

Jack Adam
jack.adam@yale.edu

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

Contents

| | |
|--------------------------------|----------|
| Abstract | 2 |
| From August to December | 2 |
| Tracker | 2 |
| Tracer | 3 |
| Tracker: Grapher | 5 |
| Tracker: Mapper | 6 |
| Tracker: Reader | 6 |
| Journal | 8 |
| To Conclude, | 8 |

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, among other things, a user’s mouse, in-use application, and keystrokes. 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, but I didn’t have an end goal in sight. Now, in December, I have two tools: one which closely resembles the aforementioned tool (this is Tracker) and another which was inspired by 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. I’ve added new 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, 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. I combined these with more common libraries such as datetime. Below is a snippet from Tracker’s record() function, which is called every second to take a snapshot of a user’s computer usage.

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

mouse = Controller().position

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

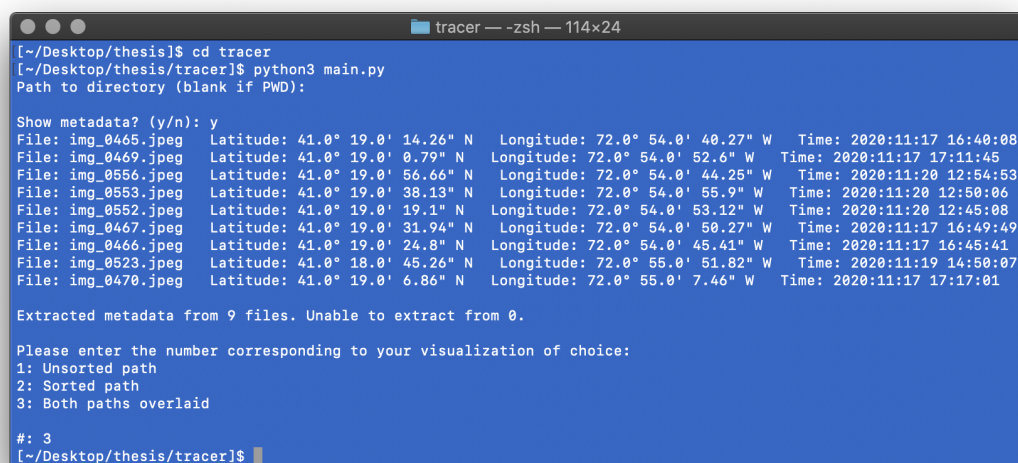
I (re-)learned sqlite3 so that I could write the collected data to both a database and a csv file. In an earlier iteration, I used a program (now: UNUSED-query.py) to remove from the database all applications which in total accounted for less than 0.3% of a user’s activity over a given session. I mistakenly thought that this data was irrelevant because it didn’t have much bearing on the overall shape of the data, but I decided that it would be dishonest to remove even one datapoint from a user’s journey. The word “journey” is emblematic of how my thinking shifted over the months: data is not just a list of isolated points, but a story to be told.

I also considered logging a user's keystrokes because I did (and still do) feel that the mouse only reveals one part of a user's experience navigating their virtual environment. I decided against this in the interest of privacy and because as Tracker stands, I can reasonably claim that the information I'm collecting isn't sensitive or personal. The word "personal" has haunted me throughout these months: how can we claim that this data is personal when we've never before seen it, and certainly don't recognize it?

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 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 Reader without the need for another Python program. With this change made, no steps are required between ending Tracker and visualization—a user can upload their data file to my site and generate all three visualizations. This uploaded data file is passed between HTML pages with an object URL.

Tracer

Tracer is also written in Python and uses the Python Image Library (PIL) to extract an image's Exif tags. I hadn't considered 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 so I added an option to print this data to the terminal, bringing it to the forefront of the user's experience with this tool. I later used these cleanly formatted coordinates, as well as their visualizations, in my final zine for ART 368.



```
tracer — zsh — 114x24
[~/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 bug is something I’d eventually like to fix in code, but it’s an easy manual fix. But this error gets at one of my central questions: if these visualizations only somewhat resemble maps to begin with, what is gained by rotating them to more closely resemble their intended shape? What is gained by trying to make legible a purposefully untraceable map? Spoiler: I do not think there is any purpose in this. But had I aimed to make a legible map, I would never have asked the question—so I’m glad that my process panned out as it did.

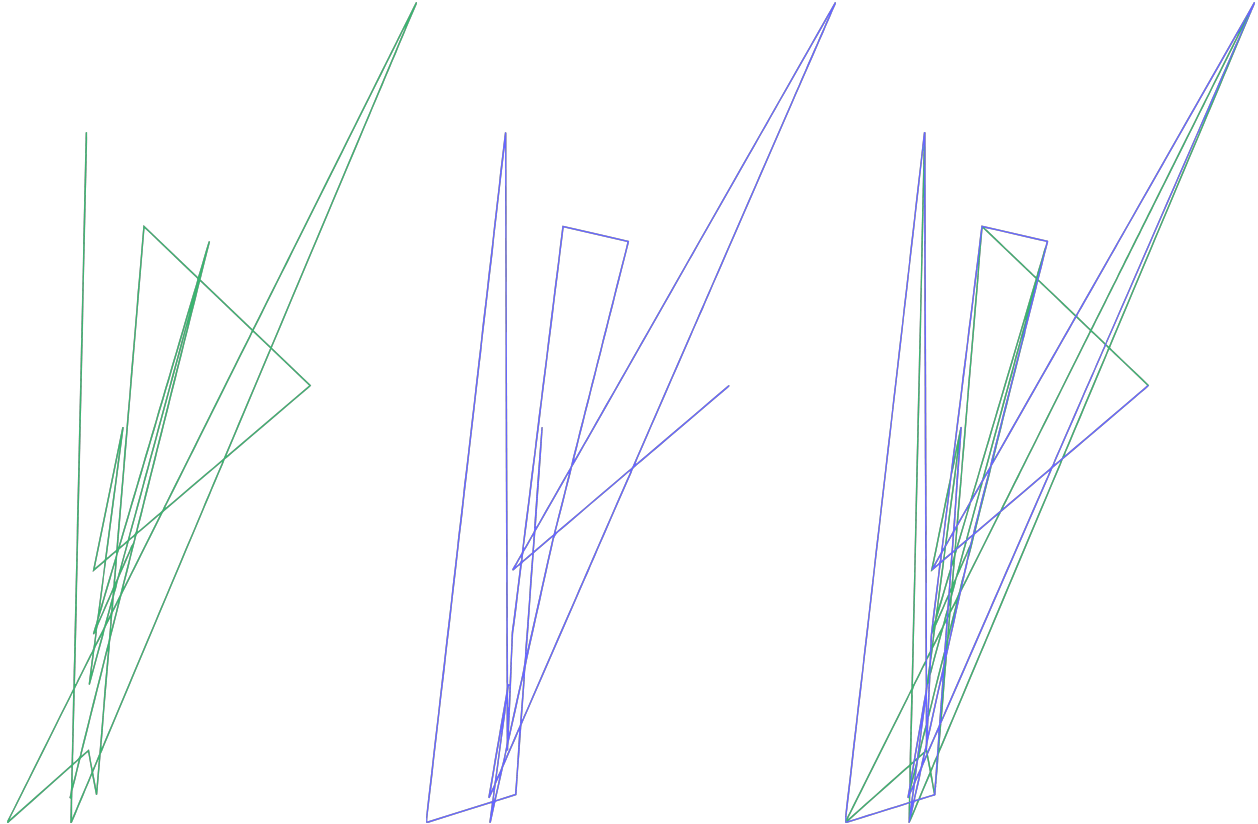


Figure 2: Locations of all police reports issued via the Yale Office of Emergency Management from March 23 to November 29, 2020. #6666FF, which is the identity color for the project site, is used to identify sorted paths, whereas #3CB371 is used to identify unsorted paths.

On the other hand, to call these visualizations “maps” is somewhat misleading, because map implies direction(s), geography, borders, and so I came to the name Tracer because of the program’s ability to trace distance in its purest sense. This is where my logic becomes hypocritical because I’ve pushed back on the notion that data is defined by functionality as opposed to, say, feeling? I explored this question at length in the Journal.

Tracker: Grapher

Grapher, which is essentially a scatterplot, seems like the most logical visualization for data of this nature, which is formatted 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, which establishes a sense of hierarchy. These paths allow for analysis, inference, and assumption.

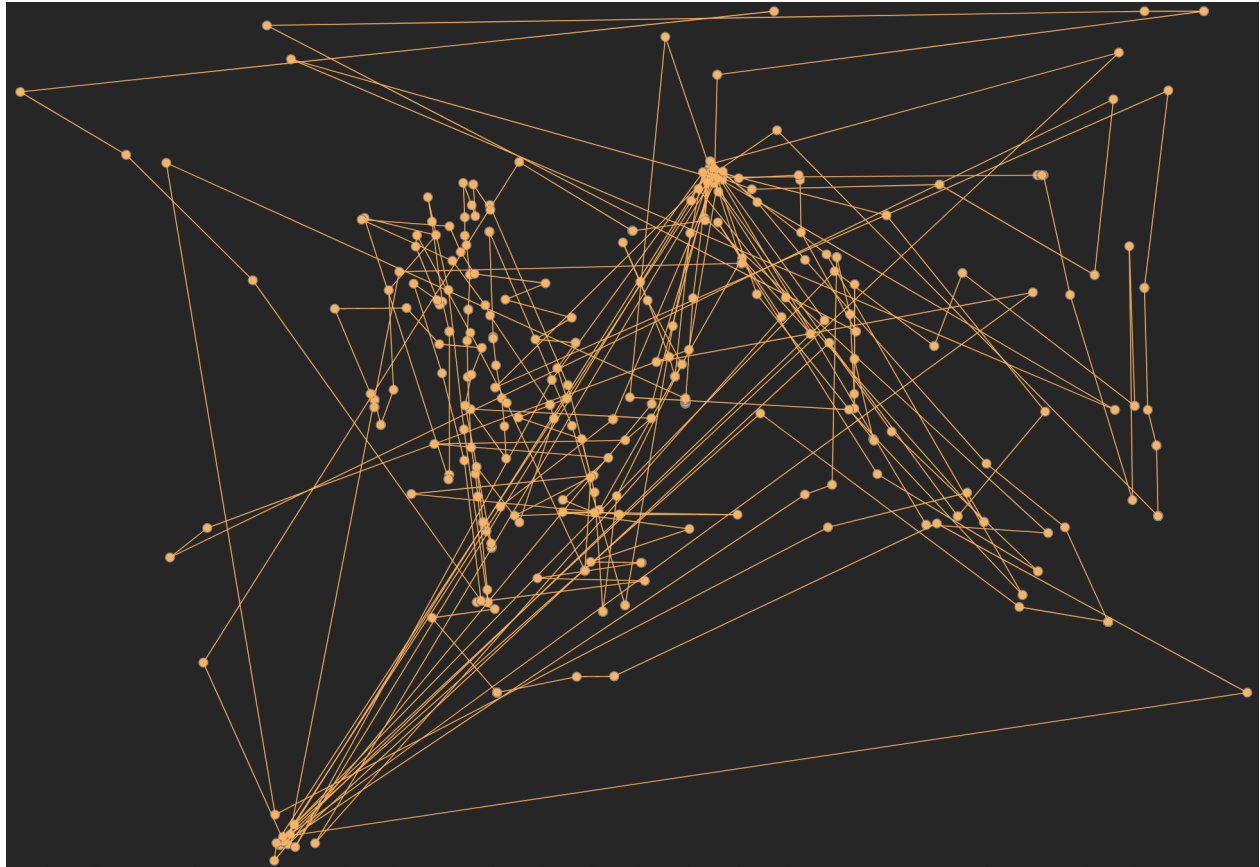


Figure 3: A connected graph showing a user's interaction with the Messages application over 277.07 minutes.

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 Messages was the only in-use application? How does the shape of this data compare to that of other users using a similar slate of applications over a similar span of time? Are these two theoretical datasets even comparable, and what would be gained from understanding their differences and similarities? Is data a raw material or a portrait of its creator?

Tracker: Mapper

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. But the idea of a heatmap felt too similar to Grapher and so it wasn't until after writing Reader that I wrote Mapper. The visualization allows a user to hide all quadrants without data—the biggest shock of this project has been seeing how many points on the screen our mouses never enter, or how much space is unused. Perhaps this is a fault of my Tracker for waiting one second in-between each entry. Regardless, removing these data-less quadrants reveals that our screens may be rectangular, but our screen activity lives inside of polymorphous blobs. By creating two scales, as seen below, the data can be re-layered.

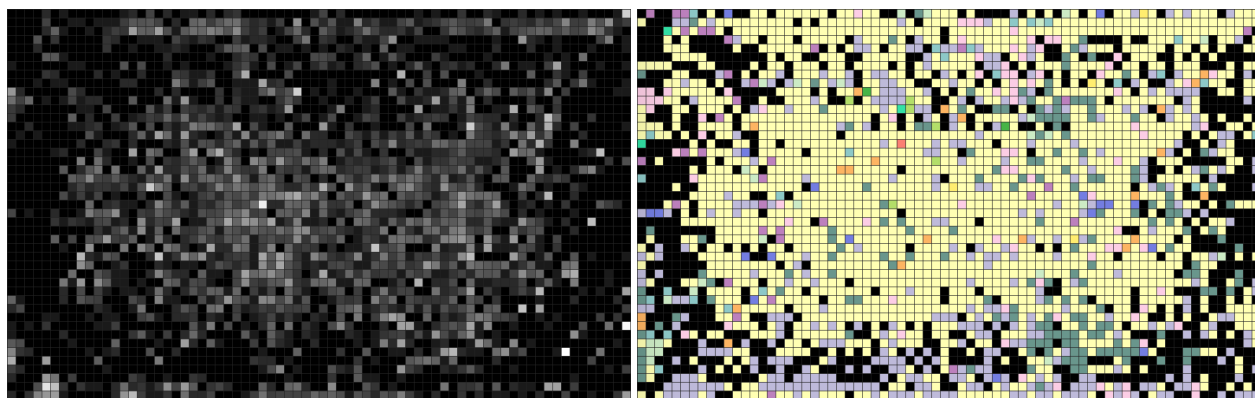


Figure 4: One dataset visualized with two scales: a logarithmic scale in which each quadrant is colored as a proportion of the greatest number of seconds spent in any one quadrant, and a winner-takes-all color coding, in which each quadrant is colored according to its most used application.

Tracker: Reader

Reader began as 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. The two lists 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.

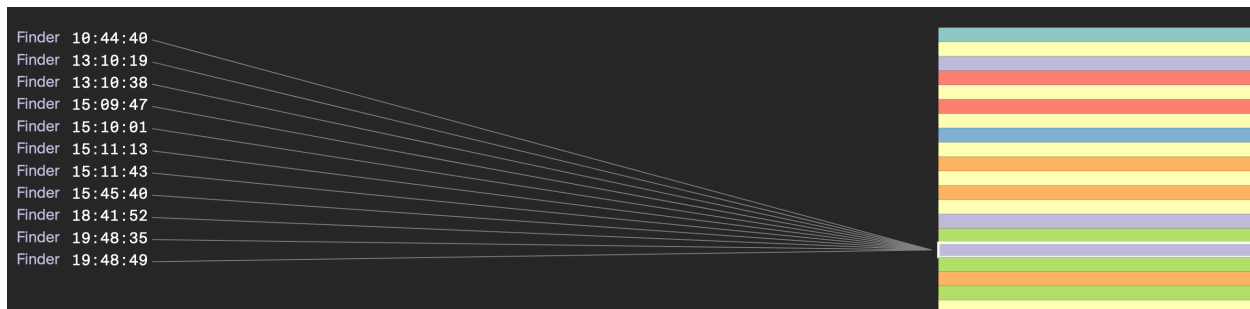


Figure 5: All instances of a user switching into Finder from another application.

When a user clicks on a colored rectangle, it locks the connected lines so that they can scroll and appreciate the data’s shape. This functionality is especially useful for larger datasets where it may be necessary to scroll.

Reader also performs data analysis and provides a summary of each dataset. The analysis is not unique to the data presented in Reader, but the visualization’s form makes it the most appropriate venue. The analysis algorithm isn’t perfectly elegant, but it has an optimization that I’m rather proud of. When calculating the longest period of continuous switching between two applications, Reader first calculates all potential patterns and the number of occurrences of each pattern. It then sorts the patterns and begins searching for continuous occurrences of the pattern with the most total occurrences (these patterns are more likely to be the eventual winner), keeping track of the longest pattern as it does so. It skips all patterns with fewer occurrences than the length of the longest pattern, thus once it reaches a pattern with fewer occurrences, the algorithm effectively stops.

Below are the results of analyzing the five datasets collected with Tracker. Columns 1–3 summarize the datasets and 4–7 quantify how a user switches between applications.

| | Total Number of Apps Used | Total Span of Data Collection | Active Time | Total Number of Switches Between Apps | Average Time Spent per App | Length of Longest Period of Continuous Switching | Number of Switches in Longest Period |
|--------------|---------------------------|-------------------------------|-----------------|---------------------------------------|----------------------------|--|--------------------------------------|
| jackadam | 17 | 583.07 | 420.42 (72.10%) | 435 | 0.97 | 29.62 | 25 |
| ruth | 12 | 1560.93 | 319.95 (20.50%) | 182 | 1.76 | 30.03 | 8 |
| GiaG | 12 | 515.60 | 297.50 (57.70%) | 124 | 2.40 | 8.73 | 6 |
| feliciachang | 7 | 88.02 | 87.57 (99.49%) | 34 | 2.58 | 44 | 9 |
| riannaturner | 9 | 555.47 | 277.07 (49.88%) | 157 | 1.76 | 20.68 | 8 |

Table 1: Summary of data analysis performed by Reader. All times measured in minutes.

Journal

I knew from the beginning that I wanted to write about the topics of my project in order to more clearly formulate questions on them. I’ve long subscribed to the notion that knowing what questions to ask can be more important than answering those same questions—and I also like telling stories. The result is a series of 10 journal entries, each of which considers distance in a specific context such as nature or cities, as a measurement, or as an abstract phenomenon.

I use JavaScript’s Intersection Observer API to set the first word of a given paragraph as a target which triggers a change in the image’s source. Also of note is the navigation bar, which is dynamically generated in JavaScript: the content div is scanned for `<text>` tags which are assigned unique IDs based on their `innerText`, and links are added to the nav div for each of these tags. This script runs on all pages with a navigation bar, allowing me to reserve the `<text>` tag as an automatically linkable header and to give the website a consistent navigation.

To Conclude,

I’d like to quote myself: “And when I look out into the vast whiteness of the tundra, or up at the murky gray sky, or into the blackness of a screen, I don’t see the space between myself and it—it which has no depth, no limits or ends. Sure, distance can be a measure of pixels or picas or points, just as sometimes it can be nothing more than a feeling.”

Thanks to advisors Dr. Benedict Brown, Justin Berry, and Dr. Julie Dorsey, housemates Felicia Chang and Rianna Turner, friends Shayna Sragovicz, Helen Kauder, Trey Lewis, Gia Grier, Harry Jain, Michaela Shelton, to my parents, and to the Yale Office of Emergency Management.