# The History of Interaction Techniques in Data Visualization Systems

**Dominik Fankhauser**
Human-IST Institute
University of Fribourg, Switzerland
dominik.fankhauser@unifr.ch

**ABSTRACT**
Nowadays, a whole bunch of data visualization tools exist. They all have in common that they want to offer the user an easy and powerful implementation to create graphical representations of data. But how can users interact with these tools? What sort of interactivity do the systems offer apart from the creation of graphics?

Based on previous work that categorizes the different interaction techniques into seven main categories, we try to validate how Vega-Lite, a software to visualize information, fulfills these interactivity taxonomies. It turns out that, even after more than ten years, the interaction taxonomies are still up-to-date and implemented in current visualization systems.

In a second part, we have a glance at the future by exploring a new visualization software that has evolved recently. The basic idea is that the user experience can be improved by automatically proposing new visualizations based on what the user has explored so far. Although we like this idea of new interaction, the concept is only a prototype until today.

**Author Keywords**
Information visualization systems; data visualization; interaction techniques.

**ACM Classification Keywords**
H.5.2. [User interfaces] – Interaction styles.

**INTRODUCTION**
Data visualization tools aim at providing the user meaningful data representations as well as some way to modify them. For the representation, the software needs to offer some functionality to plot the underlying data in different graphical forms. For example, pie charts, histograms, scatter diagrams, and so on.

The second component, the modification of data representations, asks the information visualization system to offer some sort of interactivity. Users want to, for example, explore interesting areas of the graphic or get some additional information about a specific point on the visualization.

How the two previously introduced purposes of information visualization systems are implemented differs from software to software. But there is one commonality: most tools and papers within the domain focus on the representation component, i.e. which new graphics are available. The interactive part often slides in the background even if it is an important factor for users according to Lee et al. [2]. The best representation is of limited worth if it cannot be explored in detail to find important patterns and information.

In this paper, we therefore focus on the evolution of the interactive component of information visualization systems. In a first step, we discuss the seven general categories of interaction techniques for visualization systems that Yi et al. identified in their work [4]. Afterwards, we introduce Vega-Lite, a high-level grammar of interactive graphics proposed by Satyanarayan et al. [3]. We are going to discuss how the different interaction taxonomies are implemented within Vega-Lite. In a last chapter, we take a quick look at a new system that has evolved a short time ago and makes use of Vega-Lite. The system has been proposed by Lahmar et al. [1] and is named *EVLIN*.

**INTERACTION TAXONOMIES**
In their work, Yi et al. [4] tried to identify the different interaction techniques by a literature review and exploring existing information visualization tools. In a first step, the authors found over 300 interaction techniques. Step by step, they refined the list, clustered similar interaction techniques, and finally came up with the decision to classify the interaction techniques. This classification happened based on the user intent, i.e. what the user wants to achieve with the action. The idea behind this approach is that interactions that seem to be different can pursuit the same goal depending on the underlying visualization. In the following subsection, we are going to explain briefly each of the seven techniques that were finally proposed. The short sentence next to the title is a short description the authors have chosen for each technique in their initial paper.

**Select (Mark something as interesting)**
This interaction technique aims at offering the user a possibility to mark an item of interest so that it can be identified easily when the representation changes. This is especially helpful when the data set is very large or the representation changes massively. In general, most systems use some sort of labeling or coloring to make the selected data values distinguishable from the rest of the visualization.

**Explore (Show me something else)**
Due to the large number of items in a data set and the limited screen size, users can usually just take a glance at a subset of the items. Exploring means that one should be able to change the currently displayed area (without necessarily changing the view). Normally, some data items will disappear from the screen while other ones enter the currently visible area.

**Reconfigure (Show me a different arrangement)**
By rearranging the spatial representation of the displayed data items, we reconfigure our visualization. So the underlying representation stays the same, but we slightly adapt it. For example, in a bar chart, we could filter out some items or rearrange the order in which they appear on the diagram. Another use case are 3D graphics where changing the camera angle can reveal items that were initially hidden behind other items.

**Encode (Show me a different representation)**
In contrast to the previous interaction technique, the encode interaction means that we really change the underlying representation in terms of appearance of the data items. As an example, we can think of changing the graphic from a histogram to a pie chart. But also changing the color range that is used to distinguish the items in one dimension is an example of changing the encoding of our visual representation.

**Abstract/Elaborate (Show me more or less detail)**
As the name already suggests, each data visualization tool should provide a mechanism to change the level of detail that is represented. Obviously, zooming is one way to implement this interaction technique by allowing the user to scale the representation. Another way to implement this taxonomy is, for example, to show additional information as soon as an item is hovered in the visualization.

**Filter (Show me something conditionally)**
The idea of this taxonomy is that a user should be able to query the current representation to display only items of interest. Querying means that we can specify a condition and the system will then display (or highlight) just the items that fulfill this condition. Another application of filtering is the use of sliders to specify a certain range. The system will then only show items lying within that range.

**Connect (Show me related items)**
For the last interaction technique, we need to distinguish between connection within a single view and connection amongst multiple views. The former means that visualization software can allow users to see hidden connections to other items when clicking or hovering an item. These connections can be based on one or more similar attributes between the items.

For the second use case, we suppose that several representations of the same data set exist within a tool. By clicking on one item of interest, the system should be able to highlight the same point in all other representations so that one can see how the position of the point changes depending on the attributes that are used to display it.

**VEGA-LITE**
Vega-Lite is a high-level grammar to create interactive data visualizations that has been introduced by Satyanarayan et al. [3]. The term high-level means that users do not need necessarily to care (in the sense of specify) low level details such as color encodings, axis range, and so on. The tool of course offers possibilities to specify all these things by hand, but if one does not specify anything, Vega-Lite will automatically calculate and assign the values.

In general, Vega-Lite consists of two grammars: the grammar of graphics and the grammar of interaction. In the following sections, we are going to have a quick look at both of them.

**Grammar of Graphics**
The grammar of graphics is used to create the data visualizations. The smallest components are the so called *unit specifications* which describe a single Cartesian plot. We have to specify a mark type to indicate which encoding we want (e.g., bar, pie, histogram) and which attributes shall be plotted (see figure 1). Optionally, we can define additional information such as color encodings and size.

*View Composition Algebra*
The previously introduced unit specifications can be used to create composite views by applying the built-in view composition algebra. Vega-Lite offers four operators to combine views. We can also use them to combine two composite views (instead of unit specifications).

The *layer* operator allows users to plot several units on top of each other. Vega-Lite will try to compute shared scales (if possible) and automatically adapt the ranges of the axis to fit optimally.

To put views side by side (either horizontally or vertically), Vega-Lite offers the *concatenation* operation. If possible, the tool will use optimized shared scales for this operator as well.

The *facet* operator works similar to the concatenation operator since it puts views side by side. But the idea of faceting is to create one chart for each distinct value of a particular attribute. For example, we can plot the average temperature for each month. By faceting over the location attribute, Vega-Lite will create one such plot for each different location that is stored in our data.

Last but not least, there is the *repeat* operator. This one works similar to facet, but instead of creating a single plot for each different value of an attribute, it creates different plots for a set of attributes that the user can specify.

**Grammar of Interaction**
Similar to the unit specification before, the interaction grammar knows the selection as a basic unit. Vega-Lite

knows three different selection types that can be defined on a unit specification.

The *point selection* allows one to select and color exactly one unique point in the visualization. With *list selections*, the user can select an arbitrary set of points while the *interval selection* only allows the selection of continuous points (i.e., points where the value of a specific attribute falls into an interval). The compiler will then automatically add a rectangle to the visualization so that one sees which points lie in the interval.

When these selections are triggered depends on the specification. Almost any keyboard or mouse event can be used to initialize a certain selection type.

*Selection transforms*

To modify the elements of a selection, Vega-Lite offers five different selection transforms that can be applied. Similar to the composite views, the selection transforms can be combined to create more powerful interactivity, but they are always defined on the unit specification.

The *project* function allows users to modify the predicate function of a given selection so that only the explicitly specified fields are matched.

To add or remove a point from a given list selection when a particular event occurs, one can use the *toggle* function. The event can be any of the available mouse or keyboard events.

In case that we would like to move an interval selection on a given visualization, we can apply the *translate* operator. A translation needs to be initialized by an event as well.

The *zoom* translation is more or less self-explanatory. By defining it on a representation, the user has the option to zoom-in and zoom-out on it.

The last available function is called *nearest*. It selects the nearest point when the triggering event occurs in the visualization.

## HOW DOES VEGA-LITE SUPPORT THE INTERACTION TECHNIQUES?

In the following sub-sections, we will discuss how Vega-Lite supports each of the seven previously introduced interaction taxonomies.

```
1  {
2    "$schema":
     "https://vega.github.io/schema/vega-lite/v2.
     json",
3    "data": {"url": "data/movies.json"},
4    "mark": "bar",
5    "encoding": {
6      "x": {
7        "field": "IMDB_Rating",
8        "type": "quantitative"
9      },
10     "y": {
11       "aggregate": "count",
12       "type": "quantitative"
13     }
14   }
15 }
```

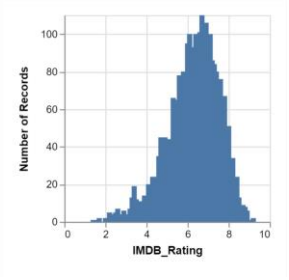


**Figure 1. Unit specification to create a bar diagram in Vega-Lite. The chart displays the number of movies that received a particular rating.**

### Select

The different selection types that we discussed in the previous section offer exactly what this taxonomy requires. Users can select single or multiple points which will then be colored so that they can be followed when the visualization changes.

### Explore

By applying the translate and zoom transform operations, we get exactly the functionality that the explore technique demands. The combination of these two functions allows users to zoom-in and zoom-out as well as moving the currently visible area of the graphic.

### Reconfigure

There exists no built-in function that covers all functionality this interaction technique asks for. In our opinion, it is even easier to simply rearrange the data in the underlying data source to achieve a rearranging of the items than trying to implement it directly in Vega-Lite.

### Encode

The way data is represented can be changed easily with Vega-Lite. For example, we could change the underlying representation in figure 1 easily from a bar diagram to a line chart by just changing the value for the “mark” on line 4 from “bar” to “line”. Or, if we would like to have a histogram, we only need to add one line of code.

As already mentioned earlier, Vega-Lite automatically determines color and axis ranges during compilation. But we can override all these properties to adapt the visualization. But we do not go into more detail here concerning these adaptions.

### Abstract/Elaborate

Vega-Lite perfectly fulfills the zooming aspect of this technique by offering the zoom translation. The second aspect, displaying additional information about a selected or hovered item, can be achieved as well with a little bit more work.

### Filter

Filtering is enabled by the fact that each selection in Vega-Lite is based on a predicate function to determine whether a given data point has been selected or not. So one can use these predicate functions to allow the user a filtering of the data set.

### Connect

With Vega-Lite, both aspects of the connect interaction technique can be covered. When we want to display all points within a view that have the same property as the selected point, we can proceed as follows: we define a point selection on the corresponding view and then use a project function to include all points that have the same property as the selected point.

In the second scenario, we have different visualizations of the same data and want that the selected point(s) are selected in all representations. Vega-Lite already offers this

function by default since it will automatically create a single selection across all views. This selection is configured in such a way that it behaves as the desired function described above.

## NEXT GENERATION INTERACTION

As with all software systems, the way we want to interact with a data visualization system changes over time. Throughout the last years, two main approaches existed to expand data visualization systems: query recommendation and data visualization recommendation. As the name already suggests, the former tries to suggest the user queries that could be interesting to explore while the latter directly creates new visualizations based on what the user explored earlier.

The new approach introduced by Lahmar et al. [1] tries to combine the previously introduced techniques to improve the data exploration and makes use of Vega-Lite for the data visualizations. In the following subsection, we quickly outline how the system of the authors (called *EVLIN*) is supposed to work.

### Provenance-based query and data recommendations

As soon as the user has specified a query to process, the rendering recommendation unit of *EVLIN* tries to find an appropriate Vega-Lite visualization of the resulting data set returned by the query. In a next step, the user can explore this visualization to find interesting components.

When the user has selected some interesting points in the visualization, the data recommendation unit of *EVLIN* tries to figure out which data could be of interest to the user in the future. Based on this information, the query reformulation unit of the system adapts the initial query. The user can then choose amongst a set of queries that have been recommended by the system to start a new exploratory step. The query recommendation happens based on a scoring schema to provide as useful queries as possible.

Without having a closer look at the theoretical background of the recommendation processes and algorithms, we can state that the evolution provenance model of *EVLIN* takes into account the interactions, visualizations, and recommendations of the previous exploration steps to propose new queries.

### Is *EVLIN* a new interaction technique?

We think that this question can be answered with "no" (at least today). We are aware of the fact that the system exists just as a prototype so far. Therefore, it is too early to answer this question in its entirety. But we found the following two arguments which led us to this answer.

First of all, the interaction taxonomies from Yi et al. [4] are more than ten years old and have not been extended up to now. Many software visualization tools have evolved during the last years, but none of them was able to actually change the interaction techniques. We see this as an indication that it will be hard for the provenance-based query and data recommendation approach to become a new, eight, interaction taxonomy.

Our second argument is that the way a user can interact with *EVLIN* goes further than what the interaction taxonomies cover. As introduced above, the taxonomies were created based on the user intent, i.e. what the user wants to achieve. Furthermore, they basically cover one single action that the user executes. But in the context of recommendation-based systems, the user has not really a clear goal that he wants to achieve. Additionally, the way one interacts with the system we introduced before is not based on a single action. The interaction is more a sequence of actions that depend on each other.

## CONCLUSION

Throughout the past years, the interaction taxonomies that have been proposed by Yi et al. [4] have become a good (however, not quantitative) measurement to determine whether or not an information visualization system provides the necessary function users need. Even ten years after the publication, current software tools like, for example, Vega-Lite still offer everything needed to interact with the system in a way the authors proposed it. We think that these interaction taxonomies will still be used in the future since every higher-level application for data visualization will be based on an underlying software that needs to provide exactly these techniques to interact with it.

However, there exist attempts to move the user interaction into a new direction by trying to automatically detect interesting data and visualizations and provide them to the system user. One such attempt from Lahmar et al. [1] tries to combine query and data recommendations based on the provenance, i.e., the interactions and visualizations the user has explored earlier. Although there exists just a prototype of this system, we think that the future movement of interaction in visualization systems will go into this direction.

Another field that has evolved throughout the past years and asks for powerful information visualization tools is called big data. Gartner, Inc. define big data in their IT glossary as follows: "Big data is high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation."[1]. We think that especially in this context, users are not able to explore exhaustively a whole data warehouse by hand. Instead, they need tools and assistance to automatically detect interesting patterns and subsets of the data. Such assistance can come from provenance-based query and data recommendation systems like *EVLIN*.

[1] https://www.gartner.com/it-glossary/big-data/

## ACKNOWLEDGMENTS

This paper is a student work, written for Seminar SS2018 "Data Visualization", under the supervision of Prof. Denis Lalanne and Florian Evequoz. Human-IST Institute, University of Fribourg, Switzerland, 2018.

## REFERENCES

1. H. B. Lahmar, and M. Herschel. 2017. Provenance-based Recommendations for Visual Data Exploration. 9th {USENIX} Workshop on the Theory and Practice of Provenance (TaPP 2017).
2. B. Lee, P. Isenberg, N. Henry Riche, and S. Carpendale. 2012. Beyond Mouse and Keyboard: Expanding Design Considerations for Information Visualization Interactions. IEEE Transactions on Visualization and Computer Graphics, 18(12): 2689-2698.
3. A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. 2017. Vega-Lite: A Grammar of Interactive Graphics. IEEE Transactions on Visualization and Computer Graphics, 23(1): 341–350.
4. J. S. Yi, Y. ah Kang, J. T. Stasko, and J. A. Jacko. 2007. Toward a deeper understanding of the role of interaction in information visualization. IEEE Transactions on Visualization and Computer Graphics, 13(6): 1224–1231.