Jack Adam
jack.adam@yale.edu

jackadam.cc/thesis
github.com/jckdm/thesis

Abstract

"Digital Distancing" considers the relationship between distance and technology in the era of social distancing in order to define a framework for understanding ourselves through our data. The project is comprised of a journal, a pair of data collection and extraction tools, and visualizations of this data.

From August to December

In August, I proposed a tool which would track a user's mouse, in-use application, keystrokes, and more. I didn't yet know what visual encodings I would use to display the data, or exactly what data would be collected. I had a list of central questions regarding the functionality, relevance, and interpretability of data. Now, in December, I have two tools: one which closely resembles the aforementioned tool (this is Tracker), and another which came out of a discussion in ART 368 (this is Tracer). Many of the questions outlined in my initial proposal have remained at the forefront of my project, and I've included questions which instead focus on the human experience with data, asking most basically if we like our data—what it does, how it looks and feels?

Tracker

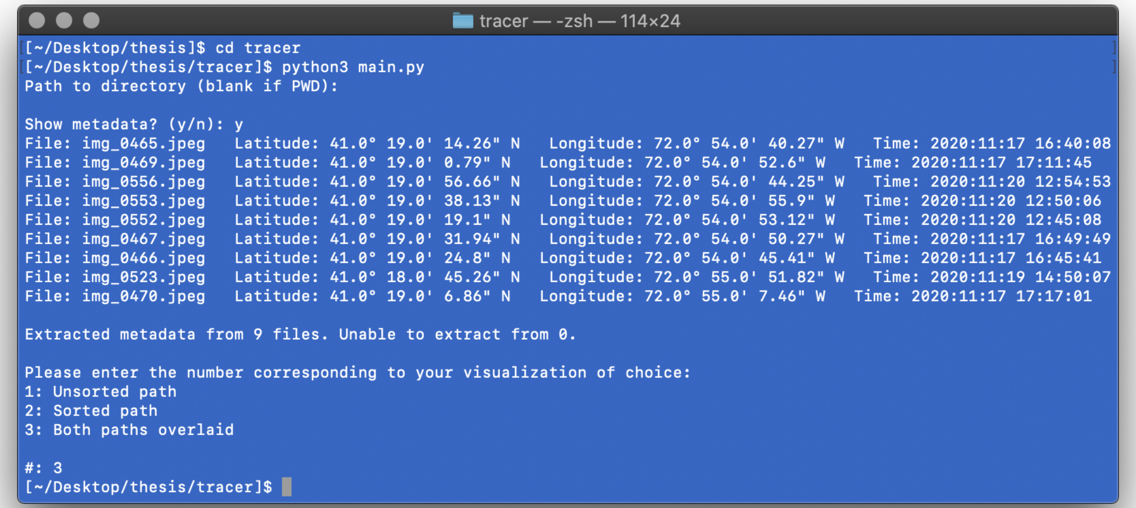
Tracker, which is written in Python, came together rather quickly. Once I had found the AppKit and pynput libraries, I was most of the way to my goal, and combined these with more common libraries such as datetime. Below is a snippet from Tracker's record() function, which is called every second to create a snapshot of a user's computer usage.

```
dt = datetime.now()
date = dt.strftime('%m/%d/%Y')
time = dt.strftime('%H:%M:%S')

a = NSWorkspace.sharedWorkspace().activeApplication()
app = a['NSApplicationName']
pid = a['NSApplicationProcessIdentifier']

m = Controller().position

return [date, time, app, pid, m[0], m[1]]
```

I (re-)learned sqlite3 so that I could write the collected data to both a database and a csv file. In an earlier iteration, I used a program (now: UNUSED-query.py) to remove from the database all applications which accounted for less than 0.3% of a user's total activity over a given session. In its current form, the database files (which are identical to the csv files) aren't used during the visualization process, but I've kept them as backups. I also wrote a program (now: UNUSED-reader.py) which reads in a csv file and outputs a csv file which removes all but the first instance of a user's presence within an application, thus keeping only the entries at which a user switches between applications. I later recreated this functionality in Tracker: Reader.

Tracer

Tracer is also written in Python and uses the Python Image Library (PIL) to extract an image's Exif tags. I originally had little interest in the legibility of the paths which the program rendered, but Professor Brown pointed out that I could sort the coordinates before connecting them. I began extracting the DateTimeOriginal tag and suddenly had two opposite visualizations of the same data, which led me to overlay them to see how they complemented each other. I also enjoyed the idea that Tracer could provide a user with a more legible summary of their images' metadata, and I used these formatted coordinates, as well as their visualizations, in my final zine for ART 368.



```
[~/Desktop/thesis]$ cd tracer
[~/Desktop/thesis/tracer]$ python3 main.py
Path to directory (blank if PWD):

Show metadata? (y/n): y
File: img_0465.jpeg   Latitude: 41.0° 19.0' 14.26" N   Longitude: 72.0° 54.0' 40.27" W   Time: 2020:11:17 16:40:08
File: img_0469.jpeg   Latitude: 41.0° 19.0' 0.79" N   Longitude: 72.0° 54.0' 52.6" W   Time: 2020:11:17 17:11:45
File: img_0556.jpeg   Latitude: 41.0° 19.0' 56.66" N   Longitude: 72.0° 54.0' 44.25" W   Time: 2020:11:20 12:54:53
File: img_0553.jpeg   Latitude: 41.0° 19.0' 38.13" N   Longitude: 72.0° 54.0' 55.9" W   Time: 2020:11:20 12:50:06
File: img_0552.jpeg   Latitude: 41.0° 19.0' 19.1" N   Longitude: 72.0° 54.0' 53.12" W   Time: 2020:11:20 12:45:08
File: img_0467.jpeg   Latitude: 41.0° 19.0' 31.94" N   Longitude: 72.0° 54.0' 50.27" W   Time: 2020:11:17 16:49:49
File: img_0466.jpeg   Latitude: 41.0° 19.0' 24.8" N   Longitude: 72.0° 54.0' 45.41" W   Time: 2020:11:17 16:45:41
File: img_0523.jpeg   Latitude: 41.0° 18.0' 45.26" N   Longitude: 72.0° 55.0' 51.82" W   Time: 2020:11:19 14:50:07
File: img_0470.jpeg   Latitude: 41.0° 19.0' 6.86" N   Longitude: 72.0° 55.0' 7.46" W   Time: 2020:11:17 17:17:01

Extracted metadata from 9 files. Unable to extract from 0.

Please enter the number corresponding to your visualization of choice:
1: Unsorted path
2: Sorted path
3: Both paths overlaid

#: 3
[~/Desktop/thesis/tracer]$
```

Figure 1: Formatted metadata extracted from 9 images.

Now that I had a system for differentiating between these two datasets even when presented together, I realized that Tracer's outputted visualizations need to be rotated 90 degrees to match up with a map—this is something I'd like to fix in code. But this is also part of the central question: if these visualizations only somewhat resemble maps, what is gained by rotating them to more closely resemble their intended shape?

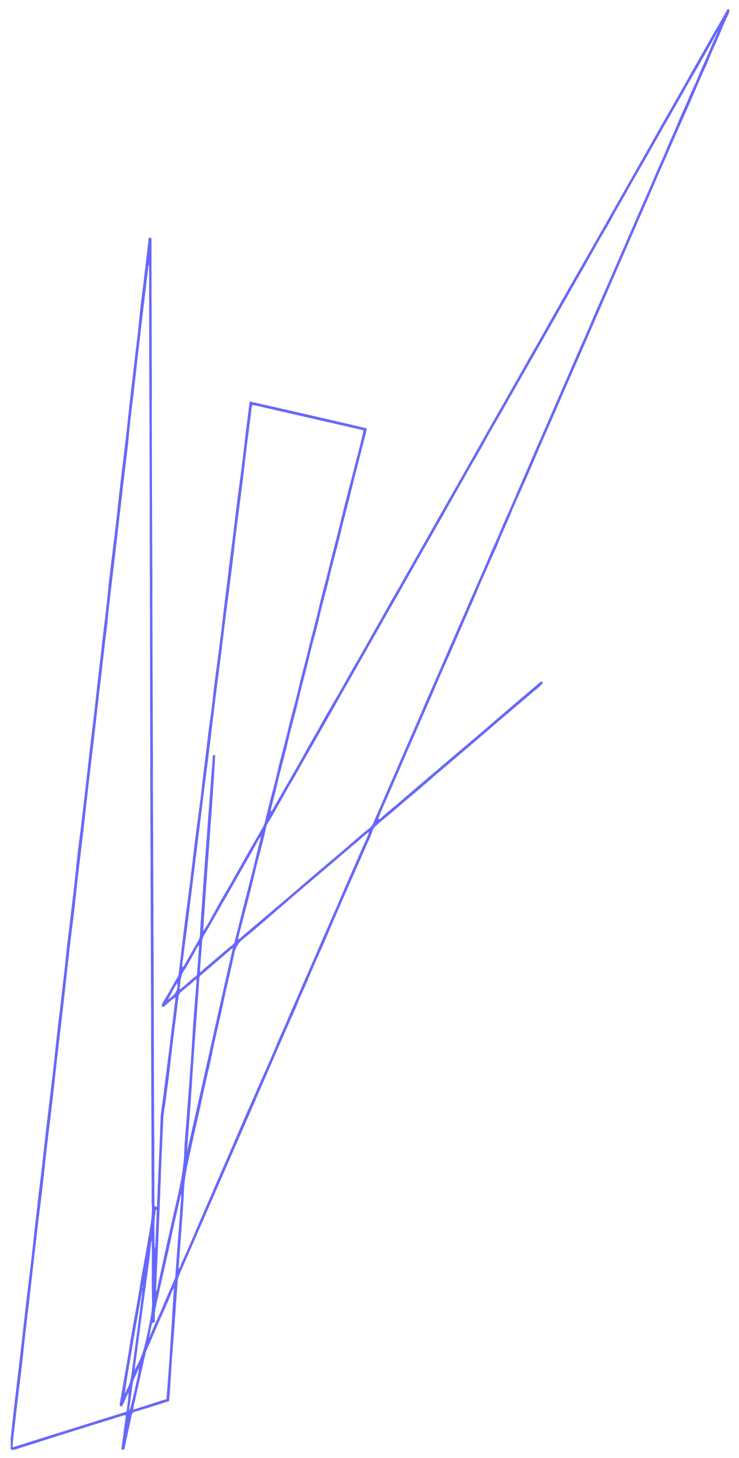


Figure 2: Locations of all Yale Police reports issued via the Yale Office of Emergency Management from March 23 to November 29, 2020. #6666FF which is the identity color for the project site, is used to identify sorted paths, whereas #3CB371 (not pictured) is used to identify unsorted paths.

Tracker: Grapher

Grapher, which is essentially a scatterplot, seemed like the most logical visualization for data of this nature, which comes as (x,y) pairs. It wasn't until I wrote the functionality to connect the points of a given application that the visualization took on an extra dimension, showing everything in-between the recorded data. These paths allow for analysis, inference, and assumption.

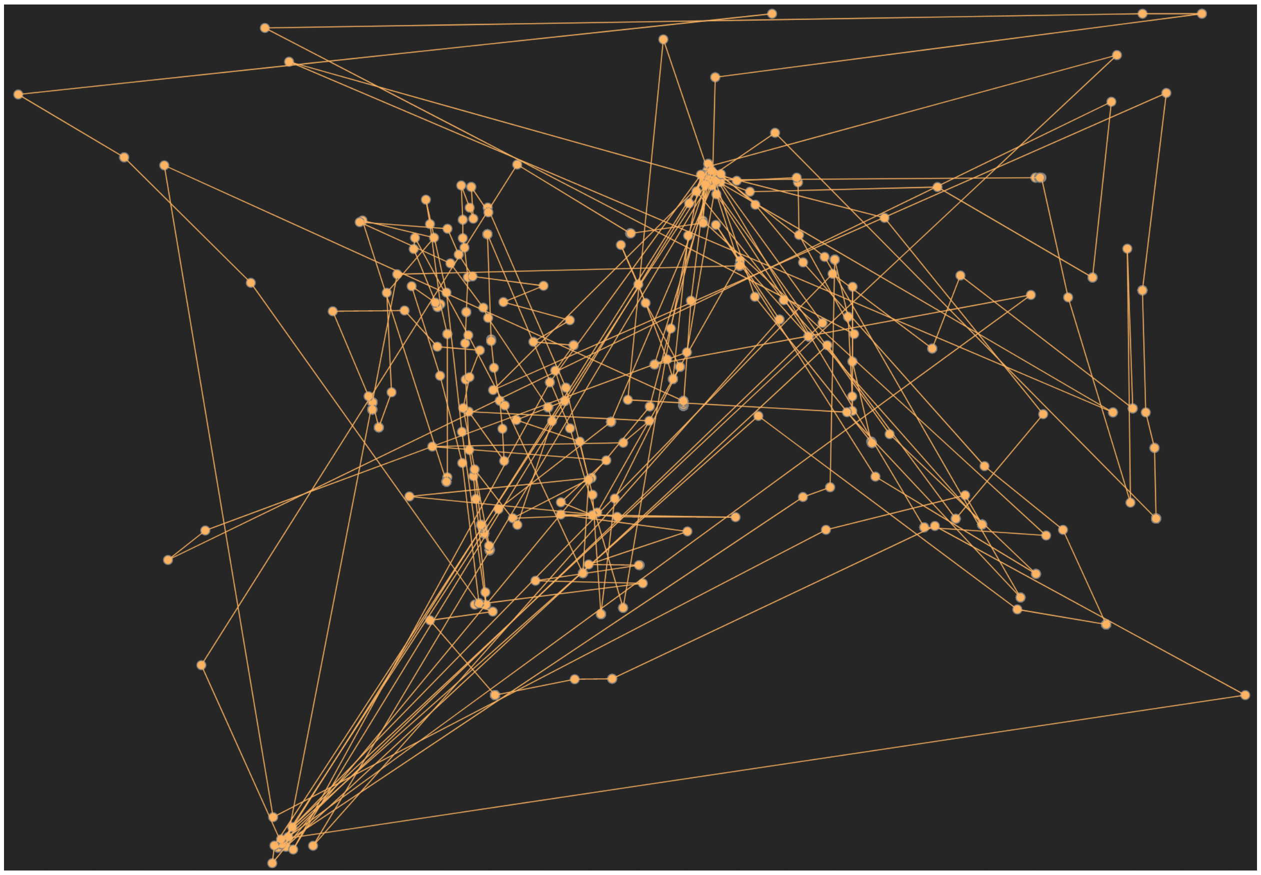


Figure 3: A connected graph showing a user's interaction with the Messages application over 555.47 minutes.

The above visualization, for example, has two main loci: one at the bottom-left (the application icon in the user's dock), and one at the top-middle (possibly the button to close the application window?). Was this user unfocused, continually closing the Messages application only to later open it? How would the shape of this visualization change over a different period of time? Or if it were the only in-use application?

Tracker: Mapper

Mapper, which is essentially a heatmap, was not the next visualization, but it is more similar to Grapher. I found that Grapher felt cluttered because of closely overlapping points and so I hoped that a more two-dimensional map would make patterns and hotspots clearer. By creating two scales, as seen below, the data can be re-layered. The visualization also allows a user to hide all quadrants without data—the biggest shock of this project has been to see how many points on the screen our mouses never enter, and how much space is unused. Perhaps this is a fault of my Tracker for waiting one second in-between each entry.

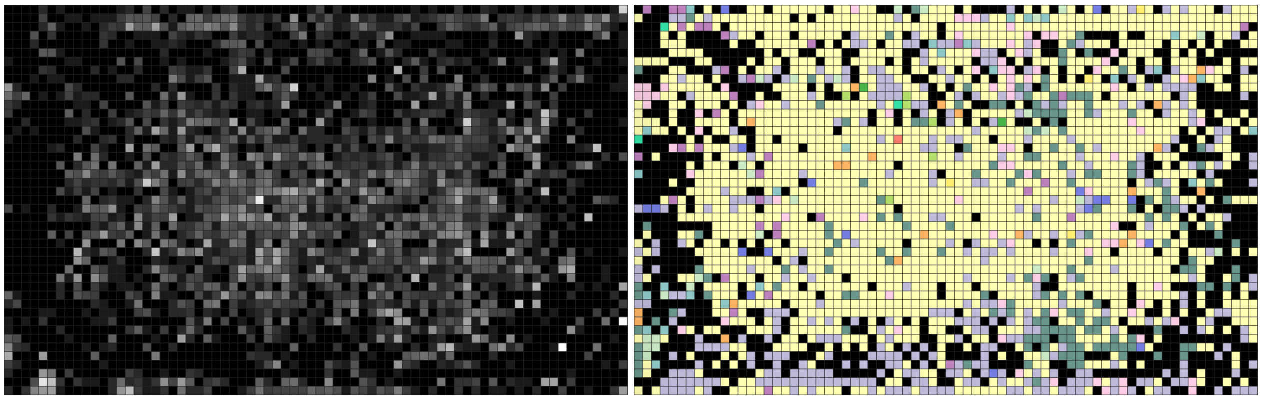
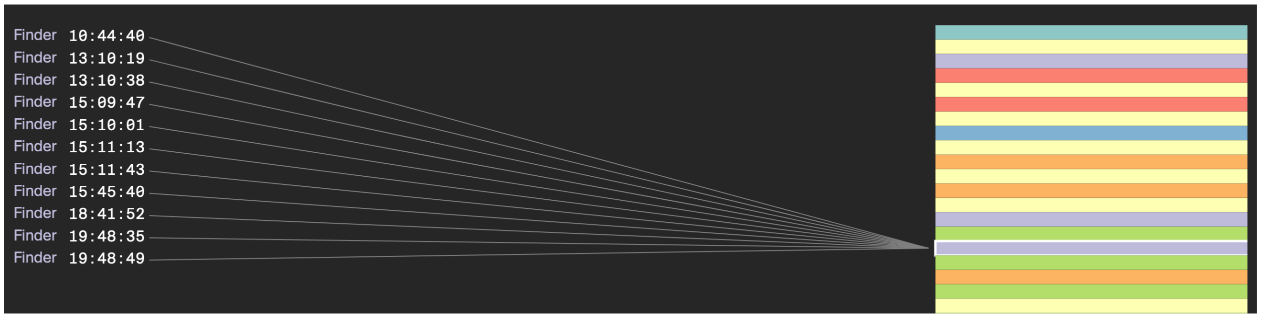


Figure 4: One dataset visualized with two scales: a logarithmic scale in which each quadrant is colored as a proportion of the most seconds spent in any one quadrant, and with a winner-takes-all color coding, in which each quadrant is colored according to its most used application.

Tracker: Reader

Reader was initially two separate visualizations with Reader being just the text-based visualization and Spacer being the color-based visualization. While I miss the mobile friendliness of these minimalistic visualizations, I think they're necessary partners. They do not quite align with each other, almost to a frustrating degree, but the colored rectangles act as a control panel for the text, allowing a user to bridge the two pieces. When a user clicks on a colored rectangle, it locks in place the many connected lines so that they can scroll and appreciate the data's shape.



```
Finder 10:44:40
Finder 13:10:19
Finder 13:10:38
Finder 15:09:47
Finder 15:10:01
Finder 15:11:13
Finder 15:11:43
Finder 15:45:40
Finder 18:41:52
Finder 19:48:35
Finder 19:48:49
```

Figure 5: All of the instances in which a user switched into Finder from another application.