About the Beiwe Sample Android Dataset

Mathew Kiang

12/20/2017

Introduction

The Beiwe Research Platform collects high-density data from a variety of smartphone sensors including GPS, WiFi, Bluetooth, and accelerometer. To learn more about Beiwe, check out the Onnela Lab page, the paper introducing the platform, or the Beiwi wiki.

In order to help (current and potential) collaborators understand the structure and format of Beiwe data, I am making my personal data available to the public. These data were collected on an Android phone from May 30th, 2016 at 08:00 (UTC) to August 25th, 2016 at 07:59 (UTC).

The hope is that access to real data will allow researchers to (1) facilitate coding and debugging for ETL, data ingestion, and other parts of their pipeline before data is collected, (2) create functions to help inspect raw data, and (3) test new methods or functions on real data.

Legal: Citation and License

The Digital Object Identifier of this dataset is `10.5281/zenodo.1120327`. When using these data, please cite the dataset as

When referring to the Beiwe Research Platform or how these data were collected, please cite the JMIR-Mental Health paper as:

> Torous J, Kiang MV, Lorme J, Onnela JP, New Tools for New Research in Psychiatry: A Scalable and Customizable Platform to Empower Data Driven Smartphone Research, JMIR Ment Health 2016;3(2):e16. URL: https://mental.jmir.org/2016/2/e16, DOI: 10.2196/mental.5165

Study parameters

Researchers may specify different data collection parameters for every study. For these data, the Beiwe app collected accelerometer, Bluetooth, call, GPS, power state, text, and WiFi data. These data were collected at the following rates:

- Accelerometer (on/off): 600 seconds / 60 seconds
- GPS (on/off): 7,140 seconds / 60 seconds
- Bluetooth (on/total): 60 seconds / 300 seconds
- WiFi (on and record): 10 seconds

Call, power state, and text data were collected when events occurred, which was relatively infrequently.

Original data

The original 8,468 data files are stored in 13 folders. The structure of this layout can be found in `original_data_tree.txt`.

Splitting the original data

Realistic Beiwe data will come from multiple users. To simulate the structure of real data, I split up the data in the following way:

1. For each file with ≥ 50 lines of data, I randomly assign each line to one of five fake users.
2. For each file with < 50 lines of data, I simply copy that entire file to each of the five fake users.

The code that performed the data splitting can be found in `./code/split_original_data.R`.

Combining fake users into one

By construction, the majority of files will each contain distinct, non-overlapping parts of the original data. To reconstruct the original data, just read in the file from each user, remove duplicate rows, and sort by timestamp. An example of this is in `./code/plotting_chicago_data.R` with the germane parts here:

```
library(tidyverse)

## Helper functions ----
bash_merge <- function(folder, awk = TRUE, joined_file = "0000-merged.csv") {
    # Uses bash `awk` or `cat` to merge files.
    # In general, faster than looping `fread()` for `read_csv`.
    # Note: `cat` doesn't work if there is a header row.
    original_wd <- getwd()
    setwd(folder)
    if (awk){
        system(paste0("awk 'FNR==1 && NR!=1{next;}{print}' *.csv > ",
                      joined_file))
    } else {
        system(paste0("cat *.csv > ", joined_file))
    }
    setwd(original_wd)
}

## Merge all gps files within each user ----
for (f in list.dirs('./data', recursive = FALSE)) {
    bash_merge(paste0(f, "/gps"))
}

## Now import all five merged files ----
all_gps <- NULL
for (f in list.files("./", recursive = TRUE, pattern = "0000-merged.csv")) {
    all_gps <- rbind(all_gps, read_csv(f))
}

## Sort it and remove duplicates (none in the default case, but possible on
## other data)
all_gps <- all_gps %>%
    arrange(desc(timestamp)) %>%
    distinct() %>%
    ungroup()
```

In this code, I merge all GPS files for all users separately. I then read in each of these merged files, remove duplicates with the `distinct()` function, and then sort by `timestamp`. The resulting dataframe is identical to reading in all original GPS files.

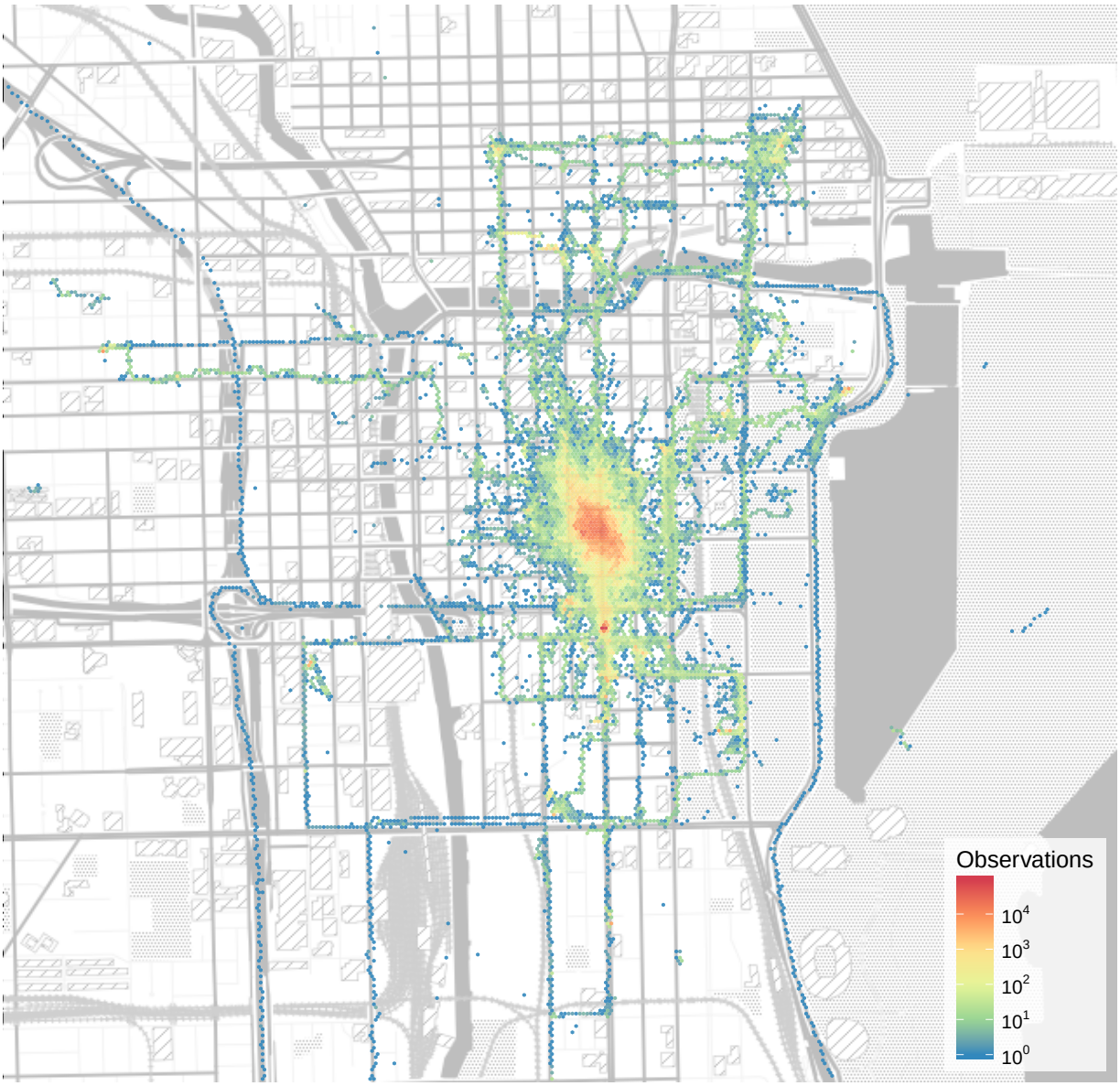
An example of GPS data

We can view the rate of data collection as a heatmap where each row is an hour and each column is a day.

```
## Code can be found in "./code/plotting_hourly_heatmap.R"
knitr::include_graphics("./plots/hourly_heatmap.jpg")
```

Similarly, we can aggregate over time and show heatmap of all GPS points on a map of Chicago.

```
## Code can be found in "./code/plotting_chicago_data.R"
knitr::include_graphics("./plots/chicago_map.jpg")
```



See `http://mkiang.carto.com` to interactively visualize these data, including movement over time.