Defining an epidemiologically meaningful contact from phone proximity events: uses for digital contact tracing

Christophe Fraser, Luca Ferretti, David Bonsall, Robert Hinch & Anthony Finkelstein

4 June 2020.

Version 3

Contents

Dynamic recursive contact tracing

The setting

We have previously proposed that digital contact tracing could play a significant role in the control of COVID-19 [Ferretti et al 2020]. The central mechanism is automated contact tracing: Individuals in a population install an app that recalls proximity events with other similarly equipped users, and then notifies past contacts if and when the individual is diagnosed. Many implementation and design issues need to be addressed, but this note concerns **only the issue of correctly identifying the most relevant contact events**, and ignoring the others. This is essential to increase the sensitivity and specificity of any intervention that relies on digital contact tracing. High sensitivity is achieved by maximising the number of people with COVID-19 in isolation; high specificity is achieved by minimising numbers of isolations for people without COVID-19

Overarching question

Consider a population of phone users who have installed an app that records proximity events with other phones, via an exchange of IDs with those neighbouring phones. The overarching question is how to map the proximity events between phones, and to convert that to the risk of transmitting infection. We propose:

1. an initial algorithm which *a priori* - based on first principles - will detect and respond to the most high risk events whilst imposing a maximum number of events, avoiding uncontrolled cascades.
2. We then propose a method by which the algorithm can be improved as the intervention continues.

The two basic events. Phone proximity, and infectious transmission of pathogens.

We start by considering a population of well-mixing individuals, a number of whom $N1$ are using phones with the tracing app installed, and a number of whom $N2$ are not. The overall user uptake of the app is $U=N1/(N1+N2)$.

Label individuals in the population who are using the app, `id1,id2=1...N1`, and assume that `id1,id2…` also label the networked phones.

If two individuals come into physical proximity, then each app will be detecting the presence of the other in the form of discrete 'pings'. These take place every `dt` time units - maybe fifteen seconds for the sake of argument. We cannot assume that `dt` is constant across phones, and so we denote `dt[id1]` the bluetooth scan frequency specific to each phone.

Some data may be exchanged by these pings: distance between the phones, approximate location, indication of whether indoors or outdoors, orientation, etc.

If one of the two individuals is infected and infectious, there is a probability that the virus will be transmitted to the other. The probability will also be a function of distance, time spent in proximity, and the other indicator variables.

The sensitivity and specificity of contact tracing will be improved by the best possible mapping of the data encoded and exchanged in the bluetooth pings onto the probability of transmission. It seems that *a priori* biologically plausible that this should be possible, since there are direct physical analogies between the transmission of low energy radio signals between phones, and the transmission of viral particles by aerosol.

Recording pings

Pings are recorded on each phone, along with corresponding data. A suggestion for the ping object design includes

```
ping <- vector{id, contact_id, time, distance,
                      context_a, context_b, …}
```

If `id=1,...N` identifies all the individuals, `contact.id` is the id of the other phone detected, time is the calendar time at which the ping occurred, and `context_a, context_b, etc..` are contextual variables. Each ping is uniquely identified, such that

```
ping[id,j]
```

Is the j.th ping recorded by the app of person id.

We recommend that values of `distance` of up to 5m be considered. These will be discounted by quadratic diffusion model later, so not inconsistent with current guidelines.

Deleting pings

Pings are recorded on each phone by the app, and deleted after a certain time t_max, i.e. ping j is kept if and only if

```
time - ping[id,k].time < t_max
```

`ping[id,k].time` records the time of the k.th ping, and so on. (NB: `time` and `ping[id,k].time` are time-date values)

`t_max` should be at least 14 days to allow for individuals reporting diagnosis some days after the appearance of symptoms.

Diagnosis, and uploading pings

If by whatever mechanism, individual i is diagnosed at time `t_diagnosis[id]`, and records the onset of their symptoms as `t_onset[id]` all of their pings are uploaded to a server for analysis. The server will store all pings from all diagnosed individuals.

Checking reciprocal pings to improve accuracy and calibration of ping data

It will occur that two individuals who have been diagnosed have experienced contact events, which will lead to a pair of users being identified as pairs {id1,id2} such that

```
ping[id1,.].contact_id == id2
ping[id2,.].contact_id == id1
```

It is then possible to match both sets of pings recorded for `id1` and `id2`, to check for consistency, and so develop an error model and/or contextual adjustments for the pings. This approach will serve to improve the accuracy and calibration of the measurements to different handsets and situations.

Becoming a case

After diagnosis (by self report, by test, by clinical diagnosis) individual `id1` is identified as an infected individual. `status[id1]` changes from `uninfected` to

```
status[id1] == {infected, time_diagnosis, time_since_infection}
```

Uploading contact information

Contact tracing is the dissemination of information (‘notifications’) to individuals listed in the vector of contact ids. This is done in two stages. First, each case uploads their contact list to the database. Without filtering the following list is uploaded:

```
{contact[id1]} == unique[{ping[id1,k].contact_id | for all k}]
```

for all possible k. This is the list of all people who have been pinged as contacts, excluding those that have expired. Pings expire if they occur more than 7 days before the onset of symptoms declared by the person reporting symptoms.

A naive approach to contact tracing (‘notification’) that should not be used

Each individual user in the population, `id2` ,regularly queries the central database to see whether they are listed as a contact:

```
is_a_contact == function(id2)
     if id2 in  {contact[id1]} return(yes) else return(no)
```

Person `id2` will then be identified as an at-risk contact.

```
status[id2] == {at_risk, time_notified, test_schedule_time}
```

`test_schedule_time` is the amount of time before a test is scheduled, not always zero since tests increase in sensitivity over time. (Not discussed here)

Problems, and risk scoring.

The problem with this approach is that there is filtering of contacts: all contacts are notified, potentially resulting in an uncontrolled process of contact tracing. This problem is addressed by introducing risk scores. An example risk score is ‘stayed within 2 metres for 7 minutes in one day’ This kind of risk score is well adapted for recollection by humans, but is suboptimal for processing phone pings, likely inaccurate [note from Google], and presents a risk of uncontrolled malfunction.

The uses of a risk score for controlling the number of notified individuals.

We propose to augment the list of contacts with a risk score

```
contact[id1] <- vector{id, risk_score, ...}
```

Where risk_score is a measure of the probability of transmission having occurred to the contact.

It would then be possible to gate notifications on a minimum risk score

```
notify.gated:
      if  contact[id1].risk_score > min_risk_score
      then notify contact[id1]
```

The minimum risk score could be set such as to control the total number of notifications given in the population. It is not however appropriate to control the number of notifications from a specific individual: if an individual has had unusually many high risk contacts, it is appropriate and correct to notify an unusually high number of people. Likewise, total numbers of notifications will change throughout the course of the epidemic, in relation to quantities of transmission and the proportion of people in social distancing.

Our proposed approach is therefore to choose `min_risk_score` such that the mean number of people notified per diagnosed individual is controlled.

To do this, first compute

```
for min_risk_score in 0 to max(risk_score)
      count = 0
      for id1 in {id1}
            if  contact[id1].risk_score > min_risk_score
            then count=count+1
      mean.notifications[min_risk_score] = count/length{id1}
```

And then choose `min_risk_score` such that `mean.notifications[min_risk_score]` has the desired value, `K`.

The parameter `K` is the parameter that tunes the contact tracing approach. The higher the value of `K`, the larger the mean number of people notified per index diagnosed individual.

If the basic reproduction number $R_0 \sim 3$, then on average each person will infect 3 and have been infected by 1. This process is however highly variable. A desirable mean number of people to notify might be `K = 20`. As long as the top ranked contacts include the source case and the three recipient cases, this will lead to a mean specificity of 4/20 = 20%, higher in a densely connected network. Increasing `K` will result in more people being quarantined, and so

necessarily more false positive notifications, but also reduced risk of false negatives, that is of people who were infected not being notified.

General interventions, such as lockdown, or contact tracing itself, will affect the reproduction number, reducing it to R_t. For example for the UK, the current estimate of R_t will be just below 1. Following the logic above, the desirable mean number of people to notify might be $K = 5*(R_t+1)$.

The user uptake of the app can also be taken into account in terms of choosing K. So for example if the user uptake is U in this particular subpopulation, then the proportion of contacts that will be recorded will be U and so an improved choice of K will be $K=K_0*U$, where $K_0 = 5*(R_t+1)$ is the number of people who should be notified in the case where all individuals are using the app. A further improvement is to account for assortative mixing, since 'being an app user' is likely to be an assortative trait on contact networks. If we label θ the assortativity, for which a prior may be 0.5, then the formula above becomes

$$K=5*(R_t+1)*(U(1-\theta)+\theta)$$

A starting configuration, with very low usage, and with Rt~1, suggests K=5 as a starting value. With maximum coverage and free mixing of the population, this would increase to K=20.

The next challenge is to develop risk scores.

The relationship between distance and transmission risk

As discussed in more details in a well-known study on the dynamics of droplets generated by coughing [Bourouiba et al, J. Fluid Mech. (2014), vol. 745, pp. 537-563], there are two contributions to the expirated cloud generated by cough, namely a buoyant cloud and droplets suspended inside it. The droplets travel within the cloud until a distance of 0.2-2 meters and reach a distance of 1-2 meters from the source, depending on the size of the droplet. We model this process as a constant risk within a minimum distance DMIN from the source. Normal speech has a similar distribution of droplet size as coughing [Xie et al, J. R. Soc. Interface (2009) 6, S703–S714] but the initial velocity of the droplets is about a quarter of the one measured when coughing [Kwon et al, Chemosphere 87 (2012) 1260–1264]. This reduces the initial momentum of the cloud. For this reason, we choose a final estimate of DMIN = 1 meter. Beyond that distance, the virus could be transported in aerosol [Nature 580, 175 (2020)] within the buoyant cloud, whose transverse section grows quadratically with the distance from the source [Bourouiba et al, J. Fluid Mech. (2014), vol. 745, pp. 537-563]. The amount of virus that can be deposited from the cloud on a small surface decreases therefore quadratically with distance. Under the assumption that risk is a continuous function of the distance from the source, we finally obtain the formula

risk(d) =I(dDMIN)+I(d>DMIN) (DMIN/d)2

where I(true)=1, I(false)=0.

Furthermore, very close proximity contacts (<1 m) are expected to be associated with physical contact between individuals (handshakes, hugging etc) and exchange of physical objects that may transfer the pathogen via fomites; this is a further reason to assume saturation of risk at very close proximity.

Proximity risk scoring, with epidemiological weighting

A simple model for risk scoring is to weight by the distance between individuals, and by the expected infectiousness of the individual.

```
ping[id,k].risk_score =
     if ping[id,k].distance < d_min
          risk.adjustment(context.a, context.b, ...)
     else if ping[id,k].distance > d_min
          risk.adjustment(context.a, context.b, ...)
               * d_min^2 / distance^2
```

where

```
risk.adjustment(context.a, context.b, ...)
```

is an adjustment for context, for example sharing household, being inside/outside etc. and `d_min` is ‘breathing volume’ where air is well mixed, before regular diffusion starts.

Examples might be

```
risk.adjustment(context.a=”household”, context.b=”indoors”, ...)=2
risk.adjustment(context.a=”household”, context.b=”outdoors”, ...)=1
risk.adjustment(context.a=”workplace”, context.b=”indoors”, ...)=1.5
risk.adjustment(context.a=”other”, context.b=”indoors”, ...)=1
risk.adjustment(context.a=”other”, context.b=”outdoors”, ...)=0.5
```

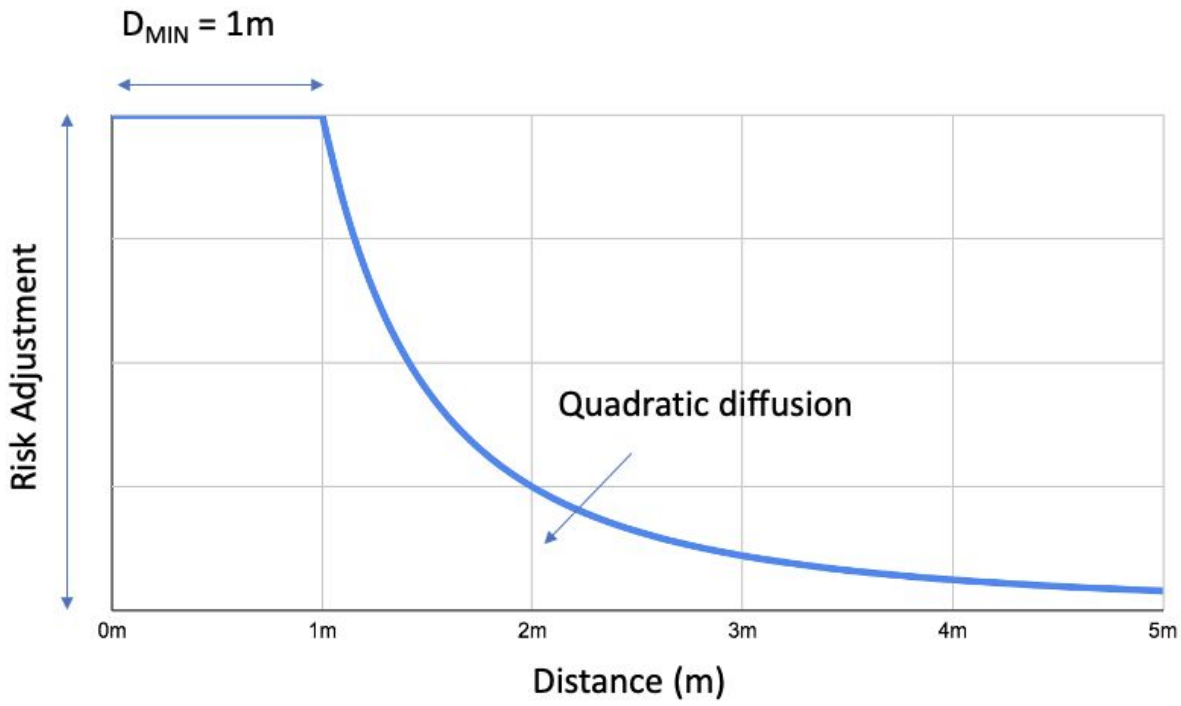

Figure 1: Proximity risk score. The distance between two individuals is used to estimate the potential for transmission. Within a short distance (<D_min; we proposed 1m), the potential risk is high and relatively constant (risk.adjustment$_{max}$). Potential risk decays proportional to the square of the distance at values greater than D_min. The scale of this function is controlled by additional adjustments (eg. context.a and context.b (see above)

A more realistic model will take into account not just the proximity and duration of contact, but also how infectious the person was likely to be at the time of contact, which is determined by the time elapsed between the ping and the onset of symptoms. Start as before

```
ping[id,k].epi_risk_score =
     ping[id,k].risk_score *
          infectiousness(time_diagnosis - time_since_onset -
ping[id,k].time)
```

For this purpose, we modify the approach of [Ferretti, Wymant et al, Science 2020] for the Maximum Likelihood inference of generation time, in order to include the distribution of time wrt symptoms rather than the generation time distribution. Beyond this modification, the approach is the same. We combine two different datasets with information about exposure periods and

onset of symptoms for both the infector and infectee, namely the 40 pairs in [Ferretti, Wymant et al, Science 2020] and the 36 clusters of size 2 included in the preprint [Xia et al, medrXiv, https://doi.org/10.1101/2020.03.06.20031955]. Combining both datasets, we find that the gaussian parametrization of the time wrt symptoms is a much better parametrization than the ones for the generation time (AIC=421 vs 435) and that the gaussian distribution has mean=-0.3 and SD=2.3. Similar conclusions are also true for the first dataset alone.
Considering an additional uncertainty of 1.5 days on the actual time of symptom onset, we finally obtain a time dependence for the risk normally distributed with mean=-0.3 and SD=2.75.

```
infectiousness(x) = normal_distribution(mean = -0.3, sd = 2.75)
```

We propose that the onset of symptoms be considered as midday on the day of onset, and that the infectiousness be measured by the probability distribution function evaluated at the exact time relative to the onset of symptoms.

The cumulative risk for a contact is then

```
contact[id1].epi_risk_score = function({ping[id1,.]})
     { score = 0
     for ping[id1,k] in {ping[id1,.]}
          if ping[id,k].contact_id == contact[id1].id
               score = score + ping[id,k].epi_risk_score*dt[id]
     return score }
```

Further epidemiological adjustments could be made here, the two most useful would be the age of person id1 and the symptom score. Adjusting for the age of the contact would also be useful if available.

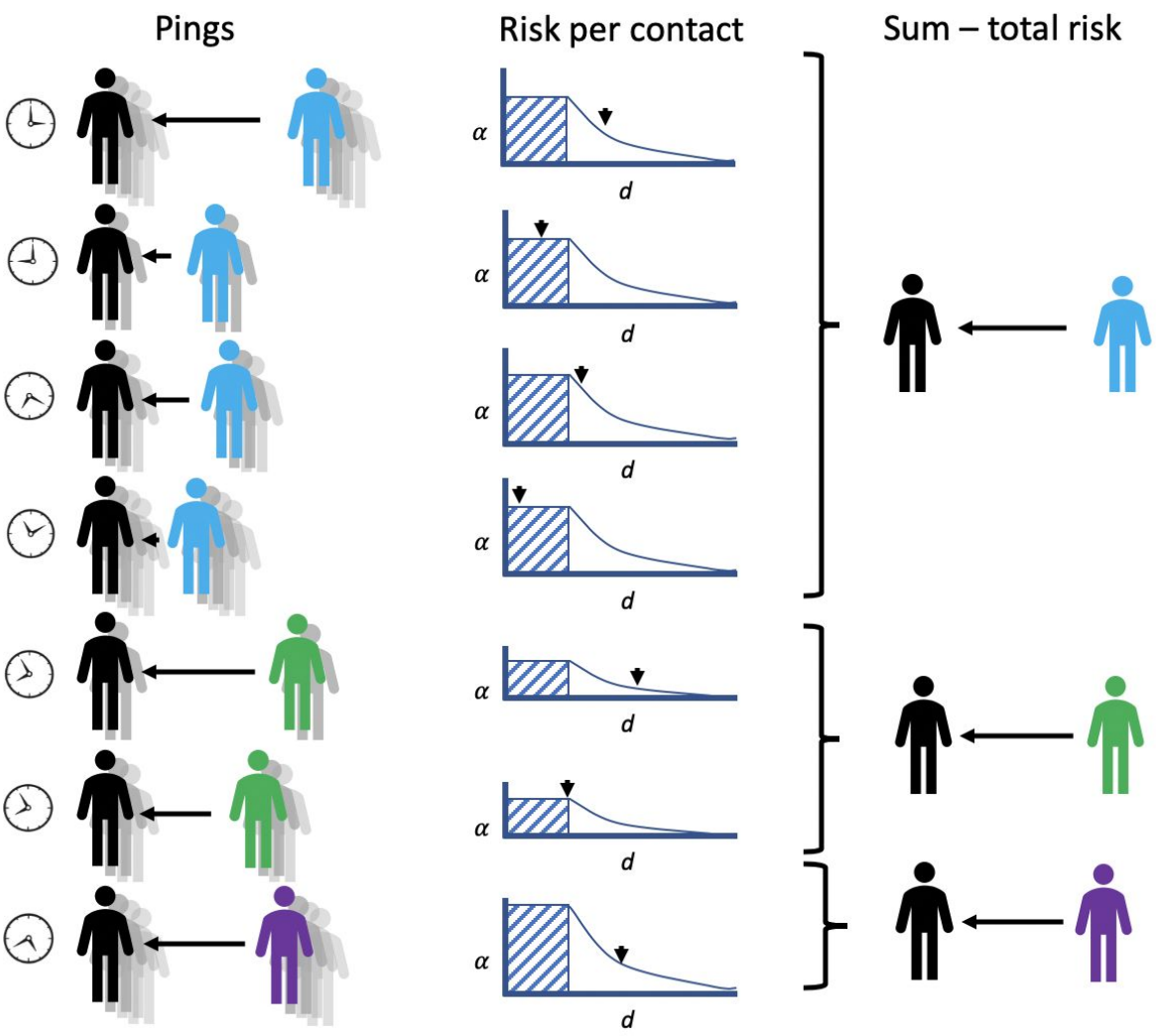

Figure 2: Diagram to explain how total risk scores between individuals are calculated for one individual (black icon). At different times the individual meets 4 individuals. Each encounter lasts a certain number of ‘pings’, and the potential risk adjustment for each ping is calculated (see fig 1). The total risk between two individuals is an integral function between all pings, for all separate encounters during the study period. We propose that all pings should be summed for the 14 days prior to the onset of symptoms in the reporting individuals.

Saturation

In epidemiology, it is often observed empirically that transmission rates saturate in prolonged contacts. This can be represented with a Hill function to saturate risk scores. There may be little benefit in this however, since only the rank of scores informs notification, and saturation does not affect rank scores.

Extending the reach of a single contact-tracing event

In Ferretti et al. we show that a proportion of direct, first-order contacts of symptomatic cases are likely to have transmitted by the time they have been traced. We therefore propose to extend the reach of a single contact-tracing event to the house-hold members of proximity-contacts. Early analyses suggest that this will achieve adequate sensitivity to control the epidemic and is a safe starting point for the point of app release. Specificity may be optimised as data is acquired. Integration of virological testing (eg. PCR) would also increase specificity.

The recursion, to the household members of contacts, can be done in an automated way, or simply by providing information that self-isolation and quarantine requirements need

Evaluation of risk scoring methods

The aim of scoring is to notify those who are or become infected (to correctly notify and quarantine true positives), to avoid failing to notify those who are or become cases (to avoid missing false negatives), to avoid notifying those are not and don't become cases (to avoid quarantining false positives), and to successfully avoid notifying those who are not and don't become true cases (to minimise quarantine of true negatives).

No system will be perfect, and the optimal trade-off between these four outcomes in terms of most rapid control of the epidemic can be explored by simulation.

There may be cases where false positive quarantines may still result in fewer total number quarantined, if the epidemic is controlled more quickly. This is currently being explored by simulation.

Improvement of risk scoring methods

The system can be analysed in terms of contacts probability of appearing as an index case, to improve the risk scoring models used.

Dynamic recursive contact tracing

Contact tracing can be recursed to speed up epidemic control.

One way to do this is to recurse in the presence of a high risk score for transmission.

```
initiate.recursion:
      if  contact[id1].risk_score > recursive_risk_score
      then ask.recursion contact[id1]
```

The function `ask.recursion` will notify the contact that they are at high risk in the usual manner, but also ask them to upload their own contacts, essentially acting as if they were presumed to be a case.

The `recursive_risk_score` will be higher than min_risk_score, and can be determined in the same way. An additional threshold can also be placed on second order contacts identified through recursion.