Agent-Based Model of Covid-19

Lecture 12

Peter Steiglechner

26 November 2020

What we will cover today.

- An Agent-Based Model of the spread of Covid-19 in a small society in order to test local policies
- How to make agents heterogeneous: Drawing from probability distributions
- How to let agents interact with each other: Social Networks for the interaction of agents

What we will cover today.

- An Agent-Based Model of the spread of Covid-19 in a small society in order to test local policies
- How to make agents heterogeneous: Drawing from probability distributions
- How to let agents interact with each other: Social Networks for the interaction of agents

What we will cover today.

- An Agent-Based Model of the spread of Covid-19 in a small society in order to test local policies
- How to make agents heterogeneous: Drawing from probability distributions
- How to let agents interact with each other: Social Networks for the interaction of agents

Coronavirus and Covid-19

- Appearance of a new virus 'Severe Acute Respiratory Syndrome Coronavirus-2 (SARS-CoV-2)' in December 2019 in Wuhan, China.
- The virus causes the disease Covid-19
- Virus spreads mainly airborne from infectious humans to other humans (aerosols, ...)
- Declared pandemic by WHO on 31st January, 2020.
- How dangerous is the Covid-19?
  - In March strongly increased death rates
  - In beginning, large reproductive number, R_0
  - $\rightarrow$ We still don't know that much about Covid-19 and Coronavirus-2

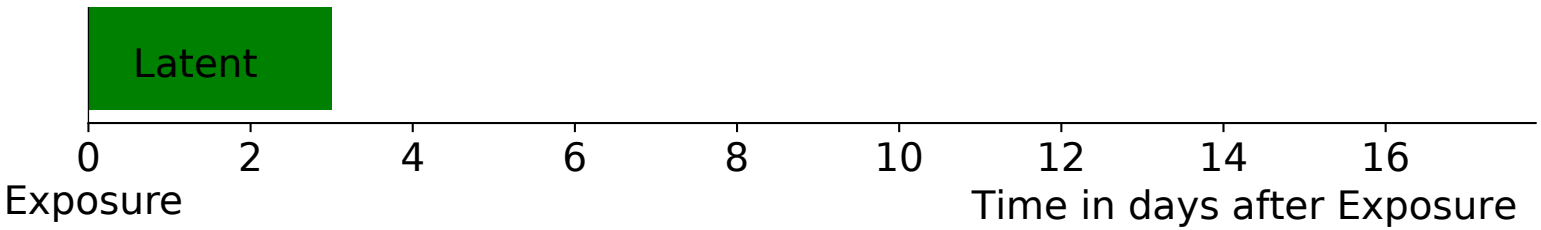
Sources e.g. https://www.ecdc.europa.eu/en/covid-19/facts/questions-answers-basic-facts

Covid-19: Course of an infection

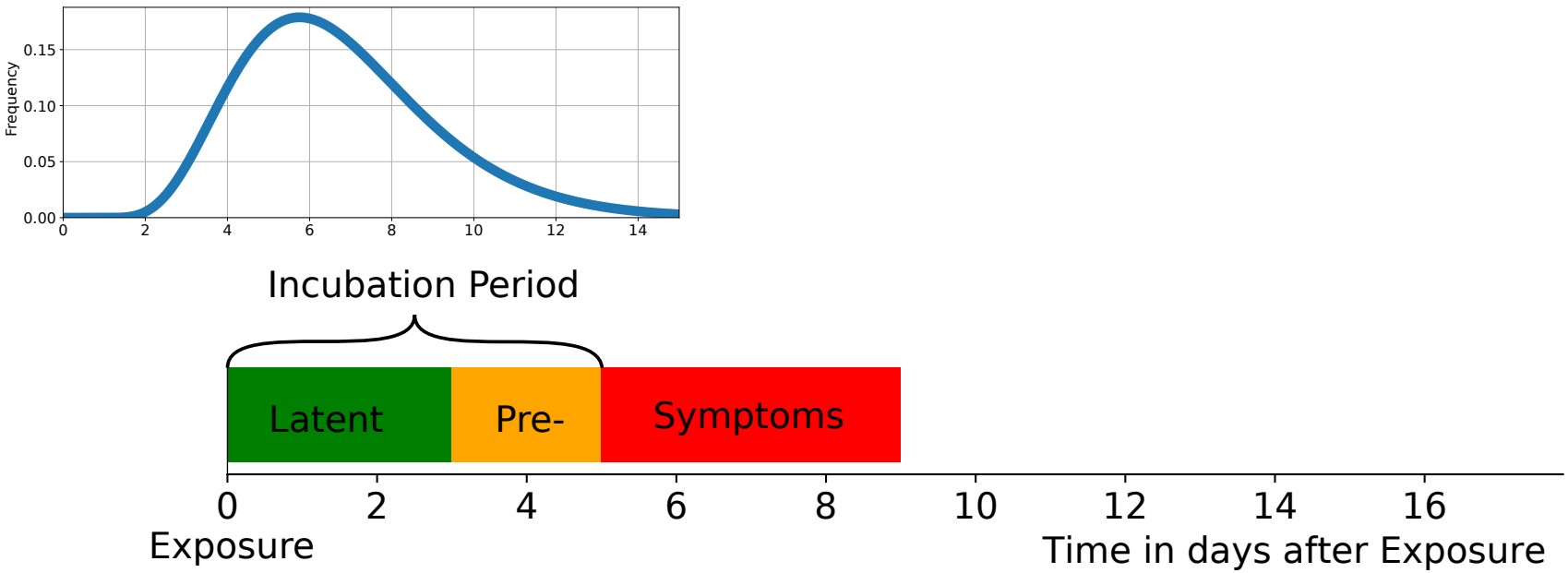
Latent, Pre-Symptomatic, Symptomatic, or Asymptomatic?

- Not all infected people develop symptoms (i.e. asymptomatic infection), but they might still transmit the virus. This is age dependent (e.g. children barely show symptoms).
- After a latent phase, the virus can be transmitted in **every** phase of the infection. In particular, the infection period starts **before** the onset of symptoms (pre-symptomatic).
- Incubation time:
  Delay between exposure ("catching the virus") and symptom onset. Ranging between 1 and 14 days (Mean ca. 5 days.)

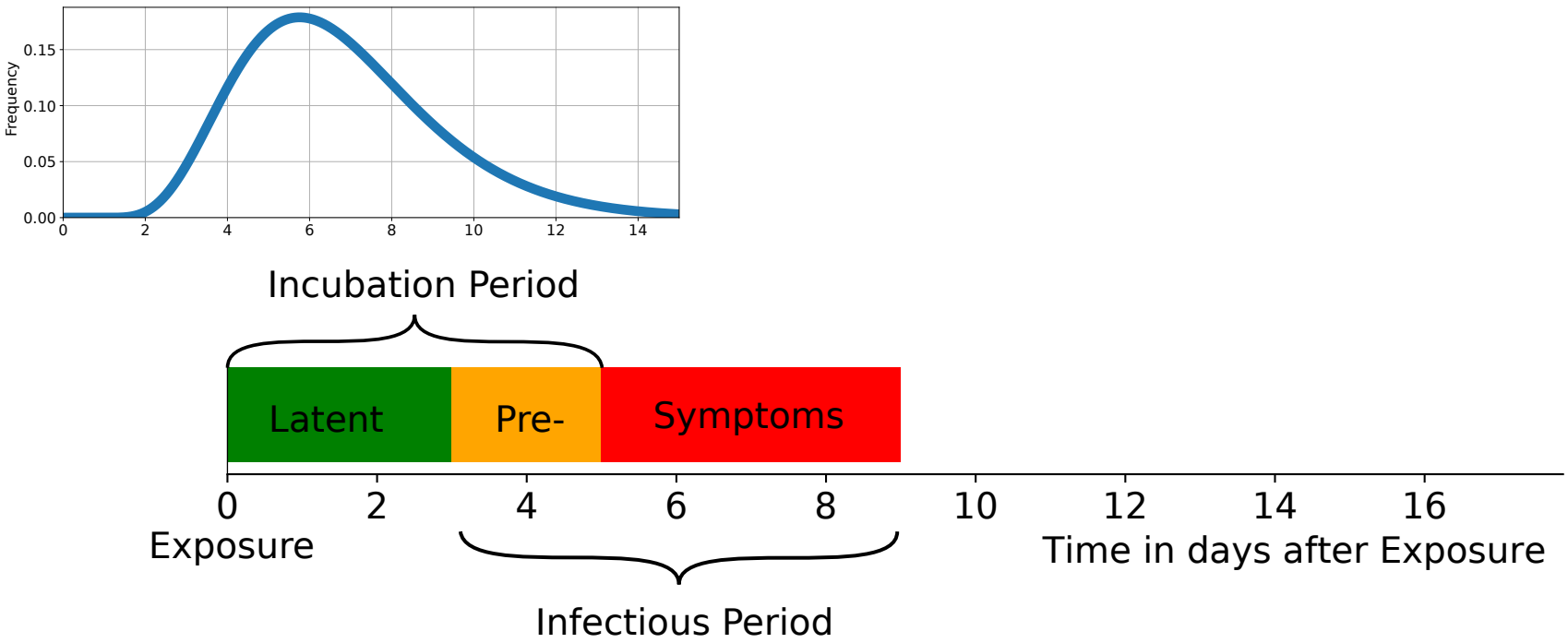
Covid-19: Course of an infection



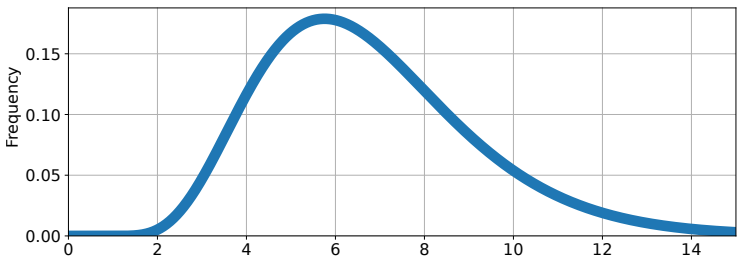
Covid-19: Course of an infection

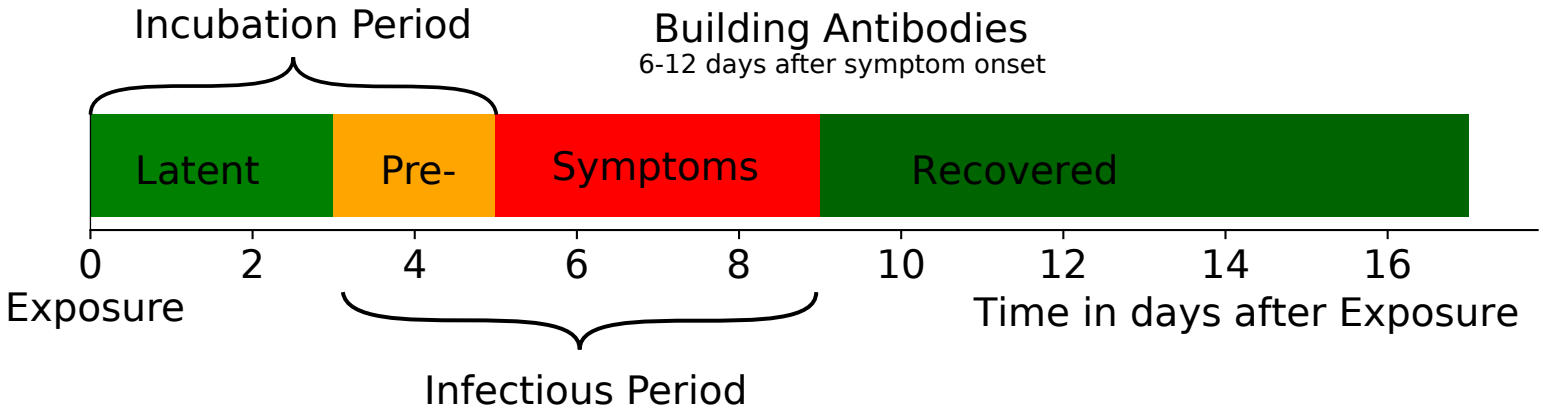


Covid-19: Course of an infection



Covid-19: Course of an infection





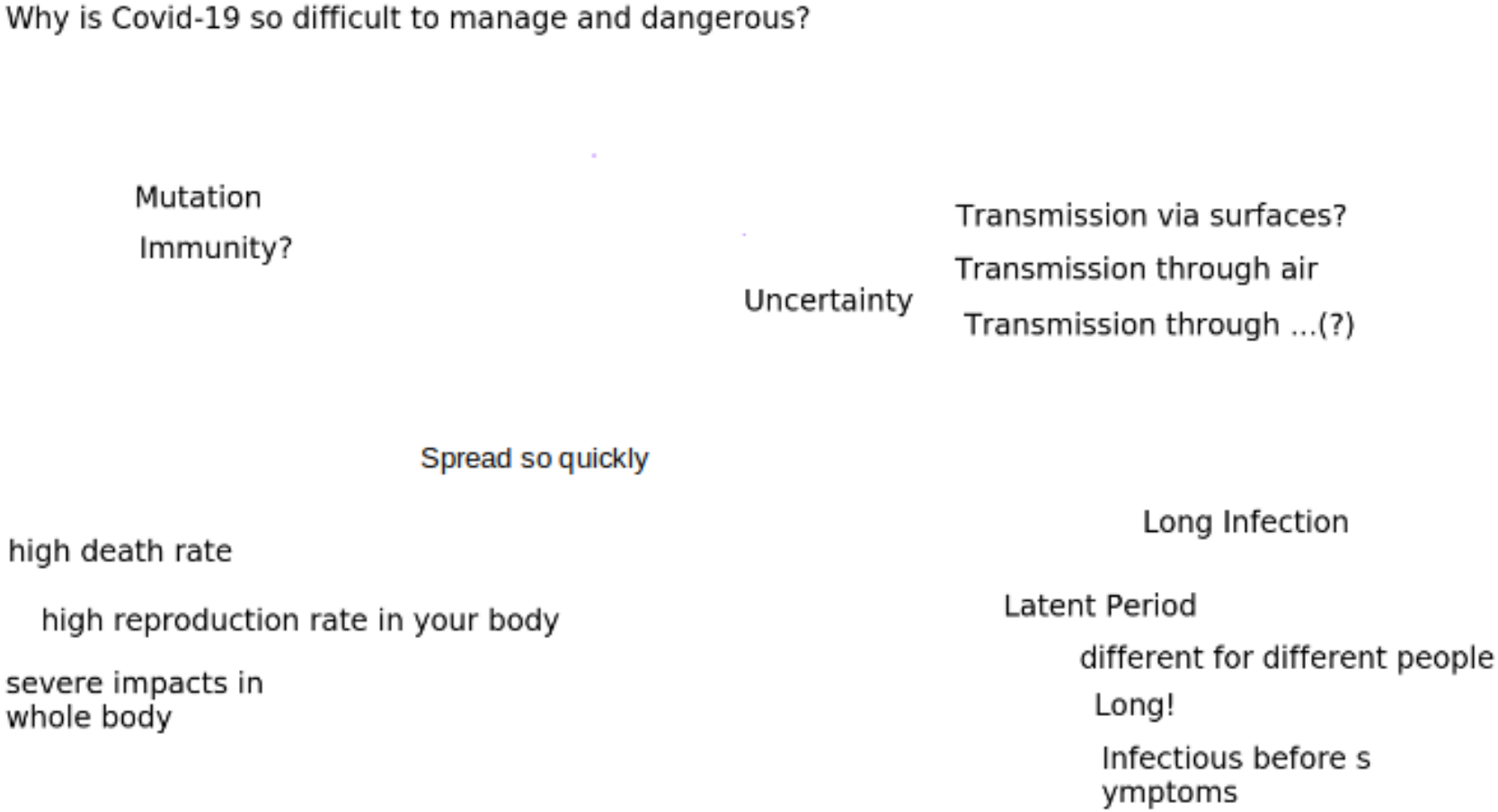
Covid-19: Stats

- Recovery or Death
  The Case Fatality Rate $cfr = \frac{\#\text{ deaths}}{\#\text{ confirmed cases}}$ depends strongly on age.
  Note, this is not "how deadly the virus is" ($\rightarrow$ Infection Fatality Ratio $ifr = \frac{\#\text{ deaths}}{\#\text{ infected cases}}$) but is often used as a proxy.
- R_0-Value
  How many people catch the virus from one infected person (on average) in a **fully susceptible society**. Observed R_0-values ranged between $1.5 - 2.5$. However, it is assumed that infectiousness varies strongly among different people[1].
- Serial Interval
  The time between symptom onsets of consecutive infections (person A infected person B):
  $t_{\text{Symptom, B}} - t_{\text{Symptom, A}} \approx 5.8$ days (He et al., 2020)

(1): Adam, D. (2020) "A Guide to R — the Pandemic's Misunderstood Metric." in *Nature*.

Question

Why is Covid-19 so difficult to manage and dangerous? **Your answers**



Specifics of Covid-19

Why is Covid-19 so difficult to manage and dangerous?

- long delay between start of infectiousness period and symptom onset
- Asymptomatic infection (?)
- High death rate for a part of the society
- Uncertainty!!

E.g. He et al. (2020)

Policy approaches

- Total Lock-down (e.g. Italy. But never perfect. E.g. supermarket, police, households)
- Strict quarantine for positive cases and first contacts (Germany, spring 2020)
- Social distancing
  - social part: encouraged to reduce contacts (i.e. reduce Nr of potential new infected people)
    - Close schools (reduce number of asymptomatic transmissions)
    - ...
  - distancing part: avoid close contacts (i.e. reduce chance of transmitting virus)
- Herd Immunity
- Wear a mask (i.e. reduce chance of transmitting virus to random people)

Policies vary in their epidemiological, social and economic implications

Role of Modeling - SIR

- Initial projections: Based on R_0 estimates/measurements. Became known via 'flatten the curve'.
- More complex, realistic models emerged (e.g. age-structured, which is important for projecting the pressure on the health system)
- Hindsight analysis: Which policy strategy was efficient? E.g. Reiner et al. (2020)
- Assessing impacts of various political decisions: The recent suggestion of 'circuit-breaker' again is fueled by mathematical modelling (Keeling et al., 2020)

Role of Modeling - An ABM shapes national politics

This article is more than **7 months old**

New data, new policy: why UK's coronavirus strategy changed

New quarantine and social distancing 'suppression' measures are based on modelling by Imperial College

- Coronavirus - latest updates
- See all our coronavirus coverage

The Guardian, 16 Mar 2020

In the beginning of the pandemic, the British government changed its control strategy mainly due to research from Imperial College and in particular one agent-based model exploring the effectiveness of policies w.r.t. Covid-19: Ferguson et al. (2020).

Different types of Models

- SIR Models
  - Differential Equations for aggregate variables: Size of the population of susceptible/(exposed)/infected/recovered people
- Agent Based Model:
  - List of discrete entities "agents", with rules specifying their individual 'trajectories' or behaviour (e.g. what they do when infected)

What are advantages/disadvantages of both approaches (in general and related to Covid-19)?

Different types of Models

What are advantages/disadvantages of both approaches (in general and related to Covid-19)? **Your answers**

t

What are advantages/disadvantages of both approaches (in general and related to Covid-19)?

ABM

track people

if you have one infection, you can follow its movement

can't find parameters of SIR easily

hard to find equations in SIR

diverse environment, e.g. different cities

What is the best action that we do.

ODE Models

rough relationship between amount of people

Why is ABM useful in this pandemic

- The Virus and Humans act non-homogeneously:
  - The pandemic is not homogeneously distributed in space
  - Each person has a different health response to catching the virus
  - Will they quarantine? How many social contacts do they have?
- The modeled world is complex and we need to simulate the actual sequence of events.

  If we have two entirely segregated societies, an outbreak of Covid-19 in society A does not impact society B at all.
- Single/Microscopic events can play a crucial role!

  E.g. carnival in Heinsberg (Germany). The pandemic wouldn't exist if "patient 0" in Wuhan had somehow avoided all contacts while infectious
- Uncertainty and random events play a role:

  Can't predict what consequences an action (like a large social event) has.
- The course of the events/decisions changes the 'rules of the game' (non-ergodic system)

  Wuhan citizens react differently to a 2nd wave than New Zealanders.

Concept 0 Summary:

Consider ABM when:

- Microscopic behaviour can cause **emergent** macroscopic phenomena.
  Example: Fish swarm or bird flock
- People, space, or responses (processes) are **non-homogeneous** with potentially non-linear feedbacks. $\rightarrow$ We can't reduce the population to a representative agent (**irreducibility**)
  Il mondo bello per que se vario.
- **Uncertainty** plays a dominant role.
- Context matters (**non-ergodic system**)

If interested, read the very enlightening book 'The end of theory' by Bookstaber (2017)

What we will cover today.

- An Agent-Based Model of the spread of Covid-19 in a small society in order to test local policies
- How to make agents heterogeneous: Drawing from probability distributions
- How to let agents interact with each other: Social Networks for the interaction of agents

Overview ABM

Let's develop an ABM (Slide 14/56 from Lecture 10 on ABM)

1. Specific problem to be solved by the ABM.
2. Design of agents and their static / dynamic attributes.
3. ~~Design of an environment and the way agents interact with it.~~
4. Design of agents' mutual interactions.
5. Design of agents' behaviours.
6. Availability of data.
7. Method of model validation.

Overview ABM

Let's develop an ABM (Slide 14/56 from Lecture 10 on ABM)

1. Specific problem to be solved by the ABM.
   How do a few infected agents affect a small, interconnected, simple society split into three age-/riskgroups? What are the impacts of certain local policies?
2. Design of agents and their static / dynamic attributes.
3. ~~Design of an environment and the way agents interact with it.~~
4. Design of agents' mutual interactions.
5. Design of agents' behaviours.
6. Availability of data.
7. Method of model validation.

Overview ABM

Let's develop an ABM (Slide 14/56 from Lecture 10 on ABM)

1. Specific problem to be solved by the ABM.
   How do a few infected agents affect a small, interconnected, simple society split into three age-/riskgroups? What are the impacts of certain local policies?
2. Design of agents and their static / dynamic attributes.
   Class *Agent()*, function *initialise()*, that creates heterogeneous agents, and helper-function *catch_virus* that creates an agent's infection course (when the agent catches the virus)
3. ~~Design of an environment and the way agents interact with it.~~
4. Design of agents' mutual interactions.

5. Design of agents' behaviours.

6. Availability of data.
7. Method of model validation.

Agent and Initialisation

```
1
2
3 class Agent:
4       pass
```

An empty class.

Agent and Initialisation

```
1  N_AGENTS = 1000  # Number of agents in total
2
3  class Agent:
4        pass
5
6  def initialise():
7        global agents
8        agents = []
9
10       for id in range(N_AGENTS):
11             ag = Agent()
12             ag.id = id                              # Unique ID
13             ag.state = "susceptible"          # health state
14             ag.group = ???                          # Age-/Rsik Group
15             agents.append(ag)
```

Create all agents.

Problem: We want e.g. 20% children, 50% adults, 30% high-risk people in our society.

Concept 1: Drawing from Discrete Distributions

Concept 1: Discrete Distributions: Simple Example

Example

Problem: We want e.g. equal male and female proportions (50% each) in our society. How can we assign a sex to any random agent to obtain such a society?

Concept 1: Discrete Distributions: Simple Example

Example

Problem: We want e.g. equal male and female proportions (50% each) in our society. How can we assign a sex to any random agent to obtain such a society?

✓For each agent randomly draw a sex with probability 50% → Use the function *np.random.choice(options, probability, size)* that chooses between options with given probabilities.

```
for ag in agents:
      ag.sex = np.random.choice(
            ["male", "female"]        # List/Array of possible options
            p=[0.5, 0.5],             # Probabilities assigned to these
            size=1                    # Number of draws (default=1)
      )
```

Agent and Initialisation

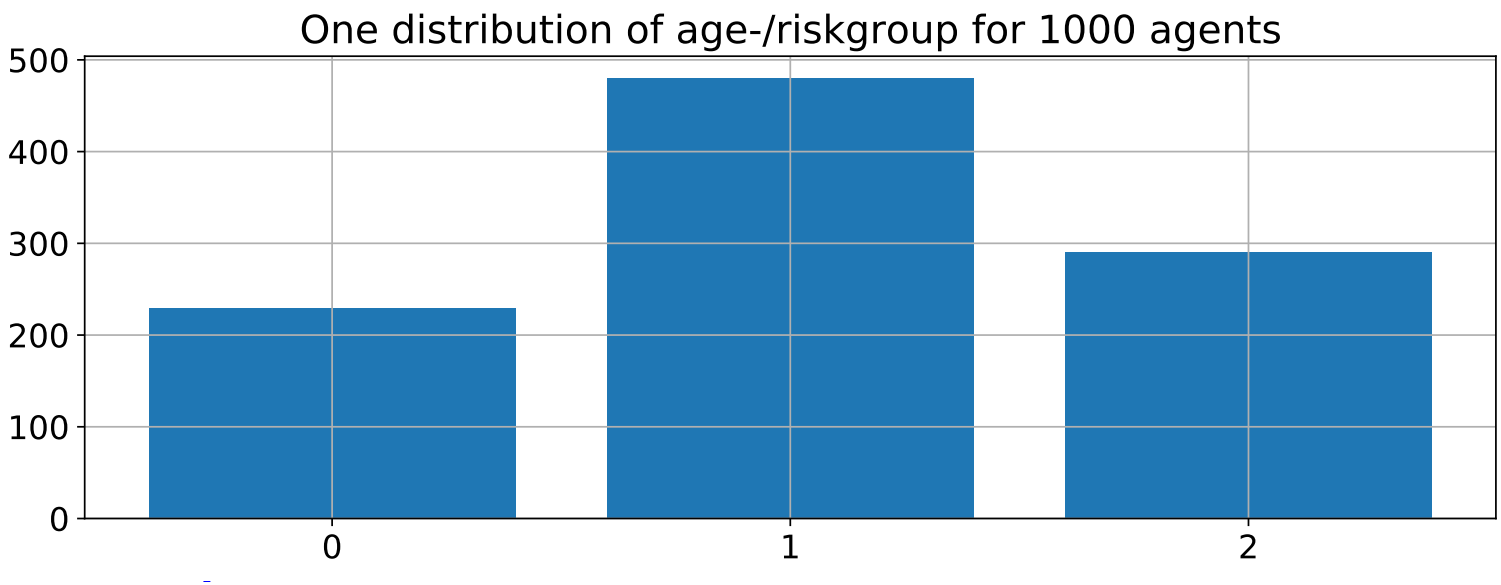
```
1  N_AGENTS = 1000  # Number of agents in total
2
3  #  Group:   0 = children (no risk)          --> 20%
4  #           1 = adults (small risk)         --> 50%
5  #           2 = high-risk group             --> 30%
6  RISKGROUPS = [0, 1, 2]
7  FRAC_RISKGROUPS = [0.2, 0.5, 0.3]
8
9  class Agent:
10       pass
11
12 def initialise():
13       global agents
14       agents = []
15
16       for i in range(N_AGENTS):
17            ag = Agent()
18            ag.id = i                          # Unique ID
19            ag.state = "susceptible"           # health state
20            ag.group = np.random.choice(RISKGROUPS, FRAC_RISKGROUPS)
21            agents.append(ag)
```

Problem: We want e.g. 20% children, 50% adults, 30% high-risk people in our society.

✓Draw from discrete probability distribution for each agent.

Agent and Initialisation

```
1  N_AGENTS = 1000  # Number of agents in total
2
3  #  Group:   0 = children (no risk)        --> 20%
4  #           1 = adults (small risk)       --> 50%
5  #           2 = hig
6  RISKGROUPS = [0, 1,
7  FRAC_RISKGROUPS = [
8
9  class Agent:
10       pass
11
12 def initialise():
13       global agents
14       agents = []
15
16       for i in rang
17             ag = Ag
18             ag.id =
19             ag.stat
20             ag.group = np.random.choice(RISKGROUPS, FRAC_RISKGROUPS)
21             agents.append(ag)
```



Problem: We want e.g. 20% children, 50% adults, 30% high-risk people in our society.

✓Draw from discrete probability distribution for each agent.

Agent and Initialisation

```
1  N_AGENTS = 1000  # Number of agents in total
2  N_INFECTED_INIT = 2  # Number of agents in state "exposed" at t=0.
3
4  class Agent:
5        pass
6
7  def initialise():
8        global agents
9        agents = []
10
11       for id in range(N_AGENTS):
12             ag = Agent()
13             ag.id = id                              # Unique ID
14             ag.state = "susceptible"          # health state
15             ag.group = np.random.choice(RISKGROUPS, FRAC_RISKGROUPS)
16             agents.append(ag)
17
18       # Let N_INFECTED_INIT randomly chosen agents catch the virus at t=0
19       symptomatic_agents = np.random.choice(agents, size=N_INFECTED_INIT)
20       for ag in symptomatic_agents:
21             catch_virus(ag, 0)                  # agent ag catches virus at time 0.
22
23       return
```

Infect a few randomly selected agents. (we'll define function catch_virus in the next slides)

Concept 2: Drawing from Continuous Distributions

Concept 2: Continuous Distributions

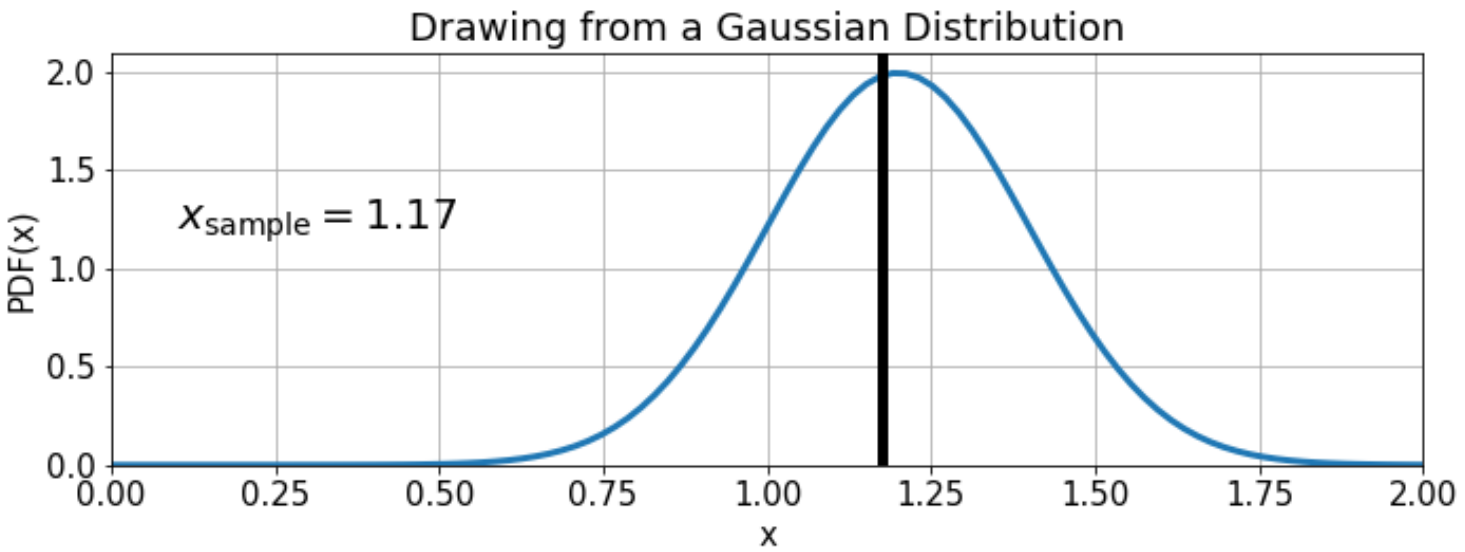
Random Variable

- Random Variable x
- Probability Distribution (or Probability Density Function PDF): $p(x)$ or $\mathrm{dist}_{\mathrm{PDF}}(x)$
- Needs to integrate to 1: $\int_{-\infty}^{\infty} p(x)\, dx = 1$
- Now we simply sample from this distribution!

Concept 2: Continuous Distributions

Random Variable

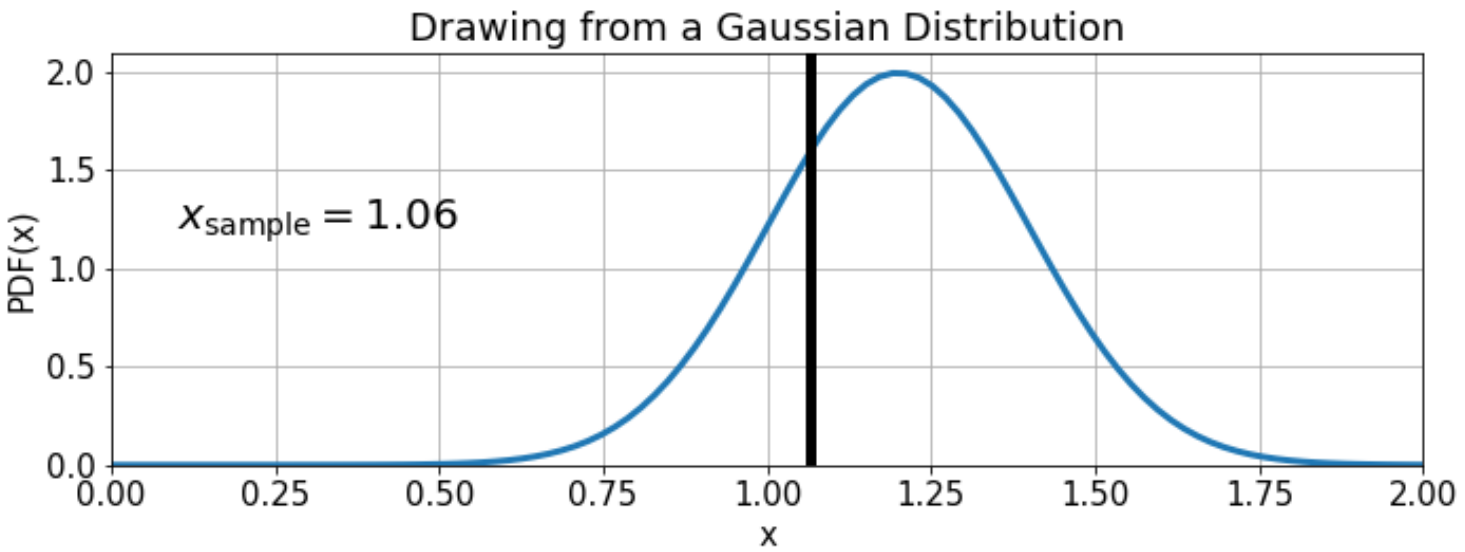
- Random Variable x
- Probability Distribution (or Probability Density Function PDF): $p(x)$ or $\mathrm{dist}_{\mathrm{PDF}}(x)$
- Needs to integrate to 1: $\int_{-\infty}^{\infty} p(x)\, dx = 1$
- Now we simply sample from this distribution!



Concept 2: Continuous Distributions

Random Variable

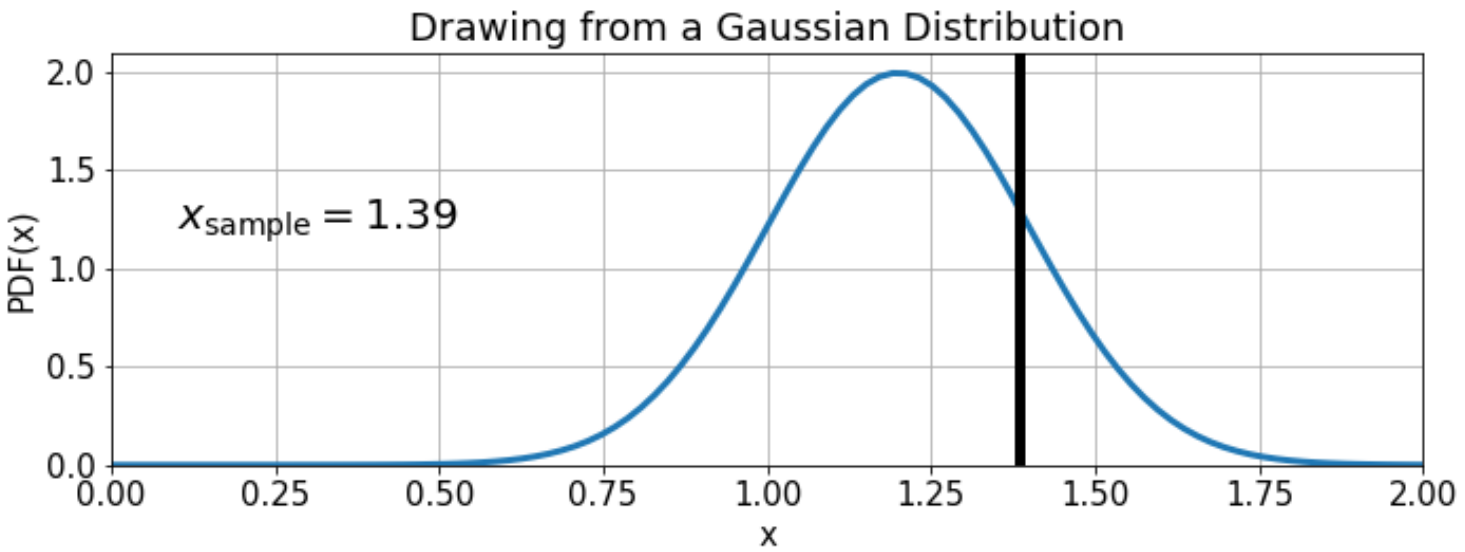
- Random Variable x
- Probability Distribution (or Probability Density Function PDF): $p(x)$ or $\mathrm{dist}_{\mathrm{PDF}}(x)$
- Needs to integrate to 1: $\int_{-\infty}^{\infty} p(x)\, dx = 1$
- Now we simply sample from this distribution!



Concept 2: Continuous Distributions

Random Variable

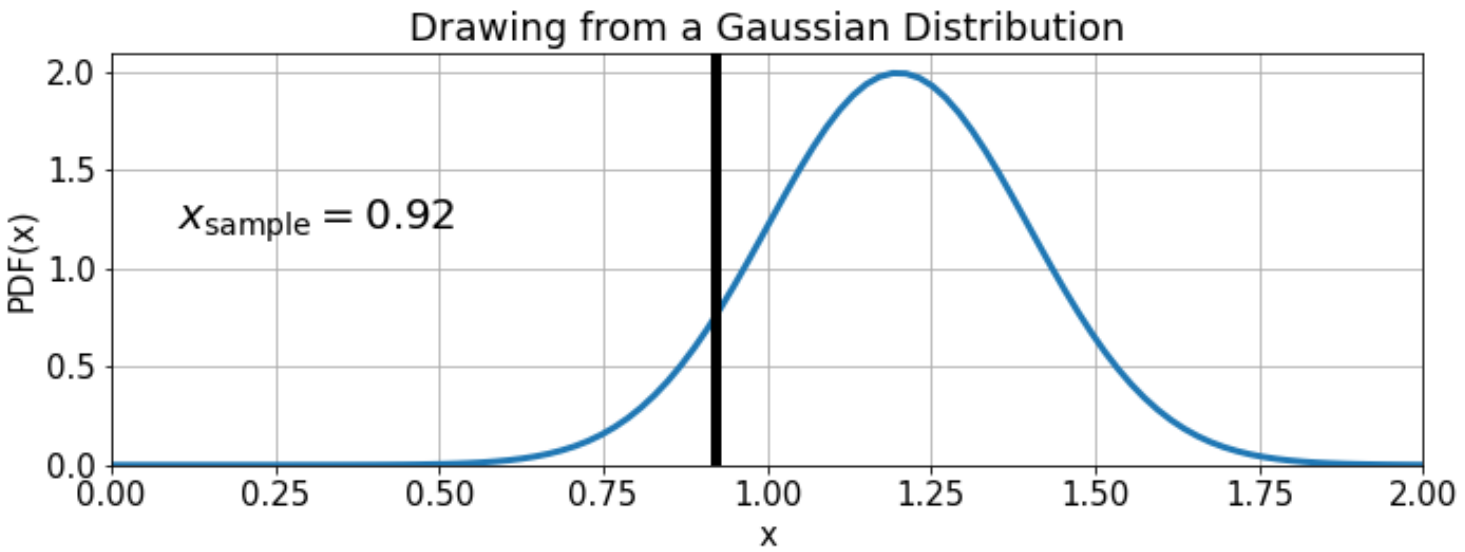
- Random Variable x
- Probability Distribution (or Probability Density Function PDF): $p(x)$ or $\mathrm{dist}_{\mathrm{PDF}}(x)$
- Needs to integrate to 1: $\int_{-\infty}^{\infty} p(x)\, dx = 1$
- Now we simply sample from this distribution!



Concept 2: Continuous Distributions

Random Variable

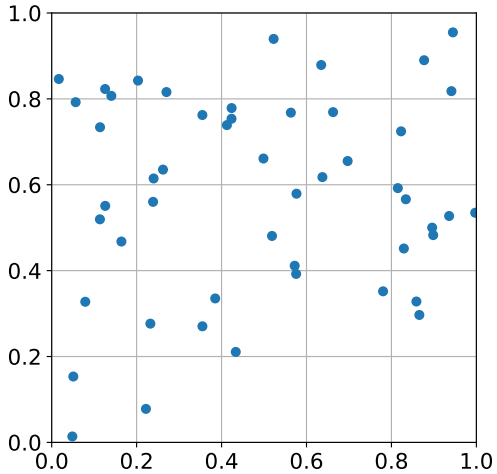
- Random Variable x
- Probability Distribution (or Probability Density Function PDF): $p(x)$ or $\mathrm{dist}_{\mathrm{PDF}}(x)$
- Needs to integrate to 1: $\int_{-\infty}^{\infty} p(x)\, dx = 1$
- Now we simply sample from this distribution!



Concept 2: Repetition

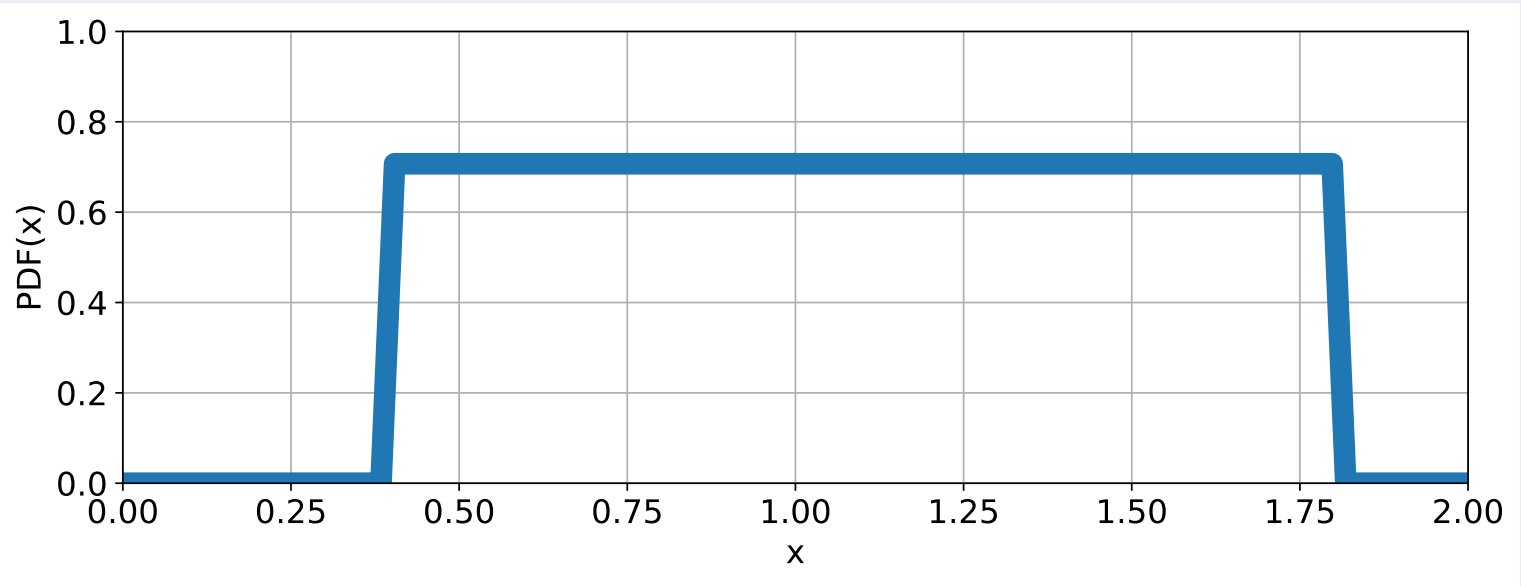
This is exactly what we have done for both previous ABMs in Lecture 10 and 11:

Agents (e.g. foxes and rabbits) were randomly spawned on a 2D space (x, y) with $x, y \in [0, 1]$. This is called the *Uniform Distribution*. Each allowed value for the random variables x and y (i.e. between 0 and 1) is equally likely.



Concept 2: Common cont. distributions

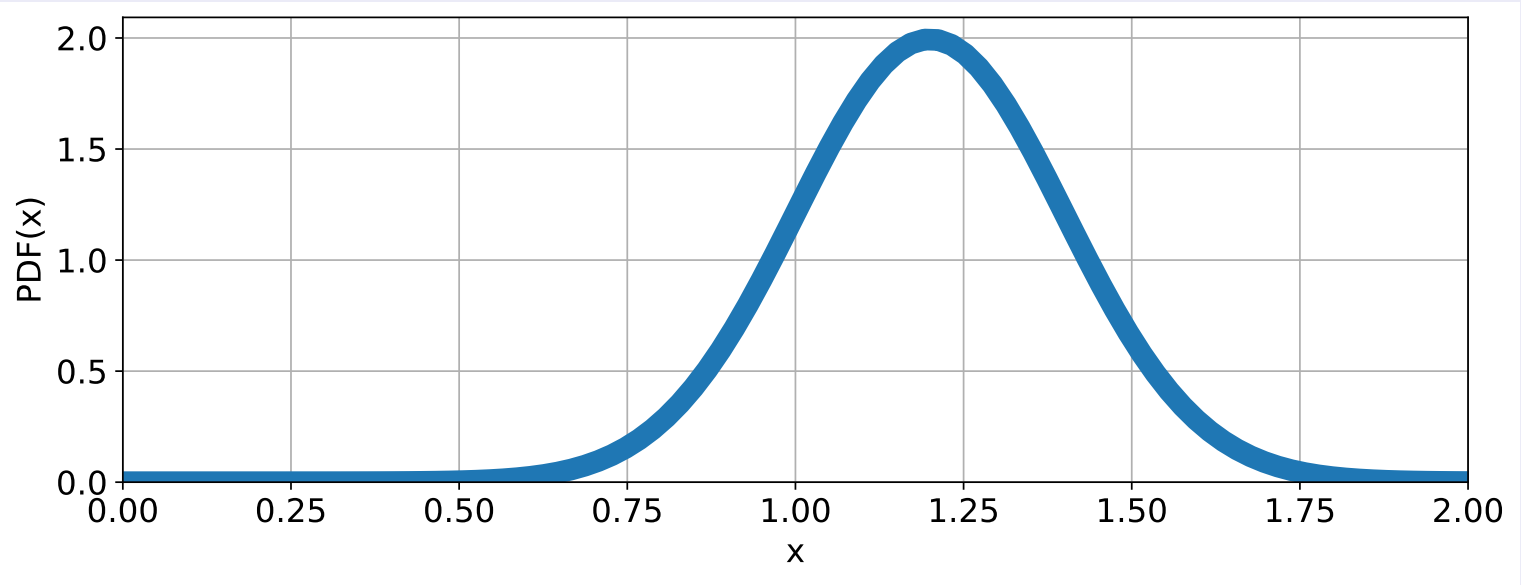
Uniform Distribution



What	continuous, bounded range (maximum and minimum is known)	
PDF	$\mathcal{U}(x_{\min}, x_{\max}) = \frac{1}{x_{\max} - x_{\min}}$	
Usage	Uninformative, when nothing is known about the random variable	

Concept 2: Common cont. distributions

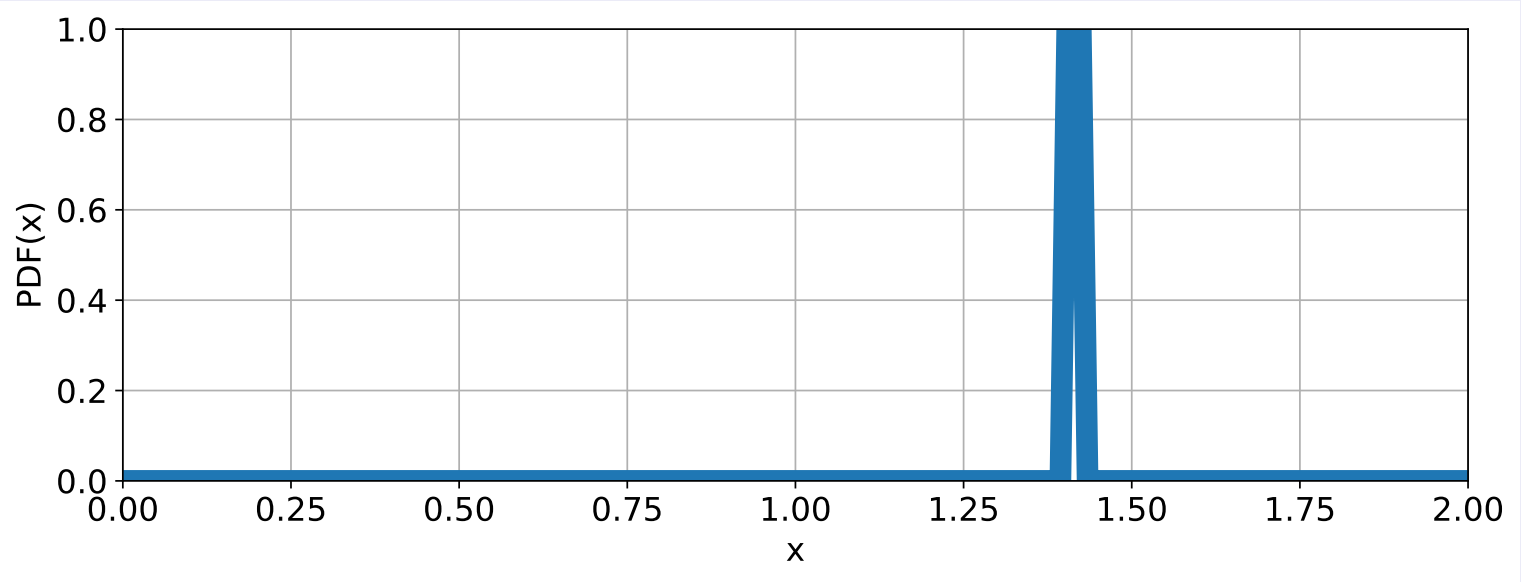
Gaussian/Normal Distribution



What	continuous, infinite range	
PDF	$\mathcal{N}(\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$	$\rightarrow$ very easy to use analytically
Usage	Most often observed in nature $\rightarrow$ Law of large numbers.	

Concept 2: Common cont. distributions

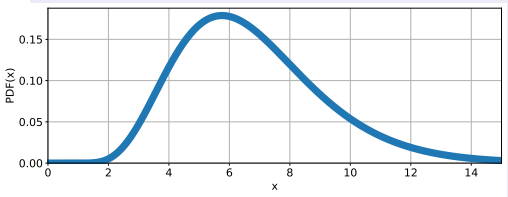
Delta Distribution



What	continuous, inifinte/bounded range	
PDF	$\delta(\tilde{x}) = \delta_{\tilde{x}} = \begin{cases} \infty \text{ if } x = \tilde{x} \\ 0 \text{ else} \end{cases}$	(integral is 1 per def.)
Usage	We know the value for sure! We use this whenever we set a parameter to a fixed value without uncertainty.	

Concept 2: Common cont. distributions

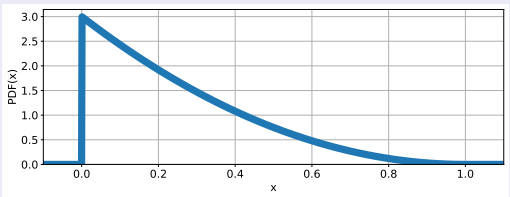
Gamma Distribution



$\Gamma(5.9807,\ 0.948)$

- x semi-bounded $[0, \infty]$
- often used as distribution "close to Gaussian with long tail" (e.g. salary of people)

Beta Distribution



$\mathrm{Beta}(1, 3)$

- x bounded between $[0, 1]$

$\rightarrow$ There're soo many distributions. Wikipedia is informative and well structured

https://en.wikipedia.org/wiki/List_of_probability_distributions

Concept 2: Python Package "scipy.stats"

Scipy Stats

- `import scipy.stats as stats`
- Create distribution e.g. via `stats.norm(mu, sigma)`
-
-

Create distribution:

```
1 import scipy.stats as stats
2
3 mu = 1.2
4 sigma = 0.2
5 some_normal_dist = stats.norm(mu, sigma)
```

- For other distributions simply replace "norm" e.g. with "beta" and look up what parameters you need to specify!!
- (as always in python) https://docs.scipy.org/doc/scipy/reference/stats.html (incl examples and explanations of the parameters to specify, ...)

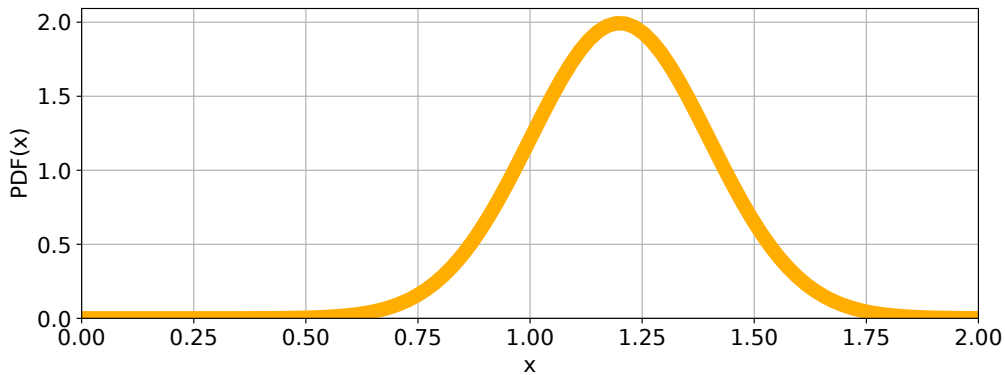
Concept 2: Python Package "scipy.stats"

Scipy Stats

- import scipy.stats as stats
- Create distribution e.g. via stats.norm(mu, sigma)
- Probability Density Function via .pdf(x)
-

Get the PDF:

```
1 x = np.linspace(-2,2)
2 plt.plot(x, some_normal_dist.pdf(x))
3 plt.title('PDF of a normal distribution with mu='+str(mu)+', sigma='+str(sigma))
4 plt.show()
```



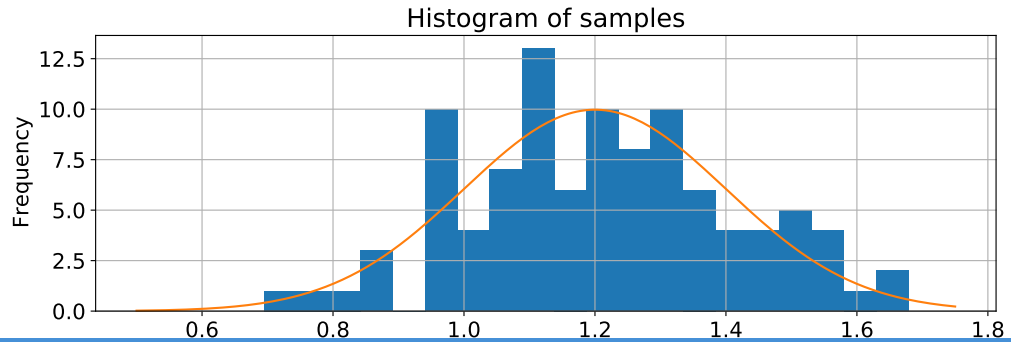
Concept 2: Python Package "scipy.stats"

Scipy Stats

- import scipy.stats as stats
- Create distribution e.g. via stats.norm(mu, sigma)
- PDF via .pdf(x)
- Sampling via .rvs(samplesize)

Draw samples from the distribution:

```
1 samples = some_normal_dist.rvs(100)  # Argument = Nr of samples
2 plt.hist(samples)
3 plt.xlabel("x")
4 plt.ylabel("Frequency")
5 plt.title("Histogram of samples")
6 plt.show()
```



Back to the agent

We have defined some **general properties** for an agent. When the agent is infected, it will need further, **infection-specific properties**:

Back to the agent

We have defined some **general properties** for an agent. When the agent is infected, it will need further, **infection-specific properties**:

1. Time of exposure

Back to the agent

We have defined some **general properties** for an agent. When the agent is infected, it will need further, **infection-specific properties**:

1. Time of exposure
2. Symptomatic or asymptomatic?

Back to the agent

We have defined some **general properties** for an agent. When the agent is infected, it will need further, **infection-specific properties**:

1. Time of exposure
2. Symptomatic or asymptomatic?
3. How long is the incubation period? When do symptoms begin?

Back to the agent

We have defined some **general properties** for an agent. When the agent is infected, it will need further, **infection-specific properties**:

1. Time of exposure
2. Symptomatic or asymptomatic?
3. How long is the incubation period? When do symptoms begin?
4. From when to when is the agent infectious?

Back to the agent

We have defined some **general properties** for an agent. When the agent is infected, it will need further, **infection-specific properties**:

1. Time of exposure
2. Symptomatic or asymptomatic?
3. How long is the incubation period? When do symptoms begin?
4. From when to when is the agent infectious?
5. Will the agent die from the infection?

Back to the agent

We have defined some **general properties** for an agent. When the agent is infected, it will need further, **infection-specific properties**:

1. Time of exposure
2. Symptomatic or asymptomatic?
3. How long is the incubation period? When do symptoms begin?
4. From when to when is the agent infectious?
5. Will the agent die from the infection?
6. How infectious is the agent? Is the agent a superspreader?

Catch the virus – Determine course of infection

```
1
2
3
4 def catch_virus(ag, t_exposure):
5     ag.state = "exposed"
6     ag.t_e = t_exposure
```

1: The agent *ag* catches the virus (*catch_virus*)at time *t=t_exposure*, i.e. after being infected with the virus by another agent. First, the *state* of the agent changes to "exposed" and the time of infection is saved in the internal variable *t_e*.

Catch the virus – Determine course of infection

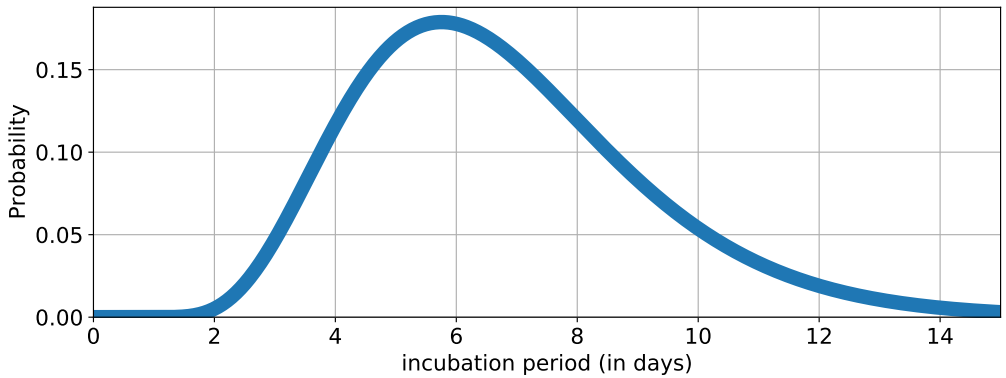
```
1 P_SYMPTOMATIC = [0.1, 0.5, 0.8]
2
3
4 def catch_virus(ag, t_exposure):
5       ...
6     p_s = P_SYMPTOMATIC[ag.group]
7     ag.symptomatic = np.random.choice([True, False], p=[p_s, 1 - p_s])
```

2: We determine randomly whether the course of the infection is going to be symptomatic (*ag.symptomatic = True*) or asymptomatic (*ag.symptomatic = False*) (☞ Discrete Distribution for two options).
The probability depends on the agent's age-/riskgroup.

Catch the virus – Determine course of infection

```python
INCUBATION_PERIOD_DIST = stats.gamma(5.807, 0.948)


def catch_virus(ag, t_exposure):
    ...
    incubation_period = INCUBATION_PERIOD.rvs()
```

3: Draw one incubation period from gamma distribution (fitted to data by Lauer et al. (2020)) (☞ Draw from continuous distr.).



Note: For asymptomatic cases the incubation has no meaning. It's just used as a characteristic value to for the infectiousness period (next slide).

Catch the virus – Determine course of infection

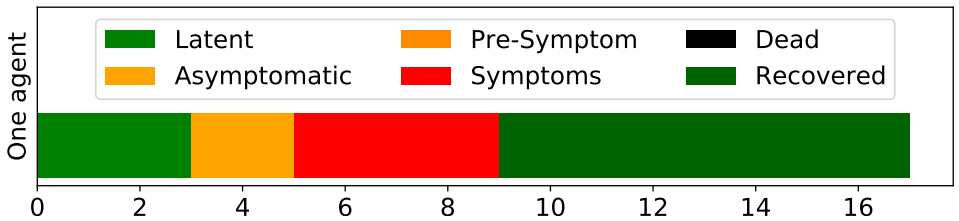
```
1
2
3
4  def catch_virus(ag, t_exposure):
5          ...
6          if ag.symptomatic:
7                  # Symptomatic Case
8                  ag.t_onset_symptoms = ag.t_e + incubation_period
9          else:
10                 # Asymptomatic Case
11                 ag.t_onset_symptoms = np.nan
```

3: Symptom onset after incubation period (if *ag*'s infection will be symptomatic), otherwise, ignore.

Catch the virus – Determine course of infection

```
1 TIME_I_PRESYMPT = 2
2 TIME_I_POSTSYMPT = 4
3
4 def catch_virus(ag, t_exposure):
5       ...
6       ag.infectious_period = [
7         ag.t_e + incubation_period - TIME_I_PRESYMPT,
8         ag.t_e + incubation_period + TIME_I_POSTSYMPT
9     ]
```

4: The agent is only infectious a few days before and after the onset of symptoms.
Here, I assume, that the period in which *asymptomatic* people are infectiousness is similarly distributed as for the symptomatic cases.



Catch the virus – Determine course of infection

```
1  CFR = [0.0001, 0.005, 0.05]    # for each age-/riskgroup
2
3
4  def catch_virus(ag, t_exposure):
5        ...
6        if ag.symptomatic:
7              # Symptomatic Case, might die
8              p_d = CFR[ag.group]
9              ag.fatal_case = np.random.choice([True, False], p=[p_d, 1-p_d])
10       else:
11             # Asymptomatic Case, can not die
12             ag.fatal_case = False
```

5: If the agent is symptomatic, there's a chance the agent might die ($ag.fatal\_case = True$).
The probability depends strongly on the age-/riskgroup.
Asymptomatic agents do not die in this model.

Catch the virus – Determine course of infection

```
1 BASE_I = stats.beta(1, 3)
2
3
4 def catch_virus(ag, t_exposure):
5       ...
6       ag.base_infectiousness = BASE_I.rvs()      # factor in probability to infect
        others
```

6: Each agent has a heterogeneous *ag.base_infectiousness* ($\in [0, 1]$), which factors in the probability of infecting another agents when they are in contact and, thus, exposing them.

Drawn from a beta distribution: few agents are highly, but most hardly infectious.

Catch the virus – Determine course of infection

Summary

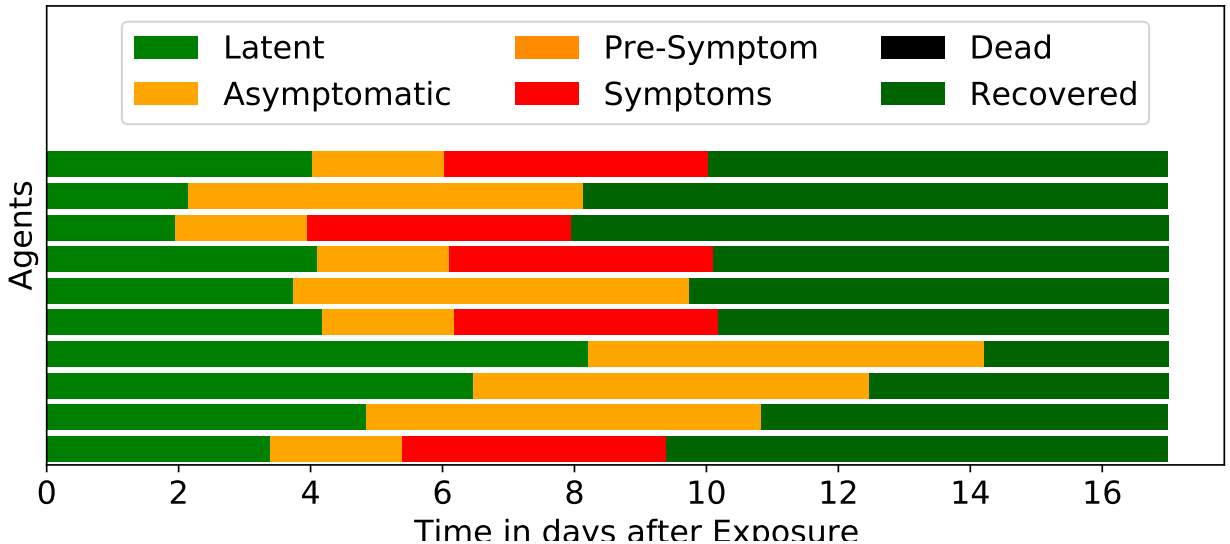
```python
def catch_virus(ag, t_exposure):
    ag.state = "exposed"
    ag.t_e = t_exposure

    p_s = P_SYMPTOMATIC[ag.group]
    ag.symptomatic = np.random.choice([True, False], p=[p_s, 1 - p_s])

    incubation_period = INCUBATION_PERIOD.rvs()

    ag.infectious_period = [
        ag.t_e + incubation_period - TIME_I_PRESYMPT,
        ag.t_e + incubation_period + TIME_I_POSTSYMPT
    ]

    if ag.symptomatic:
        ag.t_onset_symptoms = ag.t_e + incubation_period
        p_d = CFR[ag.group]
        ag.fatal_case = np.random.choice([True, False], p=[p_d, 1 - p_d])
    else:
        ag.t_onset_symptoms = np.nan
        ag.fatal_case = False

    ag.base_infectiousness = BASE_I.rvs()    # * FACTOR_INFECTIOUSNESS

      return
```

A few examples of infection courses

1. Create an agent
2. Let it catch the virus (i.e. call *catch_virus*)
3. Plot the course of the infection.
4. Repeat action 1-3

A few examples of infection courses

1. Create an agent
2. Let it catch the virus (i.e. call *catch_virus*)
3. Plot the course of the infection.
4. Repeat action 1-3



Concept 1 and 2 Summary

Drawing from distributions

- We can create heterogeneous agents / courses of infections /... by drawing new parameters or properties from probability distributions when we initialise the agent/determine the course of the infection/ ...
- For discrete choices:

```
for ag in range(N_AGENTS):
        ag = Agent()
        ag.property1 = np.random.choice(all_choices, p = probs_for_choices)
```

- For continuous random variables (here normally distributed):

```
import scipy.stats as stats
for ag in range(N_AGENTS):
        ag = Agent()
        ag.property2 = stats.norm(mu, sigma).rvs()
        # distributed according to stats.norm(mu, sigma).pdf(x)
```

Repetition: When do we want to exploit this?

Overview ABM

Let's develop an ABM (Slide 14/56 from Lecture 10 on ABM)

1. Specific problem to be solved by the ABM.
   How do a few infected agents affect a small, interconnected, simple society split into three age-/riskgroups? What are the impacts of certain local policies?
2. Design of agents and their static / dynamic attributes.
   Class *Agent()*, function *initialise()*, that creates heterogeneous agents, and helper-function *catch_virus* that creates an agent's infection course (when the agent catches the virus)
3. ~~Design of an environment and the way agents interact with it.~~
4. Design of agents' mutual interactions.
   Via social network of agents
5. Design of agents' behaviours.

6. Availability of data.
7. Method of model validation.

What we will cover today.

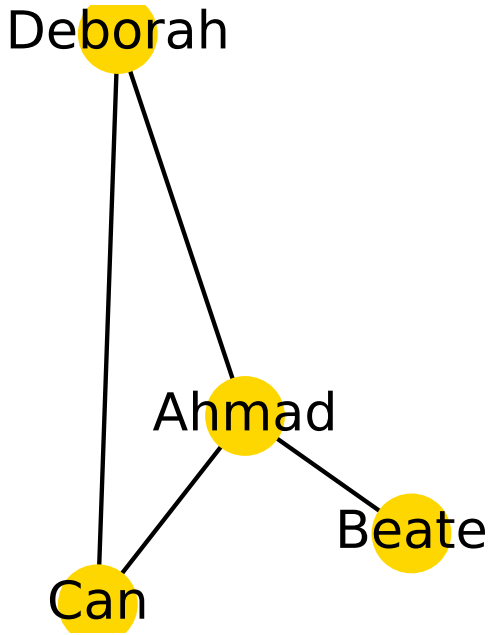
- An Agent-Based Model of the spread of Covid-19 in a small society in order to test local policies
- How to make agents heterogeneous: Drawing from probability distributions
- How to let agents interact with each other: Social Networks for the interaction of agents

Concept 3: Networks

Social Network – Basics

With which agents do agents interact? I.e. who will be possibly infected by a infectious agent? → Social Network

- Nodes: Each node represents one agent
- Link/Edge between nodes: Agents are in 'physical contact'



Social Network – Basics

With which agents do agents interact? I.e. who will be possibly infected by a infectious agent? $\rightarrow$ Social Network

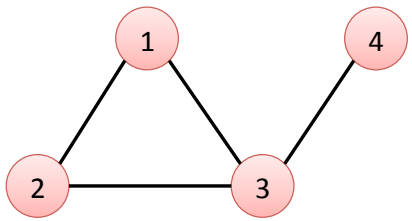
- Nodes: Each node represents one agent
- Link/Edge between nodes: Agents are in 'physical contact'
- (averageg) Node degree = (avg) Nr of links from the agent

Social Network – Basics

With which agents do agents interact? I.e. who will be possibly infected by a infectious agent? $\rightarrow$ Social Network

- Nodes: Each node represents one agent
- Link/Edge between nodes: Agents are in 'physical contact'
- (averageg) Node degree = (avg) Nr of links from the agent
- Adjacency Matrix and List: Alternative network representation. Entry in matrix is 1 = "there is a link between the nodes with index corresponding to row and column of the entry".

Network



Adjacency matrix

i \ j	1	2	3	4
1	0	1	1	0
2	1	0	1	0
3	1	1	0	1
4	0	0	1	0

Adjacency list

i	
1	$\rightarrow$ {2, 3}
2	$\rightarrow$ {1, 3}
3	$\rightarrow$ {1, 2, 4}
4	$\rightarrow$ {3}

Social Network - Topology

- We will use one particular network: A 'Watts-Strogatz network', often also referred to as 'small-world network'.

D. J. Watts & S. H. Strogatz, Collective dynamics of 'small-world' networks, *Nature*, 393:440–442, 1998.

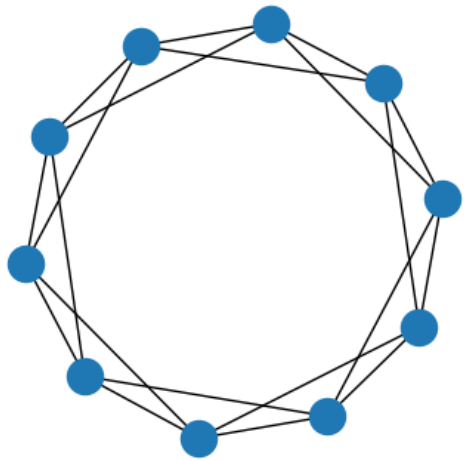
Social Network - Topology

- We will use one particular network: A 'Watts-Strogatz network', often also referred to as 'small-world network'.
  - All n nodes/agents are aligned in a ring.

D. J. Watts & S. H. Strogatz, Collective dynamics of 'small-world' networks, *Nature*, 393:440–442, 1998.

Social Network - Topology

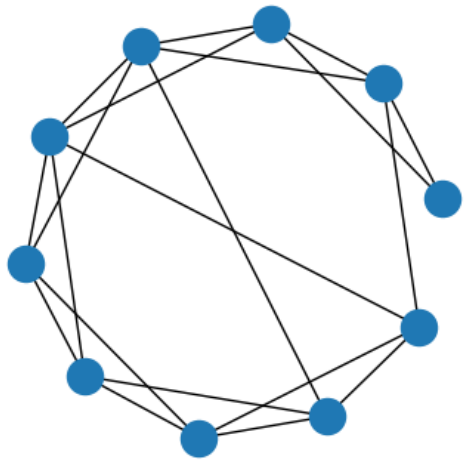
- We will use one particular network: A 'Watts-Strogatz network', often also referred to as 'small-world network'.
  - All n nodes/agents are aligned in a ring.
  - They are connected to their k nearest neighbours (left/right)



D. J. Watts & S. H. Strogatz, Collective dynamics of 'small-world' networks, *Nature*, 393:440–442, 1998.

Social Network - Topology

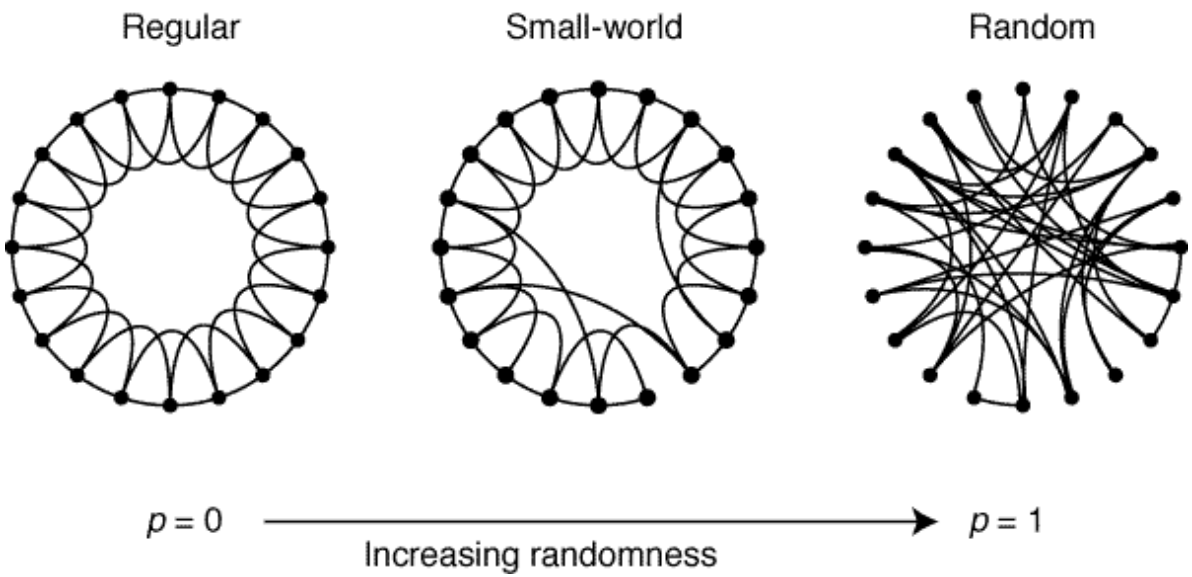
- We will use one particular network: A 'Watts-Strogatz network', often also referred to as 'small-world network'.
  - All n nodes/agents are aligned in a ring.
  - They are connected to their k nearest neighbours (left/right)
  - With probability p, each link is capped and re-drawn to a random node/agent anywhere on the ring.



D. J. Watts & S. H. Strogatz, Collective dynamics of 'small-world' networks, *Nature*, 393:440–442, 1998.

Social Network - Topology

- We will use one particular network: A 'Watts-Strogatz network', often also referred to as 'small-world network'.
  - All n nodes/agents are aligned in a ring.
  - They are connected to their k nearest neighbours (left/right)
  - With probability p, each link is capped and re-drawn to a random node/agent anywhere on the ring.



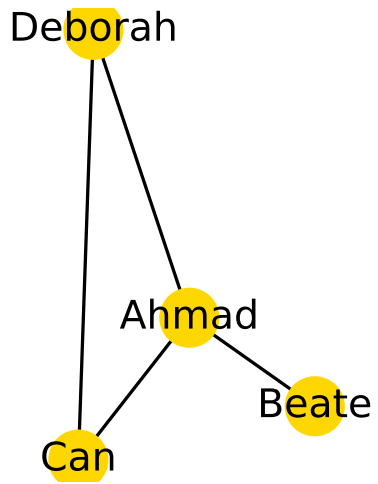
D. J. Watts & S. H. Strogatz, Collective dynamics of 'small-world' networks, *Nature*, 393:440–442, 1998.

Social Network – Perspective

- Network Theory is one of the 'hottest' topics in complexity science (e.g. neural networks)
- The method can be applied on various systems, topics, problems
- Some extensions:
  - Directed and weighted links
  - Topologies of different networks:
    - Scale-free network $\rightarrow$ e.g. Barabási-Albert-Modell
    - Random Graph
  - Clustering, Clustering Coefficient.
  - Co-evolving networks, i.e. that change over time depending on the state of the system.

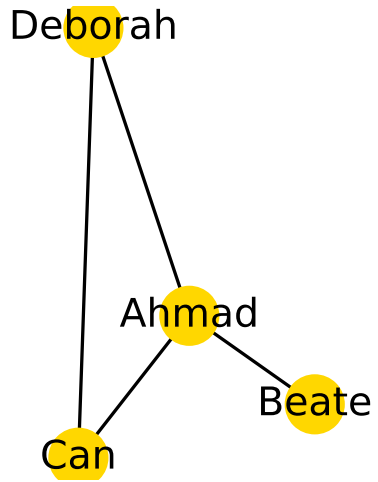
The *Networkx* Package in Python

```
1 import networkx as nx
2 G = nx.Graph()
3 G.add_node("Ahmad")
4 ...
5 G.add_edge("Ahmad", "Can")
6 ...
7 pos = nx.spring_layout(G)      # Just a 'nice' way of arranging the nodes
8 nx.draw(G, pos, with_labels=True)
```



The *Networkx* Package in Python

```python
import networkx as nx
G = nx.Graph()
G.add_node("Ahmad")
...
G.add_edge("Ahmad", "Can")
...
pos = nx.spring_layout(G)      # Just a 'nice' way of arranging the nodes
nx.draw(G, pos, with_labels=True)
```



What's the adjacency Matrix?

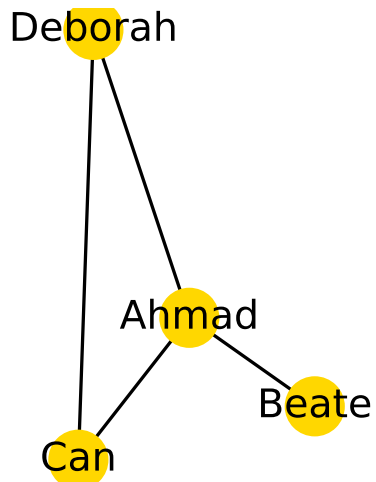
$$(A \quad B \quad C \quad D)$$

$$\downarrow$$

$$\begin{pmatrix} Ahmad\ A \\ Beate\ B \\ Can\ C \\ Deborah\ D \end{pmatrix} \Rightarrow \begin{pmatrix} & & & \\ & & & \\ & & & \\ & & & \end{pmatrix}$$

The *Networkx* Package in Python

```python
import networkx as nx
G = nx.Graph()
G.add_node("Ahmad")
...
G.add_edge("Ahmad", "Can")
...
pos = nx.spring_layout(G)      # Just a 'nice' way of arranging the nodes
nx.draw(G, pos, with_labels=True)
```



What's the adjacency Matrix?

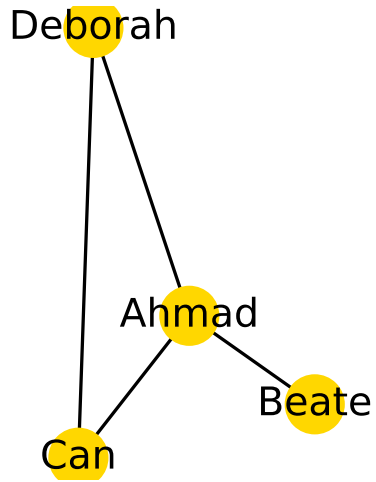
$$(A \quad B \quad C \quad D)$$

$$\downarrow$$

$$\begin{pmatrix} Ahmad\ A \\ Beate\ B \\ Can\ C \\ Deborah\ D \end{pmatrix} \Rightarrow \begin{pmatrix} 0 & 1 & 1 & 1 \\ & & & \\ & & & \\ & & & \end{pmatrix}$$

The *Networkx* Package in Python

```python
import networkx as nx
G = nx.Graph()
G.add_node("Ahmad")
...
G.add_edge("Ahmad", "Can")
...
pos = nx.spring_layout(G)      # Just a 'nice' way of arranging the nodes
nx.draw(G, pos, with_labels=True)
```



What's the adjacency Matrix?

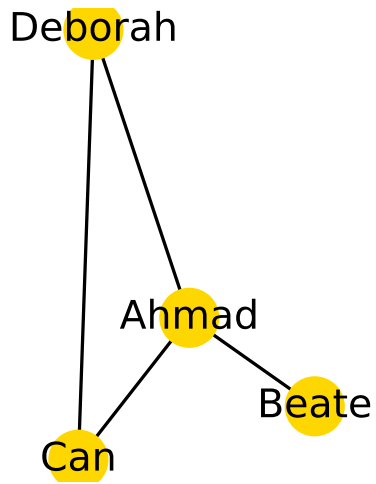
$$(A \quad B \quad C \quad D)$$

$$\downarrow$$

$$\begin{pmatrix} Ahmad\ A \\ Beate\ B \\ Can\ C \\ Deborah\ D \end{pmatrix} \Rightarrow \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & & \\ 1 & 0 & & \end{pmatrix}$$

The *Networkx* Package in Python

```python
import networkx as nx
G = nx.Graph()
G.add_node("Ahmad")
...
G.add_edge("Ahmad", "Can")
...
pos = nx.spring_layout(G)      # Just a 'nice' way of arranging the nodes
nx.draw(G, pos, with_labels=True)
```



What's the adjacency Matrix?

$$(A \quad B \quad C \quad D) \Downarrow$$

$$\begin{pmatrix} Ahmad\ A \\ Beate\ B \\ Can\ C \\ Deborah\ D \end{pmatrix} \Rightarrow \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

The *Networkx* Package in Python

```
import networkx as nx
G = nx.Graph()
G.add_node("Ahmad")
...
G.add_edge("Ahmad", "Can")
...
pos = nx.spring_layout(G)      # Just a 'nice' way of arranging the nodes
nx.draw(G, pos, with_labels=True)
```

Building a Watts-Strogatz Network is even easier:

```
WS = nx.watts_strogatz_graph(
                    N_AGENTS,        # n = How many nodes
                    4,       # k = How many nearest neighbours
                    0.1)     # p = Probability for each link to be rewired

pos = nx.spring_layout(WS)
nx.draw(WS, pos, with_labels=True)

print("The adjacency matrix is: ", nx.adjacency_matrix(WS))
print("The adjacency list for agent 'ag' is: ", WS.adj[ag.id])

```

Concept 3 Summary

Social Network

- Social networks can be used to represent communication/interaction between agents
- Topology of the network matters (in particular w.r.t. clustering and node degree)
- The package networkx is wonderful and simple to use:

```
1 import networkx as nx
2 G = nx.watts_strogatz_graph(100, 4, 0.1)
3 nx.draw(G)
4 print("Agents have contacts to the nodes/agents with these indices: ", G.adj)
```

Overview ABM

Let's develop an ABM (Slide 14/56 from Lecture 10 on ABM)

1. Specific problem to be solved by the ABM.
   How do a few infected agents affect a small, interconnected, simple society split into three age-/riskgroups? What are the impacts of certain local policies?
2. Design of agents and their static / dynamic attributes.
   Class *Agent()*, function *initialise()*, that creates heterogeneous agents, and helper-function *catch_virus* that creates an agent's infection course (when the agent catches the virus)
3. ~~Design of an environment and the way agents interact with it.~~
4. Design of agents' mutual interactions.
   Via social network of agents
5. Design of agents' behaviours.
   Via function *update()* and for single agent *catch_virus*
6. Availability of data.
7. Method of model validation.

Update Function and Run Function

Update Function

- Choose queuing order of agents (*np.random.choice*).

```
1 def update(t_now):
2       queue = np.random.choice(agents, size=N_AGENTS, replace=False)
3       for ag in queue:
```

Update Function

- Choose queuing order of agents (*np.random.choice*).
- For each agent:
  - Check and potentially update state → state-dependent action
  - Potentially infect others in network with certain probability

```
1 def update(t_now):
2       queue = np.random.choice(agents, size=N_AGENTS, replace=False)
3       for ag in queue:
```

Update Function

- Choose queuing order of agents (*np.random.choice*).
- For each agent:
  - Check and potentially update state $\rightarrow$ state-dependent action
  - Potentially infect others in network with certain probability

```
1 def update(t_now):
2       queue = np.random.choice(agents, size=N_AGENTS, replace=False)
3       for ag in queue:
4             if ag.state == "susceptible":
5                   pass  # Do nothing
```

Update Function

- Choose queuing order of agents (*np.random.choice*).
- For each agent:
  - Check and potentially update state → state-dependent action
  - Potentially infect others in network with certain probability

```
1  def update(t_now):
2        agent_list = np.random.choice(agents, size=N_AGENTS, replace=False)
3        for ag in queue:
4              if ag.state == "susceptible":
5                    pass  # Do nothing
6              if ag.state == "exposed":
7                    # Potentially become infectious
8                    if t_now >= ag.infectious_period[0]:
9                          if ag.symptomatic:
10                               ag.state = "inf_presympt"  # pre-symptomatic
11                         else:
12                               ag.state = "inf_asympt" # asymptomatic
```

Switch from latent to infectious *state* (pre-symptom or asymptomatic) when *infectious_period* starts

Update Function

- Choose queuing order of agents (*np.random.choice*).
- For each agent:
  - Check and potentially update state $\rightarrow$ state-dependent action
  - Potentially infect others in network with certain probability

```
1  def update(t_now):
2        queue = np.random.choice(agents, size=N_AGENTS, replace=False)
3        for ag in queue:
4              if ag.state == "susceptible":
5                    ...
6              if ag.state == "exposed":
7                    ...
8              if ag.state == "inf_presympt":
9                    if t_now >= ag.t_onset_symptoms:
10                         ag.state = "inf_sympt"
```

Switch from pre-symptomatic to symptomatic when incubation period is over.

Update Function

- Choose queuing order of agents (*np.random.choice*).
- For each agent:
  - Check and potentially update state → state-dependent action
  - Potentially infect others in network with certain probability

```
1  def update(t_now):
2        queue = np.random.choice(agents, size=N_AGENTS, replace=False)
3        for ag in queue:
4              if ag.state == "susceptible":
5                    ...
6              if ag.state == "exposed":
7                    ...
8              if ag.state == "inf_presympt":
9                    ...
10             if ag.state[0:3] == "inf":  # "inf_asympt", "inf_presympt", "inf_sympt"
11                   if t_now >= ag.infectious_period[1]:
12                         if ag.fatal_case:
13                               ag.state = "dead"
14                         else:
15                               ag.state = "recovered"
```

If agent in *state* = "*inf_...*" and the infectious period is over, then either recover or die.

Update Function

- Choose queuing order of agents (*np.random.choice*).
- For each agent:
  - Check and potentially update state → state-dependent action
  - Potentially infect others in network with certain probability

```
1  I_SYMPT, I_PRESYMPT, I_ASYMPT = 1, 0.5, 0.2
2
3  def update(t_now):
4        queue = np.random.choice(agents, size=N_AGENTS, replace=False)
5        for ag in queue:
6              ...
7              if ag.state[0:3] == "inf":
8                    # Get Infectiousness:
9                    if ag.state == "inf_sympt":
10                         p_i = ag.base_infectiousness * I_SYMPT
11                   elif ag.state == "inf_presympt":
12                         p_i = ag.base_infectiousness * I_PRESYMPT
13                   elif ag.state == "inf_asympt":
14                         p_i = ag.base_infectiousness * I_ASYMPT
15
16                   # Loop through contacts and expose them
17                   linked_contacts = list(Net.adj[ag.id])  # Indices of neighbours
18                   for c in linked_contacts:
19                         contact_person = agents[c]
20                         if np.random.random() < p_i:
21                               if contact_person.state == "susceptible":
22                                     catch_virus(contact_person, t_now)
```

Run and Observe Function

Goal: Perform simulation and track how many agents are in state "susceptible", "exposed", ... over time.

```
initialise()  # Initialise all agents
Net = initialise_network(agents, k=K_WS_NETWORK, p=P_WS_NETWORK)

T_ARRAY = np.linspace(0, 100, 0.5)

states = ["susceptible", "exposed", "inf_asympt", "inf_presympt", "inf_sympt", "
    recovered", "dead"]
results = np.empty([len(T_ARRAY), len(states)])

for n, t in enumerate(T_ARRAY):
      update(t)

      states_of_agents = [ag.state for ag in agents]
      N_s = states_of_agents.count("susceptible")
      N_e = states_of_agents.count("exposed")
      ...
      results[n, :] = np.array([N_s, N_e, N_ia, ...])
```

Run and Observe Function

Goal: Perform simulation and track how many agents are in state "susceptible", "exposed", ... over time.

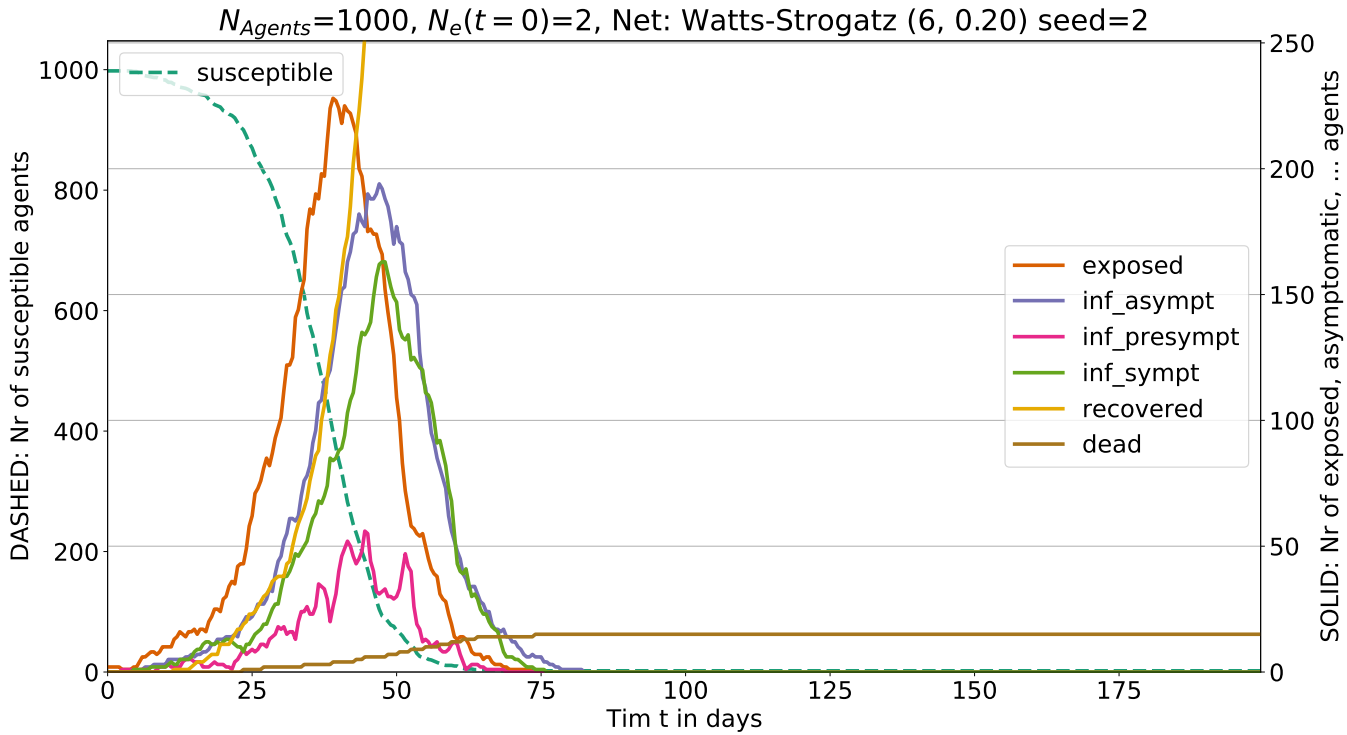


Figure: An outbreak in a network with k=6, p = 0.2, 1000 agents of which 2 are exposed at t=0.

Run and Observe Function

Goal: Perform simulation and track how many agents are in state "susceptible", "exposed", ... over time.

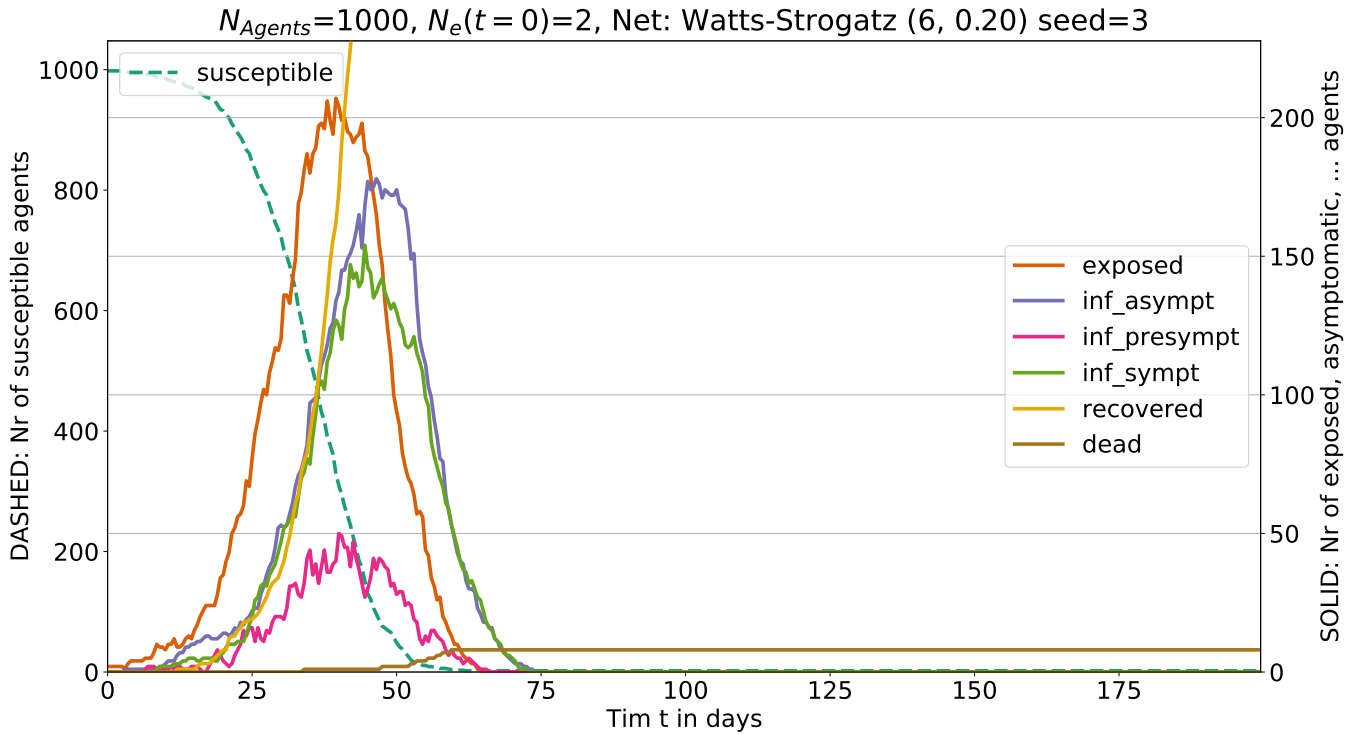


Figure: A *different* outbreak in a network with k=6, p = 0.2, 1000 agents of which 2 are exposed at t=0.

Run and Observe Function

Goal: Perform simulation and track how many agents are in state "susceptible", "exposed", ... over time.

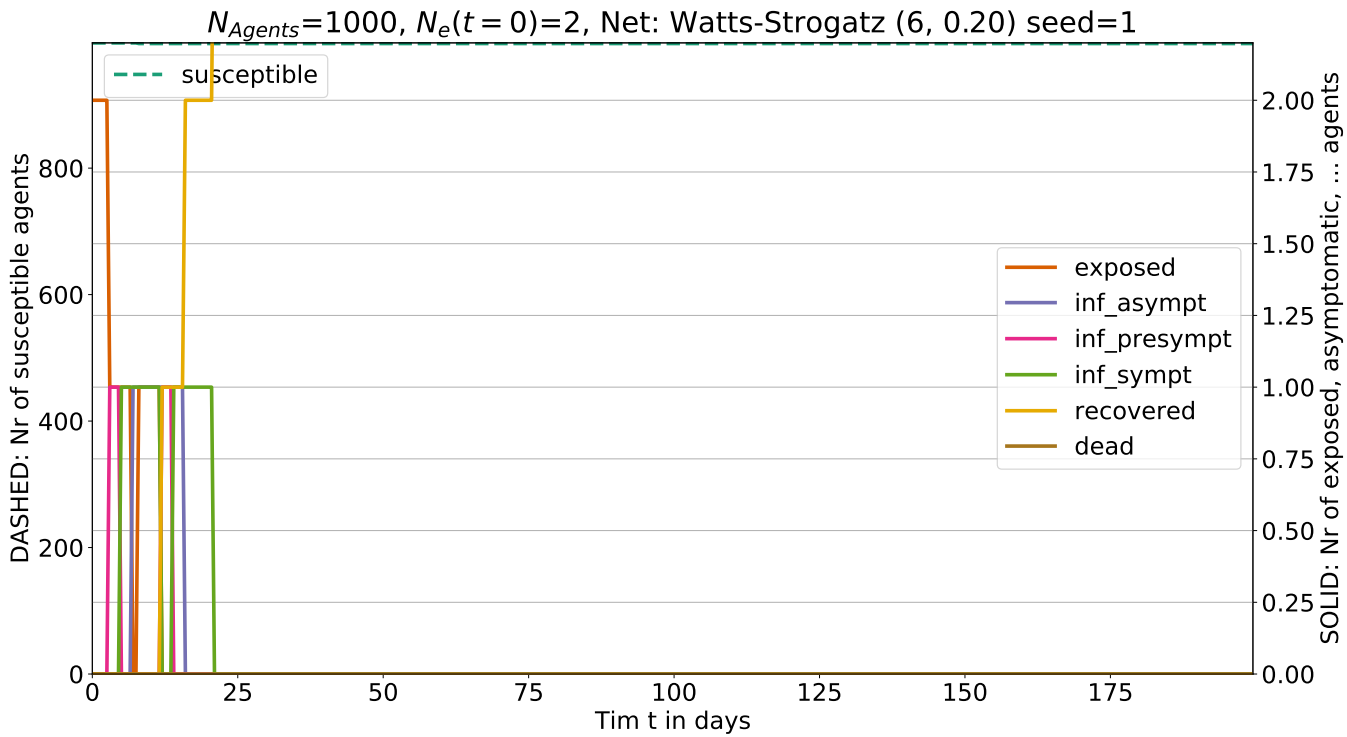


Figure: This simulation is stochastic! Even though the same model setup is used it might not lead to an outbreak (a qualitatively difference!). Model output depends on the actual course of microscopic events.

Overview ABM

Let's develop an ABM (Slide 14/56 from Lecture 10 on ABM)

1. Specific problem to be solved by the ABM.
   How do a few infected agents affect a small, interconnected, simple society split into three age-/riskgroups? What are the impacts of certain local policies?
2. Design of agents and their static / dynamic attributes.
   Class *Agent()*, function *initialise()*, that creates heterogeneous agents, and helper-function *catch_virus* that creates an agent's infection course (when the agent catches the virus)
3. ~~Design of an environment and the way agents interact with it.~~
4. Design of agents' mutual interactions.
   Via social network of agents
5. Design of agents' behaviours.
   Via function *update()* and for single agent *catch_virus*
6. Availability of data. AND
7. Method of model validation.
   E.g. via the obtained reproductive number R_0 (not shown here)

Validation

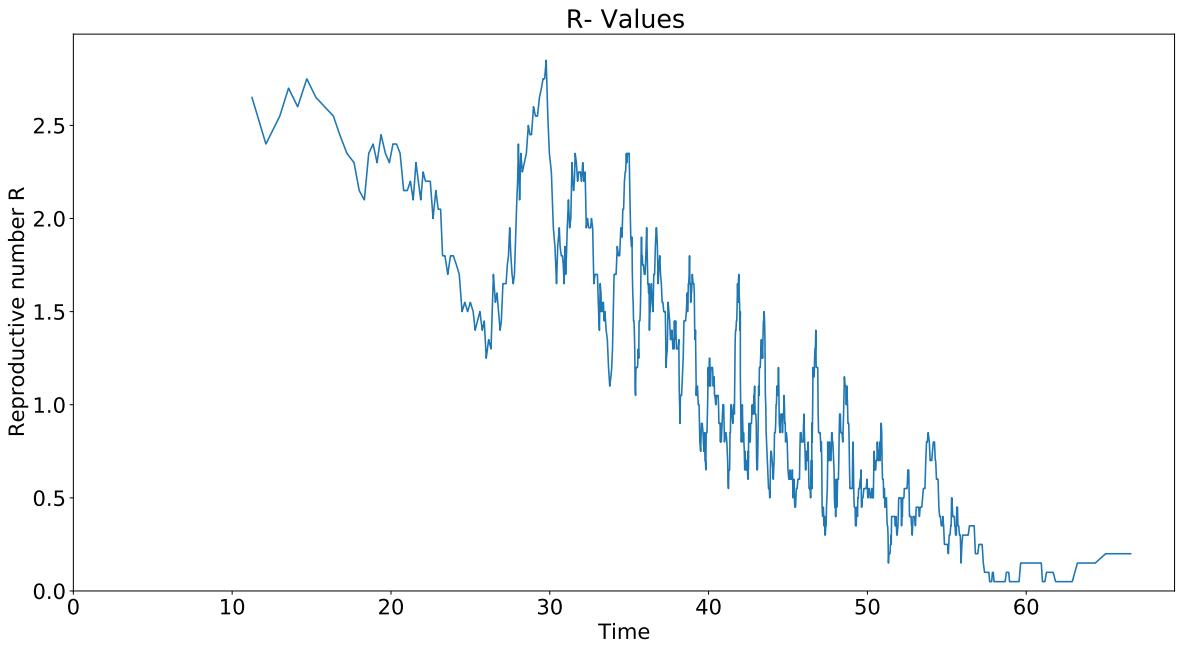
How can we verify if these results make sense? E.g.

- Ensemble Runs: Do many different simulation runs.
- Observe aggregate indicators
  - The initial reproductive number R_0.
  - Serial Interval (not done here)

Validation

How can we verify if these results make sense? E.g.

- Ensemble Runs: Do many different simulation runs.
- Observe aggregate indicators
  - The initial reproductive number R_0.
  - Serial Interval (not done here)



Policies

Policies

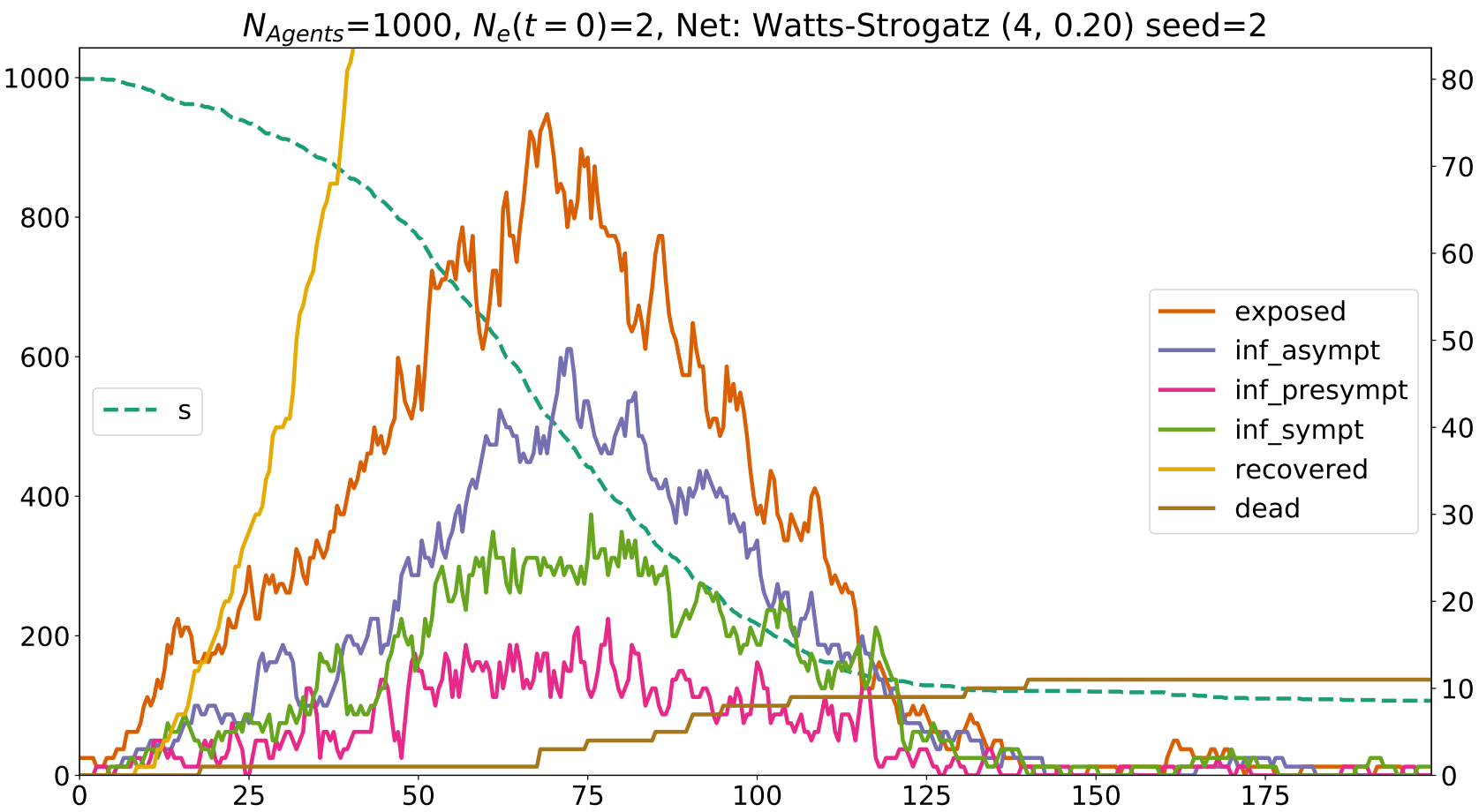
The major idea of such a model was to test the impact of local policies.
E.g. What happens

- if the number of contacts is reduced (due to a socially distancing public)?
- or if the people keep contacts within tight clusters (households, neighbours)?
- or if the agents reduce their *base_infectiousness*, by wearing a mask?

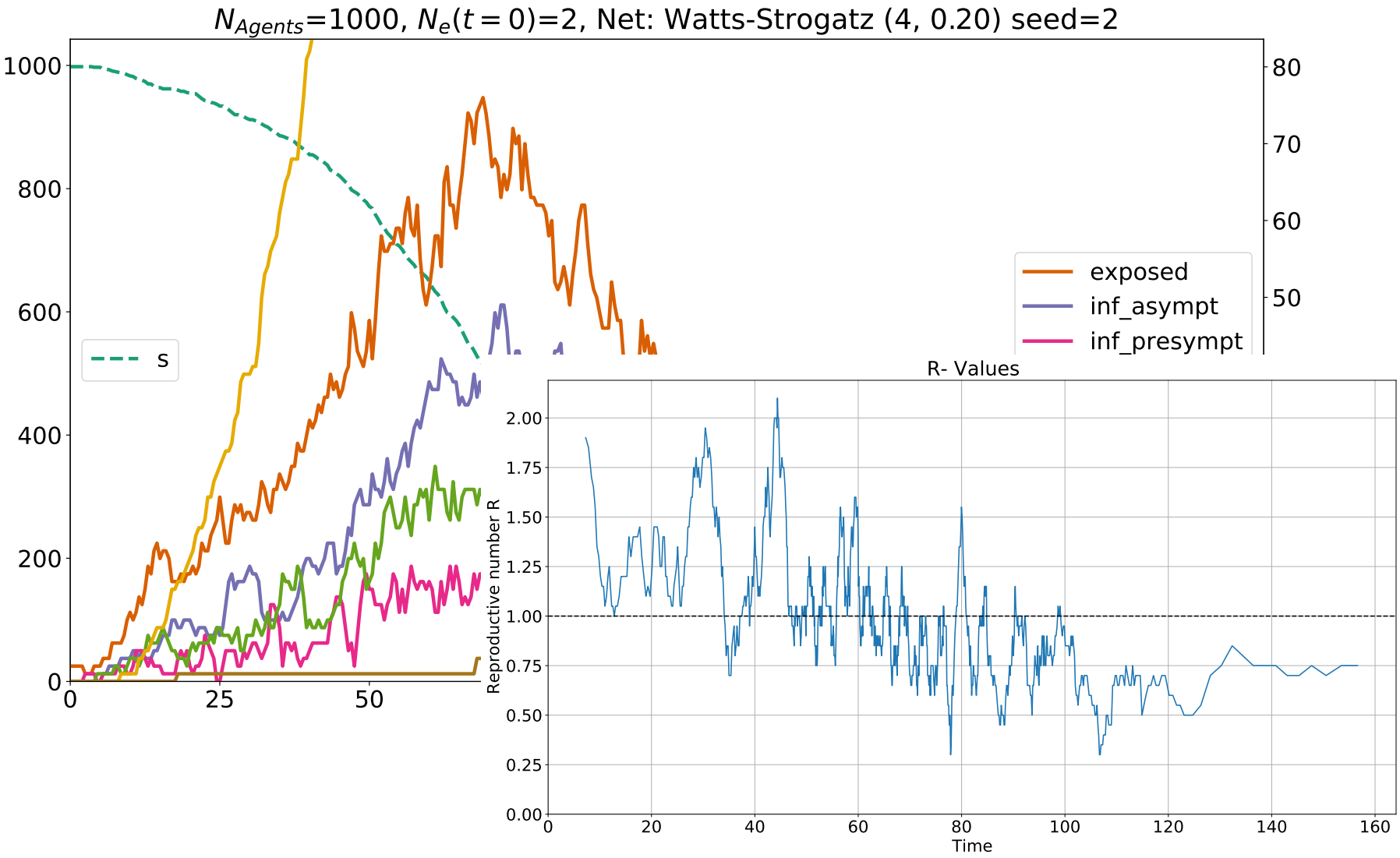
Parameters of the Model

Parameter	Description	current value
FRAC_RISKGROUPS	Percentage of each risk group	[0.2, 0.5, 0.3]
BASE_I	Distribution of base infectiousness of the agents	beta(1, 3)
P_SYMPTOMATIC	Probability to develop symptoms	[0.1, 0.5, 0.8]
INCUBATION_PERIOD	Distribution of the incubation time	gamma(5.807, 0.948)
CFR	Case fatality ratio for each age group	[0.0001, 0.005, 0.05]
I_SYMPTOMATIC	Strength of infectiousness for a symptomatic infection	1
I_ASYMPTOMATIC	-"- for an asymptomatic infection	0.2
I_PRESYMPTOMATIC	-"- for a pre-symptomatic infection	0.5
TIME_I_PRESYMPT	infectious days before symptom onset	2
TIME_I_POSTSYMPT	infectious days after symptom onset	4
K_WS_NETWORK	Number of (nearest) neighbours in networks for nodes	6
P_WS_NETWORK	Probability of each link to be rewired	0.2
N_AGENTS	Nr of agents	1000
N_INFECTED_INIT	Nr of agents set to "exposed" at t=0	2

Policy: Decrease k of Network to $k = 4$



Policy: Decrease k of Network to $k = 4$



What we have COVERED today.

- An Agent-Based Model of the spread of Covid-19 in a small society in order to test local policies
- How to make agents heterogeneous: Drawing from probability distributions
- How to let agents interact with each other: Social Networks for the interaction of agents

Assignment

- Basic:
  - Read and understand the code!
  - Run it with several different *seed* values
  - Change the network topology and properties (e.g. k or p of the Watts-Strogatz Network). What policy could this correspond to?
  - Try an entirely different network ($\rightarrow$ networkx documentation).
  - Take the distributions for *incubation period* or *base infectiousness*, draw samples from them and plot their PDF and the histogram of frequencies (see Concept 2). Note: You may want to create a separate script for this.
  - Select one of these distributions in the Covid-19 Model and change it (e.g. decrease the incubation period or make all agents equally infectious). Find something that interests you!

Deadline: Wednesday, 2nd December 2020, send relevant code, a PDF with your conclusions (including figures) to peter.steiglechner@leibniz-zmt.de. As always, please engage if you have any questions or ideas that you want to discuss.

Assignment

- Intermediate:
  - Implement a soft isolation policy: I.e. when an agent turns symptomatic, she/he will quarantine and strongly reduce further contacts.

    Hint: For this you might want to add a line in the *update* function that reduces the number of contacts when an isolated, symptomatic agent is about to infect these.
  - This policy may not be in place immediately, but only some time after the outbreak has been noticed. Implement a delay of the policy implementation. How does such a delay impact the effectiveness of a policy?

Deadline: Wednesday, 2nd December 2020, send relevant code, a PDF with your conclusions (including figures) to peter.steiglechner@leibniz-zmt.de. As always, please engage if you have any questions or ideas that you want to discuss.

Assignment

- Hard, if you feel like exploring:
  - Implement your own (time-dependent) policy strategy
    - This could include dynamic changes in the network
    - Or different behaviour of each age-group.
  - Let a few agents be defectors that do not adhere to the policies.

Note: Whenever you program anything, make sure you test (for yourself), e.g. by varying the crucial parameters, whether the results are what you expect.

Deadline: Wednesday, 2nd December 2020, send relevant code, a PDF with your conclusions (including figures) to peter.steiglechner@leibniz-zmt.de. As always, please engage if you have any questions or ideas that you want to discuss.

References

Bookstaber, R. (2017). *The End of Theory: Financial Crises, the Failure of Economics, and the Sweep of Human Interaction*. Englisch. Illustrated Auflage. Princeton, NJ: Princeton Univers. Press. ISBN: 978-0-691-16901-9.

Ferguson, N. et al. (Mar. 2020). *Report 9: Impact of non-pharmaceutical interventions (NPIs) to reduce COVID19 mortality and healthcare demand*. en-US-GB. Report. Accepted: 2020-03-17T09:57:15Z Publication Title: 20. DOI: 10.25561/77482. URL: http://spiral.imperial.ac.uk/handle/10044/1/77482 (visited on 10/28/2020).

He, X. et al. (May 2020). "Temporal dynamics in viral shedding and transmissibility of COVID-19". en. In: *Nature Medicine* 26.5. Number: 5 Publisher: Nature Publishing Group, pp. 672–675. ISSN: 1546-170X. DOI: 10.1038/s41591-020-0869-5. URL: https://www.nature.com/articles/s41591-020-0869-5 (visited on 10/28/2020).

Keeling, M. J. et al. (Oct. 2020). "Precautionary breaks: planned, limited duration circuit breaks to control the prevalence of COVID-19". en. In: *medRxiv*. Publisher: Cold Spring Harbor Laboratory Press, p. 2020.10.13.20211813. ISSN: 2021-1813. DOI: 10.1101/2020.10.13.20211813. URL: https://www.medrxiv.org/content/10.1101/2020.10.13.20211813v1 (visited on 10/28/2020).

Lauer, S. A. et al. (Mar. 2020). "The Incubation Period of Coronavirus Disease 2019 (COVID-19) From Publicly Reported Confirmed Cases: Estimation and Application". In: *Annals of Internal Medicine*. ISSN: 0003-4819. DOI: 10.7326/M20-0504. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7081172/ (visited on 10/28/2020).

Reiner, R. C. et al. (Oct. 2020). "Modeling COVID-19 scenarios for the United States". en. In: *Nature Medicine*. Publisher: Nature Publishing Group, pp. 1–12. ISSN: 1546-170X. DOI: 10.1038/s41591-020-1132-9. URL: https://www.nature.com/articles/s41591-020-1132-9 (visited on 10/28/2020).

Further reading

Hiroki Sayama 2015
Introduction to the Modeling and Analysis of Complex Systems
Open SUNY Textbooks

NetworkX Reference 2.0
https://networkx.github.io/documentation/stable/_downloads/networkx_reference.pdf
Sep 20, 2017