Chris Lab Notebook

Chris Hoover

5/11/2020

TODOs

5/11/20 Clarify source of iterations/updating with Josh
5/11/20 Get R_0 equation as function of model parameters from NextGen matrix
* Completed 5/11/20; see `LEMMA_model`

5/11/2020

Goals:
* Familiarize with model
* Code model in STAN and try some initial fits to get idea of runtimes, outputs
* Start brainstorming potential improvements to model

Package/Model functionality

`R/InputsFromSpreadsheets.R`

Contains functions to read user inputs and translate them into model inputs, generate parameter sets to sample from. Also main wrapper function that users run to generate report: `CredibilityIntervalFromExcel`

`R/CredibilityInterval.R`

contains functions to run model, check which model runs "fit" input data (e.g. which parameter sets generate predictions that agree with input data)

`R/CombinedModels.R`

contains functions to run seir model

`R/CreateOutputs.R`

Takes model posteriors and generate plots of projections, posterior distributions and comparison to prior distributions

Overall process

1. User calls `CredibilityIntervalFromExcel` which processes data in excel spreadsheet and then calls `CredibilityInterval`

2. `CredibilityInterval` does some more preprocessing of model inputs and then calls `RunSim1`

3. `RunSim1` calls `RunSim` which uses `FitSEIR` to generate reasonable estimates of when the outbreak started based on comparison of model generated outputs to observed data done with `CalcError` function. Once FitSEIR has found a best start data for each parameter set, `Seir` is run.

4. The output of `RunSim1` comes from running `Seir` which produces model estiamtes from the input parameter estimate and best guess start date. `InBounds` is then called to determine if the output of `Seir` is within the user provided uncertainty bounds of the observed data.
** There seems to be some sort of iteration updating going on that I can't quite figure out

Test Run

```
file.copy(system.file("extdata", "SF-April13.xlsx", package = "LEMMA", mustWork = TRUE), "example.xlsx")

LEMMA::CredibilityIntervalFromExcel("example.xlsx")
```

Works for the most part except for a fairly obscure ggplot error

5/15/2020

Didn't make it to STAN on the 11th, so want to try and get an idea of some of the advantages and disadvantages STAN has compared to other approaches and maybe get around to coding up a version of the model in STAN

STAN Pros:

- Hamiltonian MCMC in STAN is quite fast and would generate more posterior samples
- Could incorporate priors to constrain some of the parameters like hospitalization rate
- Could place a prior on t_0 and fit epidemic start time more explicitly

STAN Cons:

- Not sure how to implement the same type of model fit in STAN that incorporates the min and max guess (i.e. confirmed hospitalizations and hospitalizations + 30% suspected cases). Could put a distribution on it, but could end up being a bit wonky

5/18/2020

Fixed some algebra slip-ups in the $\mathcal{R}_0$ calculation for the model in the `LEMMA_model` document

5/19/2020

Working on some initial network-based agent based models. read Maya and Mark's `Adaptive Surveillance` doc. From a coding perspective, making the model run quickly and having fairly flexible implementation

of different interventions (e.g. testing, isolating) will be the most challenging. From a practical standpoint, having a realistic contact/network structure may be the most challenging. The `EpiModel` package seems like a good place to start, gonna toy around with that a bit now. Specifically, going to try constructing a stochastic network model with the package which uses the `tergm` and `networkDynamic` packages to implement separable-temporal exponential-family random graph models (STERGMs; whatever that means...) Basically following the tutorial on the EpiModel website

Working with `EpiModel` package

- Good background on networks and terminology here
  - Network characteristics to consider:
    - **degree distribution**: distribution of number of connections per node
    - **geodesic distribution**: distribution of the shortest path between nodes
    - **component size distribution**: distribution of the size of different components if multiple unconnected components in the network

Step 1) Develop network characteristics

Set network size and characteristics

```
require(EpiModel)

# Initialize network with n=1000
N <- 1000

nw <- network.initialize(n = N, directed = FALSE)

# Add characteristics to the vertexes of the network
  # add race characteristic; here 500 A and 500 W
  nw <- set.vertex.attribute(nw, "race", c(rep("A", 500), rep("W", 500)))
  # Add household characteristic (assumes mean HH size of 5, this could be more data-informed)
  nw <- set.vertex.attribute(nw, "house", sample(c(1:200), 1000, replace = T))
```

NOTE: could add these characteristics from a separate data frame in which household, race, occupation, dormitory, major, etc. could be correlated and drawn from an empirical distribution of some sort

Set network partnership formation formula

Here we declare characteristics of the network in order to generate edges. This consists of a `formation` formula that declares relationships between nodes based on their characteristics and a `target.stats` vector which gives the expected number of edges given the number of nodes, the network characteristics between nodes (e.g. their mean degree) and their relationships as declared in the `formation` formula.

```
formation <- ~edges + nodefactor("race") + nodematch("race") + concurrent

tot.mean.degree <- 2
raceA.mean.degree <- 1
prob.same.race.edge <- 0.75
prob.concurrent <- 0.7
```

```
target.stats <- c(N/2*tot.mean.degree, 500*raceA.mean.degree, (N/2*tot.mean.degree)*prob.same.race.edge
```

Get network dissolution formula

Rate at which network dissolves

```
coef.diss <- dissolution_coefs(dissolution = ~offset(edges), duration = 10)
coef.diss
```

```
## Dissolution Coefficients
## ======================
## Dissolution Model: ~offset(edges)
## Target Statistics: 10
## Crude Coefficient: 2.197225
## Mortality/Exit Rate: 0
## Adjusted Coefficient: 2.197225
```

Fit network

```
netfit1 <- netest(nw,
                  formation = formation,
                  target.stats = target.stats,
                  coef.diss = coef.diss,
                  edapprox = FALSE)
```

The above took forever to run, and I'm also beginning to question whether all this fuss is really worth it when the contact networks we want to simulate aren't necessarily all that complicated. `EpiModel` seems to be more suited for e.g. sexually transmitted diseases where contact networks relevant to transmission are clearly defined and are fairly dynamic and structured through time. For SARS-CoV2, relationships that lead to contacts that lead to transmission probably fall into one of two categories: highly persistent (e.g. households) or highly random (e.g. the person that I passed on the street that wasn't wearing a mask)

5/20/20

Bummed that I spent the entire day trying to get the `EpiModel` package to work only to give up on it, but I think it's best to just start with an ABM from scratch at this point. There are a couple good places to start, including in Maya and Mark's document, but also this code from epirecipe and this covid abm from oxford

Started coding the ABM (`COVID_ABM_V1.R`). Have code to generate a network as a matrix, but would be good to inform network characteristics with more data. Would be even better to code so that users could input network characteristics and resulting transmission probabilities and then simulate from there. Could be implemented as a spreadsheet with three columns: number of edges in this part of the network, transmission probability along these edges, and now I've forgotten the third one... Maybe specific transmission characteristics associated with each part of the network? Anyways, basically finished code to generate network structure, now working on simulating transmission on top of the network and then will incorporate interventions.

5/21/20

Potential improvements/ideas (running list)

Network models to examine sub-epidemics (like Mission study), implications for effectiveness of SiP, opening back up

Intro paper on network models here with corresponding R package and website with tutorials

Questions

- What are the characteristics of a network that gives rise to the findings of the Mission study?
- What are the implications of such a network for efficacy/limitations of SiP intervention? What are the implications for lifting SiP? i.e. how does the network change once SiP is lifted and how does transmission then propogate through this post-SiP network?
- How can test-trace-isolate be used to limit transmission through such a network?

Concepts

- **Degree** or **connectivity** (k) - number of neighbors an individual has; **degree distribution** (p_k) population level characteristic of connectivity
- **Size** of the network described by **distance** between two nodes - the length of the shortest path between them in the network; **diameter** of the network is the largest **distance** in the network
- **Mixing matrix** describes how individuals of type i within the network are connected to individuals of type j where "type" can refer to any characteristic that differs between individuals. **Assortative mixing** describes a network pattern in which individuals are more likely to interact with like individuals
- **local clustering** (ϕ) - how many pairs are shared between pairs of individuals in the network, e.g. given two pairs, how many of the networks between the two pairs will look like triangles in which A relates to B, B relates to C, AND A also relates to C.
- **Betweenness/Centrality** - Importance of particular individuals within the network; number of paths between i and j that pass through a particular node

Continuous time, stochastic model versions

I've got a continuous time stochastic model built with the `adaptivetau` package that's got a slightly different structure that I could work on building out a bit more/make more user friendly with similar input/output functionality. Would having three models all running projections help or make things more confusing? Maybe have this going on behind the scenes and convey where there's more or less agreement between the models to DPHs?

Quantifying impact: how many cases/hospitalizations/deaths avoided due to SiP?

Erlang distributed latent period, hospitalization length

Maybe a bit more of an academic exercise, but compartmental models make an implicit assumption that wait times between model states are exponentially distributed, which is rarely the case. Implementing "box cars", we can relax this assumption to make wait times Erlang (special case of a gamma distribution in which the shape parameter is a positive integer) distributed. Would be good for both E and H states given we have data on the distributions of latent period and hospitalization times we could fit to

Make some priors endogenous (e.g. non-adjustable), focus users on things most likely to vary by region/area/population:

- Population Size
- Start date of projection
- End date of projection
- Percent infected who are hospitalized
- Average hospital length of stay
- Timing and effectiveness ($\mathcal{R}_e$) of interventions