Basic Info

Title: Hashtag Misleading
Our names: Max Lisnic, Jack Wilburn
Email addresses: maxim.lisnic@utah.edu, jack.wilburn@utah.edu
UID: u1317463, u0999308
Project URL: https://github.com/JackWilb/dataviscourse-pr-hashtag-misleading

Background and Motivation

The motivation for this project stems from an ongoing research project. One of the authors manually labeled a set of nearly 10,000 tweets containing data visualizations to describe the ways visualizations can mislead the audience. This class project serves to augment the research project by providing a platform to both get an overview of high-level patterns in the data and at the same time to explore individual data points in detail.

Project Objectives

In this project, we hope to create a visualization tool with multiple coordinated views to show twitter data with some manually coded attributes. The tool should both provide a high-level overview and also allow the user to dive deeper into individual tweets. We hope that the tool will also help uncover interesting patterns in the data and inspire future analysis.

Data

Our data comes from Max's recent paper submission to CHI. The data is tabular and contains information about tweets, images contained within the tweets, and a by-hand encoding of engineered variables that describe: the source of the visualizations, the polarization of the tweet text, visualization design violations, and any reasoning errors that the authors made.

Data Processing

Our data is in a tidy format and is mostly ready to be visualized. There will be some work to pull in all the required images to the application, but we anticipate using either a custom server that we set up, or the Twitter API. There are pros and cons to each approach:

Pros of our own server

- Our infrastructure
- Easy to modify
- No rate limits
- All tweets will be available

Cons of our own server

- Have to keep the app private because of Twitter's ToS

- Added complexity with DNS/CORS
- Have to manage more infrastructure

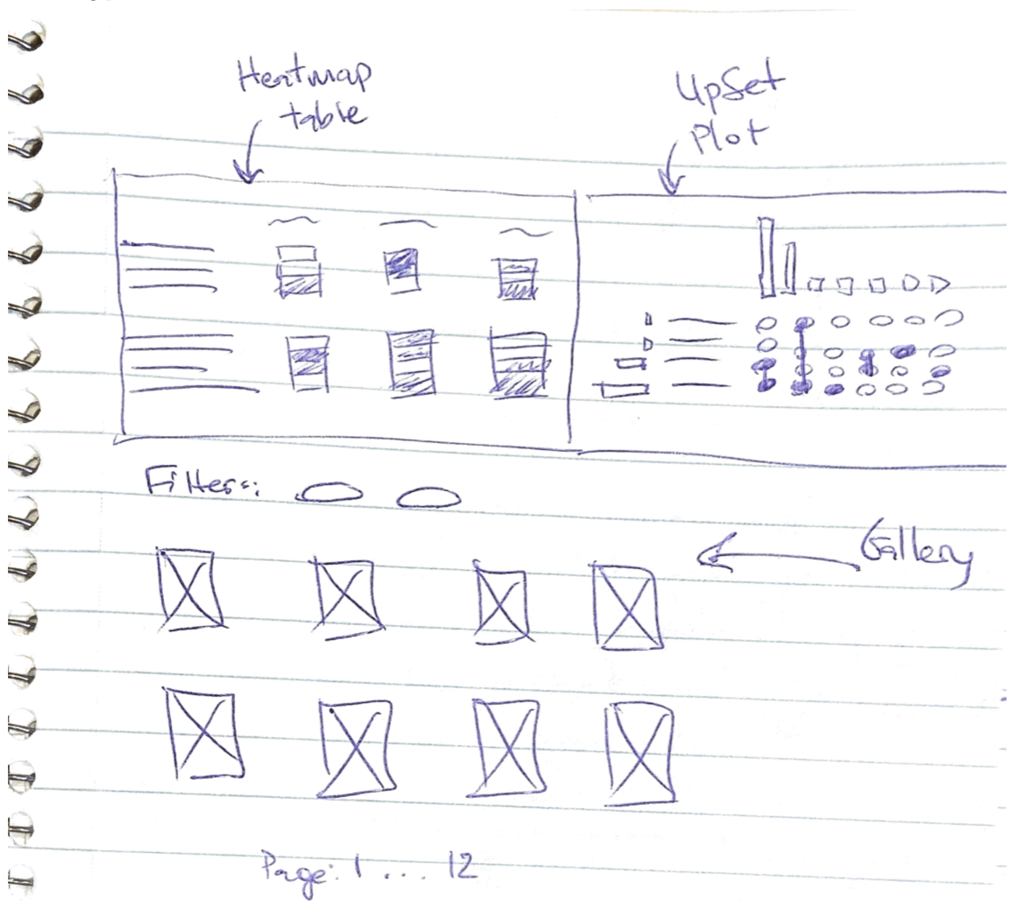
Pros of using the Twitter API
- Can make the app public according to Twitter's ToS
- Don't have to manage any infrastructure
- We can mark tweets as deleted
- Might be able to host the full app as just a frontend with some twitter API calls

Cons of using the Twitter API
- Might not be able to retrieve deleted tweets
- Rate limits

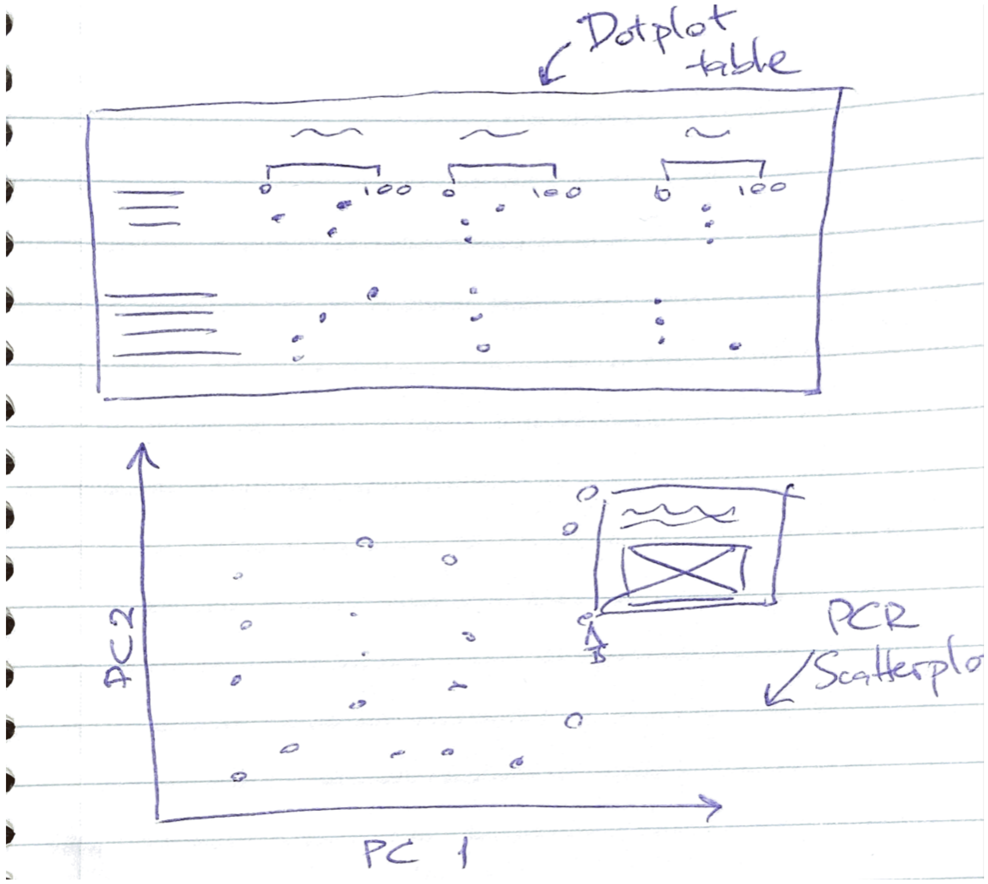
Visualization Design

Please see attached sketches.

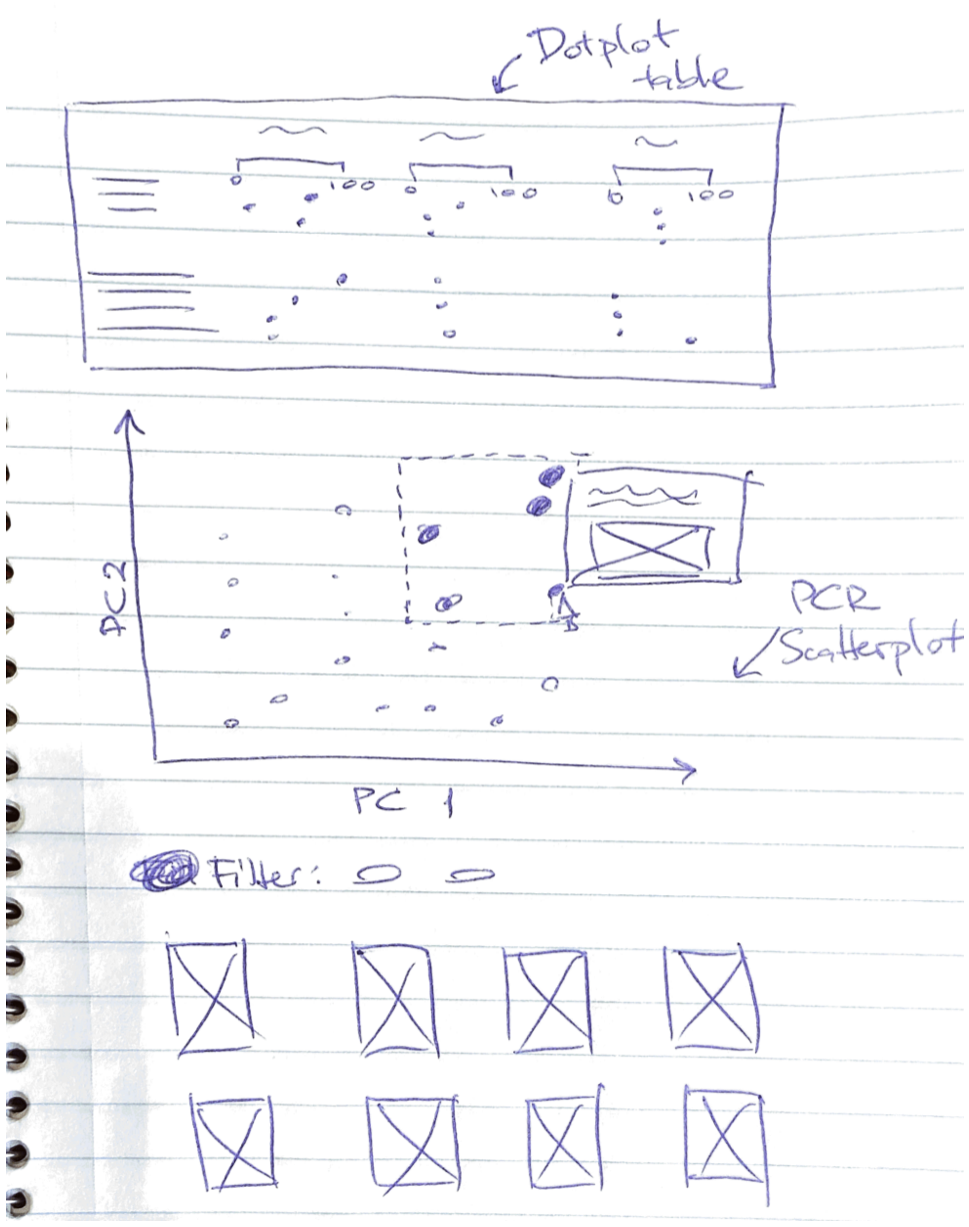
Prototype 1:



Prototype 2:



Prototype 3:



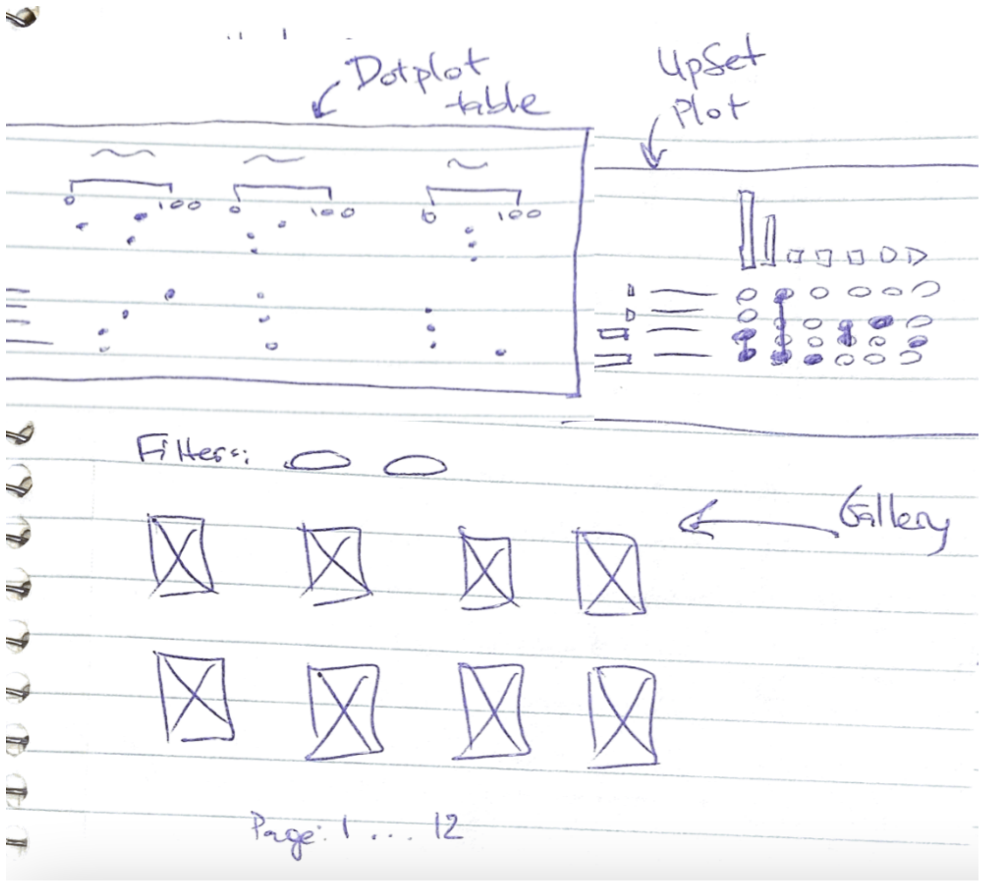
We intend to have 3 coordinated views of the data. We proposed a few different options for each view

The first is a tabular view of the data. Here we can implement the relative counts as a heatmap with percentages/counts overlaid (Prototype 1), or with multiple dot plot visualizations (Prototype 2, 3). The dot in each “cell” would represent the percentage of visualizations that contain this attribute. On hover we intend to gray out/mute unrelated information (unless this causes performance issues), and on click we would update the visualization to filter the data to the selected categories (using AND logic if multiple categories are clicked).

The second view shows the co-occurence of categories for tweets or the relatedness of tweets. We can achieve this by using an upset plot (Prototype 1) or by doing a principal component analysis on the tweets (Prototype 2, 3). If we use upset, we can implement the visualization using the standard upset library. The interaction for this visualization would be similar to the first view — muting on hover and filtering on click. If we show a scatterplot of the PCA results, we can allow for brushing to subset the gallery view (Prototype 3).

The third view of the data is a view of the unstructured tweets/images (Prototype 1, 3). We propose that a gallery would be the best way to visualize this data. Potentially, this view could be dropped in favor of interacting with the PCA plot (Prototype 2). This data would filter down as the other visualizations are interacted with. When clicking on these tweets, we would show the tweet in a modal for closer inspection.

Final prototype:



In the end, we consider the PCA view to be slightly overly complicated and imprecise. So our final prototype is similar to Prototype 1, however we plan to go with the dotplot table instead of the heatmap table. The dotplot will allow the viewer to more accurately compare the prevalence of various components.

Must-Have Features

- Tabular data view

- Upset/PCR data view (one or the other)
- Scrollable gallery of all tweets
- Open individual tweets to show detail
- Filter the gallery by any attribute in our data

Optional Features

- Filter the gallery by clicking on the overview visualization

Project Schedule

- Week of 10/17: Proposal due. VIS 22.
- Week of 10/24: Peer feedback. Meet to review feedback.
- Week of 10/31: Finish setting up the backend, decide on whether to serve data ourselves or query the Twitter API. Make progress on frontend.
- Week of 11/7: Finalize the gallery and one visualization view. Milestone due 11/11.
- Week of 11/14: Review with TA. Meet to review feedback.
- Week of 11/21: Finalize the second visualization view. Have a full working prototype.
- Week of 11/28: Final due 12/2.