**Data Wrangling**

**[andrew]**

Data wrangling has been a big part of the first stage of our project. On the one hand, the SE dataset is, as datasets go, beautiful. It’s complete, well-organized, and pretty well documented. So we were able to avoid issues that often come with uneven or incomplete datasets, such as sparsity and imputation. On the other hand, our design implementations address the data in specific ways which the original structures weren’t always well-equipped to serve. For instance, the chord layout requires a square matrix to render pairwise edges. That sort of wrangling can be done online, in the javascript that loads as part of our visualization.

Other restructuring required more heavy-duty wrangling, which we carried out using Python’s Pandas module. (See the “munging” folder for code files.) In particular, generating periodicity and aggregate measures out of time series data took some figuring out .

In the case of proximity data, we have geo-location measurements taken every six minutes for more than 18 months, for several dozen individuals. As our interest in this dataset lies in the social networks that formed between study participants over time, we needed to come up with periodic pairwise frequency estimates for the number of times any pair of participants were in geographic proximity of one another. We settled on two-week intervals as a reasonable set of checkpoints for an 18-month-long time series. We carried out the same type of time series transformation for communication data as well, for which we combined two separate datasets (calls and sms logs) and generated pairwise aggregate frequencies over two-week periods.

You can see this time series animated in our node graph demo, where we use the communication data series for edge weighting the network structure, here:

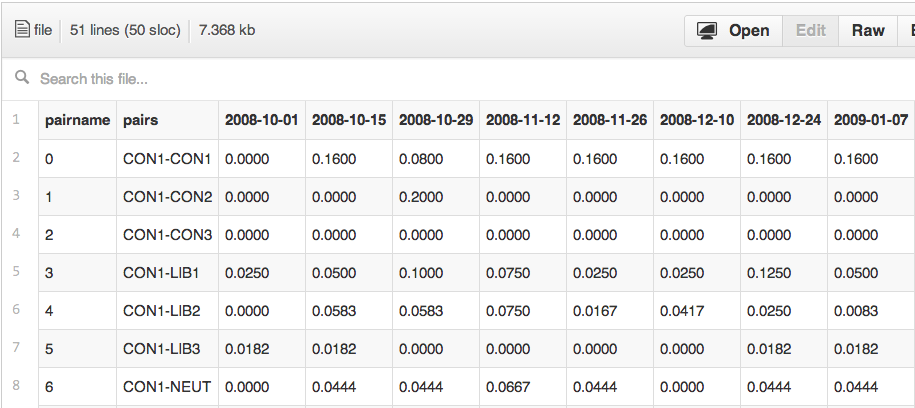
<http://goo.gl/33YJha>

We are also interested in giving users the ability to better understand interpersonal relationships based on individual participants’ personal preferences and behavior. For instance, it may be that individuals of a given political orientation (eg. liberal or conservative) are more likely to communicate with others of a similar orientation. The same may be true for people with similar tastes in music, or for people who exercise regularly, etc. The wide array of datasets in this study all contain bits and pieces of personal information along these lines, but for both speed and coherency of implementation, we wanted to assemble a central dataset of individual characteristics. We used Pandas’ groupby functionality to collate and aggregate various user metrics, which are assembled in our subjects-master.csv file.

Having a central repository for individual user profiles allows us to represent pairwise relationships on extra measures, aside from the basic time series variables (proximity, communication). Our first effort at representing this added layer of connectivity is in the heatmap implementation we have on our force-directed graph. Each heatmap requires pairwise data in a different structure - similar to the main time series datasets, but different in that the heatmap grid responds only to changes in frequency between any two given time points. (So whereas the edges of our main time series graphs always represent a running total, heatmaps instead reflect new activity at each checkpoint. If a pair of individuals had communicated 30 times by T1 and 32 times by T2, their contribution to the heatmap grid would be less impactful than a pair which had only 5 interactions at T1 but 15 at T2. The edge weight for the former pair in the main visualization, however, would be considerably larger at both time points than for the latter pair. We’re still discussing whether it’d be better to represent absolute changes, derivative change, or other measure of temporal change, in our various graphs. The difference between our primary graph and the heatmap for now are largely for proof-of-concept, rather than a final design choice.)

As such, the heatmap implementation required several steps along the way to getting the data in the shape we needed it in. First, we needed pairwise frequency of occurrences at each timepoint. We have that now in the main time series datasets we created. But then, we need to reach into the subjects master file to get whatever personal profile variables we want to relate pairs on (eg. political orientation). That creates a new kind of matrix, with row indices being pairwise combinations of variable categories (eg. veryliberal-mildconservative, mildliberal-neutral, etc.), and column indices as checkpoints in the time series. Each cell represents the number of new instances of a given pairwise relationship that occurred since the previous timepoint.

The number of instances of each pair occurrence is a biased metric, however, as the total number of possible pairings for each pair is unequal. (To continue the political orientation example, there are more liberals than conservatives in the dataset, so there are more possible chances for liberal-liberal pairings to occur than there are chance for conservative-conservative pairings.) Thus, some normalization of frequency is appropriate here - we chose simply to divide each cell count in the matrix by the total number of possible pairings, resulting in a proportion between 0 and 1.



In conclusion, the three main munging operations we carried out were:

1) assembling time series data on two different variables (proximity and communication)

2) generating a master file for subject profiles

3) creating a template for generating “heatmap”-style matrices of pairwise associations across specific profile variables

**Time Series Representation**

**[andrew, apr 10]**

We’ve been talking about how to best represent the passage of time. One way is to animate the entire series. Another way is to provide a slider that the user can use to move manually across time checkpoints. So far, we’ve implemented both - the slider is on the chord layout, and the animated version is on the node layout.

We discussed pros and cons to both. In the end, we may just end up offering both, as it’s not too distracting for the user to have a slider and a play button, and people like to play with things.

One interesting point that Brian brought up is that a sliding scale based on time data is best represented using a quantized scale, which isn’t something we’ve looked at in class.

(quantized scales: github.com/mbostock/d3/wiki/Quantitative-Scales#quantize-scales)

**Drill-down per user features**

**[andrew, apr 10]**

We’d like to offer some drill-down capability to get more information on individual subjects, or inter-subject relationships, or both. One idea we’ve discussed is to have some kind of hover feature for both nodes and edges which brings up a focus box with details to the side of the main visualization (or, alternately, a tooltip-style box). A pairwise focus box might show how subjects match up on different variables, along with their total proximity/communication data. Individual focus boxes might display data on subject attributes and preferences. This dataset is formally anonymized, but it doesn’t mean we can’t offer at least some window into the personal features of its participants.

**Filtering subjects by magnitude of activity**

**[andrew, apr 10]**

We have quite a few participants - 80-something in total - and so both the chord and force-directed layouts are a bit messy when we render the full dataset. So we’ve been thinking about ways to allow the user to filter out subjects they don’t care about. This could be done by setting a minimum activity threshold for the time series, which we partially implemented in the force-directed layout (we set a gradient color scale on edges that exceeded a certain size in the course of the animation).