

Project 3: Telling stories about Chicago Crime

Process Book

0: Proposal

Participants

- Jeff Adams
 - jeffisadams@gmail.com
 - Programming comfort: 4
- Matt Lutze
 - mail@mattlutze.com
 - Programming comfort: 4

Research questions and hypothesis

We propose to study the changes in crime in Chicago, IL USA over the last ten years. We plan to track trends in types of crime, increases or decreases in crime by location, and attempt to identify interesting, descriptive and statistically significant correlations between the crime data and demographic/socioeconomic vectors.

- How has crime changed in Chicago since 2003?
 - Where did it occur, and where does it now?
 - What forms of crime have risen and fallen?
 - What have the “designer crimes” been in the last 10 years?
- Are there macroeconomic motivations that are demonstrably related to the commission of crime in Chicago?
 - How do convictions correlate to demographic markers in Chicago?
 - If a kind of crime, or crime in general, has moved geographically, has it followed other migratory trends in the city?
 - Can we make any predictive observations about the trends?
- Is there correlation between changes in policing methodologies and the commission or conviction of crimes in Chicago?
- Can we correlate crime trends to historical public or notorious events?

Our general hypothesis is that crime in Chicago will show moderate to strong correlation with some socioeconomic and geographic variables. We further hypothesize that a correlation between crime commission and socioeconomic migratory trends may extend to the prediction of future criminal activity locations.

Motivation

Our purpose is to come up with an important and pertinent visualization to the general public, or at least Chicago. Visualizing trends about social media is interesting and useful, and sometimes very compelling to the human experience, however, a visualization about current and historical crimes committed has an immediate and lasting impact on the viewer in making future living decisions.

One of us is a recent Chicago transplant while the other grew up only a few hours away; Chicago is both home and icon, frontier and cherished memory, important for various reasons to both of us. The city is a bustling center of opportunity, yet continues to experience significant social loss from violent and egregious crime. Constructing accurate, accessible, and rhetorically persuasive visualizations of such addressable blight can help inform new resident decision-making, while also empowering local leaders and influencers to engage their communities.

Data

<https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2>

Further data regarding demographic and socioeconomic statistics will be collected from US Census, State of Illinois and/or City of Chicago information sources. We'll collect the data either as a static download or, if time permits and it's appropriate for the source, through automation scripts to call source APIs.

One notable type of data lacking in this data api are murders committed. If time allows, that would be an interesting category for our analysis, and more effort should be made to find data that would comply with our existing data set for murders committed in Chicago.

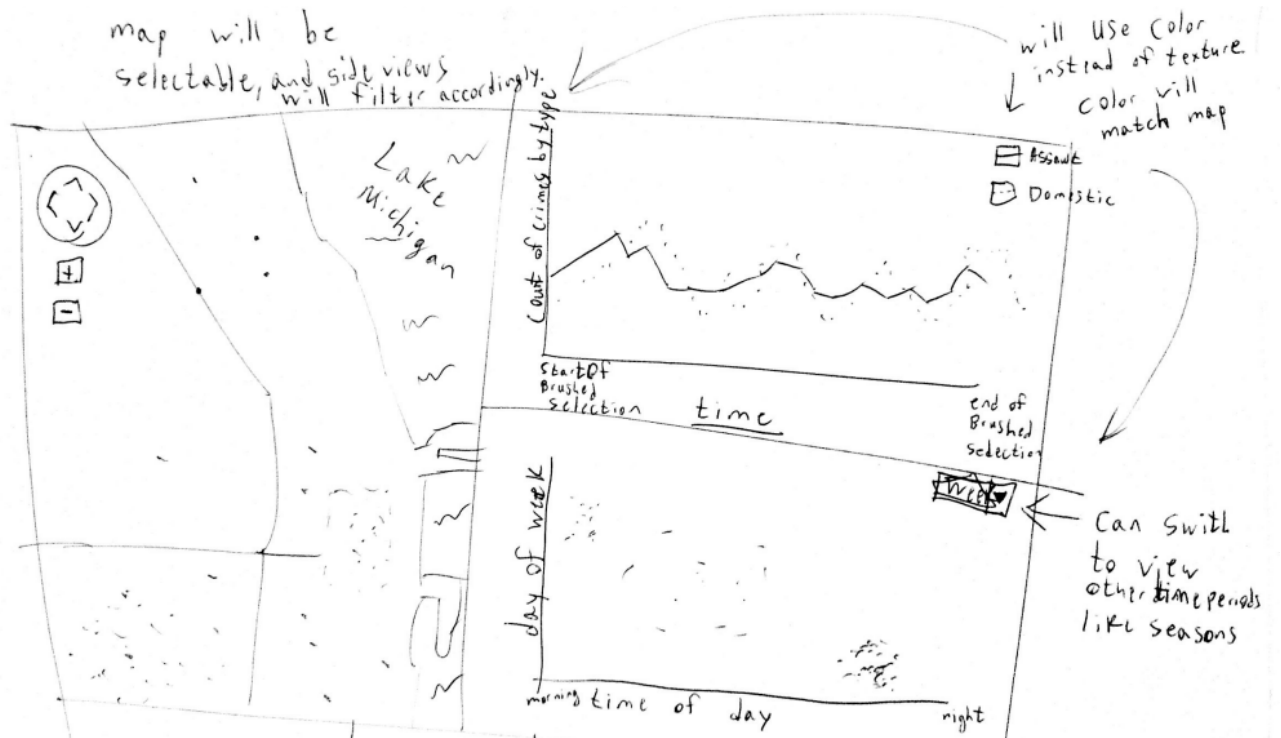
Visualization

We plan to encode aggregate data trends on a geographic map as a central component of our design. Accompanying the map will be scatter plots surrounding trends established in numbers of crimes separated categorically by type over time. Particular crimes may be isolated and their change tracked over time as clustered bar or line graphs.

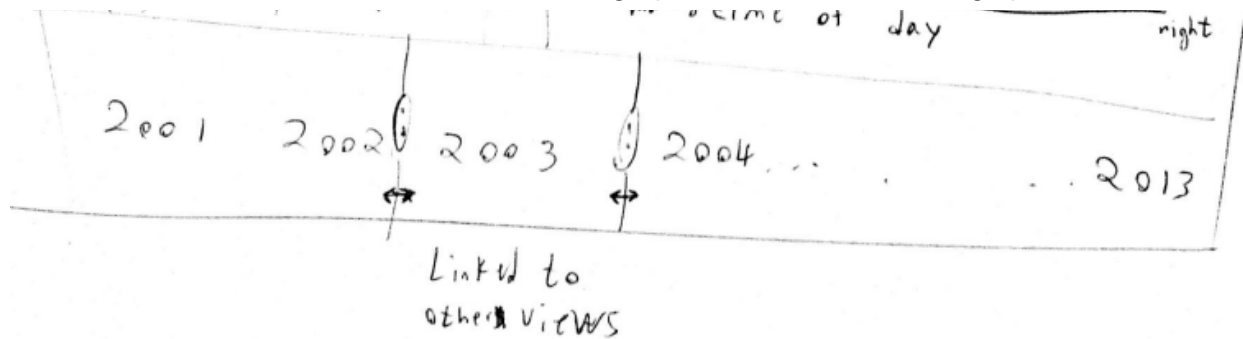
Ideally: One possible compelling visualization would be a map with crime locations, and then allow the user, or automatically advance time. The ability to see hotspots for large amounts of crime could be particularly compelling. Maps and charts may also be viewed as summed counts per locality—ward, neighborhood, etc.—to map the change in crime between time units. With even more time available, we'll would attempt and pursue visualizations which extend trends into the future, based on status.

Sketches & Concepts

The main dashboard view with side by side views. Left being geographic, Right being aggregate measures.

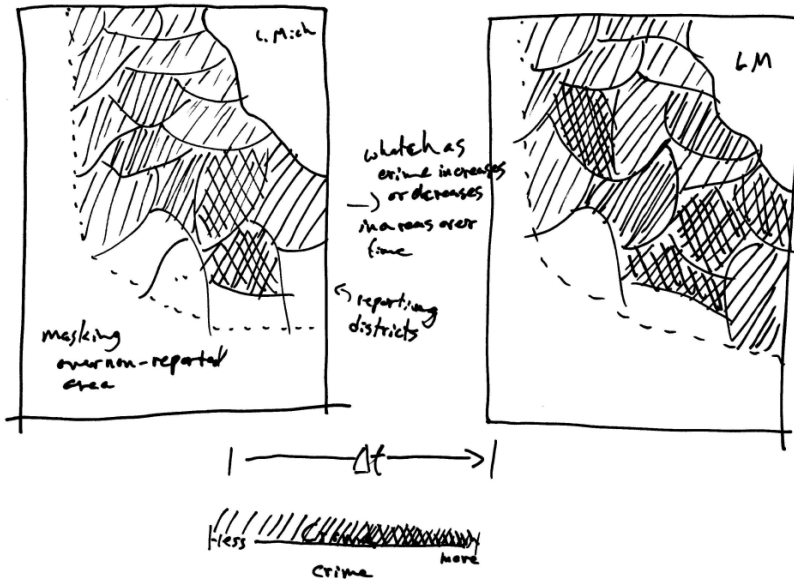


The main control housed at the bottom of the graph and used for brushing specific timeframes.

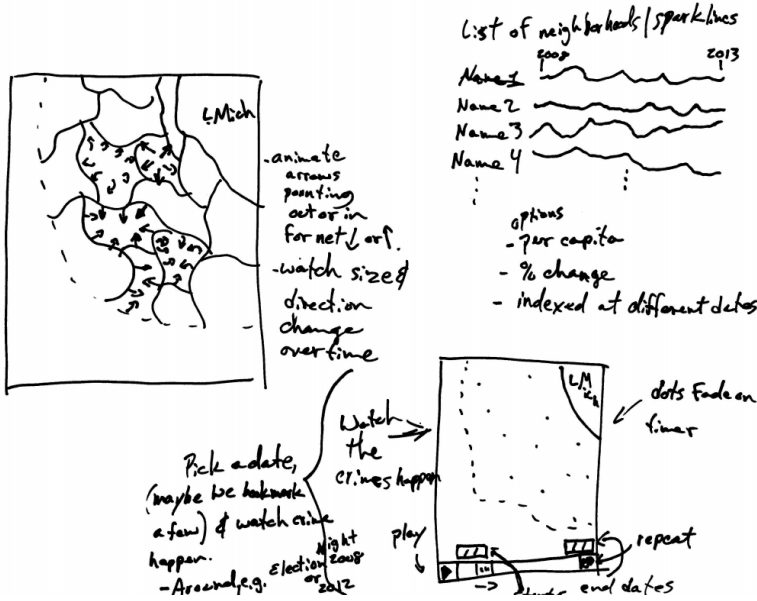


The idea for the main Hypothesis is to see how crime is changing over time.

over time, where has crime migrated?



Another possible overall dashboard view. The tabular data in the upper right might be useful on more specifically filtered selections of data.



1: Data acquisition and cleaning

The foundation of a good visualization is clean, fast data. A slow

When should we get the data?

What information is important? What information isn't? We had some deep foundational questions to answer, but what turned out to be particularly pernicious was the question of *when*.

Discussion:

Does the browser make an API call, or do we store the data locally and send “old” information to the user? The important distinction we had to make was how “fresh” our visualization had to be. After close consideration of the data, we discovered that a meaningful lag time existed from when a crime was committed to when it was uploaded—meaning it could take days to weeks for the information to appear.

Additionally, we had to consider performance. Even with a trimmed dataset, we were looking at 6-7 MB of data per month. The three months that we include with our visualization here clock in just over 14MB. Having to get this data from an API call would have further increased load time and reduced the user's potential enjoyment of our application.

Decision:

The experience for this visualization argued toward caching over an API call. Technical aspects discussed below further pushed our decision toward a cached delivery.

Mitigating connection errors for live data acquisition

There are many risks to using a live connection:

- If we have the client-side poll the data source API directly, there is a threat of network failure.
- Home or public networks may block the remote API call.
- If the API is modified, our code may become unusable until we discover the change and fix it.

Discussion & Decision:

We made the decision to use flat data files for now encoded as json files. Utilizing an external API is an interesting aspect for adding real time data into our analysis, however, the time we would have spent implementing that api would have made initial progress much more difficult. Our plan was to add the data directly through json encoded files, and writer the controller for handling and parsing the data such that we could refactor that class and get the data from elsewhere without having to change the higher level visualization.

2: Application development

How should we

The primary development start was in determining the parsed data structure. We made a controller that would save live parsed objects with the aggregate data and then getters and

setters to interact later. We also made the decision to lower the footprint of the raw data by removing unused columns from the raw data.

Discussion & Decision:

One key element to the underlying dataController object was to make the load methods replaceable so we can make improvements or live update methods in the future. To this end, there is a method initialization and callback structure to get the data parsed initially. This proved itself useful later on in the development cycle by allowing us to set filters for the raw data and then just fire the parsing methods again. This allowed us to recalculate new aggregate data easily, and helped set a strong metaphor for working with the dashboard as a whole. We didn't end up using the D3 filtering functionality because we would have had to join all of our raw data to the dashboard, and compartmentalizing and separating the aggregate data objects allowed for more in depth calculations on the fly with minimal speed costs.

Resolving repository updates

The process of using a repo proved very worthwhile. I recently bought a raspberry pi and set up our own svn server for this purpose. It had a few reliability issues as I'm new to server admin stuff, but overall it made adding changes significantly easier. You can get a checkout of the codebase at <http://homebase.archiverecording.com/svn/chicagocrime/trunk/>.

When there's too much data

Jeff had a look at the full data set we were considering using and discovered it would end up being around 1 Gb of data—far, far too much to consider processing in the browser.

Discussion & Decision:

We had to look at both how much data we needed to still visualize interesting and useful information. We also had to look at how much of each record was necessary.

There is a further opportunity to do some pre processing of the full dataset for our project later on. Or there is an open api that we could make calls to get filtered data from. This option could cause trouble with sluggishness in our visualization overall, but would give us up to the minute information so long as the server is running.

Our current solution to this problem has been twofold: first we ended up using less data than we originally intended. The date starting point for our data set is the start of the calendar year in 2013 through to the present. This made the data realistically usable in the browser context. Our second solution was to add a splash loading screen to the start of the app in interest of making the experience better despite the longer load time. This added a bit of polish to the visualization, and seems to be the best solution for now.

There is a further opportunity going forward to house the 1GB full dataset on a server,

and create a service that parses the data into our more manageable aggregates. We could use these services with python scripts to cut down on the footprint of the download. There is also an opportunity to try and utilize the open API for queries in an effort to make the information always current.

Data Processing specifically for Project3

One of the initiatives I was trying to get to was to create a service that would return the data we needed on the fly decreasing load time at the expense of subsequent specific data loads. Initial attempts were made to make these calculations by iterating over the whole data set, but this proved unusable as the full data set had 5,000,000 rows.

Since the full iteration didn't pan out, I tried to run a series of "grep -c" commands to get the counts, but this proved unusable as I had to run a grep for every date, and subsequently every primary category. Then for every ward, and every category.

These attempts are catalogued in the *services/index.php* and in *services/text.py*.

Once I ruled out doing processing on the fly, I attempted to make a pre processor that would prepare the data before hand, but this similarly did not pan out as I would need two different files and a good way to essentially do a join query, so outside of a mysql db, I didn't see much that would get us the full data set.

Instead of pursuing a sql db, I tried to truncate the data down to only the columns we needed so I could get more rows into the js version. This was successful, and then I just converted the remaining part to json so it's easily read and processed in JS. I was able to get about 500,000 data points into js before the load time became too painful to work with, so we now have from October 2011 through april 24th 2013 in our dataset which is an order of magnitude improvement over the previous.

SVG sourcing considerations

The svg map of Chicago was taken from a github repository located at <https://github.com/smartchicago/chicago-svg-ward-map>. The intent of the full application code provided is to call the map with javascript code and dynamically draw it. Instead we extracted the raw svg code and use it as a static starting point on the index.html page.

What to do when an SVG takes a while to load, and messes up your layout because of it

Map is drawing over Timeline's space because timeline is drawn first and map rejects its prescribed dimensions.

Discussion & Decision:

- Setting a max-height of 100% for the SVGs in the CSS didn't work.

- Setting a height explicitly to the ID for the SVG in the CSS didn't work.
- Setting a height in the SVG's element tag in the HTML page didn't work.

Decision:

It seemed best to create static heights to our svg graphs, and to create a start animation once the data was loaded which gives us a chance to avoid foux. There is likely a more involved dynamic resizing opportunity to be had, but our primary learning objective is in visualizing the data as opposed to mastering css3 media queries or javascript resizing.

Follow-up: Some SVG's heights are set to refresh on the page height. Weird.

When we resize the browser window, the svg view window changes its height as well, and I think this is part of what's messing up the layout, above.

Discussion & Decision:

Initial attempts at making the layout entirely responsive by using css and javascript to dynamically size and respond to the resize event proved useful, but the amount of time to produce a polished outcome proved distracting to the progress of making a good visualization. We decided to split the difference and make the page more static.

3: Data visualization

Considering the individual impact of a ward on the total information

Do the most crime-heavy wards contribute the most crime universally?

The answer was both yes and no.

One surprising aspect about the visualization is how consistent proportions of crime is in general areas, and in regards to category comparison. Selecting a smaller time period changes the total count, but the wards with high crime stay relatively constant in comparison to the other wards, but more surprisingly, the proportion of types of crimes stays very consistent as well. Theft is the continual winner in number, but it's visual proportion to the next high count is relatively constant despite the selected ward or the selected time period. This implies that the crime types and geographic locations is relatively stable and constant.

Color in relation to the crime information

How do we show this information without raising unnecessary ire from the viewer?

Discussion:

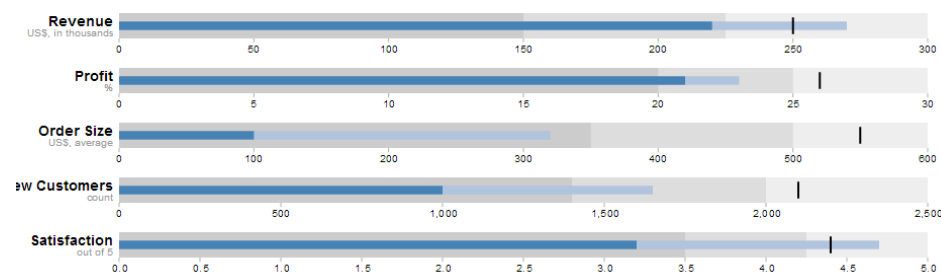
Crime is a funny thing, not in the least because it effects everyone. Our goal wasn't, however, to

upset the audience just by viewing the visualization, however. It was important to balance a color palate with high visibility against the emotion that color can inspire.

We know from our readings that the retina has a low density of blue-sensing cones, making these a poor choice for detailed graduated information. It works fine in a dark hue offering much contrast with our white background. We paired it as an accent therefore with a brown-yellow graduation, as the eye perceives these colors with much higher detail. By choosing a more neutral palette, further, we ensure that we avoid the emotional trigger that reds provide.

Design considerations in the bar chart

In order to show the proportion of crime in specific wards relative to the total, we used a bullet chart within the bar chart allowing the user to compare length according to the shared axis. This was taken from <http://bl.ocks.org/mbostock/4061961>.



4: Debrief and Action Items

Our project was nominated for one of the top project spots! It was incredibly exciting and confirming for what we'd felt was a rewarding experience. Jacob (Pritt) had quick turnaround on useful feedback from our project and the good news.

The most visible suggestion was adding a title... it's stunning sometimes how easily you can miss something so simple as a page title when you look at it without one for so many hours over so many weeks. We've added this, and will ensure it doesn't disappear when we submit our Project 3 deliverables.

There were a number of bug fixes that were suggested, and we'll be tightening up the UX for the focus locking on the wards while adding more information when locked, making the utility of the feature more apparent. We'll also be adding tooltips and measurements to our graphs to make them more accessible for browsing.

Finally, Jacob suggested that we expand the discussion of our visualizations chosen, which we plan

5: Development improvements

Project 3 realizes significant improvements on Project 2 through: improvement and expansion of the data, refinement of the visualizations presented, and new functional capabilities to support and extend the ability of our project to tell stories.

As discussed above, we built a Python-based data processing component to host on a remote server, to which we make calls and get processed slivers of our 1GB+ dataset (the total set is obviously too large to transmit and process raw in JS—even the 12MB we used for Project 2 took too long). This allowed us to use 12 years of data and support the bookmarking ability described below. We further added arrest data to support a secondary storyline in our visualization. Each data point includes whether it resulted in an arrest; the new dimension would add a new order of opportunities for discussing crime and where it happens in Chicago.

After addressing the bugs discussed above, we refined the way that brushing, wiping and locking the components worked. We analyzed the various edge cases and ensured the visualization interface was smooth and predictable. Most notable is the new slideshow navigation, which uses time bookmarks to programmatically present highlights in our dataset and support the stories we've programmed.

Core of the visualization

The data is again encoded with a geographic map as a central component of our design. Accompanying the map is an improved bar chart, using nested bullet charts to visualize the contributions of specific geographic wards to the total crime category count over a selected timeline, and put a particular ward's activity in context of total commission and arrest.

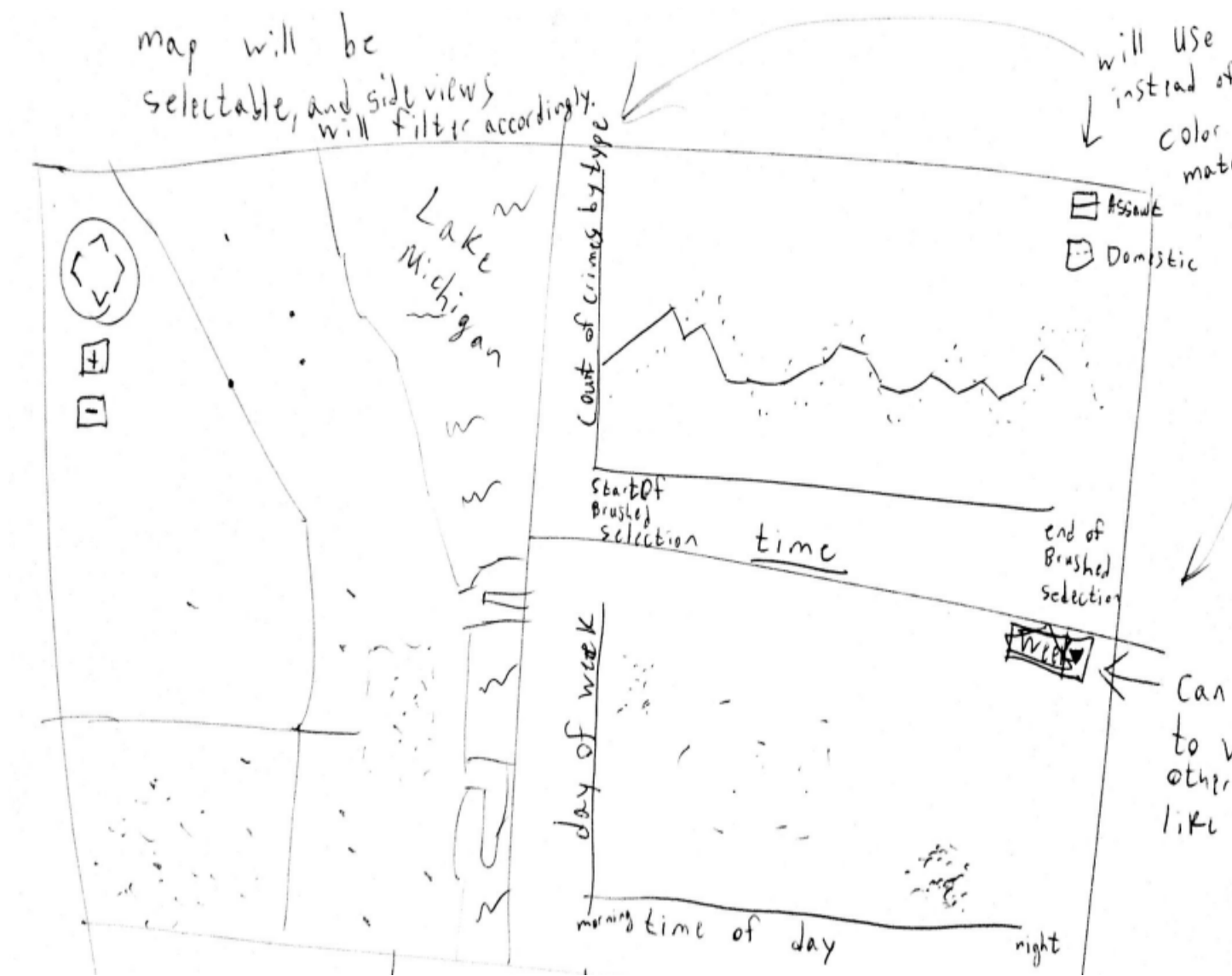
The slideshow is the core of our storytelling component.

Scope of Storytelling

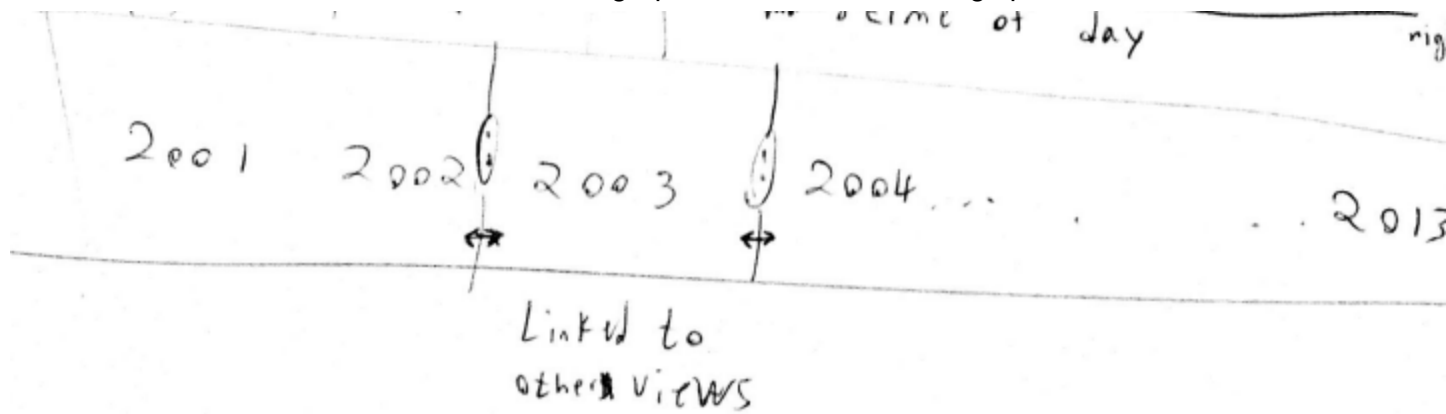
We've explored two avenues of storytelling. Our visualization introduces the scope of our story space as the page loads, explaining basic facts on crime and events in Chicago over the last few years. As the user clicks through the various slides, we discuss interesting points in time, using our bookmarking tool to advance through various points in time and displaying commentary about those events. Users will explore the data by bookmark, walking through the various points in time. Then, at the end, the experience is left open-ended for users to explore the data themselves, finding interesting meaning through locations and periods of time.

Sketches & Concepts

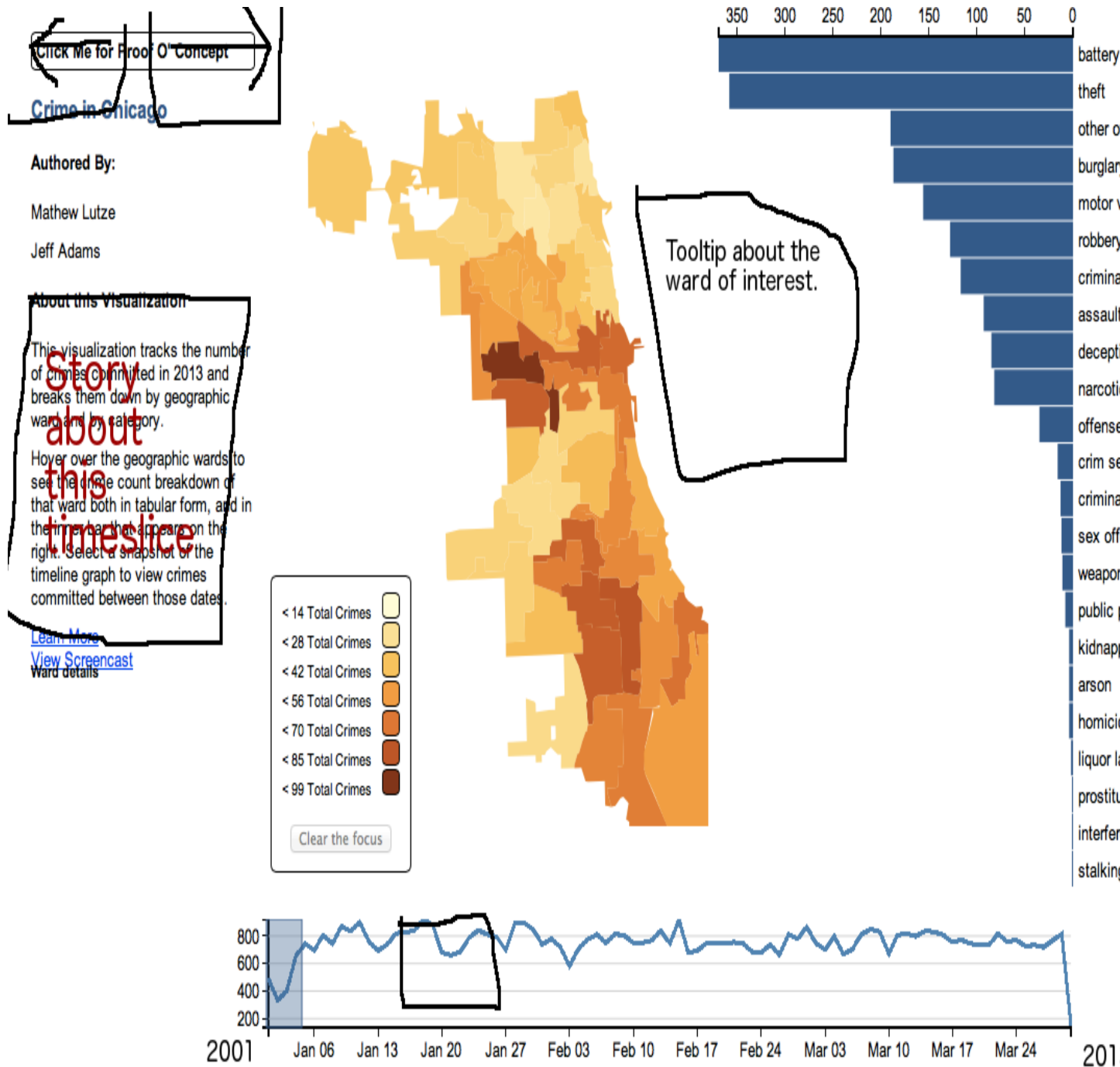
The main dashboard view with side by side views. Left being geographic, Right being aggregate measures.



The main control housed at the bottom of the graph and used for brushing specific timeframes.



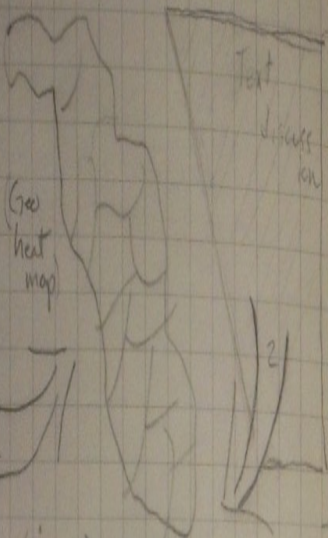
The idea for the main Hypothesis is to see how crime is changing over time. To this end, the primary interaction change is to add slideshow buttons that will tell a story through time snapshots in the visualization.



If time permits, we'll approach the arrest information additions, sketched and described below.

Title

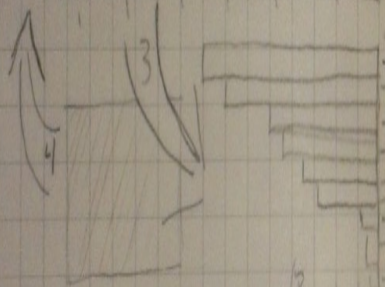
Intro
info



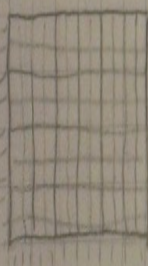
[Show me
intro]

Timeline

Crimes
Arrests

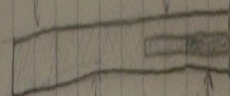


Bar chart
of crime
type



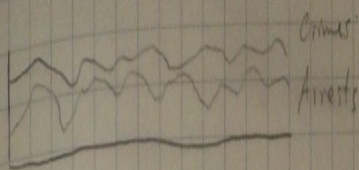
Total crime

Number of crime

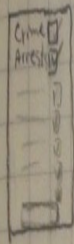


Total arrest

Number of arrest



With both "crime" & "arrests" selected, map shows percentage of arrests that resulted.



0: Page load, after splash screen, everything slightly dimmed

1-3: maximize to 1 on the three sections

6: Comments on the new development

Bookmarked date ranges

The bookmarking was a boon. It took some consideration to discover a good balance between too much time (tending toward visualizing the normal distribution) but still finding interesting pieces of information. One of the most helpful tactics here was simply making the timeline SVG taller. Allowing more room for variation in the timeline allowed us to more easily discover changes and interesting points in time within our data.

Deep links in the URI

Reading and writing to the URL—it could potentially be a solution to the locking thing for the wards, if it's easy enough. They have some window properties that you can get those pretty easily, and then just write code to handle it in the `$(document).ready` event

Finding something to talk about

It was difficult. We got to a point where we had so many great features to pursue and saw so much potential in the tools at hand, that it was hard eventually to find the few kernels of interest that we eventually landed on for our story. It took a significant amount of banging heads on walls, but the data points we discovered to talk about finally gave a necessary focus. It's far too easy, in this instance, to lose an interesting tree within the vast forest.