# Evolution of Geo-local Social Networks

**Feeny-Reece-Sulkow Project Team**
**CS-171**
**Spring 2014**

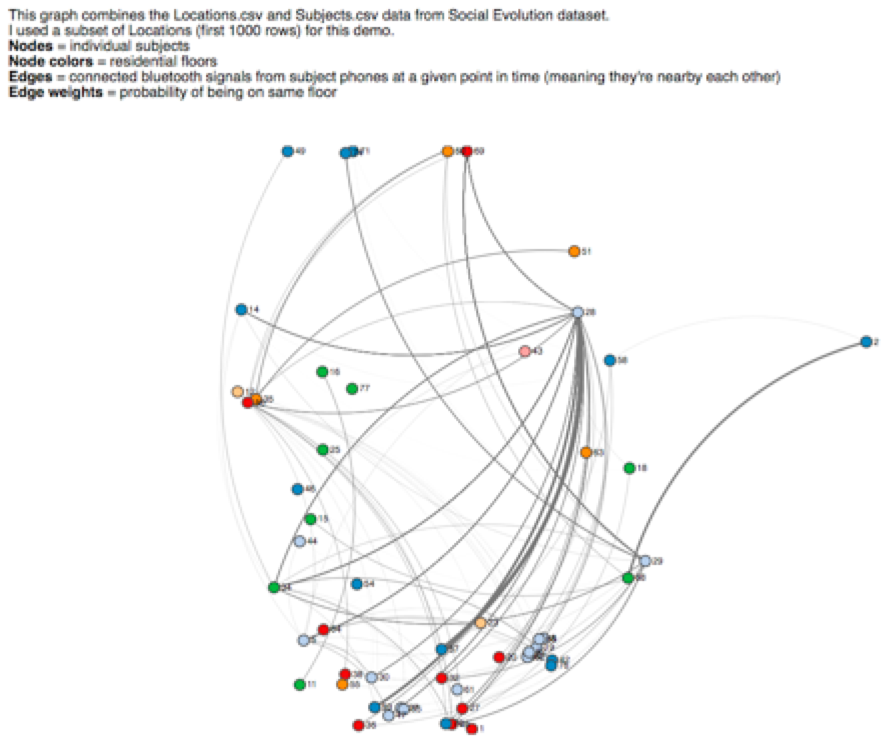
## Overview
**[jennifer mar 25]**

We are using the Social Evolution dataset from the Reality Commons project at MIT. This project used technology such as cell phones, Bluetooth devices, and WiFi access points to track the interactions of small communities of people. The Social Evolution dataset involves a dormitory of about 80 participants. Their locations, proximities, phone calls, and text messages were tracked, and they also completed surveys indicating their preferences regarding politics, music, eating habits, exercise habits, campus activities, and other topics. Our goal is to create a set of interactive visualizations showing these interactions and how some of the connections (proximity and communications) between pairs of participants may influence, or be influenced by, their personal preferences and habits.

**Preliminary Assessment of the Data**
**[jennifer mar 25]**

Participants in the experiment were tracked in a multitude of ways. Their proximity to each other was tracked via Bluetooth devices, their calls and texts were tracked via their cellphones, and different personal preferences such as musical taste, political opinions, health (cold and flu symptoms), fitness activities, friendship with each other, and eating habits were tracked via survey. We viewed the CSV data using spreadsheets and reviewed the papers and visualizations on the Reality Commons website before deciding that the most fruitful data were probably the proximity and communication (call/SMS) data, and that we would supplement it with the participants' musical taste and political opinions. The latter two items were less likely to change over time and could be seen as "attributes" of each person, while we examine the participants' changing relationships with each other by way of the changing frequency of communications between them over time.

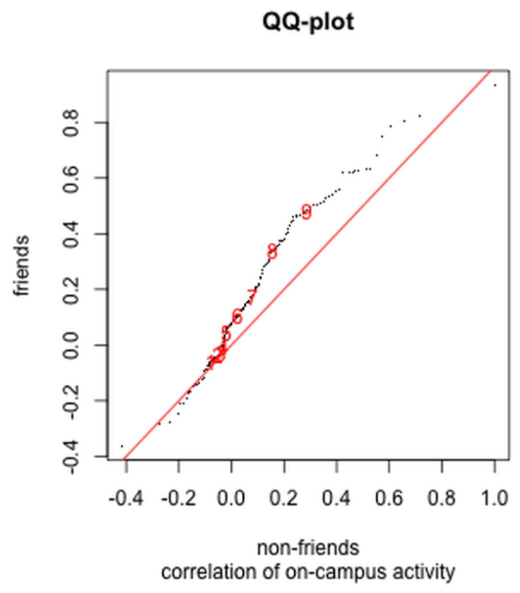
First Prototype, using proximity data:

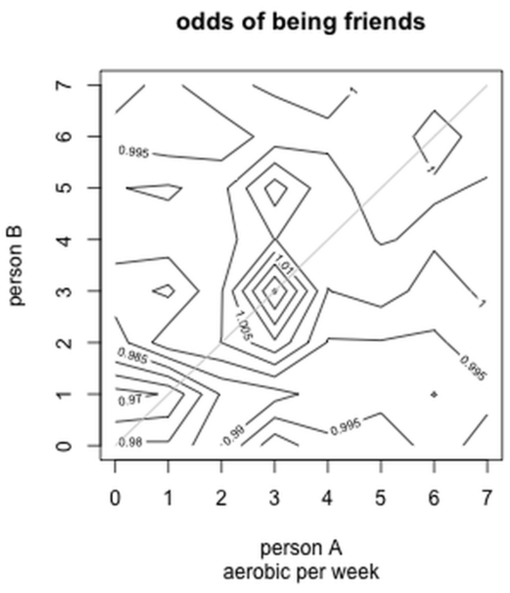


This graph combines the Locations.csv and Subjects.csv data from Social Evolution dataset.
I used a subset of Locations (first 1000 rows) for this demo.
**Nodes** = individual subjects
**Node colors** = residential floors
**Edges** = connected bluetooth signals from subject phones at a given point in time (meaning they're nearby each other)
**Edge weights** = probability of being on same floor

**Data description provided by researchers**
**[andrew mar 28]**

The MIT team provided a pretty good document explaining the details of the dataset, along with some analysis.  It leaves out description of a few parts (Activity.csv, for example, and some location info), but for the most part it's helpful.  A few interesting findings, although their visualizations are pretty ugly.





**[jennifer, mar 30]**
I created a function to create a square matrix from the SMS data and used it on a small file (the January SMS data).  As part of the data manipulation, if the file said the call was incoming, I switched the user/destination values so they would all be outgoing calls.  Also I still can't get the chord group labels to show up, even though I can see the text is with the path in the web inspector.

To get the csv data into a form the chord layout could use, I made a function to create a square matrix using user_id and dest_user_id.  First it gets all the unique users out of both fields and puts them in one 'users' array.  The length of the array is the number of arrays in the 'matrix' array of arrays, and also the length of each array within 'matrix'.   Then I walk through the SMS data checking the user_id and dest_user_id against 'users'.  When two matches are found, it increments the count in matrix at the position [u][u2] where 'u' is the index of the user's ID in the users array and 'u2' is the index of the dest user's ID in the users array.

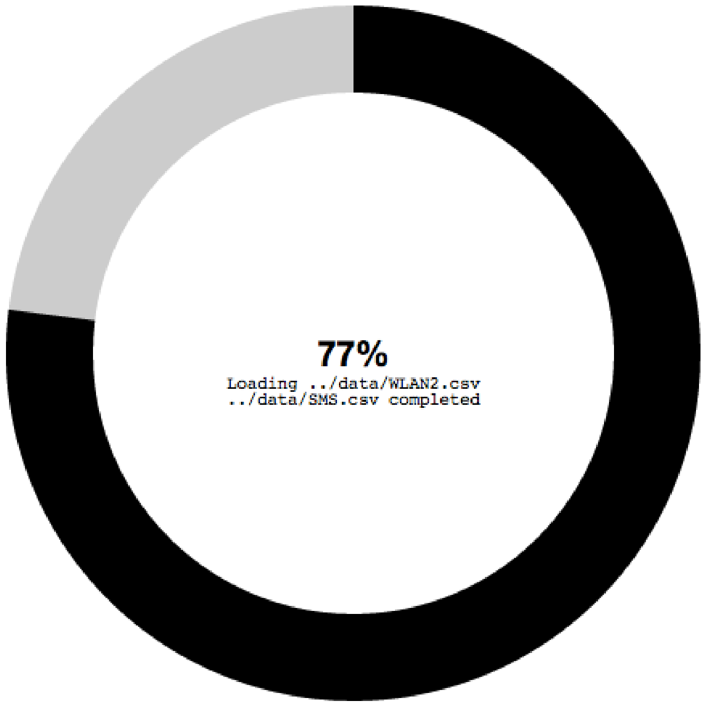
I used d3’s built-in support of colorbrewer to color the chords.

## Data loading and transformation
**[brian mar 30]**

I created a program which loads in every file of the MIT Social Evolution data (13 files). The program then does some basic transformations, such as making sure dates are proper Date objects. In doing this, I noticed issues with loading the Proximity.csv. This is a large CSV of over 2M rows. When loaded, it seemed to cause strange behavior with the rest of my program, such as freezing up or not loading other parts of the data. This could be a local problem. Because there was so many files (13) and some of them were quite large, I wanted to give some visual feedback to the user of what was going on while the data was loading. I was able to accomplish this by creating XHR objects to load each CSV, and then attaching progress event handlers to them. This resulted in a very basic way to display to the console the progress of the files being loaded.

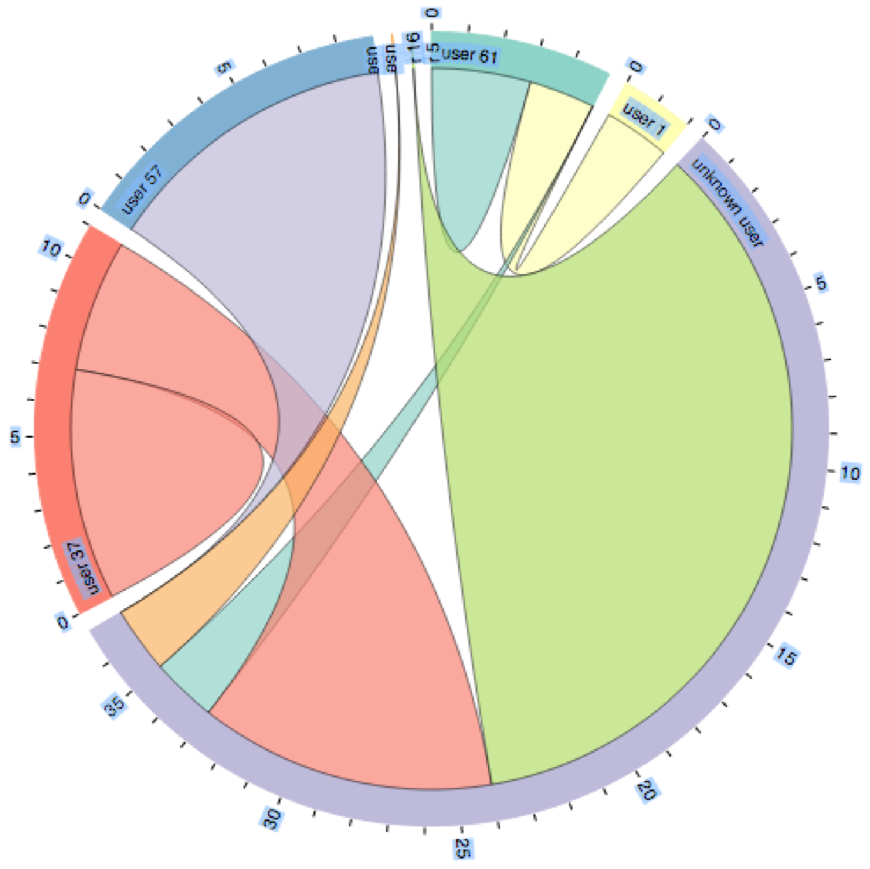
**Visual data loading**
**[brian mar 30]**

In order to improve the user experience of the files being loaded, I looked to give more of a graphic/visual feedback. I used http://bl.ocks.org/mbostock/3750941 as much inspiration on how to get this done. Rather than show the progress of each individual file, it made more sense to show the overall progress of all files being loaded. This is because the files are being loaded in parallel, using Queue.js. I came up with modified code that can display the overall progress of files being loaded and use the visual progress graph. It is not clear if we will use this in our final visualization, but if we have something that may take some time, it could be used to give visual feedback to the user.



```
Loading ../data/WLAN2.csv
../data/SMS.csv completed
```

**[jennifer mar 31]**
Got the chord labels mostly showing up (except in cases where the chord group is too small for them to fit. Probably need to make the labels go perpendicular to the chord's edge instead). Brian suggested dropping the data on calls/texts to unknown persons. I agree; we want to show the interactions between the people in the dataset. I will revise, but for now the chord looks like this:



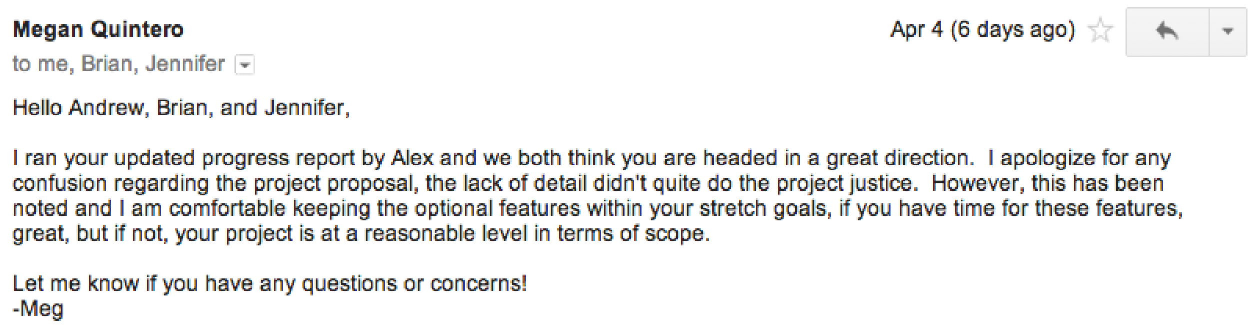
**Check-in with TF**
**[andrew apr 01]**

Megan gave us some feedback regarding our initial proposal, and we met her on a conference Hangout to discuss.  She felt she didn't have enough information in our proposal to get a good sense for the scope of our project.  We told her we'd send her a more detailed writeup of our project that she could bring to the TF meeting this Thursday, where she'd re-present our project goals and see whether they were deemed sufficient.

**[jennifer apr 2]**
Instead of working with the small data (one slice of time), I now have the chord diagram working with the data file Andrew put together, with tallies of calls between pairs at distinct two-week slices of time. I revised the matrix function to create a square matrix based on the new file format. With the time slider Brian is working on, I think things are really starting to come together.

**TF response**
**[andrew apr 05]**

Megan wrote and said that our updated proposal was good and that we were approved to move forward with our must-have and optional features as we'd described them.



**Megan Quintero**
to me, Brian, Jennifer

Apr 4 (6 days ago)

Hello Andrew, Brian, and Jennifer,

I ran your updated progress report by Alex and we both think you are headed in a great direction. I apologize for any confusion regarding the project proposal, the lack of detail didn't quite do the project justice. However, this has been noted and I am comfortable keeping the optional features within your stretch goals, if you have time for these features, great, but if not, your project is at a reasonable level in terms of scope.

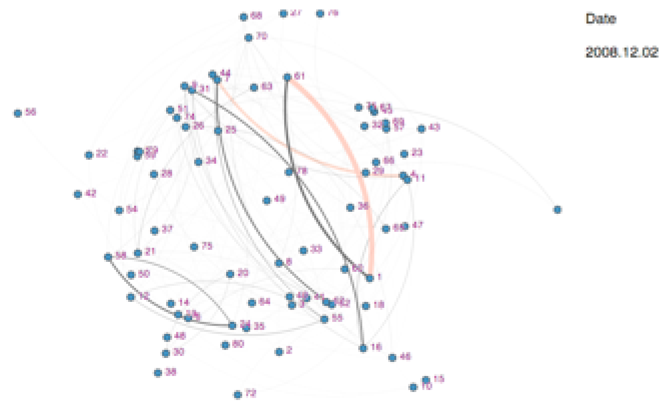
Let me know if you have any questions or concerns!
-Meg

**Visualizing the Data**

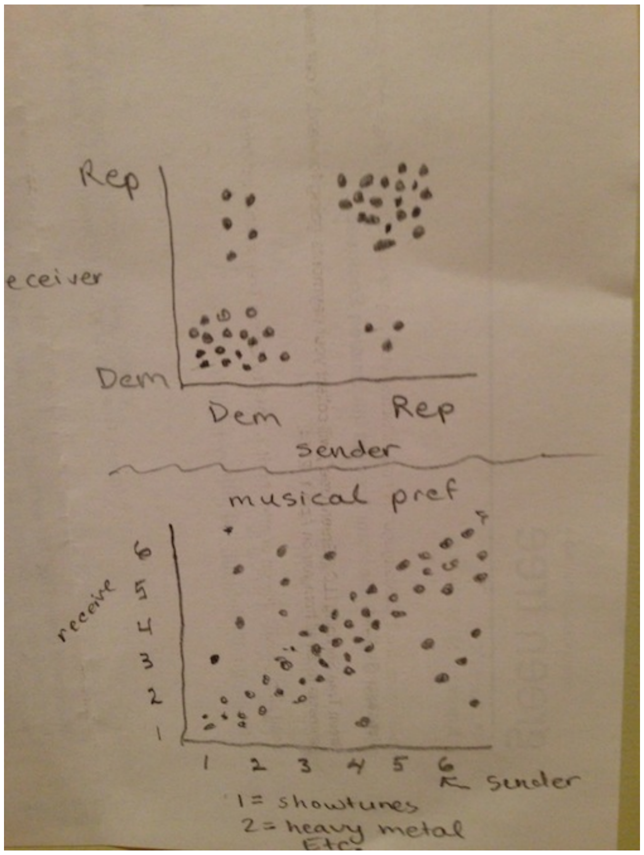
**[jennifer apr 08]**

Since we have multiple dimensions of data that we want to visualize about the Social Evolution participants, we've experimented with different types of visualizations and navigation flows.

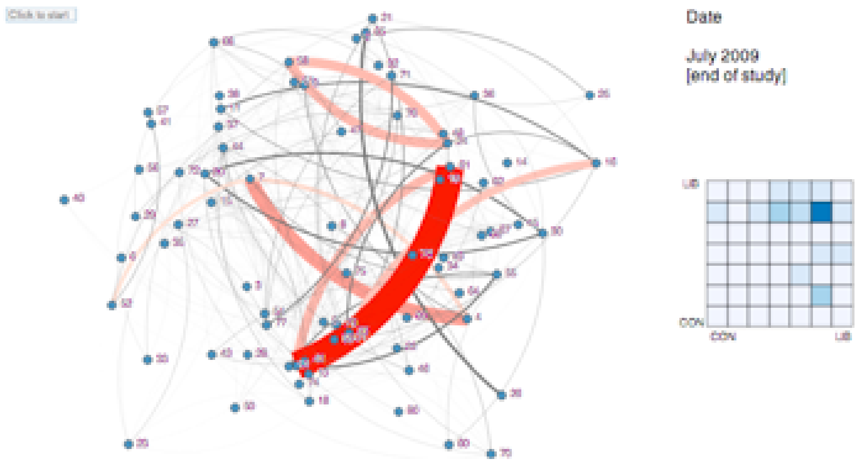
An early approach to visualizing the network connectivity of participants is shown below. The display is animated and cycles through the time period of the Social Evolution dataset. A snapshot from the middle of the cycle is shown. The edge weights indicate the number of calls/texts between pairs of participants. Alternately, we can swap communication frequency for proximity data.



We also want to show information about the pairs who are communication with each other. We have data on their musical preferences, political preferences, and other personal attributes. I've been trying to think of how to represent this sort of ordinal, multiple-choice data for a group. I suggested something like a scatterplot (sketch below) that might show up next to the chord diagram for the selected time period. Of the pairs who made calls to each other, the sender's attribute is on the x-axis and the receiver's attribute is on the y-axis. A cluster all along the diagonal means they share the preference.



Andrew pointed out that a scatterplot wouldn't be the best way to represent this data since there are only a few discrete variables and the points would all be on top of each other. He suggested a heatmap as a variation on the idea. The initial version of the heatmap shows how the calling pairs identify themselves on a spectrum from liberal to conservative, using a sequential color scale.



**Chord diagram tweaks**
**[brian apr 06]**

The chord diagram was transitioning along with a time series slider.  The transitions however were not smooth.  This is because the transition step within d3 just naively moves pixels from their current location to the new location, linearly, and this doesn't display well with a chord diagram.  What was needed in order to smoothly transition the groups (arcs) and chords was a tweening function.  We found this concept nicely documented on Stack Overflow.

A key function was also introduced to maintain object constancy based on d.index.

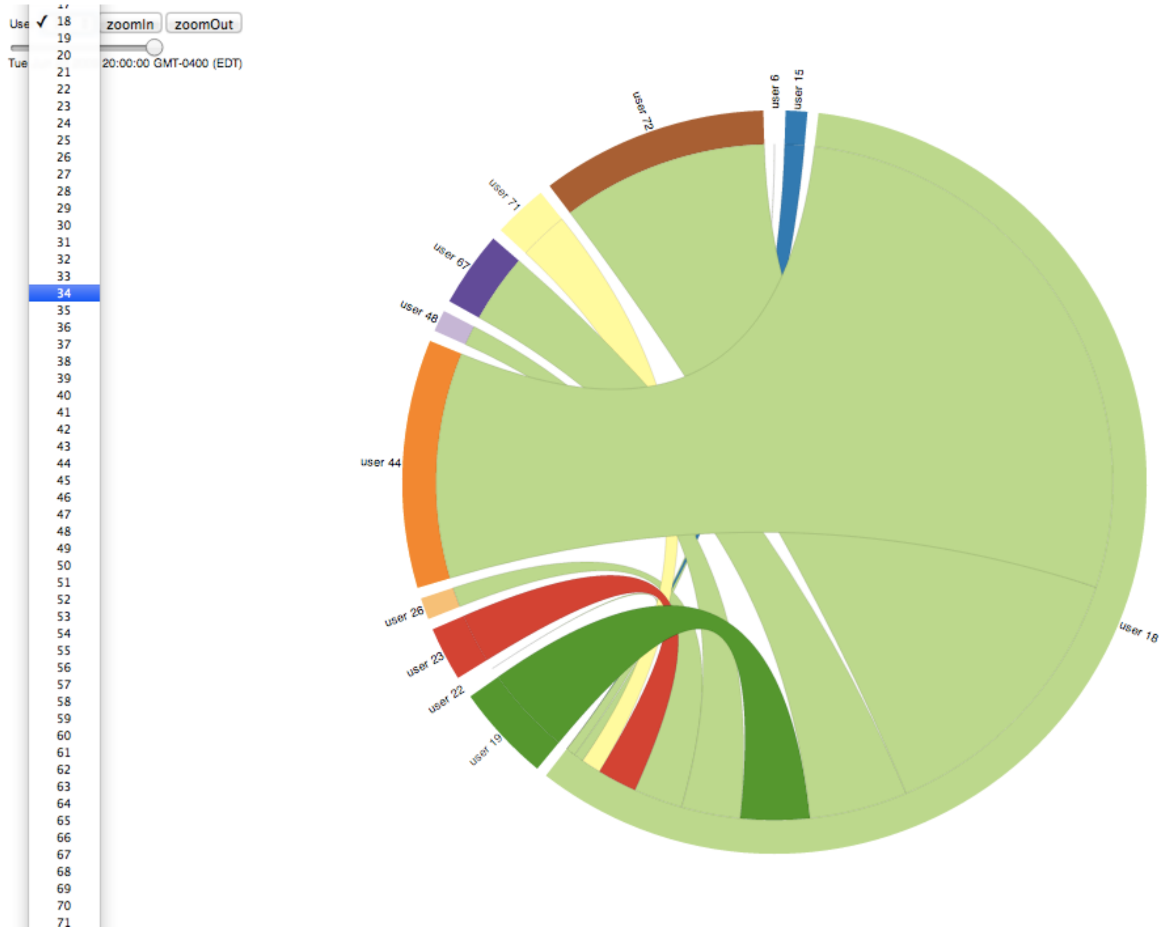
Group labels were moved to be at 90 degrees to their arcs for easier reading.  Mouseovers to the groups display total number of calls made from that user, and mouseovers to the chord display call information as well.

**[jennifer apr 8]**
I am now working on possible revisions to the force-directed graph to incorporate clustering of people who live on the same floor.  This is inspired by the feedback Andrew received in design studio 3.

There are numerous approaches to a clustered force diagram.  Some use the d3 pack layout, some use the d3 cluster layout.  I didn't want to get rid of our force layout, which has had a lot of work put into it and is currently showing communication connections with edge weights for the number of communications.  I want to add onto that layout.  The approach I'm trying involves assigning a node from each cluster to be the "cluster node" that the other nodes in the cluster will be attracted to.  So each floor (of the subjects' living quarters) is a cluster, each node is assigned to a cluster, and one node of each cluster is the "cluster node".   This approach is found in http://bl.ocks.org/mbostock/7881887.

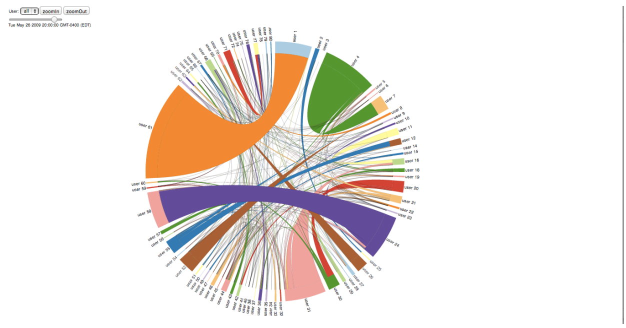
**Chord user selector**
**[brian apr 08]**

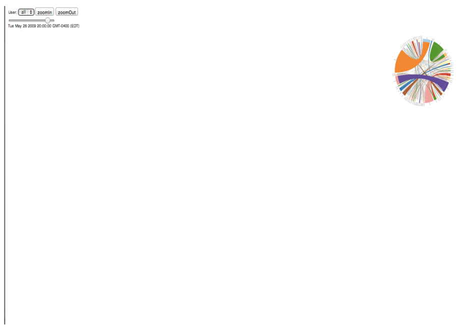
A chord will likely not serve as the main context graph in our visualization, but rather as a focus graph.  A user may be selected (clicked) and perhaps a chord built to show that user and their connections.  In order to model this on our current chord, I created a user dropdown which allows you to select "all" or a single user (1-80), and it will just show that user and their directly connected network.



**Chord graph scaling**
**[brian apr 09]**

The chord graph in its current state is fairly large (width/2, height/2).  It occupies more real estate on the screen than would be practical if we are going to have multiple graphs showing.  We discussed making the chord smaller, but were concerned that it may not be very readable.  I implemented the ability to scale the graph to essentially a fraction of its original size.  A button can be clicked to scale the graph down or scale it back to its original size.  This functionality may prove useful in our final visualization in order to make efficient use of our visual space.





**Design Studio 3 Feedback**
**[andrew apr 09]**

One interesting piece of feedback we got from DS3 was that it might be interesting to encode proximity time series data by actually having nodes on the force layout cluster together based on proximity tallies.  That's not as easy as it sounds - we'd have to come up with a representation of centrality and community parameters, for instance.  But it may be worth a shot, it's a cool idea. Bostock has some examples that may be of use.

**Data Wrangling, pt 1 - Overview**
**[andrew apr 10]**

Data wrangling has been a big part of this 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 .

**Data Wrangling, pt 2 - Time Series**
**[andrew apr 10]**

In the case of proximity data, we have geo-location measurements taken every six minutes for more than 18 months, for several dozen individuals. That's a lot of data. 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 frequency estimates for the number of times any pair of participants were in geographic proximity. We settled on two-week segments as reasonable intervals, and computed running pairwise tallies over the range of the time series. That gave us time series data for proximity estimates - we also carried out the same type of time series transformation for communication data. For that job we combined two separate datasets (calls and sms logs) and generated pairwise aggregate frequencies over the same two-week periods as in the proximity data. You can see the communication time series animated in our node graph demo, here:

http://goo.gl/33YJha

**Data Wrangling, pt 3 - Subject profiles**
**[andrew apr 10]**

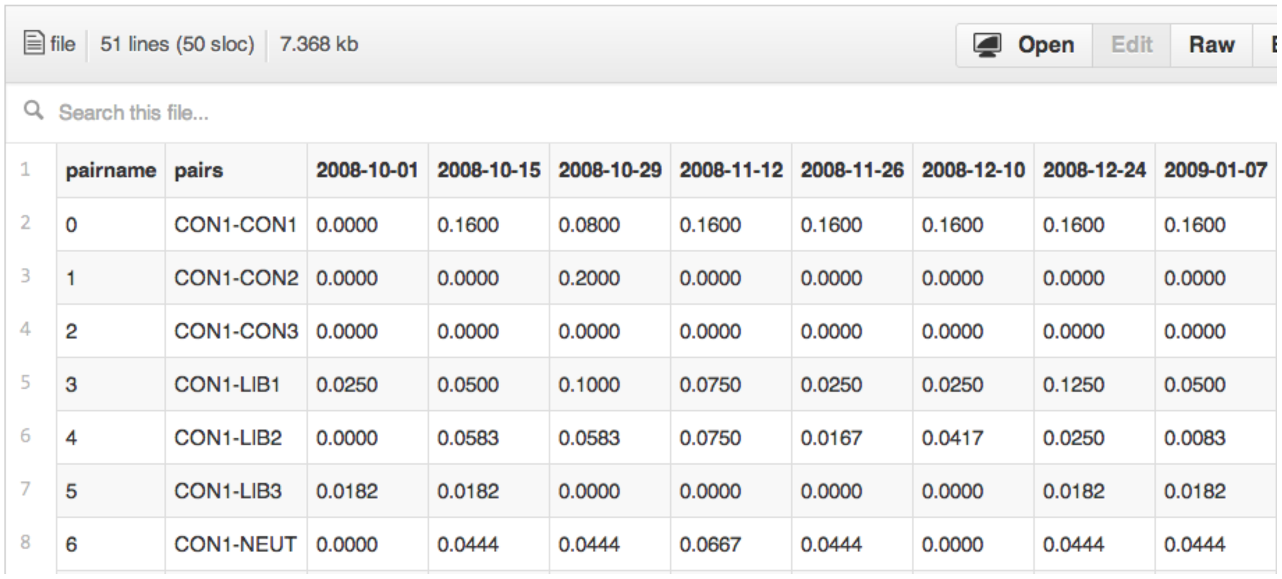
We are also interested in giving users the ability to better understand interpersonal relationships based on individual participants' personal preferences and behavior. For instance, it may be that individuals of a given political orientation (eg. liberal or conservative) are more likely to communicate 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.

**Data Wrangling, pt 4 - Heatmaps**
**[andrew apr 10]**

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.

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.


file | 51 lines (50 sloc) | 7.368 kb

Open Edit Raw

Search this file...

| pairname | pairs | 2008-10-01 | 2008-10-15 | 2008-10-29 | 2008-11-12 | 2008-11-26 | 2008-12-10 | 2008-12-24 | 2009-01-07 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | CON1-CON1 | 0.0000 | 0.1600 | 0.0800 | 0.1600 | 0.1600 | 0.1600 | 0.1600 | 0.1600 |
| 1 | CON1-CON2 | 0.0000 | 0.0000 | 0.2000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 2 | CON1-CON3 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 3 | CON1-LIB1 | 0.0250 | 0.0500 | 0.1000 | 0.0750 | 0.0250 | 0.0250 | 0.1250 | 0.0500 |
| 4 | CON1-LIB2 | 0.0000 | 0.0583 | 0.0583 | 0.0750 | 0.0167 | 0.0417 | 0.0250 | 0.0083 |
| 5 | CON1-LIB3 | 0.0182 | 0.0182 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0182 | 0.0182 |
| 6 | CON1-NEUT | 0.0000 | 0.0444 | 0.0444 | 0.0667 | 0.0444 | 0.0000 | 0.0444 | 0.0444 |

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

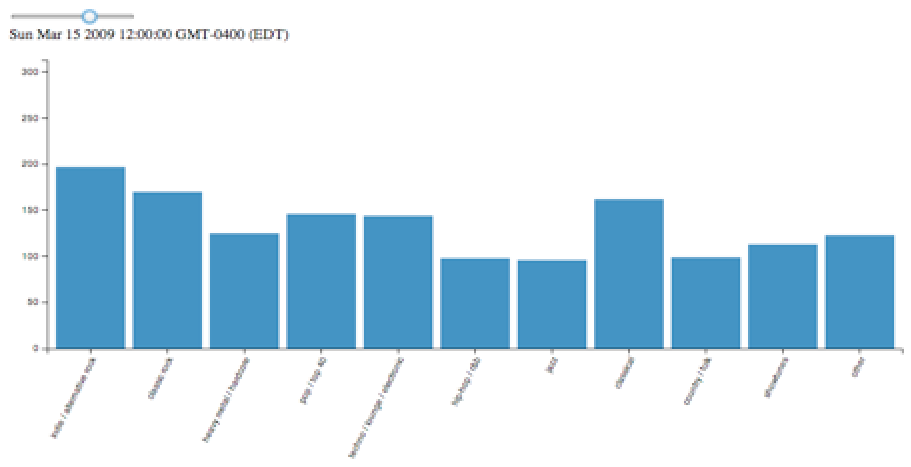
**Data Wrangling, pt 5 - Queue.js & Other issues**
**[brian apr 10]**

Wrangling the Proximity data presented some unforeseen challenges. While using Queue.js to attempt to load in several data files, Brian found that the full file of Proximity data, over 2 million rows, caused the process to crash for unknown reasons. Attempts to solve this included: using the Zap Gremlins feature in TextWrangler; loading the file into Google Refine and exporting a CSV; splitting the file into three parts and attacking it piece by piece. None of these worked, so Proximity.csv was set aside for the time being. After successfully wrangling the combined call/SMS data (above), a similar pattern was used to tame the Proximity data (using Python).

Different issues arose with MusicGenreImmersion data. In this dataset, users indicate the amount of time they spend listening to each listed genre, such as showtunes or heavy metal/hardcore. We wished to track the musical preferences as attributes of a person, but each person might select more than one genre of music. A given person's musical taste does not seem to change much over time.

Brian: "This data is a good example of data that may not be so useful as individual observations, but may be useful as an aggregate. I load the data, and materialize the aggregate."

Example of aggregate musical taste data:



Eventually we decided to weight each person’s responses on music preference and take their top choice as an attribute of the person.

**Time Series Visual 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, pt 1**
**[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.  We threw together a rudimentary implementation in the force-directed layout.  (Technically it's not filtering out low frequencies, but rather emphasizing high frequencies.)  In it, we set a gradient color scale on edges that exceeded a certain size over the course of the animation.  It may be cleaner to just fade out or remove edges that fall below threshold, rather than to emphasize edges that exceed a threshold.

**Filtering by individual selection**
**[andrew, apr 10]**

I don't know how hard this will be to do, but it also would be nice to be able to select a specific pair of nodes, or even a small community of nodes, and then run the time series animation using just the subset the user selected.

**Predictive models and user-adjusted stats**
**[andrew apr 10]**

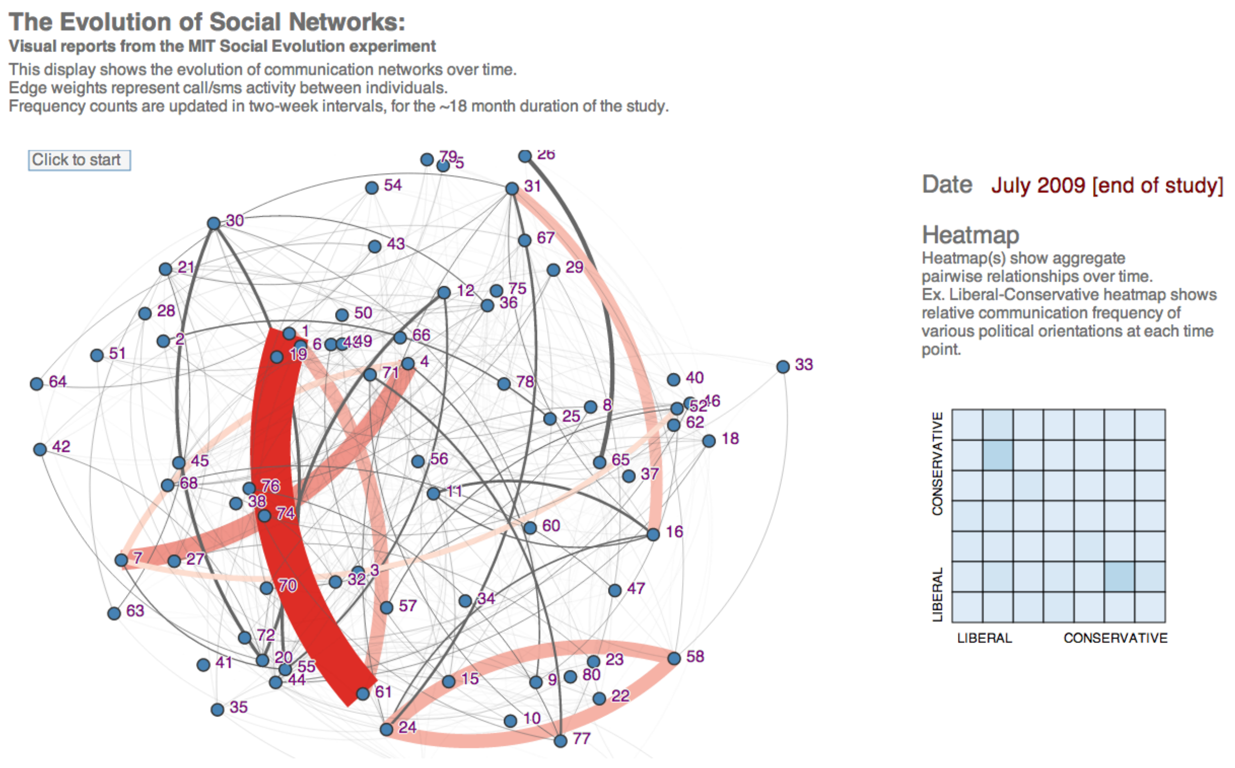
This may be more of a wishlist item, but if we really want to model the formation of social networks based on this data, then we should implement a model with adjustable parameters for individual datapoints. That would require some kind of statistical learning algorithm, which may be beyond the scope of our efforts here. But one could imagine that a clustering algorithm like k-means or hierarchical agglomerative might detect certain subject variables that indicate likely community formation. Once a model is fit, we could give the user a panel of adjustable knobs for each parameter in the subject profile (political orientation, music prefs, average mood, etc), where one could either make individual subjects more liberal or conservative, and then see what that did to social network formation given the new individual parameters. Or one might instead adjust the average level, of, say, depression, in the population, and the model would generate a prediction of network formation based on the new depression levels.

That's a lot of work! But this is the sort of merging of data viz with machine learning that turns visual storytelling into a viable predictive tool.

**What type of change to show?**
**[andrew apr 10]**

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. For example, it's clear in the screenshot below that subjects #1 and #61 communicated the most over time. And during the animation, you can see their communication tally grow over time. But that's not to say that this pair also had the biggest derivative at each new time point. It could be that they had a burst of communication early on, then didn't communicate much at all afterwards. Although the keen-eyed user might still spot that their edge width wasn't increasing much after the first few steps in the time series, the implementation as it is doesn't make derivative change obvious.

The heatmap, on the other hand, currently only shows any new updates to pairwise tallies at each new time point. It is, in a sense, memory-less. That's kind of nice during the animation, although we discussed that perhaps once the animation concludes, the full heatmap "trail" can be displayed on the grid - otherwise the user is only left with the heatmap print from the final time period, which may not be representative of the average print across the entire time series.

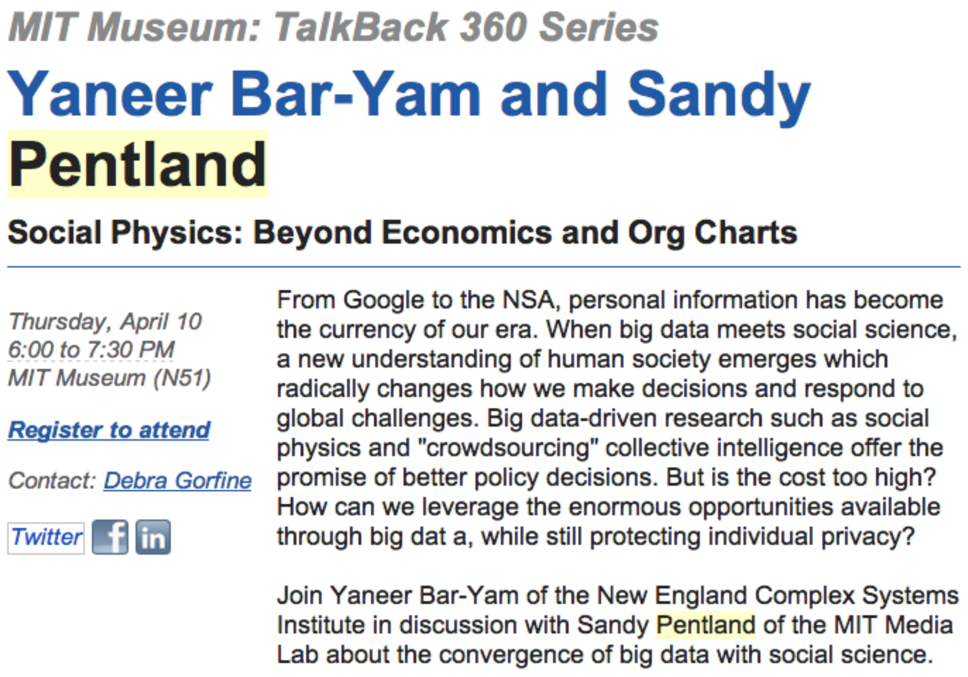


The Evolution of Social Networks:
Visual reports from the MIT Social Evolution experiment
This display shows the evolution of communication networks over time.
Edge weights represent call/sms activity between individuals.
Frequency counts are updated in two-week intervals, for the ~18 month duration of the study.

Date July 2009 [end of study]

Heatmap
Heatmap(s) show aggregate pairwise relationships over time.
Ex. Liberal-Conservative heatmap shows relative communication frequency of various political orientations at each time point.

**[jennifer, April 10]**
Today, Andrew and I attended a talk by Alex Pentland, the director of MIT's Human Dynamics Lab who created the Reality Commons project. He was pleased to hear that we're using his data, and his opinion is that it's most interesting to look at the communication/proximity data in conjunction with some of the survey data, which is what we are doing with the heat map and force-directed graph.



*MIT Museum: TalkBack 360 Series*

**Yaneer Bar-Yam and Sandy Pentland**

**Social Physics: Beyond Economics and Org Charts**

*Thursday, April 10*
*6:00 to 7:30 PM*
*MIT Museum (N51)*

***Register to attend***

*Contact: Debra Gorfine*

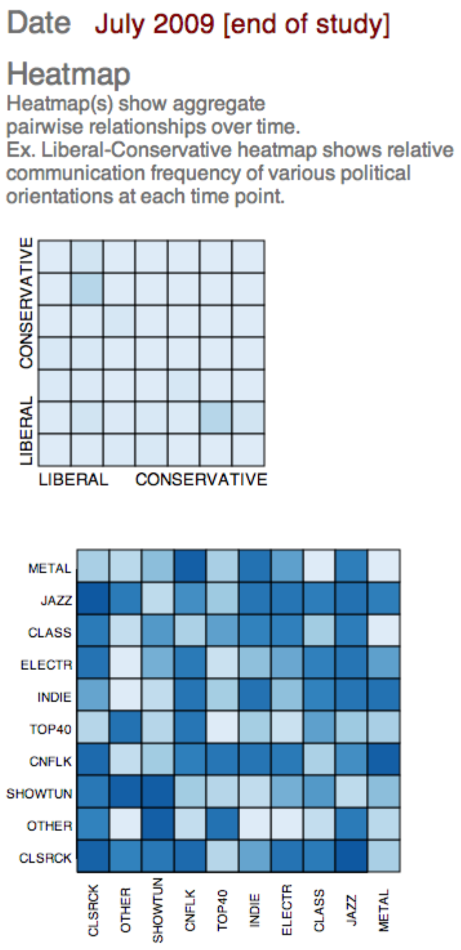
*Twitter*

From Google to the NSA, personal information has become the currency of our era. When big data meets social science, a new understanding of human society emerges which radically changes how we make decisions and respond to global challenges. Big data-driven research such as social physics and "crowdsourcing" collective intelligence offer the promise of better policy decisions. But is the cost too high? How can we leverage the enormous opportunities available through big dat a, while still protecting individual privacy?

Join Yaneer Bar-Yam of the New England Complex Systems Institute in discussion with Sandy Pentland of the MIT Media Lab about the convergence of big data with social science.

**Heatmap update**
**[andrew apr 10]**

Music preferences heatmap is working now. Also, for music pref heat map only I set the final display to be the aggregated imprint across the entire time series. So it still animates in the same way as the lib-con heat map, but once the animation is over, it displays the average pattern across time points. (Doesn't seem like much of a story there, but we can decide that later.)



Date July 2009 [end of study]

Heatmap

Heatmap(s) show aggregate pairwise relationships over time.
Ex. Liberal-Conservative heatmap shows relative communication frequency of various political orientations at each time point.