Tracing and testing the COVID-19 contact chain: cost-benefit tradeoffs

Table of Contents

Instructions for installing and running the software

1. Requirements and installation

1. It requires R v4.0.0 (or above) and it is strongly recommended to download and install RStudio from https://rstudio.com/products/rstudio/.
   - Typical install time on a 'normal' desktop computer is 30 minutes.
   - Our software is tested on Mac OS 10.15.6.
   - Non-standard hardware is not required.
2. Open RStudio.
3. Set working directory.
   - Go to `Tools > Change Working Dir...` menu (`Session > Set Working Directory` on a mac), or use `setwd()` function. e.g., type `setwd("/Users/COVID-19")` in the Console.
   - Check that the "home" directory for your project is the working directory of our current R process by typing `getwd()` in the Console.
4. Keep all the files (required input csv and txt files, R scripts) associated with a project located together in the working directory.
   - Input file 1: `university_student_raw_data.csv`
     - Download the raw data from https://www.dropbox.com/s/4q4ccsgs6e7sx37/university_student_raw_data.csv?dl=0

- Data collected by smartphones of university students, as part of the Copenhagen Networks Study. The smartphones were equipped with Bluetooth cards which recorded proximity between participating students at 5-minute resolution.
- The raw data was previously published in *P. Sapiezynski, A. Stopczynski, D. D. Lassen, and S. Lehmann. Interaction data from the Copenhagen Networks Study. Scientific Data, 6(1):1–10, 2019*.

– Input file 2: `foursquare_raw_data.txt`
  - Download the raw data from https://www.dropbox.com/s/grscylxobqa4a3c/foursquare_raw_data.txt?dl=0
  - Data that users of Foursquare service (a Location-Based Social Networking) made available in Social Media about their locations in Tokyo, Japan along with time-stamps.
  - The raw data was previously published in *D. Yang, D. Zhang, V. W. Zheng, and Z. Yu. Modeling user activity preference by leveraging user spatial temporal characteristics in LBSNs. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 45(1):129–142, 2014*.

2. Construction of contact networks

Human contact networks from real-world datasets

| University student contact network and Foursquare contact network

You can construct contact networks directly from the real data mentioned above through the R codes below. You can also use the contact networks we have already generated with the software right away to reduce computation time. For the former, follow the instruction in Method A below, and for the latter, follow the instruction in Method B.

Method A

1. Execute the code to construct contact networks from real-world datasets
   – If you want to construct `University student contact network`, type `source("construction_of_university_student_contact_network.R", echo = TRUE)` in the Console.
   – If you want to construct `Foursquare contact network`, type `source("construction_of_foursquare_contact_network.R", echo = TRUE)` in the Console. It may take a few hours.
2. Check output of this software
   – The output (`contact_network.RData`) of this software is saved in the current working directory.
   – The output returned is a list in which the total number of elements is the total number of days of the contact network plus one. Each element except the last one represents a contact between individuals in each day, and the last element represents the total number of nodes. For example, the first element in the list consists of a list of contact between nodes in the first day.

Method B

1. Rename a pre-generated specific network we are interested in for use in the next step
   - If you want to use a pre-generated `University student contact network`, type `file.rename('university_student_contact_network.RData', 'contact_network.RData')` in the Console.
   - If you want to use a pre-generated `Foursquare contact network`, type `file.rename('foursquare_contact_network.RData', 'contact_network.RData')` in the Console.
2. The renamed file (`contact_network.RData`) is a list in which the total number of elements is the total number of days of the contact network plus one. Each element except the last one represents a contact between individuals in each day, and the last element represents the total number of nodes.

Synthetic networks

| Scale-free network and Erdos-Renyi random network

1. Execute the code to construct synthetic networks
   - If you want to construct `Scale-free network`, type `source("construction_of_scale_free_network.R", echo = TRUE)` in the Console.
   - If you want to construct `Erdos-Renyi random network`, type `source("construction_of_erdos_renyi_network.R", echo = TRUE)` in the Console.
2. Check output of this software
   - The output (`contact_network.RData`) of this software is saved in the the current working directory.
   - The output returned is a list in which the total number of elements is the total number of days of the contact network plus one. Each element except the last one represents a contact between individuals in each day, and the last element represents the total number of nodes.

3. Simulation of virus transmission and multi-hop contact tracing

1. Open the script file `simulation_virus_transmission_and_contact_tracing.R` in the editor.
   - Go to `File > Open > simulation_virus_transmission_and_contact_tracing.R` or type `file.edit('simulation_virus_transmission_and_contact_tracing.R')` in the Console.
2. Setting parameters values. (set values in lines 15,21,27 of the script file)
   - `cooperativity`: cooperativity. Consider the following values: 0.2, 0.5, 1.0.
   - `prob.inf.from.symp`: probability of infection. Consider the following values: 0.04, 0.2, 0.4.
   - `I0`: the number of initially infected individuals. Set `I0=1` for university student contact network and `I0=3` for the rest of the networks.

- e.g., if `cooperativity=0.2` and `prob.inf.from.symp=0.4`, the result will be generated for 0.2 cooperativity and 0.4 probability of infection.

3. Execute the code for simulations of virus transmission and multi-hop contact tracing
   - Type `source("simulation_virus_transmission_and_contact_tracing.R", echo = TRUE)` in the Console.
4. Check output of this software
   - The output files of this software are saved in the current working directory.
   - `cumulative_infections_per_1000.csv`: the cumulative number of infections (per 1,000 population) for k-hop contact tracing for various values of k (average of 1000 simulation runs). The row names of the file indicate the value of k and the column names indicate the day. For example, if the row name is `2-hop` and the column name is `5`, the corresponding cell indicate the cumulative number of infections (per 1,000 population) on day 5 for 2-hop contact tracing.

```
cumulative_infections_per_1000<-
read.csv("cumulative_infections_per_1000.csv", row.names = 1,
check.names=FALSE)
print(cumulative_infections_per_1000[,1:7])  # display the first 7
columns

##              1        2       3        4        5        6        7
## 0-hop 1.488095 1.763393 6.81994 9.958333 29.12946 51.15179 97.30655
## 1-hop 1.488095 1.763393 6.81994 9.910714 28.15774 48.09524 87.97173
## 2-hop 1.488095 1.763393 6.81994 9.901786 26.36161 42.55506 68.93899
## 3-hop 1.488095 1.763393 6.81994 9.877976 25.01786 39.14435 61.61458
## 4-hop 1.488095 1.763393 6.81994 9.858631 24.52679 38.08482 59.75000
## 5-hop 1.488095 1.763393 6.81994 9.842262 24.38690 37.85119 59.37500
```

   - `daily_new_infections_per_1000.csv`: the daily new infection count (per 1,000 population) for k-hop contact tracing for various values of k (average of 1000 simulation runs). The row names of the file indicate the value of k and the column names indicate the day. For example, if the row name is `2-hop` and the column name is `5`, the corresponding cell indicates the daily new infection count (per 1,000 population) on day 5 for 2-hop contact tracing.

```
daily_new_infections_per_1000<-
read.csv("daily_new_infections_per_1000.csv", row.names = 1,
check.names=FALSE)
print(daily_new_infections_per_1000[,1:7])  # display the first 7 columns

##              1         2        3        4        5        6        7
## 0-hop 1.488095 0.2752976 5.056548 3.138393 19.17113 22.02232 46.15476
## 1-hop 1.488095 0.2752976 5.056548 3.090774 18.24702 19.93750 39.87649
## 2-hop 1.488095 0.2752976 5.056548 3.081845 16.45982 16.19345 26.38393
## 3-hop 1.488095 0.2752976 5.056548 3.058036 15.13988 14.12649 22.47024
## 4-hop 1.488095 0.2752976 5.056548 3.038690 14.66815 13.55804 21.66518
## 5-hop 1.488095 0.2752976 5.056548 3.022321 14.54464 13.46429 21.52381
```

- `test_per_1000.csv`: the number of tests over time (per 1,000 population) for k-hop contact tracing for various values of k (average of 1000 simulation runs). The row names of the file indicate the value of k and the column names indicate the day. For example, if the row name is `2-hop` and the column name is `5`, the corresponding cell indicates the number of tests (per 1,000 population) on day 5 for 2-hop contact tracing.

```
test_per_1000<-read.csv("test_per_1000.csv", row.names = 1,
check.names=FALSE)
print(test_per_1000[,1:7])  # display the first 7 columns

##       1        2         3          4          5          6          7
## 0-hop 0 0.360119 0.2544643  0.2157738  0.1845238  0.3169643  0.4940476
## 1-hop 0 0.360119 0.3288690  0.9642857  2.5848214  7.7083333 22.0669643
## 2-hop 0 0.360119 0.5982143  5.1666667 22.4866071 61.1592262 87.8839286
## 3-hop 0 0.360119 1.6220238 19.2157738 65.5059524 88.2217262 96.2946429
## 4-hop 0 0.360119 4.2395833 44.1294643 91.3883929 84.9226190 93.2202381
## 5-hop 0 0.360119 8.1562500 64.1726190 94.3348214 76.8660714 86.5565476
```

Example data and expected output

1. Instructions

1. Set the folder containing example data and expected output as the working directory.

2. As an example, we use a pre-generated `University student contact network` (file: `contact_network.RData`), and consider 0.2 cooperativity and 0.4 probability of infection.

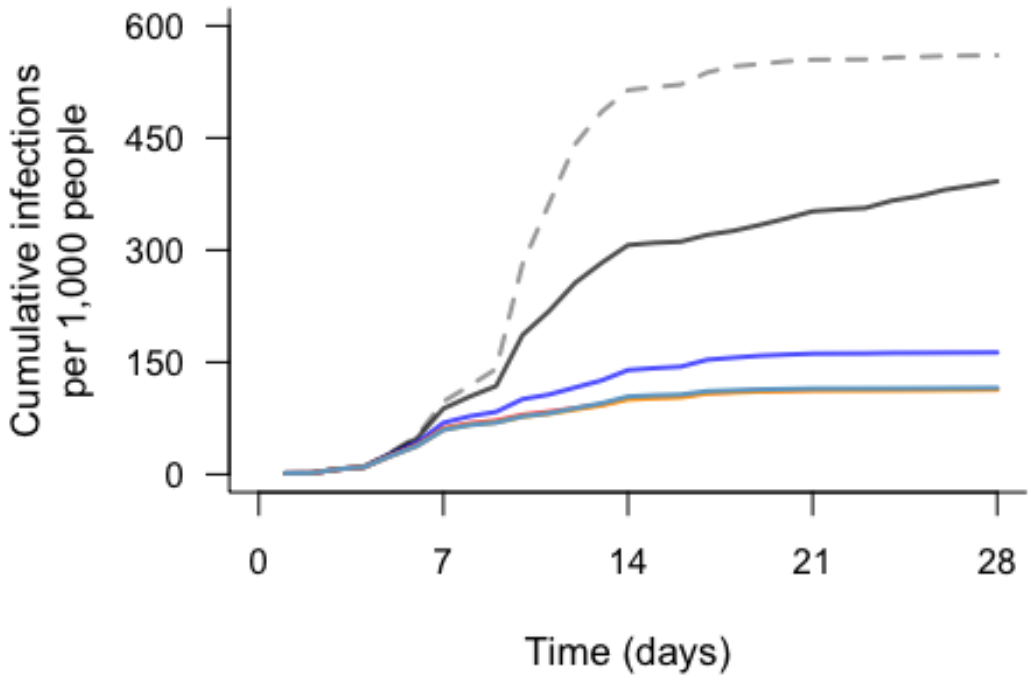
3. Execute the code for simulations of virus transmission and multi-hop contact tracing
   - Type `source("simulation_virus_transmission_and_contact_tracing.R", echo = TRUE)` in the Console.
   - Then, three different csv output files through the execution should be same as the expected output included in the folder.
   - `cumulative_infections_per_1000.csv` corresonds to the data of top-left of Figure 3a in the main paper.
   - `daily_new_infections_per_1000.csv` corresonds to the data of bottom of Figure 5a in the main paper.
   - `test_per_1000.csv` corresonds to the data of top of Figure 5a in the main paper.
   - Expected run time for the demo on a 'normal' desktop computer is 50 minutes.

2. Generation of figures from the example data

1. Top-left of Figure 3a in the main paper from `cumulative_infections_per_1000.csv`

```
par(mar=c(5,6,4,1)+.1)
time.seq<-1:28
plot(time.seq, cumulative_infections_per_1000['0-hop',], xlim=c(0,28),
ylim=c(0,600), xlab='Time (days)', bty = 'l', ylab='Cumulative
```

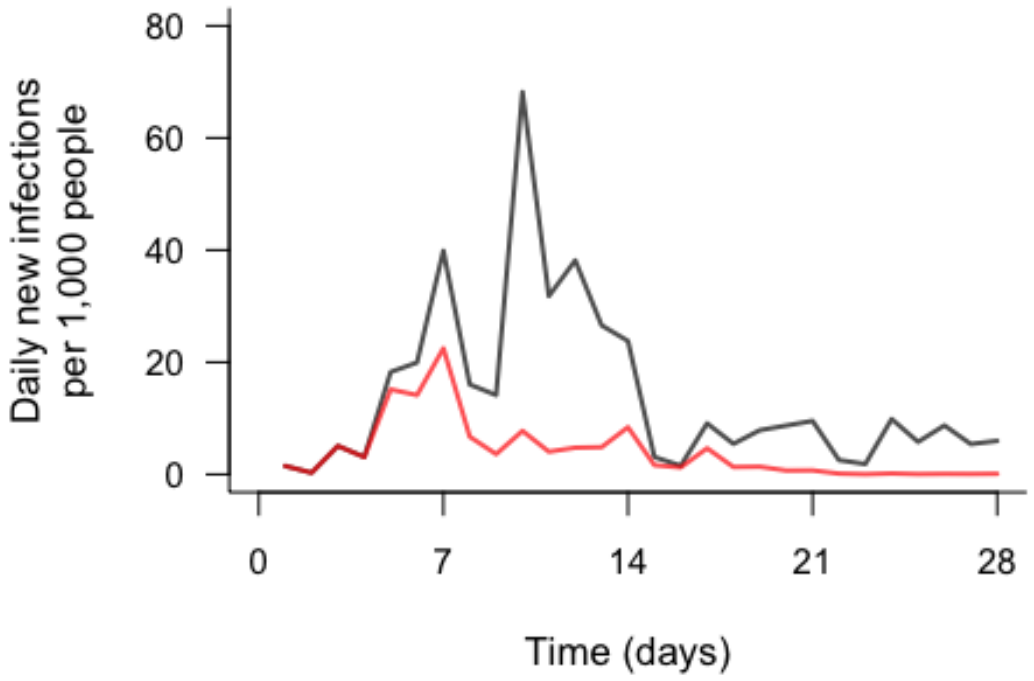
```
infections\nper 1,000 people', type='l', lwd=2, lty=2, xaxt='n', yaxt='n',
col=alpha('black',0.4), cex.axis=.95, cex.lab=1.05)
axis(2, at=seq(0,600,600/4), cex.axis=.95, las=2,
labels=formatC(seq(0,600,600/4), format="d", big.mark=','))
axis(1, at=seq(0,28,28/4), cex.axis=.95, las=1,
labels=formatC(seq(0,28,28/4), format="d", big.mark=','))
lines(time.seq, cumulative_infections_per_1000['1-hop',], lty=1, lwd=2,
col=alpha('black',0.7))
lines(time.seq, cumulative_infections_per_1000['2-hop',], lty=1, lwd=2,
col=alpha('blue',0.7))
lines(time.seq, cumulative_infections_per_1000['3-hop',], lty=1, lwd=2,
col=alpha('red',0.7))
lines(time.seq, cumulative_infections_per_1000['4-hop',], lty=1, lwd=2,
col=alpha('orange',0.9))
lines(time.seq, cumulative_infections_per_1000['5-hop',], lty=1, lwd=2,
col='skyblue3')
```



2. Bottom of Figure 5a in the main paper from `daily_new_infections_per_1000.csv`

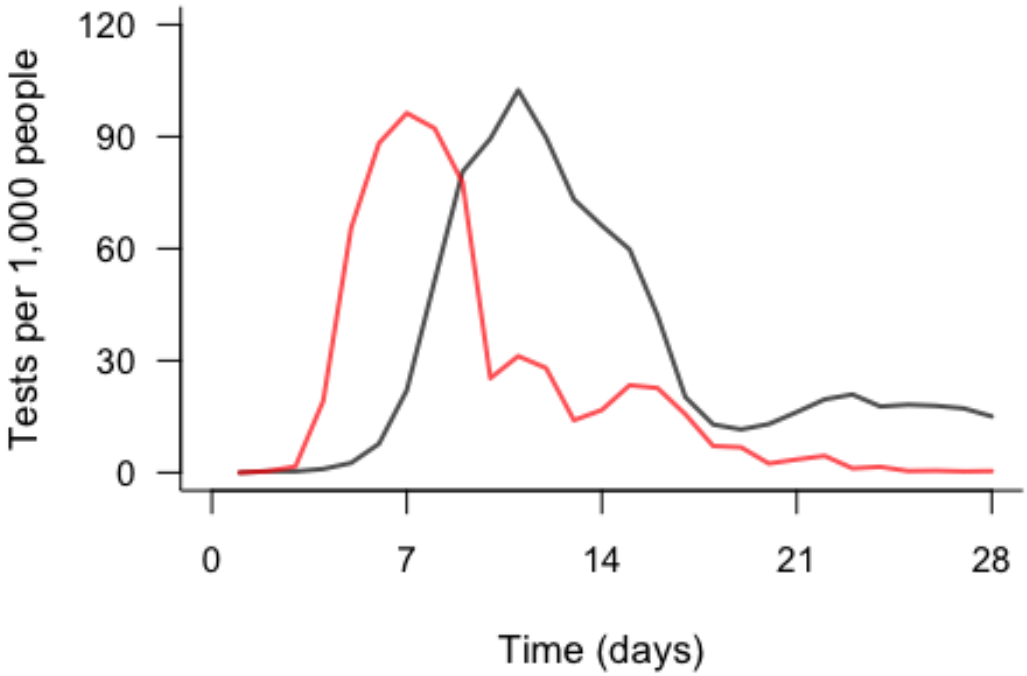
```
par(mar=c(5,6,4,1)+.1)
time.seq<-1:28
plot(time.seq, daily_new_infections_per_1000['1-hop',], xlim=c(0,28),
ylim=c(0,80), xlab='Time (days)', bty = 'l', ylab='Daily new infections\nper
1,000 people', type='l', lwd=2, lty=1, xaxt='n', yaxt='n',
```

```
col=alpha('black',0.7), cex.axis=.95, cex.lab=1.05)
axis(2, at=seq(0,80,80/4), cex.axis=.95, las=2,
labels=formatC(seq(0,80,80/4), format="d", big.mark=','))
axis(1, at=seq(0,28,28/4), cex.axis=.95, las=1,
labels=formatC(seq(0,28,28/4), format="d", big.mark=','))
lines(time.seq, daily_new_infections_per_1000['3-hop',], lty=1, lwd=2,
col=alpha('red',0.7))
```



3. Top of Figure 5a in the main paper from `test_per_1000.csv`

```
time.seq<-1:28
plot(time.seq, test_per_1000['1-hop',], xlim=c(0,28), ylim=c(0,120),
xlab='Time (days)', bty = 'l', ylab='Tests per 1,000 people', type='l',
lwd=2, lty=1, xaxt='n', yaxt='n', col=alpha('black',0.7), cex.axis=.95,
cex.lab=1.05)
axis(2, at=seq(0,120,120/4), cex.axis=.95, las=2,
labels=formatC(seq(0,120,120/4), format="d", big.mark=','))
axis(1, at=seq(0,28,28/4), cex.axis=.95, las=1,
labels=formatC(seq(0,28,28/4), format="d", big.mark=','))
lines(time.seq, test_per_1000['3-hop',], lty=1, lwd=2, col=alpha('red',0.7))
```


120
90
60
30
0
Tests per 1,000 people
0
7
14
21
28
Time (days)