Visualization Trends

Hey: Top 5 Data Visualization Trends

Video Visualization

Data Democratization

Real-time Visualization

Mobile and Social Data Visualization

Artificial Intelligence and Machine Learning Datavis

1. Video Visualization

The statistics behind data retention through video is even more impressive than those behind static visuals - and that’s saying something.

We all know the emotional impact of a good movie, or the arresting capacity of an exhilarating music video. But the truth is even simpler - even with less exciting content such as that in a business presentation, our minds are hardwired for paying attention to video.

We retain data better with video. Participants in a recent study answered questions 323% better when they had a video aid, compared with those who …

Visualizations uncertainy

Hey: Most visualizations have been designed on the

assumption that the visually represented data are

free from uncertainty. However, this is rarely the

case. Visualizing uncertainty is essential if we want

to improve how people understand statistics. We

are surrounded by summary statistics and statistical

inference on our daily lives, such as means and

probabilities in news paper, government journals,

and mobile applications. These statistics often

lead us to biased understanding of the data.

Visualizing uncertainty hidden beneath statistics

can improve the way we interpret statistics

by looking at whole datasets and/or possible

outcomes to find comparisons. This thesis focuses

on the uncertainties generated from inefficient

representation of data (summary statis…

visualizing categorical data

Hey: Visualizing categorical data

In the relational plot tutorial we saw how to use different visual representations to show the relationship between multiple variables in a dataset. In the examples, we focused on cases where the main relationship was between two numerical variables. If one of the main variables is “categorical” (divided into discrete groups) it may be helpful to use a more specialized approach to visualization.

In seaborn, there are several different ways to visualize a relationship involving categorical data. Similar to the relationship between relplot() and either scatterplot() or lineplot(), there are two ways to make these plots. There are a number of axes-level functions for plotting categorical data in different ways and a figure-level interface, catplot(), that gives unified higher-level access to them.

Visualize proportions

Hey: Visualizing proportions

We often want to show how some group, entity, or amount breaks down into individual pieces that each represent a proportion of the whole. Common examples include the proportions of men and women in a group of people, the percentages of people voting for different political parties in an election, or the market shares of companies. The archetypal such visualization is the pie chart, omnipresent in any business presentation and much maligned among data scientists. As we will see, visualizing proportions can be challenging, in particular when the whole is broken into many different pieces or when we want to see changes in proportions over time or across conditions. There is no single ideal visualization that always works. To illustrate t…

Visualize data on multi-plot grids

When exploring multi-dimensional data, a useful approach is to draw multiple instances of the same plot on different subsets of your dataset. This technique is sometimes called either “lattice” or “trellis” plotting, and it is related to the idea of [“small multiples”](https://en.wikipedia.org/wiki/Small_multiple). It allows a viewer to quickly extract a large amount of information about a complex dataset. Matplotlib offers good support for making figures with multiple axes; seaborn builds on top of this to directly link the structure of the plot to the structure of your dataset.

The [figure-level](https://seaborn.pydata.org/tutorial/function_overview.html) functions are built on top of the objects discussed in this chapter of the tutorial. In most cases, you will want to work with those functions. They take care of some important bookkeeping that synchronizes the multiple plots in each grid. This chapter explains how the underlying objects work, which may be useful for advanced applications.

Conditional small multiples

The **[FacetGrid](https://seaborn.pydata.org/generated/seaborn.FacetGrid.html" \l "seaborn.FacetGrid" \o "seaborn.FacetGrid)** class is useful when you want to visualize the distribution of a variable or the relationship between multiple variables separately within subsets of your dataset. A **[FacetGrid](https://seaborn.pydata.org/generated/seaborn.FacetGrid.html" \l "seaborn.FacetGrid" \o "seaborn.FacetGrid)** can be drawn with up to three dimensions: row, col, and hue. The first two have obvious correspondence with the resulting array of axes; think of the hue variable as a third dimension along a depth axis, where different levels are plotted with different colors.

Each of **[relplot()](https://seaborn.pydata.org/generated/seaborn.relplot.html" \l "seaborn.relplot" \o "seaborn.relplot)**, **[displot()](https://seaborn.pydata.org/generated/seaborn.displot.html" \l "seaborn.displot" \o "seaborn.displot)**, **[catplot()](https://seaborn.pydata.org/generated/seaborn.catplot.html" \l "seaborn.catplot" \o "seaborn.catplot)**, and **[lmplot()](https://seaborn.pydata.org/generated/seaborn.lmplot.html" \l "seaborn.lmplot" \o "seaborn.lmplot)** use this object internally, and they return the object when they are finished so that it can be used for further tweaking.

The class is used by initializing a **[FacetGrid](https://seaborn.pydata.org/generated/seaborn.FacetGrid.html" \l "seaborn.FacetGrid" \o "seaborn.FacetGrid)** object with a dataframe and the names of the variables that will form the row, column, or hue dimensions of the grid. These variables should be categorical or discrete, and then the data at each level of the variable will be used for a facet along that axis. For example, say we wanted to examine differences between lunch and dinner in the tips dataset: