Efficient simulation and graphical modeling of Covid-19 spread

Group 13
Camille Bean, Intekhab Hossain, Hui Li and Stephanie Wu

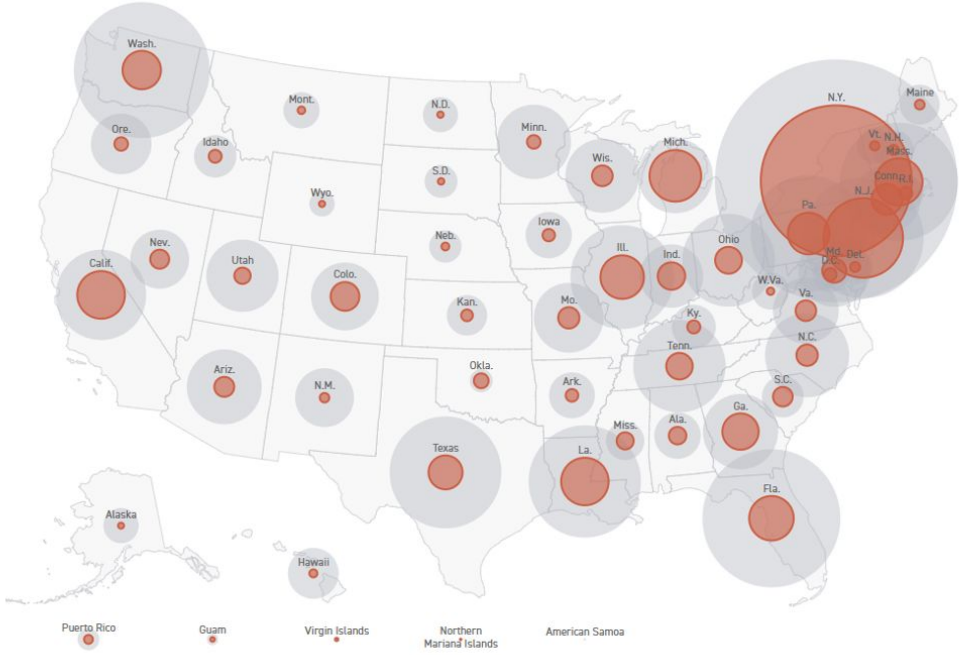
Overview

Quick Recap

Efficient simulation and graphical modeling of the spread of Covid-19 in a local community, using real Covid-19 incidents data and human interaction network data online.

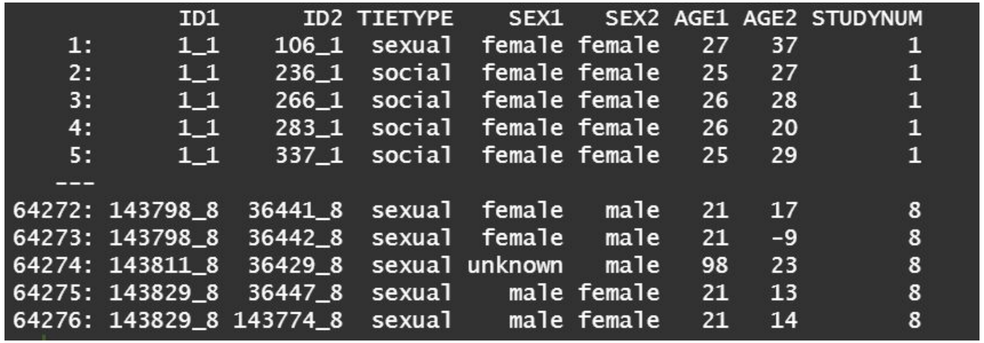
We want to apply our simulation model to:

- ➔ evaluate the impact of preventative measures on public health outcomes;



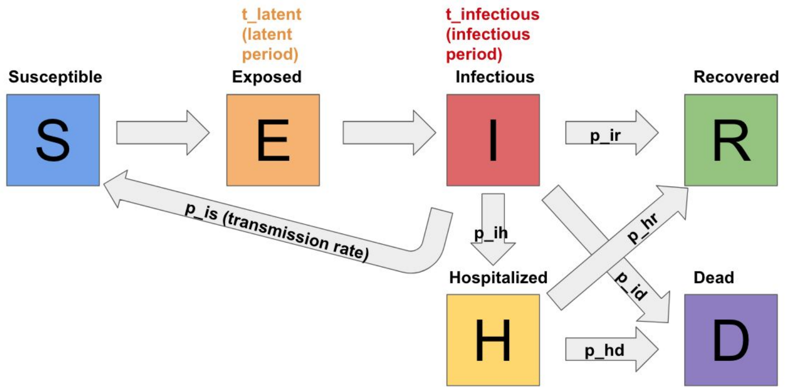
Coronavirus by state map. *"Live tracker: How many coronavirus cases have been reported in each U.S. state?" Politico.* March, 16, 2020.

Network dataset and the SEIRHD model



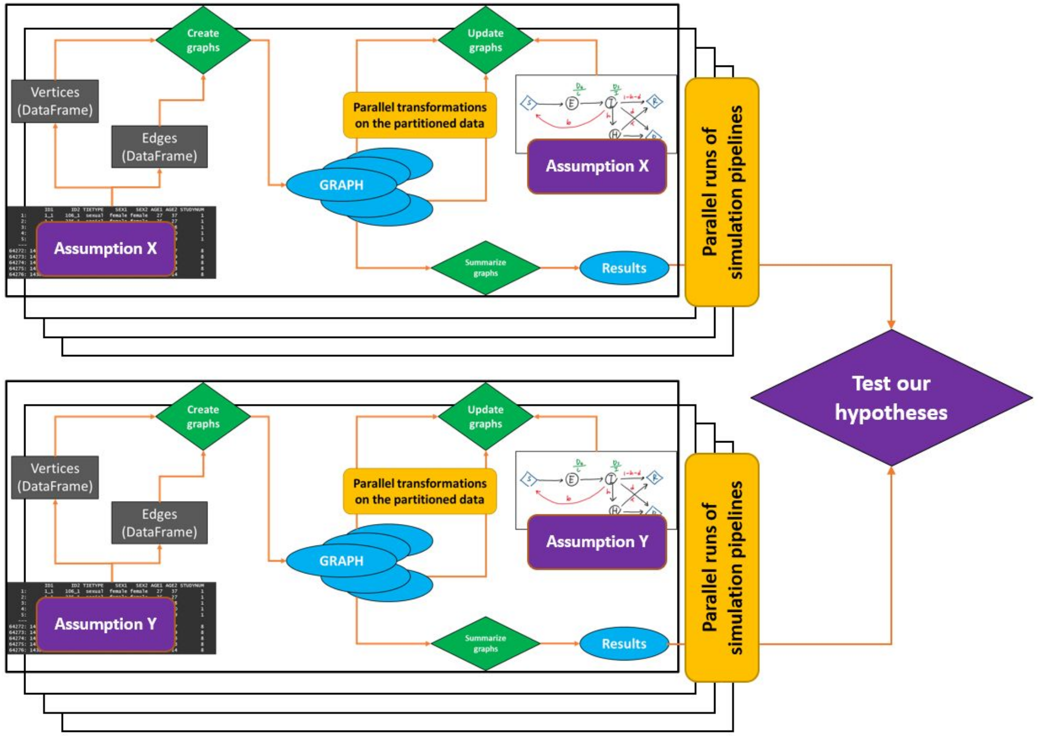
```
            ID1       ID2 TIETYPE    SEX1   SEX2 AGE1 AGE2 STUDYNUM
    1:      1_1     106_1  sexual  female female   27   37        1
    2:      1_1     236_1  social  female female   25   27        1
    3:      1_1     266_1  social  female female   26   28        1
    4:      1_1     283_1  social  female female   26   20        1
    5:      1_1     337_1  social  female female   25   29        1
   ---
64272: 143798_8   36441_8  sexual  female   male   21   17        8
64273: 143798_8   36442_8  sexual  female   male   21   -9        8
64274: 143811_8   36429_8  sexual unknown   male   98   23        8
64275: 143829_8   36447_8  sexual    male female   21   13        8
64276: 143829_8 143774_8  sexual    male female   21   14        8
```

HIV Transmission Network Metastudy Project: An Archive of Data From Eight Network Studies, 1988--2001 (ICPSR 22140)



	Estimate	Distribution	Estimation procedure
Transmission rate	1.596 (time-series data available)	Gamma	Used EpiEstim software, assuming a serial interval of 3.96 (4.75) from Du et al. 2020. U.S. national level data from Covid Tracking Project, dates ranging from 2/19 to 4/18.
Hospitalization rate	0.0678	Beta	Daily hospitalized increase / Daily case increase. U.S. national level data from Covid Tracking Project, averaged across available dates from 2/19 to 4/18.
Death rate	0.0419	Beta	Daily death increase / Daily case increase. U.S. national level data from Covid Tracking Project, averaged across available dates from 2/19 to 4/18.
Recovery rate	0.3945	Beta	Daily new recovery / Daily hospitalized increase. U.S. national level data from Covid Tracking Project, averaged across available dates from 2/19 to 4/18.
Latent period	5.2 days	Poisson	Taken from the literature (An et al. 2020)
Infectious period	2.3 days	Poisson	Taken from the literature (An et al. 2020)

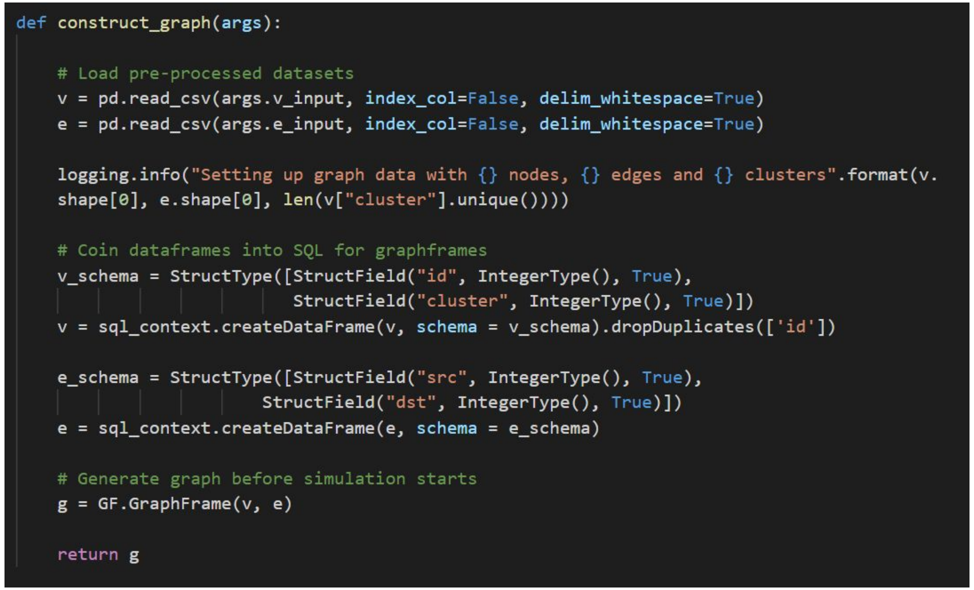
Big Data Application with a GraphFrames solution



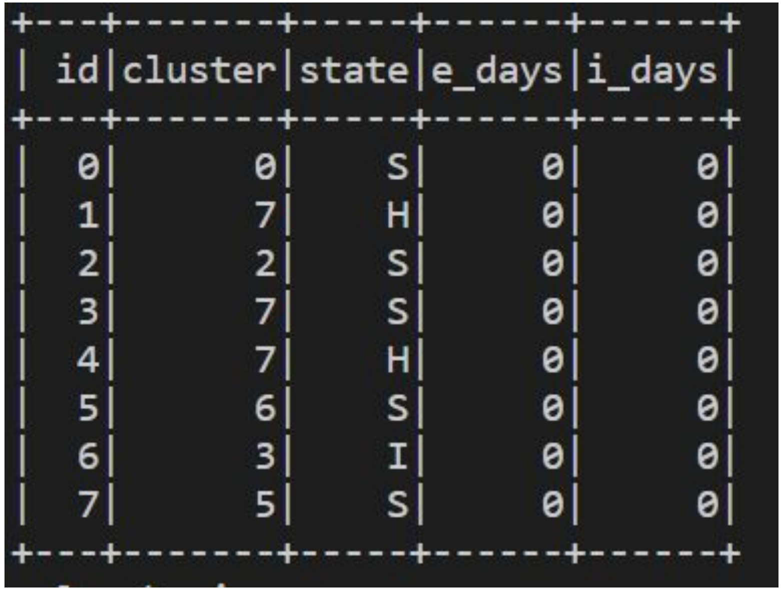
Source:http://www.bccdc.ca/health-info/diseases-conditions/covid-19/about-covid-19/how-it-spreads

Implementation Features - Graphs

- ➢ **Graph construction**
- ➢ **Use vertices attributes to keep track of state transition of individuals**



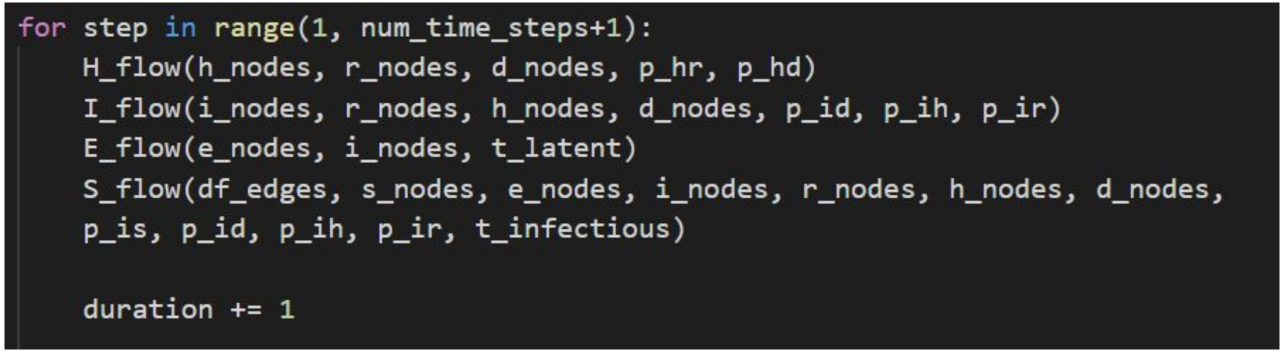
```python
def construct_graph(args):

    # Load pre-processed datasets
    v = pd.read_csv(args.v_input, index_col=False, delim_whitespace=True)
    e = pd.read_csv(args.e_input, index_col=False, delim_whitespace=True)

    logging.info("Setting up graph data with {} nodes, {} edges and {} clusters".format(v.
    shape[0], e.shape[0], len(v["cluster"].unique())))

    # Coin dataframes into SQL for graphframes
    v_schema = StructType([StructField("id", IntegerType(), True),
                           StructField("cluster", IntegerType(), True)])
    v = sql_context.createDataFrame(v, schema = v_schema).dropDuplicates(['id'])

    e_schema = StructType([StructField("src", IntegerType(), True),
                        StructField("dst", IntegerType(), True)])
    e = sql_context.createDataFrame(e, schema = e_schema)

    # Generate graph before simulation starts
    g = GF.GraphFrame(v, e)

    return g
```



```
+---+-------+-----+------+------+
| id|cluster|state|e_days|i_days|
+---+-------+-----+------+------+
|  0|      0|    S|     0|     0|
|  1|      7|    H|     0|     0|
|  2|      2|    S|     0|     0|
|  3|      7|    S|     0|     0|
|  4|      7|    H|     0|     0|
|  5|      6|    S|     0|     0|
|  6|      3|    I|     0|     0|
|  7|      5|    S|     0|     0|
+---+-------+-----+------+------+
```

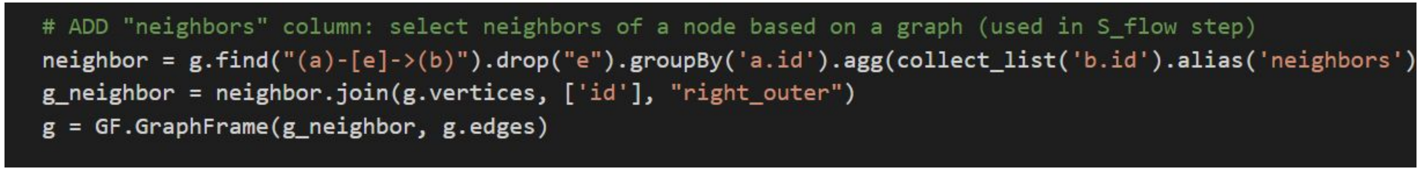
Implementation Features - Network updates

➢ **Update the graphical network at each time-step**



```
for step in range(1, num_time_steps+1):
    H_flow(h_nodes, r_nodes, d_nodes, p_hr, p_hd)
    I_flow(i_nodes, r_nodes, h_nodes, d_nodes, p_id, p_ih, p_ir)
    E_flow(e_nodes, i_nodes, t_latent)
    S_flow(df_edges, s_nodes, e_nodes, i_nodes, r_nodes, h_nodes, d_nodes,
    p_is, p_id, p_ih, p_ir, t_infectious)

    duration += 1
```

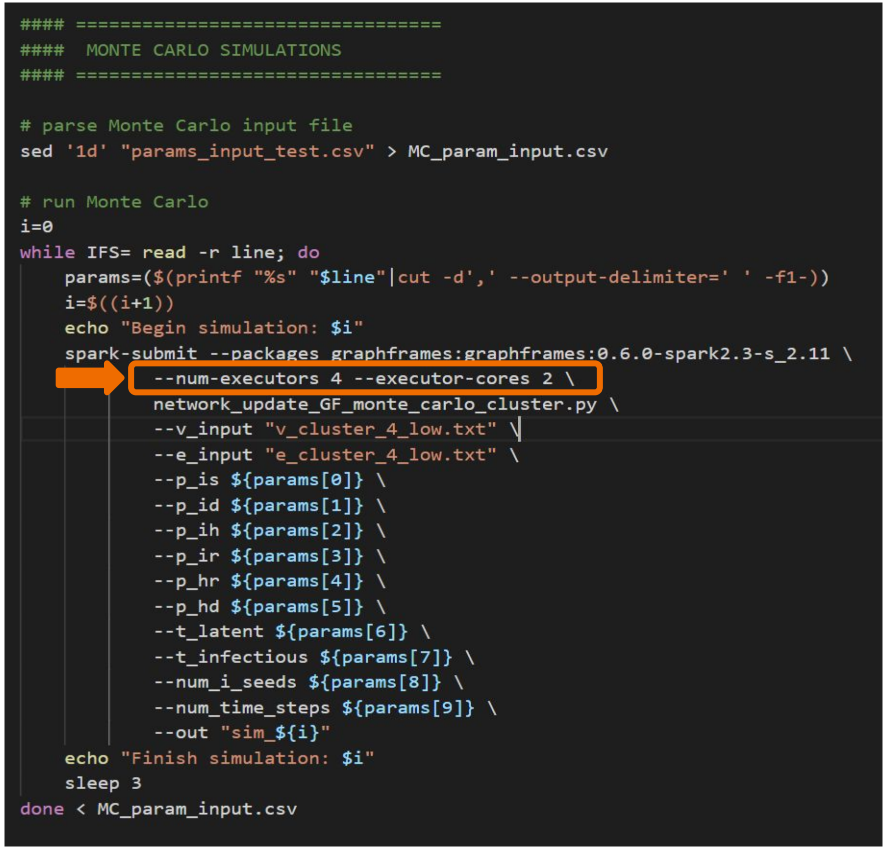
➢ **Use motif finding to locate neighbors**



```
# ADD "neighbors" column: select neighbors of a node based on a graph (used in S_flow step)
neighbor = g.find("(a)-[e]->(b)").drop("e").groupBy('a.id').agg(collect_list('b.id').alias('neighbors')
g_neighbor = neighbor.join(g.vertices, ['id'], "right_outer")
g = GF.GraphFrame(g_neighbor, g.edges)
```

Implementation Features - Parallelization levels

- Task-level parallelization
  - SPMD
  - Distributed across multiple instances and threads in the cluster

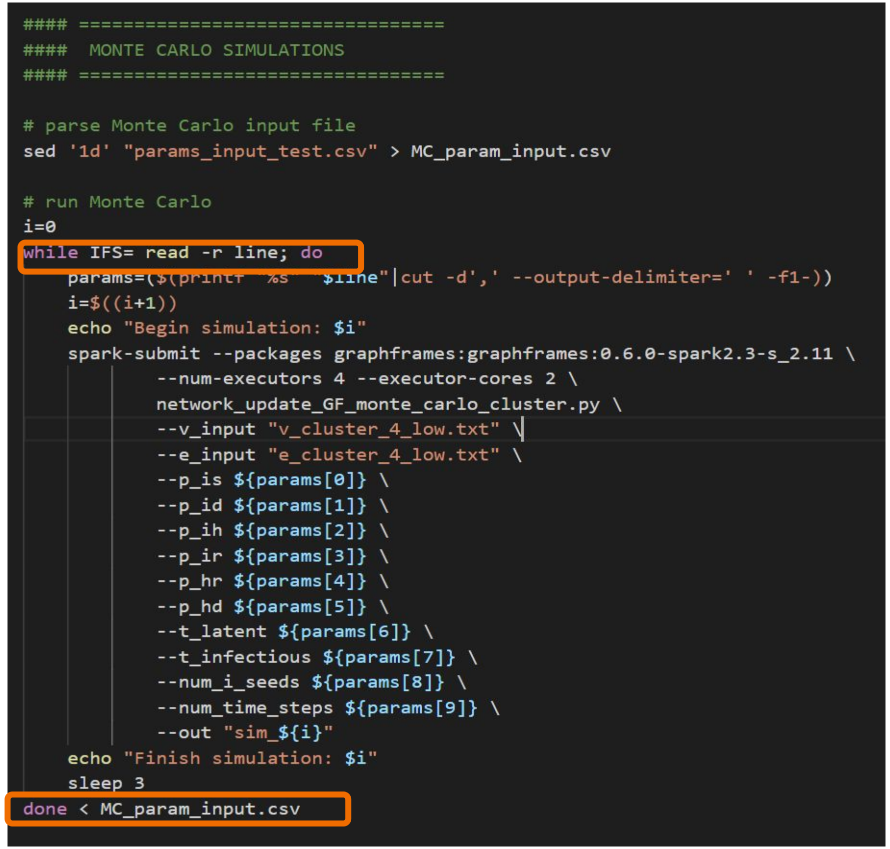


```
#### ================================
####  MONTE CARLO SIMULATIONS
#### ================================

# parse Monte Carlo input file
sed '1d' "params_input_test.csv" > MC_param_input.csv

# run Monte Carlo
i=0
while IFS= read -r line; do
    params=($(printf "%s" "$line"|cut -d',' --output-delimiter=' ' -f1-))
    i=$((i+1))
    echo "Begin simulation: $i"
    spark-submit --packages graphframes:graphframes:0.6.0-spark2.3-s_2.11 \
            --num-executors 4 --executor-cores 2 \
            network_update_GF_monte_carlo_cluster.py \
            --v_input "v_cluster_4_low.txt" \
            --e_input "e_cluster_4_low.txt" \
            --p_is ${params[0]} \
            --p_id ${params[1]} \
            --p_ih ${params[2]} \
            --p_ir ${params[3]} \
            --p_hr ${params[4]} \
            --p_hd ${params[5]} \
            --t_latent ${params[6]} \
            --t_infectious ${params[7]} \
            --num_i_seeds ${params[8]} \
            --num_time_steps ${params[9]} \
            --out "sim_${i}"
    echo "Finish simulation: $i"
    sleep 3
done < MC_param_input.csv
```

Implementation Features - Parallelization levels

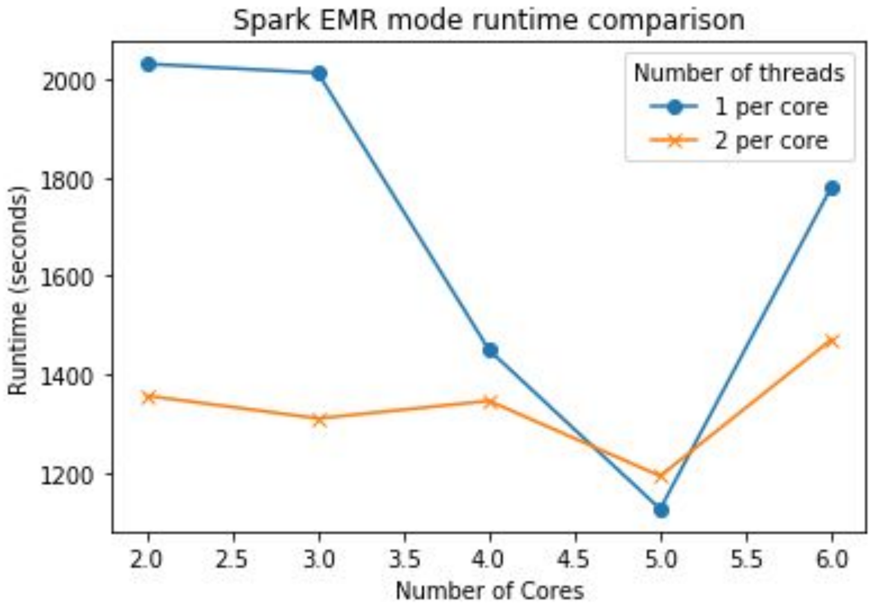
- Task-level parallelization
  - SPMD
  - Distributed across multiple instances and threads in the cluster

- Monte-Carlo-level parallelization
  - Independent runs
  - Embarrassingly parallel



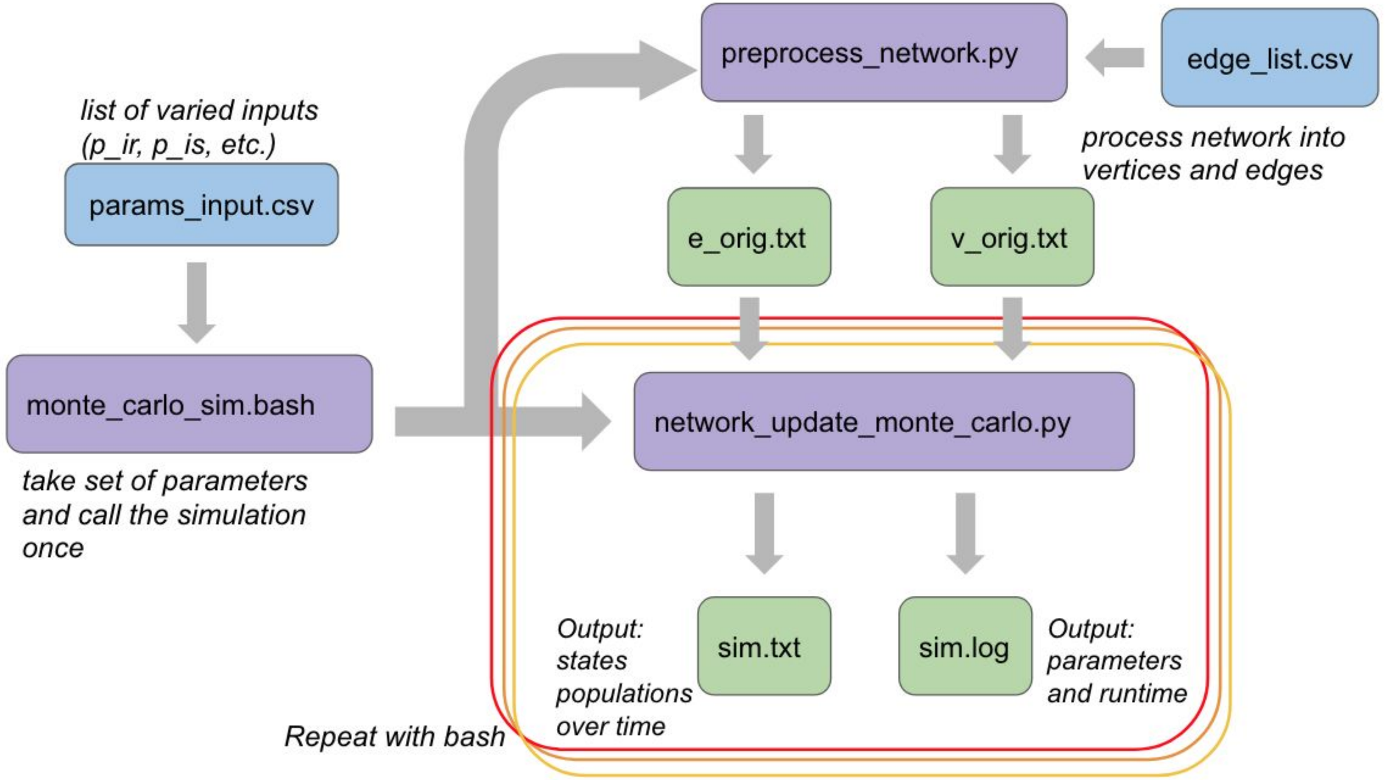
```
#### =================================
####  MONTE CARLO SIMULATIONS
#### =================================

# parse Monte Carlo input file
sed '1d' "params_input_test.csv" > MC_param_input.csv

# run Monte Carlo
i=0
while IFS= read -r line; do
    params=($(printf "%s" "$line"|cut -d',' --output-delimiter=' ' -f1-))
    i=$((i+1))
    echo "Begin simulation: $i"
    spark-submit --packages graphframes:graphframes:0.6.0-spark2.3-s_2.11 \
            --num-executors 4 --executor-cores 2 \
            network_update_GF_monte_carlo_cluster.py \
            --v_input "v_cluster_4_low.txt" \
            --e_input "e_cluster_4_low.txt" \
            --p_is ${params[0]} \
            --p_id ${params[1]} \
            --p_ih ${params[2]} \
            --p_ir ${params[3]} \
            --p_hr ${params[4]} \
            --p_hd ${params[5]} \
            --t_latent ${params[6]} \
            --t_infectious ${params[7]} \
            --num_i_seeds ${params[8]} \
            --num_time_steps ${params[9]} \
            --out "sim_${i}"
    echo "Finish simulation: $i"
    sleep 3
done < MC_param_input.csv
```

Performance Evaluation

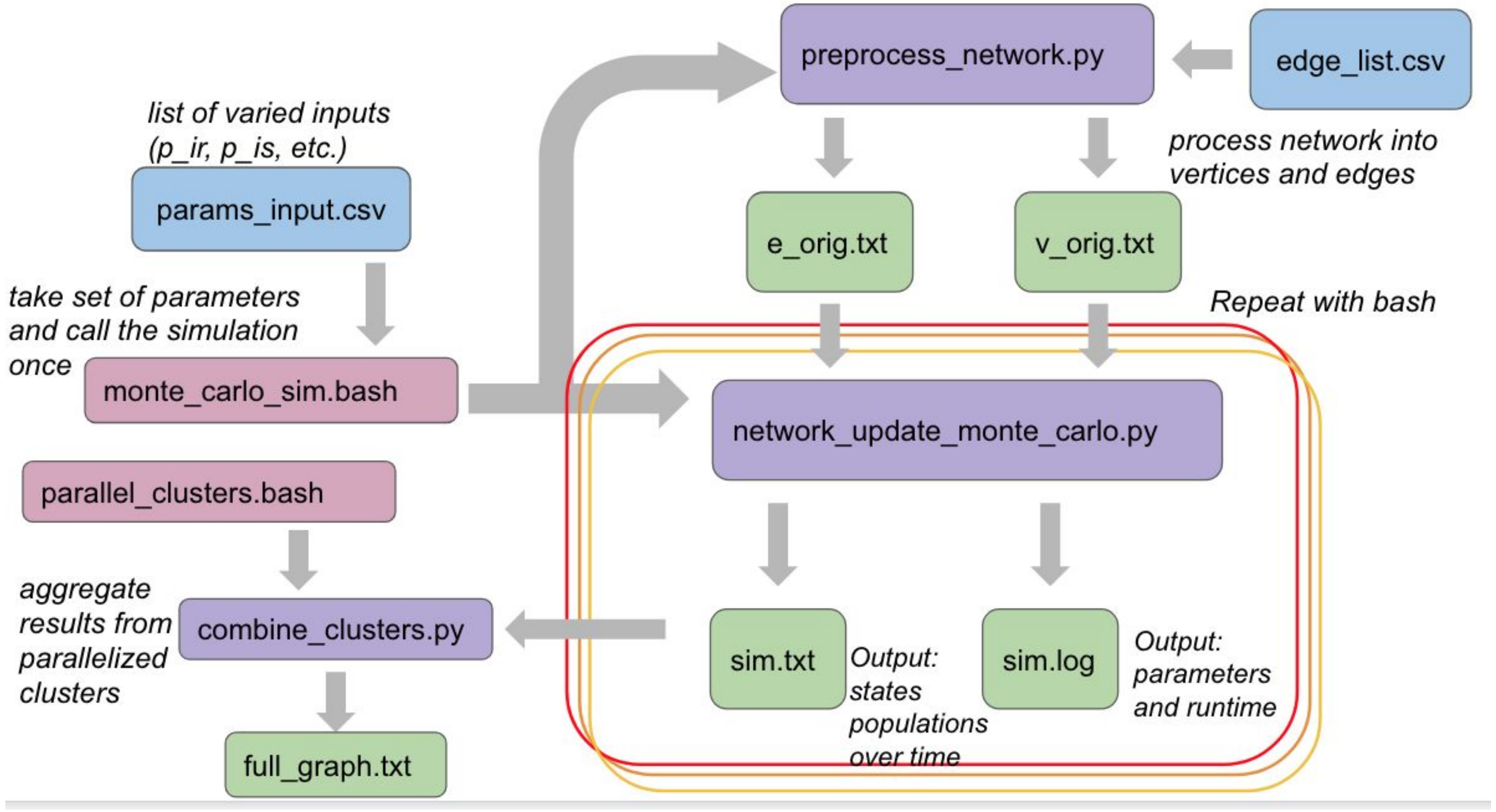


- ➢ When a limited number cores are available, it is optimal to use a double-threaded approach.
- ➢ With 5 processors available, the single-threaded approach performs just as well, if not better, than the double-threaded approach.
- ➢ The inherent stochasticity built into the graph updating step may lead to different runtimes depending on the run, so these results may differ slightly if we were to repeat the experiment.
- ➢ **We ran all scaled-up tests on AWS EMR with 4 cores x 2 threads per cores.**

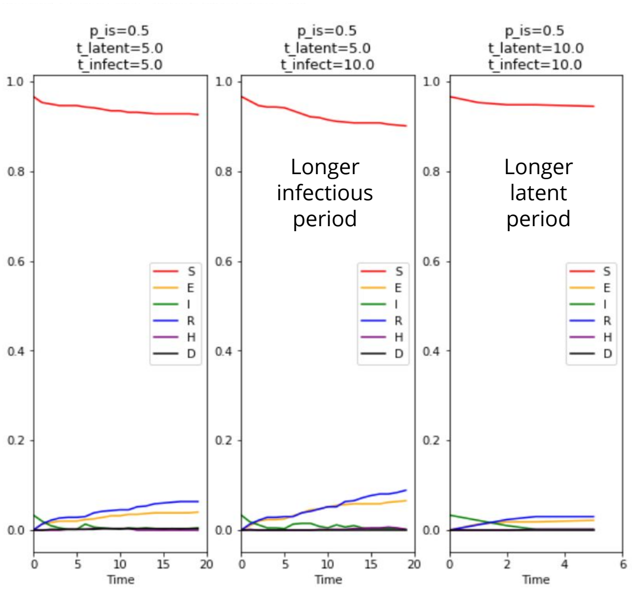
Code Structure - Monte Carlo


preprocess_network.py
edge_list.csv
list of varied inputs (p_ir, p_is, etc.)
params_input.csv
process network into vertices and edges
e_orig.txt
v_orig.txt
monte_carlo_sim.bash
network_update_monte_carlo.py
take set of parameters and call the simulation once
Output: states populations over time
sim.txt
sim.log
Output: parameters and runtime
Repeat with bash

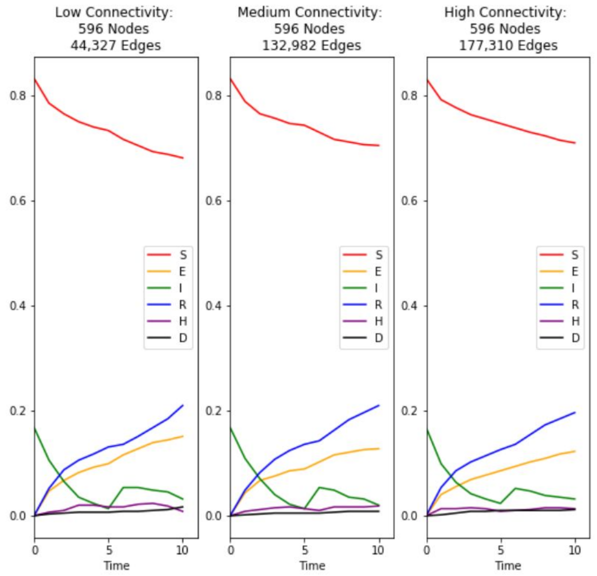
Code Structure - Monte Carlo with Study-Level Parallelization


preprocess_network.py
edge_list.csv
list of varied inputs (p_ir, p_is, etc.)
params_input.csv
process network into vertices and edges
e_orig.txt
v_orig.txt
take set of parameters and call the simulation once
Repeat with bash
monte_carlo_sim.bash
network_update_monte_carlo.py
parallel_clusters.bash
aggregate results from parallelized clusters
combine_clusters.py
sim.txt
Output: states populations over time
sim.log
Output: parameters and runtime
full_graph.txt

Pipeline results



Monte Carlo simulations with different latent and infectious periods

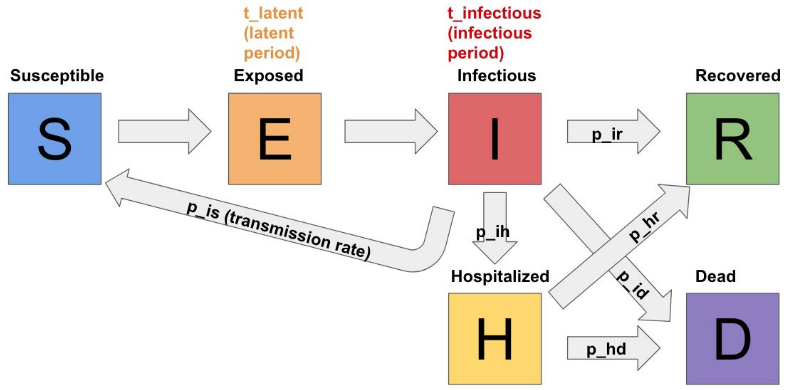


Varying connectivity levels to model airborne transmission and social distancing

- Longer infectious period increases epidemic
- Little effect of connectivity
- Epidemic dies out

Challenges and Improvements

- ➢ **SEIR model implementation in GraphFrames**
  - ○ Incorporate latent and infectious periods
- ➢ **Incorporate entire underlying network**
- ➢ **Longer maximum duration for time**



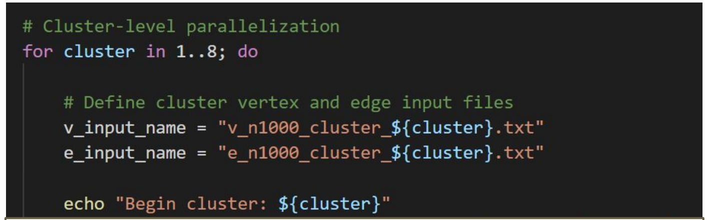
Key Takeaways

Goals Achieved:

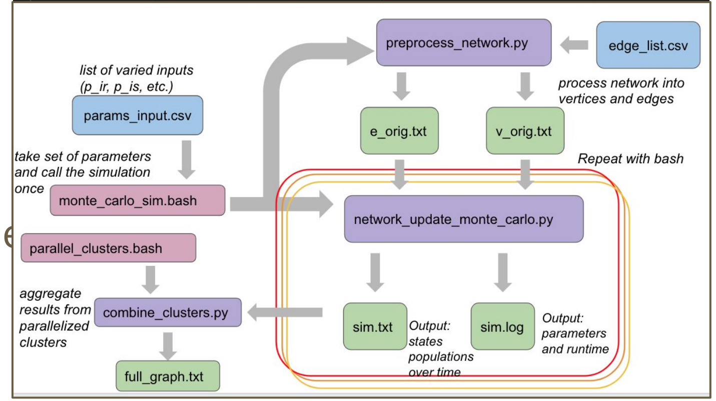
- Developed code serial and parallel and incorporated realistic network and parameters
- Speed-Up through Spark parallel-processing
- Varying connectivity, study-level parallelization code and bash pipeline established

Future Work:

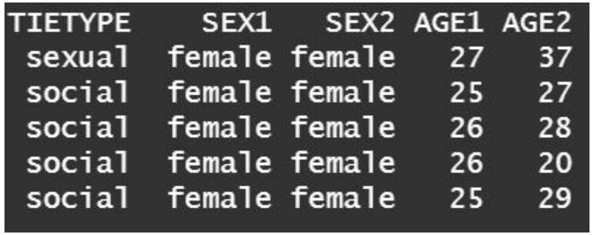
- Weighting edges by tie strength (e.g., sexual, social)
- Time-dependent transmission probabilities
- Transmission probabilities influenced by vertex attributes (e.g., age, gender)
- Study-level pipeline and MPI parallelization
- More extensive summary statistics



```
# Cluster-level parallelization
for cluster in 1..8; do

    # Define cluster vertex and edge input files
    v_input_name = "v_n1000_cluster_${cluster}.txt"
    e_input_name = "e_n1000_cluster_${cluster}.txt"

    echo "Begin cluster: ${cluster}"
```

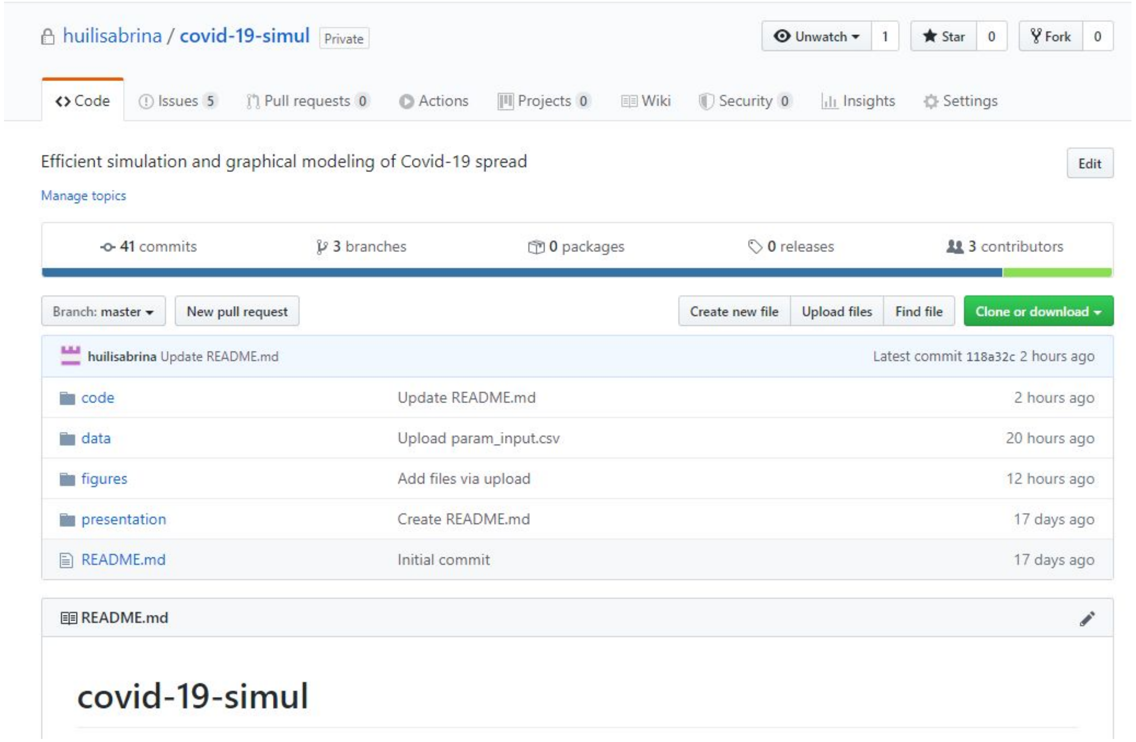


preprocess_network.py
edge_list.csv
process network into vertices and edges
list of varied inputs (p_ir, p_is, etc.)
params_input.csv
e_orig.txt
v_orig.txt
take set of parameters and call the simulation once
monte_carlo_sim.bash
Repeat with bash
network_update_monte_carlo.py
parallel_clusters.bash
aggregate results from parallelized clusters
combine_clusters.py
sim.txt
Output: states populations over time
sim.log
Output: parameters and runtime
full_graph.txt



```
TIETYPE      SEX1    SEX2  AGE1  AGE2
 sexual    female  female    27    37
 social    female  female    25    27
 social    female  female    26    28
 social    female  female    26    20
 social    female  female    25    29
```

Project Repository



huilisabrina / covid-19-simul Private

Efficient simulation and graphical modeling of Covid-19 spread

41 commits | 3 branches | 0 packages | 0 releases | 3 contributors

huilisabrina Update README.md — Latest commit 118a32c 2 hours ago

code	Update README.md	2 hours ago
data	Upload param_input.csv	20 hours ago
figures	Add files via upload	12 hours ago
presentation	Create README.md	17 days ago
README.md	Initial commit	17 days ago

README.md

covid-19-simul

GraphFrames: version 0.6.0
Spark: version 2.3-s 2.11
Python version: 2.7.17
AWS m4.xlarge instances
EMR mode: 1 master + 6 workers

https://github.com/huilisabrina/covid-19-simul