Appendix: Automatic Contact Tracing for Outbreak Detection Using Hospital Electronic Medical Record Data

Michael DeWitt*

6/14/2020

1 Introduction

The methods described in the paper can be demonstrated through a simulation study with synthetic data that mimics the type of data available in the Electronic Medical Record. Implementation of the methods described in the paper have been placed in an R package called autotracer and can be downloaded as follows:

```
remotes::install_github("conedatascience/autotracer")
```

2 Data

The autotracer package as a synthetic data set generated using the Wakefield R package (Rinker 2018). These data represent typical data that could be retrieved from an EMR at the patient level and supplemented by geocoding the patient address information.

- Latitude/ Longitude (x,y) coordinates for the patient's home address
- Patient Identifier (could be a patient Medical Record Number or some other unique identifier)
- Race
- Age (both whole years and binned in 10 year intervals)
- Sex
- Primary Language
- Test Date/ Symptom Onset Date (depending on information availability)

These data can be view as follows:

```
library(autotracer)
library(dplyr)
library(tidyr)

head(autotracer::autotracer_linelist)
```

```
## # A tibble: 6 x 9
##         x       y Race     ID      Age Sex    Language   patient_id date
##     <dbl>   <dbl> <fct>    <chr> <int> <fct>  <fct>           <int> <date>
## 1 -1.13   -0.715  White    00001    69 Female Portuguese          1 2020-06-21
## 2  1.60   -0.140  Asian    00002    43 Female Wu                  2 2020-06-29
## 3  0.936  -1.17   White    00003    57 Male   Punjabi             3 2020-06-10
## 4  0.181   0.473  Black    00004    19 Male   Min Dong            4 2020-06-23
## 5  1.05   -0.0124 Hispanic 00005    31 Male   Bengali             5 2020-06-14
## 6  0.0572 -0.494  White    00006    57 Female Swahili             6 2020-06-04
```

*Cone Health, michael.dewitt@conehealth.com

3 Implementation

The synthetic EMR data set can then be cleaned in order to establish the probabilistic transmission chain within the cluster.

For illustrative purposes the latitude and longitude coordinates are trimmed to some level of precision. This trimming is governed by local conditions based on the estimated contacts patterns in the community of interest. Typically, a good starting point is to round the geographic coordinates to four decimal places which represents a 30 foot radius to represent a "household."[1] For expediency, here rounding to two decimal places is used. These new coordinates are then combined to a single "location" parameter and used as a grouping variable representing a cluster candidate. Note that those locations with only one case are removed from the cluster candidate pool.

```
contact_matrix <- autotracer_linelist %>%
    mutate(location = paste0(round(x,2),"-",round(y,2))) %>%
    group_by(location) %>%
    mutate(ct = n()) %>%
    dplyr::filter(ct>1) %>%
    arrange(-ct) %>%
    dplyr::select(-ct)
```

The cluster candidates can then be submitted to the `connect_probable_cases` function which in turn internally calls functions from the outbreaker2 (T et al. 2014) package. The `connect_probable_cases` also has arguments that allow the user to specify the probability distribution for the relevant distribution (e.g. serial interval as shown below, but generation interval could also be used), a cluster link identifier, and a threshold for removing a patient from the transmission chain. The threshold argument is used to remove patients from a cluster based on if the time between the transmission is larger than one would expected (e.g. it is unlikely that someone transmitted a respiratory infection after 60 days). Again, subject matter expertise and local conditions should be used to determine this threshold because it could also be affected by testing delays.

Continuing with the above example the transmission chains can be estimated using the following code applied to the first ten candidate for expediency:

```
contact_matrix <- contact_matrix %>%
    nest() %>%
    .[1:10,] %>%
    mutate(connections = purrr::map(data, connect_probable_cases,
                                                    exposure_link = location,
                                                    threshold = 30)) %>%
    dplyr::select(connections) %>%
    unnest(connections) %>%
    ungroup() %>%
    dplyr::select(from, to, diff_dt_onset, exposure_link)
```

After the contact matrix is established, the epicontacts package (Nagraj et al. 2017), a part of the R Epidemics Consortium suite of packages can be used for further analysis and visualization of clusters.

```
library(epicontacts)
x <- make_epicontacts(linelist = autotracer_linelist,
                                            from = "from", to = "to",
                                            id = "patient_id",
                                            directed = TRUE,
```

[1]There are drawbacks to this method as people living in multifamily homes or apartment buildings could be combined into a cluster where there may not be any real interactions between these persons. In this case it is important to understand transmission pathways and to balance the false positive rate for cluster identification versus returning fewer, but more exact clusters.

```
                                        contacts = contact_matrix)
```

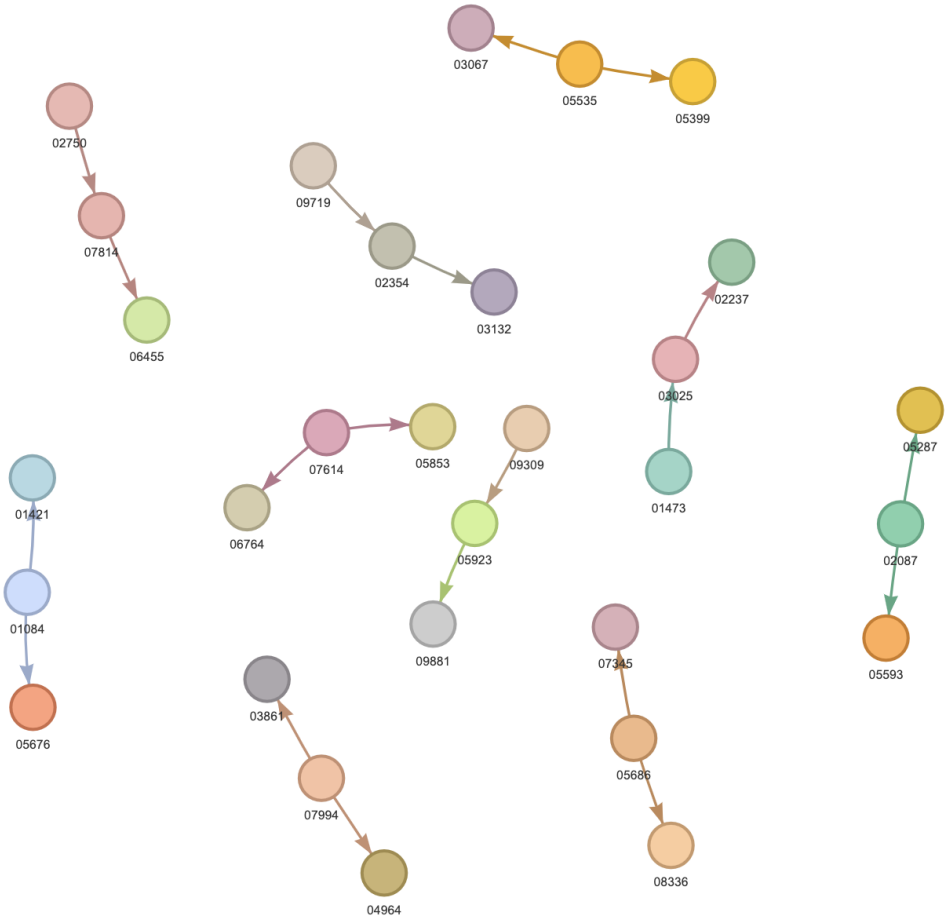

Figure 1: Example Network Graphic Generated Using epicontacts

Automatic cluster emergence can be accomplished by running this program each day and writing out the cluster identifiers and associated members. Each day the cluster sizes and associated members can be done in order to identify which clusters are growing.[2] Additionally, the tools provided by the epicontacts package allow for secondary analysis such as establishing contact patterns and estimated the over-dispersion of the transmission chains.

References

Nagraj, VP, Thibaut Jombart, Nistara Randhawa, Bertrand Sudre, Finlay Campbell, and Thomas Crellen. 2017. *Epicontacts: Handling, Visualisation and Analysis of Epidemiological Contacts.* https://CRAN.R-project.org/package=epicontacts.

Rinker, Tyler W. 2018. *wakefield: Generate Random Data.* Buffalo, New York. https://github.com/trinker

[2] Another method would be to run this algorithm iteratively, sub-setting the data on date (e.g. run the algorithm for all cases less than date n_t, then repeat for date n_{t+1}, and so on. Emerging clusters could then be identified in memory; however, running this algorithm in such a way would likely be computationally costly.)

/wakefield.

T, Jombart, Cori A, Didelot X, Cauchemez S, Fraser C, and Ferguson N. 2014. "Bayesian Reconstruction of Disease Outbreaks by Combining Epidemiologic and Genomic Data." *PLoS Computational Biology*.

Wickham, Hadley. 2020. *Tidyr: Tidy Messy Data*. https://CRAN.R-project.org/package=tidyr.

Wickham, Hadley, Romain François, Lionel Henry, and Kirill Müller. 2020. *Dplyr: A Grammar of Data Manipulation*. https://CRAN.R-project.org/package=dplyr.