model_current_state

April 19, 2020

1 Extended Network Model

1.0.1 States

$S_s, S, E, I_n, I_a, I_s, I_d, R_d, R_u, D_d, D_u$

1.0.2 Transitions

$$S \to S_s$$

$$S \to E$$

$$S_s \to S$$

$$S_s \to E$$

$$E \to I_n$$

$$E \to I_a$$

$$I_n \to R_u$$

$$I_a \to I_s$$

$$I_s \to R_u$$

$$I_s \to D_u$$

$$I_s \to I_d$$

$$I_d \to R_d$$

$$I_d \to D_d$$

$$I_a \to I_d$$

$$E \to I_d$$

1.0.3 Seirsplus propensity

$$P[X_i = S \to E] = \left[p\frac{\beta I}{N} + (1-p)\frac{\beta \sum_{j \in C_{G(i)}} \delta_{X_j = I}}{|C_{G(i)}|} \right] \delta_{X_t = S}$$

where I is a number of infected, N total number of living individuals, β rate of transmission (may be individual for each node), $\delta_{X_j=I}$ is 1 if $X_j = I$, 0 otherwise, X_i state of individual i, $C_{G(i)}$ is a set of close contacts (nodes j, that edge (i,j) is in the given graph G).

1.0.4 Extended propensity

We use the same formula for $P[X_i = S - > E]$ with these modifications:

- I is the total number of I-states, i.e. sum of I_n, I_a, I_s, I_d
- $\delta_{X=I} = 1$ if and only if $X \in \{I_n, I_a, I_s, I_d\}$
- edges has weights
- $|C_{G(i)}|$ is not the number of edges from i, but the sum of their weights (!there is only one edge between (i,j) in G)
- $\sum_{j \in C_{G(i)}} \delta_{X_j=I}$ is replaced by $\sum_{j \in C_{G(i)}} w_{(i,j)} \delta_{X_j=I}$

1.0.5 Constraction G from G_multi

Graph G_{multi} may contain more edges between i, j. Each edge has type t and subtype s and weight $w_{i,j,t,s}$ (intensity).

Graph G has max one edge (i,j) between nodes i, j. Edge as weight $w_{i,j}$.

$w_{i,j,t} = \sum_s w_{i,j,t,s}$ (sum over intensities of sublayers)

Weight $w_{i,j} = 1 - \prod_t (1 - l_t * w_{i,j,t})$ (weights on layers t are taking as probabilities of contact on that layer and are the final weight is probabily of contact on any layer)

l_t is coeficient of layer t

1.0.6 SEIRS + testing

in fact, we use the variant with testing, where I_d states has their own probs, TODO rewrite formulas

```
propensities[("S", "S_s")] = model.false_symptoms_rate*(model.X == "S")

#  "S" -> "E"
numI = model.current_state_count(
    "I_n") + model.current_state_count("I_a") + model.current_state_count("I_s")

S_to_E_koef = (
    model.p * (
        model.beta * numI +
        model.q * model.beta_D * model.current_state_count("I_d")
    ) / model.current_N()
    +
    (1 - model.p) * np.divide(
        model.beta * numContacts_I +
        model.beta_D * numContacts_Id, model.degree, out=np.zeros_like(model.degree), wher
    )
)
propensities[("S", "E")] = S_to_E_koef * (model.X == "S")

propensities[("S_s", "S")
             ] = model.false_symptoms_recovery_rate*(model.X == "S_s")

# becoming exposed does not depend on unrelated symptoms
propensities[("S_s", "E")] = S_to_E_koef * (model.X == "S_s")

exposed = model.X == "E"
propensities[("E", "I_n")] = model.asymptomatic_rate * \
    model.sigma * exposed
propensities[("E", "I_a")] = (
    1-model.asymptomatic_rate) * model.sigma * exposed

propensities[("I_n", "R_u")] = model.gamma * (model.X == "I_n")

asymptomatic = model.X == "I_a"
propensities[("I_a", "I_s")
             ] = model.symptoms_manifest_rate * asymptomatic

symptomatic = model.X == "I_s"
propensities[("I_s", "R_u")] = model.gamma * symptomatic
propensities[("I_s", "D_u")] = model.mu_I * symptomatic
```

```
    detected = model.X == "I_d"
    propensities[("I_d", "R_d")] = model.gamma_D * detected
    propensities[("I_d", "D_d")] = model.mu_D * detected

    # testing  TODO
    propensities[("I_a", "I_d")] = (
        model.theta_Ia + model.phi_Ia * numContacts_Id) * model.psi_Ia * asymptomatic

    propensities[("I_s", "I_d")] = (
        model.theta_Is + model.phi_Is * numContacts_Id) * model.psi_Is * symptomatic

    propensities[("E", "I_d")] = (
        model.theta_E + model.phi_E * numContacts_Id) * model.psi_E * exposed
```

2 Engine

2.1 Sequential implementation

Iterates in days, and each day for each node decides to which state it moves (including current state).

```
for t in 1, ... , T:
     propensities = calculate_propensities()
     # for each node select one action (including X->X) based on propensities[node_id]
     trainsitions = select_transitions(propensities)
     # update node states according selected transitions
     update_states(transitions)
```

3 Model paremeters overview

Scalars: (but last four can be also made to be vectors)

parameter	desription
p	probability of interaction outside adjacent nodes
q	probability of detected individuals interaction outside adjacent nodes
false_symptoms_rate	probability of S -> S_s
false_symptoms_recovery_rate	probability of S_s -> S
asymptomatic_rate	rate of going to I_n instead of I_a from E
symptoms_manifest_rate	probability of I_a -> I_s

Vectors: (each node has its own value, if only a single float is given than all nodes share the same value)

parameter	desription
beta	rate of transmission (exposure)
sigma	rate of infection (upon exposure)
gamma	rate of recovery (upon infection)
mu_I	rate of infection-related death
beta_D	rate of transmission (exposure) for detected inds
gamma_D	rate of recovery (upon infection) for detected inds
mu_D	rate of infection-related death for detected inds
theta_E	rate of baseline testing for exposed individuals
theta_Ia	rate of baseline testing for Ia individuals
theta_Is	rate of baseline testing for Is individuals
phi_E	rate of contact tracing testing for exposed individuals
phi_Ia	rate of contact tracing testing for Ia individuals
phi_Is	rate of contact tracing testing for Is individuals
psi_E	probability of positive test results for exposed individuals
psi_Ia	probability of positive test results for Ia individuals
psi_Is	probability of positive test results for Is individuals)

[ ]:

```

```