Visualization Analysis and Design

Tamara Munzner

July 18, 2016

1 Lecture 1

Disclaimer: taking notes for lectures which use slides is harder because I can't take advantage of the time the speaker writing on the board. So this will probably contain a lot less information than the previous one. Fortunately the notes by the speaker is available here.

1.1 Visualization

Lets start with the definition of visualization. Computer-based visualization systems provide visual presentatinos of datasets designed to help people carry out tasks more effectively. When there is fully automated solutions, visualization is not very useful, but when we have an ill-specified problem, we need to use visualization to help us understand the problem better to build the model. It could also help developers and end users of automatic solutions to determine parameters / build trust. Visualization is a way to help us out of our jobs: towards ultimate automatic solutions.

But why do we use an external representation? If we have a table of numbers it is hard to keep track of them in our minds, but if we have a short-cut through the cognitive act of memorizing and analyzing numbers by plotting them, then we can recognize patterns and understand trends much more easily. We don't want to burden our cognition with low-level book-keeping, but save them for much higher level understanding, and that is where visualization comes in.

How does visualization help people do things faster? It summarizes lose information, details matter. We can put things together to confirm expected and find unexpected patterns, etc. The so-called Anscombe's Quartet shows 4 datasets with identical statistics, but look completely differently when plotted. These kind of situations are where it is very useful to show the whole dataset at the same time without losing details.

1.2 Framework

We can build an analysis framework by putting it under 4 levels and ask 3 questions. The first level is *domain*, i.e. "who are the target users"? There is a vocabulary for a domain (domain specific language), to communicate with the users of that specific domain. Then we come down to the next level which is *abstraction*. Here we want to abstract/translate the specifics of domain to the vocabulary to visualization. We don't usually draw exactly what we are given, but transform them to a new form. This is the question of "What is shown?". We also need to address "Why is the user looking at this".

The next level is *idiom*, i.e. "how to draw the visualization and how to manipulate them?". Then the last level is the *algorithm* level which is separate from the representation levels above.

We have different ways to get things wrong at any level. At the domain level, we could assume the needs of the audience wrong, e.g. assuming the same level of domain-specific knowledge as the speaker from the audience. At the abstraction level we could be showing the wrong data. At the visual level we could be showing things in a way that doesn't work, and finally we may have a code too slow to do the job.

How to do things right? We need to use methods from different fields at each level. At algorithm level we are talking about computer science, which is technique-driven work. At idiom level we are in the regime of design and cognitive psychology. At the higher levels we start to venture into the field of anthropology/ethnography. Here we are talking about problem-driven work.

1.3 Questions

1.3.1 What

Lets start by asking the question of *What*. There are three main types of datasets: tables, networks, and spatial (fields or geometry). Tables are easy to understand, with columns and rows. Networks are like graphs with nodes and links between them. Spatial data can come as fields, like a grid of positions with values associated to them. It could also be geometry, like a map.

There are also different types of attributes that we want to represent. We could have categorical data, or those with intrinsic ordering. With the latter we also have the question of which ordering direction (sequential/diverging/cyclic). In situations where the data is cyclic it could represent some interesting challenges.

1.3.2 Why

Lets now look at the question of *Why*. We can split the question into actions and targets. On the action side, we typically want to analyze/query data. Analyzing is a way to consume data, to discover things or present them to others. We could also present new data by annotating older sets or by deriving new things from older things. Querying is changing the amount of data, to identify/compare things, typically reducing the amount of data to process in our head.

We can talk a bit more about deriving. We actually rarely draw things we are given directly! First we need to decide what the right thing to show is, create it with a series of transformations on the original data, and then draw that. A simple example is a graph of exports and imports, we could take the difference and plot the trade balance.

An example of analysis is to derive one attribute from a series of trees/networks, namely the Strahler number. Details in the slides, I don't want to summarize it on the fly...

Now lets talk about the targets. We might want to present trends, outliers, or features, where the last one is an all-encompassing way to say things for example quantitative...

1.3.3 How

Lets now talk about the important question of *How*. First thing we need to do is to encode data. There are various structures for visually encoding the data. To talk about this, we introduce marks and channels. Marks are geometric primitives, and channels control the appearance of marks, e.g. color, position, size, etc. We can use different channels to encode the same data to bring emphasis to the data, or we could add different channels to encode different content to squeeze in more information.

1.3.4 Marks and Channels

The different idiom structures of visually encoding the data are combinations of marks and channels. A histogram is using the vertical positions of lines to encode quantities. A scatter plot is using the vertical/horizontal positions of points, etc.

There are two main types of channels, one is magnitude, and the other is identity. The magnitude channel, like position, length, saturation, are suited at showing ordered attributes, whereas the identity channel (spatial region, shape, etc.) is suited at showing categorical attributes. We need to match the channel and the data attributes we want to represent.

Different channels have different effectiveness! We are very good at judging spatial positions, but not very good at judging saturation or luminance. We want to encode the most important attributes with the highest ranked channels. Spatial position ranks the highest for both magnitude and identity channels, so it should be the first thing we think about when we determine the way to encode data. One thing to keep in mind is that there are people who respond less well to color hues (colorblindness).

Where does the channel ranking come from? We have the Steven's Psychophysical Power Law: $S = I^N$. The perceived sensation of length has index $N = 1$, with area/depth at $N = 0.7/0.67$. There were visualization experiments done by Cleveland & McGills, and experiments using crowdsourcing.

In addition to the channel to choose, we need to consider how many usable steps we have at each channel. We need to have a channel with enough different levels to show the different data we need. We also can't use all the channels at once, because sometimes channels step on each other. Refer to slide page 32 for a comparison when we have channels that interfere with each other.

One thing we need to consider is "Popout". When we have multiple clues for popout the data basically presents itself, but when we need to do a conscious search on the data representation it takes a much longer time on our cognitive system to recognize the pattern.

We can represent grouping between features. We can use containment or connection, or just put related things close to each other.

One last word about representation is relative vs. absolute judgments. The perceptual system mostly operates with relative judgments, not absolute one. Therefore to represent the comparison of data, accuracy increases with common frame/scale and alignment. The ratio of increment vs. common background determines the ease of judgment.

1.4 Common Idioms for Different Datasets

1.4.1 Tables

Now lets to talk how to use space to encode each of the data types. The first is *tables*. Lets use the computer science words key and values to denote the table. The key is an independent attribute to uniquely index the items we want to look up. A simple table has one key, whereas multi-dimensional table has multiple keys. Values are simply the data we want to represent.

As an example, the scatter plot is used to express values only, with no keys. We simply represent two quantitative attributes on each dimension. It scales very well since we can fit in thousands of points on a plot. A bar chart has one key and one value. The key is a category attribute and the value is a quantitative attribute. It is good at comparing or looking up values. A line chart also has one key and one value, with point as the mark. Now the key can be a quantitative attribute.

When should we should bar charts vs. line charts? The answer depends on the type of the key attribute. For female/male key (categorical) it is better to use bar charts, and for a number key (quantitative) it is

better to use line chart.

Lets consider heatmap. We have two keys, x-y position, and one value as color. The two keys can be categorical attributes, and the value is quantitative. It can be useful at finding clusters and outliers. However we are limited by how much display can resolve on the screen, and we are capped at $\sim 1\,\mathrm{M}$ items. We are however very limited in the quantitative attribute levels since we only have so many colors/regions to play with without interfering with each other.

We can combine different idioms together using for example scatterplot matrix, where we have a matrix of scatter plots put together. We can also have parallel coordinates that put different line charts together. Either of these we are limited to dozens of attributes.

We also have pie charts or polar area charts. In the former area marks with angle channel to represent data, and in the latter we have length channel for the area marks. Both are better replaced by bar charts. One argument for pie charts is to show parts-to-whole idea, but we could fix that by plotting a normalized stacked bar chart.

There are interesting cases where one could use radial orientation to represent data. One example is glyphmaps to represent cyclic data.

1.4.2 Spatial Data

Let's talk about spatial data now. We often want to use directly the data we are given, since the data is encoded directly in the position they are given. For example the choropleth map is to color-code the different regions of a map to represent a quantitative attribute on a geographic geometry. One important issue here however is the normalization of data.

We could also have a topographic map, which is plotting a scalar spacial field on geographic geometry using contours. Similar idea is to plot isosurfaces or directly volume rendering which is to plot scalar spatial field directly in a 3D space by rendering the isosurface of the value. Due to the 3D nature often we often use opacity as well as colors to encode the desired data.

There are a family of ways to show vector and tensor fields. The challenge is that there are many attributes to show per cell. We could have flow glyphs where we just show local directions. We could also have geometric flow that trace the trajectory of particles. We could have texture flow and feature flow. In general there is no single best way but it depends on the task we want to carry out.

1.4.3 Networks

The last data type is networks, which is when we add links to the dataset. One way is simply use node/lines to directly show the networked graph. This is good for exploring topology, locating paths and clusters, etc. It is tricky however to consider spatial positions and proximity semantics, since we may be conveying unintended information. The scalability of this kind of graph is limited by the number of edges. This kind of graph only makes sense when we have number of nodes N less than 4 times the number of edges $4E$.

We could also transform the network into a heatmap where we encode two categorical attributes with x-y positions and use color to represent link. However we need to train ourselves to recognize topological structure. The strength of this representation lies in the scalability.

We could also use a radial node-link tree. It is easy to understand topology and to follow paths.

2 Lecture 2

2.1 Mapping Channels

2.1.1 Color

We talked about arranging and aligning graphs, and now we will talk about mapping the data into colors and other channels.

We will talk about color in three channels: luminance, saturation, and hue. Hue is a categorical channel which distinguishes color by themselves. Luminance and saturation however, can represent magnitude or ordered data. In the language of computers we use RGB to encode color, but it is terrible to encode data because it doesn't align with human perception. It can be a lot safer to use the HSL language which directly uses the three channels, but beware that the L in the HSL system stands for lightness, not luminance. In the HLS system lightness is independent of the color hue, but the human eyes responds to luminance which is different for different color hues. For example, yellow looks much brighter than blue. The eye detects edges using luminance, so it is a good idea to use it to denote clusters and borders.

There are different colormap designs. There is categorical color map which map different categories of data to different color hues. We could have sequential data to represent two ends, or we could have diverging color scheme. However, beware of bivariate colormaps when we try to combine two kinds of colormaps. It is in general okay to use categorical with sequential, but other combinations might not work quite as well as you hoped. By the way, Brewer is the one who created a series of very good color schemes which are safe to use.

There are a couple of principles when using color channels. Small regions need high saturation, whereas large regions needs low saturation. Saturation and luminance is hard for the human eyes to distinguish in small increments, so it is about 3-4 bins max for us to easily distinguish.

When similar colors are next to each other, it is much easier to distinguish them, but if they are separated into small noncontiguous regions, then we could only distinguish 6-12 bins, since human eyes want to group similar colors together automatically. When we ant more bins, we can combine colors with sizes and other channels.

For ordered color maps, we often find the rainbow color scheme as the default scheme, which is a terrible default! The color hues are not perceptually ordered, and perceptively nonlinear. Certain regions in the rainbow like yellow and cyan represents more transition than other regions. If we use fewer hues we can see large-scale structures better. If we want to see fine structures then we can use multiple hues with monotonically increasing luminance (viridis). If we really want to keep rainbow, its best strength is its many hues, so we could use segmented rainbow to show binned or categorical data.

2.1.2 Other Channels

For sizes, people are good at judging lengths, less so for area, and even less so for 3D volume. People are generally good at judging angles, but the accuracy could be nonlinear. Shapes could be very useful since people are good at distinguishing different low-level primitive shapes, and combinations of them. Motion is also a very powerful channel, but it could be too strong that it draws your attention visually and might distract you from other channels. It is extremely good for highlighting.

2.2 Handling Complexity

We want to introduce 3 more strategies to handle complexities in visual representation.

2.2.1 Manipulate

Manipulating the data can be done in several different ways. One way is to change the view over time. It could be done interactively or simply showing the data twice in different views. One could also reorder data in different attribute values, which gives a sense of ranking. Another thing one can do is realign, especially when there are different things to compare.

In general, animated transitions are good because it is obviously a change of view over time. For example, when we zoom in or zoom out of a region, it can be a good way to help people understand what is going on.

For interactive selection, there are a few basic operations for basic interaction, for example clicking with a mouse, hovering, or moving across. There are also multiple ways to highlight selected targets. One way is to change the color of the subject, or we could let it bounce or change size.

We could use navigation to change the visibility of items. For example, we could move our camera to pan, translate, and rotate, in order to focus our viewpoint to some specific region. A more advanced example would be semantic zooming, where the visualization technique depends on the size of the constraining box. Example on page 97 of the notes.

However free navigation is very disorienting. It is much better to have constrained navigation on the animated transitions so that the viewer has a better sense of trajectory and trail.

Another way of navigating is slicing, cutting, or projecting the data, in order to reduce complexity. The first two are quite self-explanatory. Projection is a way to change the mathematics of image creation, like reducing the dimension from 3 to 2.

2.2.2 Facet

Faceting into multiple different views is also a very powerful technique. There are many ways to do this. *Juxtapose* is a way to share something between different datasets, for example sharing encoding, subsets of data, or navigation.

As an example, shared highlighting on different facets of the same dataset can reveal how the same subset of data is distributed in different order/view. It can also be implemented with interactivity, where you can select a subset in one panel and view the distribution in another graph on another panel.

Another example is the bird's-eye map view of Google map. The user can pan and zoom the camera, but a small bird's eye view at the corner will always tell the user where in general is the user looking at.

Why not show this in animation? The reason is that the disparate frames and regions make comparison difficult. Especially when there are different things happening, it can be very difficult for the human eye to keep track of what is going on in time. The safe special case is animated transitions, where it is naturally going from one state to another.

Why do we juxtapose views? It can be beneficial for the cognitive load because the eye can move quickly between two views, thus removing the requirement of putting the view into short term memory. The cost of this however, is that we run out area to display. We can run out of pixels very quickly on the screen.

It is an open question how many views is okay. But it is definitely important to investigate the power of multiple views. It is generally useful to have multiple linked views and reorderable, so that one can look at data simultaneously in different angles.

The second question to ask is how to *partition* the data. This is the question of how to divide data between different views. One could partition into side by side views. Check out the examples in the slides: List alignment and recursive subdivision.

One last thing to think about is to *superimpose* layers. We can superimpose different layers of spread-out objections in the same region. If we are going to do this we need to use different data channels for different layers, and not to use to many layers in the first place.

One principle for using layers is that the background layer should have low saturation. Another thing to remember is that if you should first try to make it look right in black and white, which is the ultimate luminance contrast. Superimpose and be used interchangeably with juxtapose. Some advice would be to use superimpose for local, and juxtapose for global.

2.3 Reduce

Lets finally talk about this idea of data reduction and filtering. There are many different ways to do this. Filter is the most obvious thing to do: you just throw things away. It is very straightforward both cognitively and computationally. However, the con is that the user will almost always just forget about what they don't see. This can lead to a lot of misconceptions.

Aggregation is the alternative to filtering. We don't simply discard what we don't want to draw, but show them in a somewhat reduced/averaged way. However, it is difficult to avoid losing signal and still be effective.

However these techniques are not exclusive. A lot of systems actually combine all of these techniques at the same time. A great way of doing filtering is to dynamically filter with interactive result display. One can iteratively change the filter criteria to achieve the desired complexity.

Histogram is an example of static aggregation. The shown data is derived data from a given table. In the histogram the keys are bins and values are counts. The bin size is crucial since patterns can change dramatically with discretization.

There is also an idiom of scented widgets. A danger of interaction is for it to degrade into manual search, which is wasteful of time. Scented widgets shows with color cues what would happen when people move the sliders, or show people where there might be interesting things to see.

The boxplot is also a useful aggregation tool. The box plot plots 5 derived statistical data: median, lower/upper quartile, and lower/upper fences. One could even put in outliers that do not fit in the distribution.

Another reduction technique is dimensionality reduction. When we can't directly measure what we care about, for example when the dimensionality of dataset conjectured to be much smaller than the dimensionality of measurements. If we have a 9D parameter space for measured data, we can plot the derived data in a, e.g., 2D target space, and it might reveal cluster structures, hidden variables, etc.