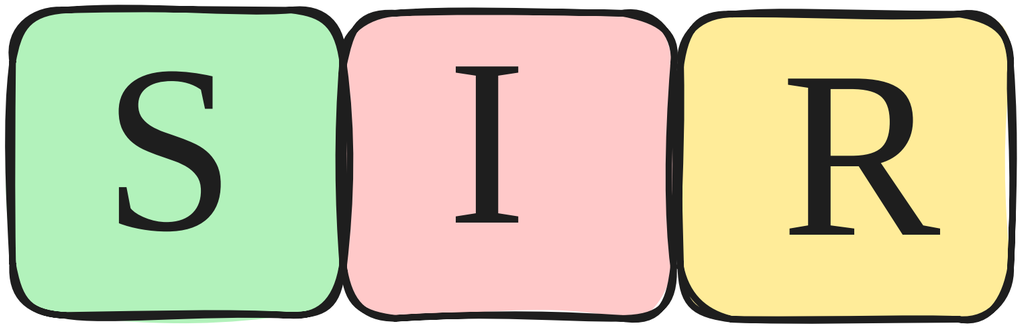
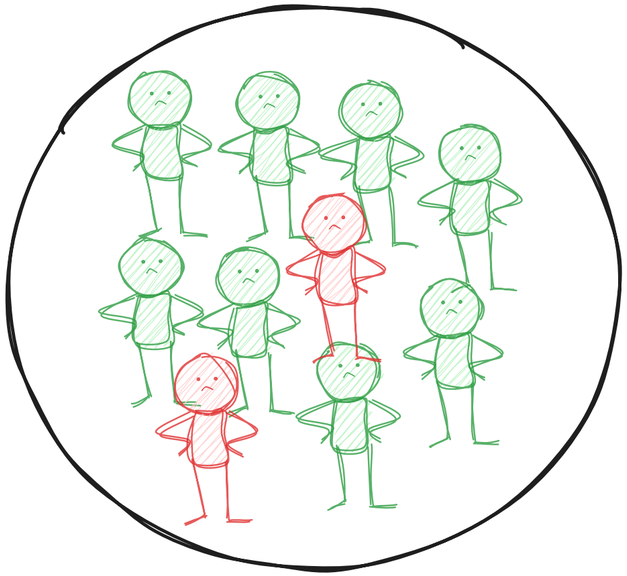
Day 2: Hands on epidemic diffusion

Marco Fernández da Silva

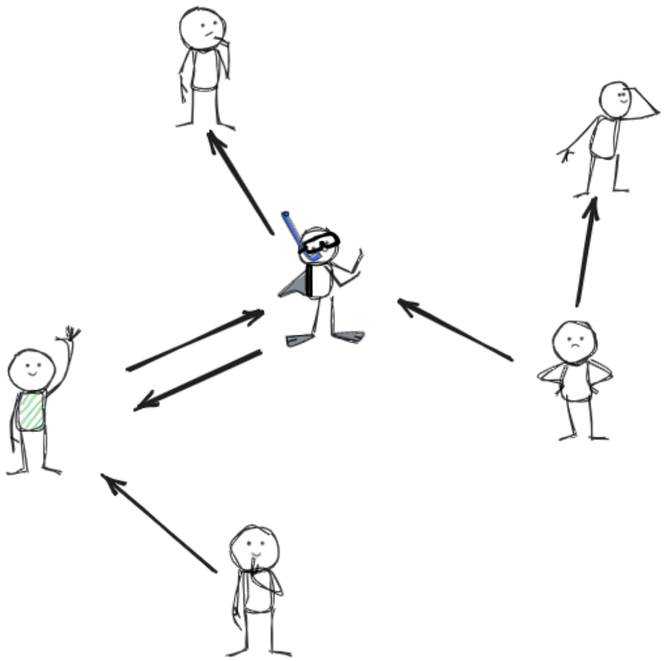


Different simulations

- **Well Mixed**



- **Networks**



Different simulations

- **Well Mixed**
  - Solve differential equations.
  - Monte Carlo Simulations.
  - Gillespie algorithm
- **Networks**
  - Monte Carlo Simulations.
  - Gillespie algorithm.
  - Optimized Gillespie.

Notebook installation

SCAN ME

1. Scan this QR code.
2. Github user: **Markfds01** repo **shenzhen_school.**
3. Clone or download the git repository.
4. Open the folder and install the packages:
    a. Use pip install -r requirements.txt.
    b. If does not work, install via pip install pandas, networkx, random and matplotlib.
5. Try to run the first cell in the jupyter notebook.

Solving ODEs

- **The differential equations of the SIR model are:**

$$\frac{dS}{dt} = -\beta S \frac{I}{N}$$

$$\frac{dI}{dt} = \beta S \frac{I}{N} - \mu I$$

$$\frac{dR}{dt} = \mu I$$

Euler’s method

We can discretize the differential equations → **EULER**

Euler's method

We can discretize the differential equations → **EULER**

URM position differential equation:

$$\frac{dx}{dt} = v(t)$$

It can be discretized as:

$$x(t+h) = x(t) + v(t) \cdot h$$

Euler’s method

Once we know how to apply Euler, we can run our simulations:

1. Apply the **Euler's** method to update the compartments based on the rates.
2. **Update** the simulation time.
3. **Store the data** for the current time step (Time, Susceptible, Infected, Recovered) in a list.
4. Repeat from 1. to 3. until the **stopping condition** is reached.
5. Convert the list of data into a **Pandas DataFrame** for easier analysis and visualization.
6. **Plot** the epidemic curves using `plot_well_mixed_ODES`.

* **NOTE**: The time series simulation data should be stored in a structured format with the columns (Time, Susceptible, Infected, Recovered)

Stochastic simulations

Epidemic diffusion is **not** deterministic.

In order to add stochasticity → **Monte Carlo** simulations

Stochastic simulations

Epidemic diffusion is **not** deterministic.

In order to add stochasticity → **Monte Carlo** simulations

We are going to **generate pseudo-random** numbers in python using the library **random** or **numpy**.

Monte Carlo simulations

Basic idea:

Susteptible + P(S → I) → **Bernouilli test**

Infected + P(I → R) → **Bernouilli test**

Monte Carlo simulations

Basic idea:

Susteptible + P(S → I) → **Bernouilli test**

Infected + P(I → R) → **Bernouilli test**

We **loop** through all the corresponding individuals and wwe throw a number, if it is **lower** than the probability, then the transition **happens**.

Monte Carlo simulations

Basic idea:

Susteptible + P(S → I) → **Bernouilli test**

Infected + P(I → R) → **Bernouilli test**

Instead of looping through all the individuals, one can **sample** the number of new individuals in the compartments from a **binomial**:

new individuals = **Binomial**(possible transitions, probability)

Optional task

As an optional task, we can randomly track the events of infection and recovery (you will need some variables to keep the tracking).

- For every **infection**, randomly sample the **infector** from the set of infected individuals and the infected from the set of susceptible individuals.

- For every **recovery**, randomly sample the **recovered** individual from the set of infected individuals.

- **Store** the data in a list and convert to a pandas Dataframe.

Monte Carlo simulations

1. Compute **infection** and **recovery** probabilities.
2. **Sample** the new number of infected and recovered individuals.
3. **Update** the population
4. **Store** the data
5. **Repeat** steps 1 to 4 until the stopping condition is reached.
6. **Repeat** steps 1 to 5 for all the simulations.

Gillespie algorithm

In the last two simulations, time was **discrete,** no matter the possible number of events, the time step is the **same.**

With Gillespie's method, we can make it **continuous** by sampling the waiting time 't' until the next event from:

$$f(t) = \tau_T e^{-\tau_T t}$$

Where the total rate is the sum of the **infection** and **recovery** rate.

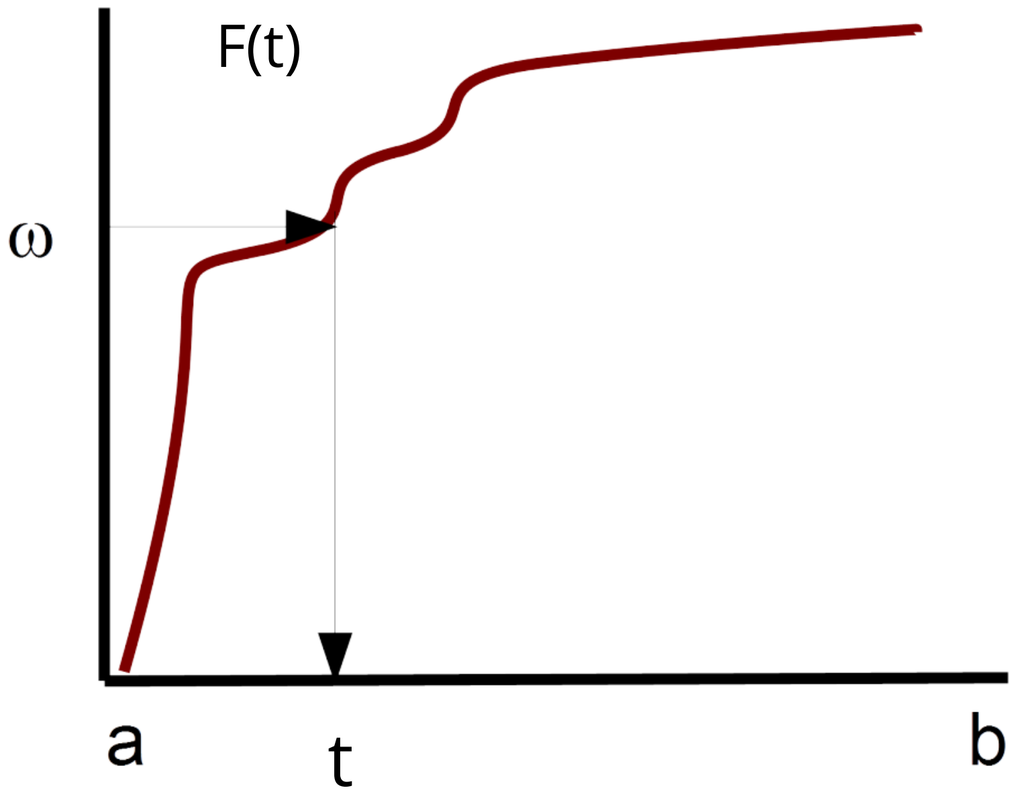
$$\tau_I = \beta S \frac{I}{N} \qquad \tau_R = \mu I$$

Cumulative distribution function

We compute the **cumulative distribution** as:

$$F(t) = \int_{-\infty}^{t} f(t)dt$$

Find the ‘t’ that **satisfies** F(t) = w, with ‘w’ being a random **uniform** number



Gillespie

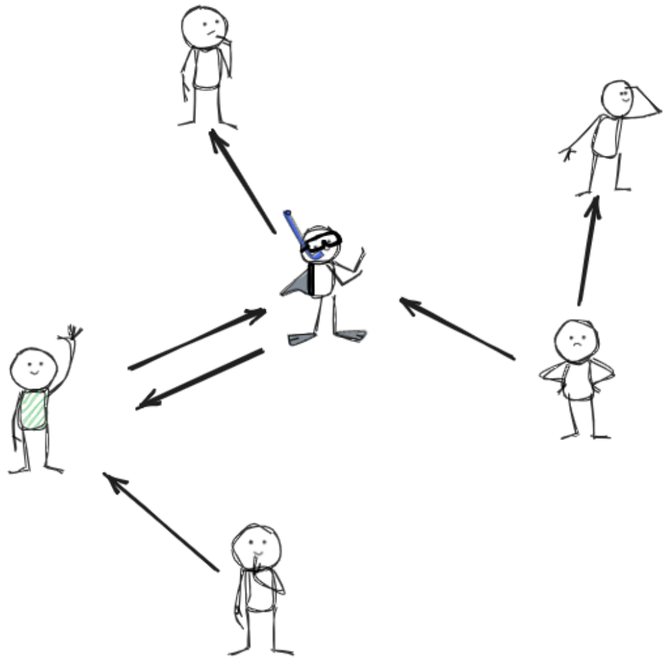
Once we know how to apply Gillespie, we can run our simulations:

1. Calculate the **total rate** of events, infections and recoveries.
2. **Sample** the time step using the **CDF** method.
3. **Choose** the event to execute based on the rates (with a uniform random number).
4. **Update** the sets based on the event executed.
5. **Store** the time series data (t, S(t), I(t), R(t)) for the Gillespie well-mixed model.
6. Repeat steps from **2 to 6** until the stopping condition is reached.
7. Repeat all the steps for more than one simulation.

Stochastic simulations in Networks

Real world **contact structure** is not a well mixed population.

Contact structure → **Networks**



Stochastic simulations in Networks

Real world **contact structure** is not a well mixed population.

Contact structure → **Networks**

Individuals can only interact with the individuals they have a link with.

Stochastic simulations in Networks

Basic idea:

- Loop through all the nodes.
    - If the node is infected:
        - Check if it infects any of its neighbours.
        - Check if it recovers.
- Store the data.
- Repeat until the simulation ends.
- If you finish, repeat the steps with different networks and more than one simulation in each.

Optional task

As an optional task, we can randomly track the events of infection and recovery (you will need some variables to keep the tracking).

- For every **infection**, randomly sample the **infector** from the set of infected individuals and the infected from the set of susceptible individuals.

- For every **recovery**, randomly sample the **recovered** individual from the set of infected individuals.

- **Store** the data in a list and convert to a pandas Dataframe.

Gillespie in Networks

How do we sample the **time step**?

Same as before

Differences:

- The **infection rate** is $\tau_I = \beta \cdot E(I \longrightarrow S)$, the infectivity rate times the **number of edges** between infected and susceptible individuals.
- We **pick** the new infected individual with a **probability proportional** to their **number of infected neighbours**.

Gillespie in Networks

Once we know how to apply Gillespie, we can run our simulations:

1. **Compute the total rate of events**
2. **Sample** the time step using the **CDF** method.
3. **Choose** the event (infection/recovery) based on the rates (with a uniform random number).
4. **Update** the states.
5. **Store** the time series data (t, S(t), I(t), R(t)).
6. Repeat steps from **2 to 5** until the stopping condition is reached.
7. Repeat **all steps** for each simulation.

* **NOTE**: The time series simulation data should be stored in a structured format with the columns (Simulation, Time, Susceptible, Infected, Recovered)

Optimized Gillespie

As Gillespie algorithm is computationally **expensive**, we can apply an **optimization**.

The infection rate is the **sum of the degrees** of all infected nodes. If an infection event occurs, choose the **infector** with probability proportional to its **degree**. Then select one of that node's neighbours uniformly at random and infect it **only if** it is susceptible.

Optimized Gillespie

Once we know how to apply Gillespie, we can run our simulations:

1. **Compute the total rate of events**
2. **Sample** the time step using the **CDF** method.
3. **Choose** the event (infection/recovery) based on the rates (with a uniform random number).
4. **Update** the states.
5. **Store** the time series data (t, S(t), I(t), R(t)).
6. Repeat steps from **2 to 5** until the stopping condition is reached.
7. Repeat **all steps** for each simulation.

* **NOTE**: The time series simulation data should be stored in a structured format with the columns (Simulation, Time, Susceptible, Infected, Recovered)

Thank you