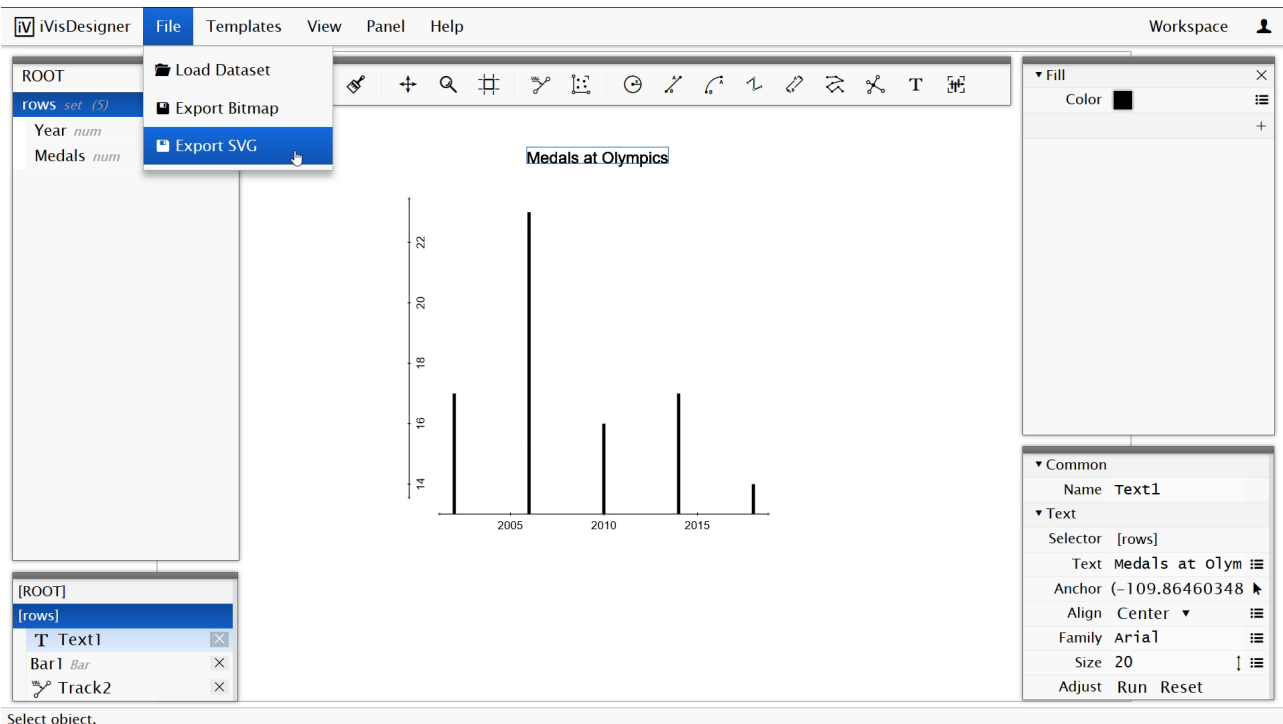


Figure 4.14: Create The Bars



Text: Select the anchor.

Figure 4.15: The finished barchart after adding a label.



Select object.

Figure 4.16: Exporting The Chart

4.3.3 Evaluation Of iVisDesigner

Below you can see the evaluation of the editor according to our defined metrics. iVisDesigner is a powerful editor and you can create many different visualizations. One is only limited by the provided toolbar, which at the moment offers bars, scatterplots, lines and some other options to create visualizations to snap data onto. That and one's skill, as the threshold before one can be called an adapt user is quite high. The documentation is very simplistic and lacking at times, which makes it hard to get into the editor. Additionally some menu items do not work, e.g. at the moment you can not export any charts. You can click the corresponding item in the menu, but no download window will be prompted.

iVisDesigner is not a finished product and its alpha status indicates that. Sadly it will probably stay there, as it will not be developed any further as far as we could tell by looking at the commit history [*iVisDesigner Github* [2018]]. It has been rated as follows according to our metrics:

1. *Platform*: hosted web application
2. *Pricing and license*: free under the BSD 3-clause license
3. *Customisability*: high
4. *Range of chart types*: high
5. *Usability*: bad
6. *Export formats*: SVG / Bitmap, but both are not working

Chapter 5

Concluding Remarks

Following we will recommend one of the editors we have surveyed for each category we defined. We considered our metrics and outside factors surrounding the respective editors to be able to make the upcoming recommendations.

5.1 Recommendations

- *Plug and Play*: **RAW Graphs**
- *Programmable*: **amCharts**
- *Data Guided Drawing*: **Data Illustrator**

It is very hard to make a clear cut recommendation for the first category as all three editors are excellent in what they offer. After looking further at the factors surrounding the actual editors, we had to chose **RAW Graphs**[2.1]. Unfortunately you must have an ongoing subscription to **Datawrapper**[2.2] to make it worthwhile using it – branding is removed and more export options are offered. Additionally we could not recommend **Liquid Diagrams**[2.3] with a clear conscience. It is an excellent editor, but Adobe Air is out of date.

For the second category, we can recommend **amCharts**[3.2], again due to factors surrounding the actual editors. **Lyra**[3.1.3] could have been an excellent editor, but sadly development got discontinued and due to bugs the hosted web application is sometimes difficult to use.

The last category has a clear winner, which is **Data Illustrator**[4.2]. It outruns both **Hyecoo**[4.1] and **iVisDesigner**[4.3]. Data Illustrator already is an excellent tool and you can see in which direction it is going to develop, as it is still a project in research. One can imagine that it will find its place alongside the other great applications in the Adobe Suite.

Bibliography

- iVisDesigner Github* [2018]. <https://github.com/donghaoren/iVisDesigner>. Last accessed: 2018-05-13. 2018 (cited on pages 37, 41).
- iVisDesigner Website* [2018]. <https://donghaoren.org/ivisdesigner/>. Last accessed: 2018-05-13. 2018 (cited on page 37).
- Kim, Nam Wook, Eston Schweickart, Zhicheng Liu, Mira Dontcheva, Wilmot Li, Jovan Popovic and Hanspeter Pfister [2017]. “Data-Driven Guides: Supporting Expressive Design for Information Graphics”. *IEEE Transactions on Visualization and Computer Graphics* 23.1 [Jan 2017], pages 491–500. doi:10.1109/tvcg.2016.2598620. <https://doi.org/10.1109/tvcg.2016.2598620> (cited on pages 29, 31).
- Mauri, Michele, Tommaso Elli, Giorgio Caviglia, Giorgio Uboldi and Matteo Azzi [2017]. “RAWGraphs: A Visualisation Platform to Create Open Outputs”. In: *Proceedings of the 12th Biannual Conference on Italian SIGCHI Chapter*. CHIItaly '17. Cagliari, Italy: ACM, 2017, 28:1–28:5. ISBN 978-1-4503-5237-6. doi:10.1145/3125571.3125585. <http://doi.acm.org/10.1145/3125571.3125585> (cited on page 5).
- RAW Graphs Github* [2018]. <https://github.com/densitydesign/raw/>. Last accessed: 2018-05-13. 2018 (cited on pages 5, 9).
- RAW Graphs Website* [2018]. <https://rawgraphs.io/>. Last accessed: 2018-05-13. 2018 (cited on page 5).