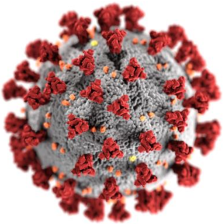
Agent-Based Model of Covid-19

Lecture 11

Peter Steiglechner

29 November 2023



So far...

Differential Equation Models	Agent-Based Models

So far...

Differential Equation Models	Agent-Based Models
SIR	Segregation (Schelling)
Population growth	Predator-prey (rabbit-fox)
HIV-cells	
Chemostat	
NPZ(D)	
Predator-prey	

So far...

Differential Equation Models	Agent-Based Models
SIR	Segregation (Schelling)
Population growth	Predator-prey (rabbit-fox)
HIV-cells	Covid-19
Chemostat	
NPZ(D)	
Predator-prey	

What we will cover today.

- Agent-Based Model that simulates the spread of Covid-19 in a small society consisting of three age groups and realistic, stochastic infection dynamics. Understand how a model like this can be used to design policies.
- *Concept 1+2:* Parametrisation of ABMs: draw heterogeneous agent features from (discrete/continuous) distributions
- *Concept 3:* Interaction in ABMs: network of agents
- *Concept 4:* Event scheduling in ABMs

What we will cover today.

- Agent-Based Model that simulates the spread of Covid-19 in a small society consisting of three age groups and realistic, stochastic infection dynamics. Understand how a model like this can be used to design policies.
- *Concept 1+2:* Parametrisation of ABMs: draw heterogeneous agent features from (discrete/continuous) distributions
- *Concept 3:* Interaction in ABMs: network of agents
- *Concept 4:* Event scheduling in ABMs

What we will cover today.

- Agent-Based Model that simulates the spread of Covid-19 in a small society consisting of three age groups and realistic, stochastic infection dynamics. Understand how a model like this can be used to design policies.
- *Concept 1+2:* Parametrisation of ABMs: draw heterogeneous agent features from (discrete/continuous) distributions
- *Concept 3:* Interaction in ABMs: network of agents
- *Concept 4:* Event scheduling in ABMs

What we will cover today.

- Agent-Based Model that simulates the spread of Covid-19 in a small society consisting of three age groups and realistic, stochastic infection dynamics. Understand how a model like this can be used to design policies.
- *Concept 1+2:* Parametrisation of ABMs: draw heterogeneous agent features from (discrete/continuous) distributions
- *Concept 3:* Interaction in ABMs: network of agents
- *Concept 4:* Event scheduling in ABMs

What we will cover today.

- Agent-Based Model that simulates the spread of Covid-19 in a small society consisting of three age groups and realistic, stochastic infection dynamics. Understand how a model like this can be used to design policies.
- *Concept 1+2:* Parametrisation of ABMs: draw heterogeneous agent features from (discrete/continuous) distributions
- *Concept 3:* Interaction in ABMs: network of agents
- *Concept 4:* Event scheduling in ABMs

Coronavirus and Covid-19

- SARS-CoV-2 (Coronavirus) appears in Wuhan, China (Dec 19); worldwide spread; pandemic declared by WHO (Jan 20)
- The virus spreads mainly airborne (aerosols, ...)

- The main question in early 2020: "How dangerous is Covid-19?"
  - High fatality from corresponding disease 'Covid-19' (March 2020, Italy/Iran/...)
  - Fast spreading in societies/over the world

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

What made Coronavirus so difficult to manage?

.

Your answers

What made Coronavirus so difficult to manage?

.

Your answers

1. Complex spreading patterns
2. Aggressive virus
3. Specific groups of people especially vulnerable to the diseases
4. Little prior experience in (political) decision-making

→ In sum: a lot of uncertainty!

E.g. He et al. (2020)

Policy approaches – 2020

- (Nearly) complete lock-down (Italy, 2020)
- Quarantine positive cases and first contacts (Germany, 2020)
- Homeschooling/-office, social distancing (Germany, 2020)
- Individual responsibility to distance (Sweden, 2020)
- Herd immunity (?)
- 'Circuit breaker' (two-week lock-down) (Germany, winter 2020)

Policy approaches – 2020

- (Nearly) complete lock-down (Italy, 2020)
- Quarantine positive cases and first contacts (Germany, 2020)
- Homeschooling/-office, social distancing (Germany, 2020)
- Individual responsibility to distance (Sweden, 2020)
- Herd immunity (?)
- ‘Circuit breaker’ (two-week lock-down) (Germany, winter 2020)

In general: **social distancing**

- ‘Social’: fewer contacts
  - Close schools (asymptomatic infections)
  - Ban large gatherings
- ‘Distancing’: reduced ‘contact’
  - Avoid close physical contact
  - Wear a mask

Policies have very different epidemiological and socio-economic implications

What to do?

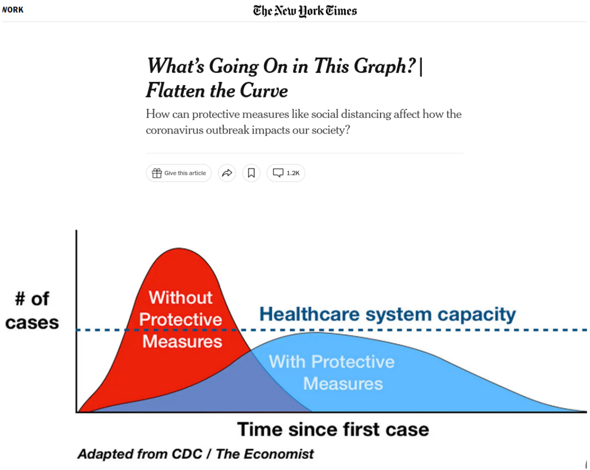
→ A model to the rescue!?

Role of Modeling

- Initially: basic models to understand epidemiology $\rightarrow$ 'flatten the curve'.

Role of Modeling

- Initially: basic models to understand epidemiology
  $\rightarrow$ 'flatten the curve'.



Role of Modeling

- Initially: basic models to understand epidemiology
  $\rightarrow$ ‘flatten the curve’.

- Later: models with more complexity, more realism
  e.g. age-structured, local resolution, realistic human behaviour.
  See also this article on why we need (local, context dependent) models in the pandemic.

Role of Modeling

- Initially: basic models to understand epidemiology
  $\rightarrow$ 'flatten the curve'.
- Later: models with more complexity, more realism
  e.g. age-structured, local resolution, realistic human behaviour.
  See also this article on why we need (local, context dependent) models in the pandemic.
- Which policies work to reduce the pressure of Covid-19 cases on the health system?
  - **Design:** suggest 'new' policies that have a higher effectiveness (Keeling et al., 2020)
  - **Hindsight analysis:** which policy strategy was efficient? (e.g. Reiner et al., 2020) ($\rightarrow$ what-if questions)

Role of Modelling - 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


EUROPE

A chilling scientific paper helped upend U.S. and U.K. coronavirus strategies

By William Booth
March 17, 2020 at 4:25 p.m. EDT

From left, chief medical officer Chris Whitty, British Prime Minister Boris Johnson and chief scientific officer Patrick Vallance give a news conference on the coronavirus pandemic in London, March 16, 2020. (Richard Pohle/Pool/EPA-EFE/Shutterstock)

Washington Post, 17 Mar 2020

- March 2020: UK changes its strategy
- Mainly due to one ABM developed at Imperial College (Ferguson et al., 2020)!

RECAP: Different types of Models

- ODE-based SIR Models
  - Aggregate variables: size of the population of susceptible/(exposed)/infected/recovered
  - Equations determine how populations co-evolve.
- Agent Based Model:
  - Discrete entities ‘agents’
  - Individual behaviour specified by rules/heuristics (e.g. what does an adult do when infected?)
    $\rightarrow$ produces individual trajectories

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

Why is ABM useful in this pandemic

- The virus and humans act non-homogeneously:
  - The pandemic is not homogeneous in space
  - Each person has a different health response to catching the virus
  - Each person behaves differently (e.g. contacts, policy compliance)
- The world is discrete. Therefore, we need to simulate the actual sequence of events.
  Same society, different dynamics.
- Single, random and 'microscopic' events are crucial. Reality is path-dependent.
  e.g. the behaviour of 'patient 0' is crucial for the evolution of the pandemic
- Uncertainty is a crucial aspect of the dynamics. The future is unpredictable.
- The 'rules of the game' evolve as events unfold (it's a non-ergodic system).
  e.g. our behaviour in the first and the second lock-down differed widely..

Concept 0 Summary:

Consider ABM when:

- Microscopic behaviour can cause **emergent** macroscopic phenomena.
  Example: Fish swarm or bird flock
- People, space, or responses (entities or 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** and **randomness** play dominant roles.
- Context matters (**non-ergodic system**)

'The end of theory' by Bookstaber (2017). A very enlightening book about the paradigm shift induced by Agent-Based Modelling

What we will cover today.

- Agent-Based Model that simulates the spread of Covid-19 in a small society consisting of three age groups and realistic, stochastic infection dynamics. Understand how a model like this can be used to design policies.
- *Concept 1+2:* Parametrisation of ABMs: draw heterogeneous agent features from (discrete/continuous) distributions
- *Concept 3:* Interaction in ABMs: network of agents
- *Concept 4:* Event scheduling in ABMs

Design an ABM - I

Let's think of an ABM (Slide 16/47 from Lecture 9 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' behaviour
5. Design of agent mutual interactions.
6. Availability of data.
7. Method of model validation.

Design an ABM - I

Let's think of an ABM (Slide 16/47 from Lecture 9 on ABM)

1. Specific problem to be solved by the ABM.
   How do a few infected agents affect a small, interconnected, simple society, which consists of agents in three age groups? 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' behaviour
5. Design of agent mutual interactions.
6. Availability of data.
7. Method of model validation.

Design an ABM - I

Let's think of an ABM (Slide 16/47 from Lecture 9 on ABM)

1. Specific problem to be solved by the ABM.
   How do a few infected agents affect a small, interconnected, simple society, which consists of agents in three age groups? 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' behaviour
5. Design of agent mutual interactions.
6. Availability of data.
   Data informs the parametrisation
7. Method of model validation.
   ...

Design an ABM - II

Let's create an ABM (Slide 17/47 from Lecture 9 on ABM)

1. Design the data structure to store the attributes of the agents.
2. ~~Design the data structure to store the states of the environment.~~
3. ~~Describe the rules for how the environment behaves on its own.~~
4. ~~Describe the rules for how agents interact with the environment.~~
5. Describe the rules for how agents behave on their own.
6. Describe the rules for how agents interact with each other.

Design an ABM - II

Let's create an ABM (Slide 17/47 from Lecture 9 on ABM)

1. Design the data structure to store the attributes of the agents.
   class *agent()* with attributes *age*, *health_state*
2. ~~Design the data structure to store the states of the environment.~~
3. ~~Describe the rules for how the environment behaves on its own.~~
4. ~~Describe the rules for how agents interact with the environment.~~
5. Describe the rules for how agents behave on their own.
6. Describe the rules for how agents interact with each other.

Design an ABM - II

Let's create an ABM (Slide 17/47 from Lecture 9 on ABM)

1. Design the data structure to store the attributes of the agents.
   class *agent()* with attributes *age*, *health_state*
2. ~~Design the data structure to store the states of the environment.~~
3. ~~Describe the rules for how the environment behaves on its own.~~
4. ~~Describe the rules for how agents interact with the environment.~~
5. Describe the rules for how agents behave on their own.
   when healthy: stay healthy
   when exposed: become infected
   when infected: go through the stages of Covid-19 following stochastic, age-dependent patterns.
6. Describe the rules for how agents interact with each other.

Design an ABM - II

Let's create an ABM (Slide 17/47 from Lecture 9 on ABM)

1. Design the data structure to store the attributes of the agents.
   class *agent()* with attributes *age*, *health_state*
2. ~~Design the data structure to store the states of the environment.~~
3. ~~Describe the rules for how the environment behaves on its own.~~
4. ~~Describe the rules for how agents interact with the environment.~~
5. Describe the rules for how agents behave on their own.
   when healthy: stay healthy
   when exposed: become infected
   when infected: go through the stages of Covid-19 following stochastic, age-dependent patterns.
6. Describe the rules for how agents interact with each other.
   agents are connected through a network. They meet 'physically' with their network neighbours, potentially infecting each other.

ABM base units

```
class agent:
    ...

def initialise():
    ...

def initialise_network():
    ...

def update():
    # (1) agents update health_status
    # (2) interactions using catch_virus, infect_others,
    ...

def catch_virus(ag, t_exposure):
    ...

def infect_others(ag, t_exposure):
    ...

# Run
initialise()
for t in range(T):
    update()
observe()
```

What do we need to model Covid-19

- Understand individual infection course
  what happens with infected people
- Understand transmissions
  how do people infect each other
- Understand social structure
  who infects who

Covid-19: Course of an infection (2020)

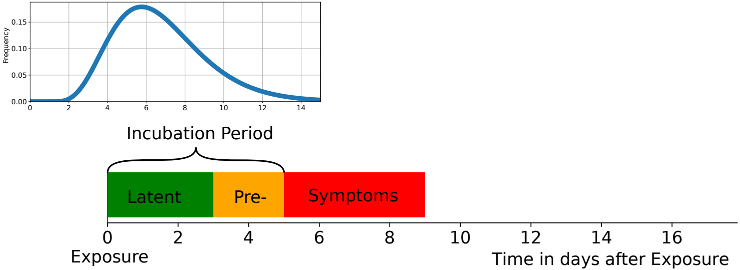
Latent period? Pre-Symptomatic and symptomatic, or asymptomatic infection?

Covid-19: Course of an infection (2020)

Latent period? Pre-Symptomatic and symptomatic, or asymptomatic infection?

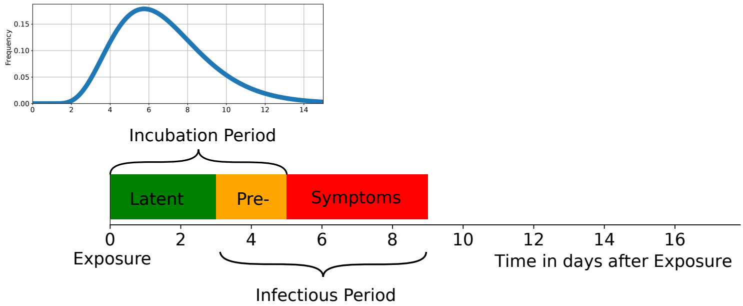
Covid-19: Course of an infection (2020)

Latent period? Pre-Symptomatic and symptomatic, or asymptomatic infection?



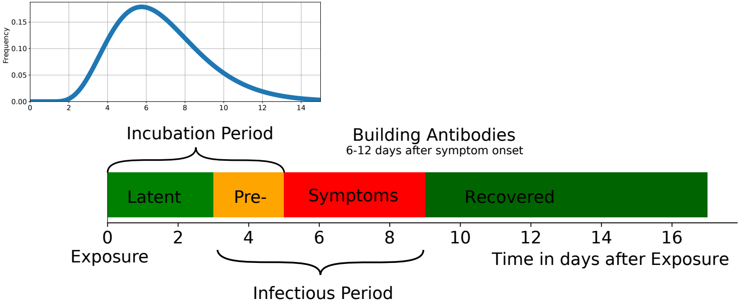
Covid-19: Course of an infection (2020)

Latent period? Pre-Symptomatic and symptomatic, or asymptomatic infection?



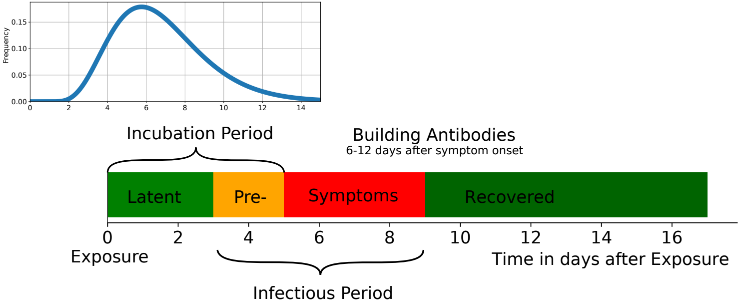
Covid-19: Course of an infection (2020)

Latent period? Pre-Symptomatic and symptomatic, or asymptomatic infection?



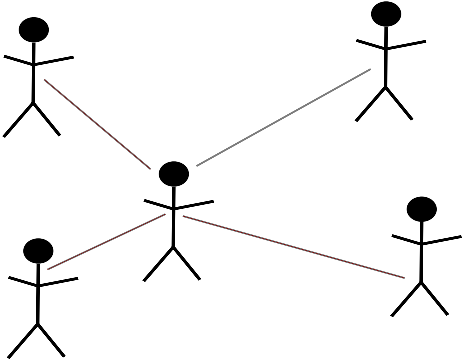
Covid-19: Course of an infection (2020)

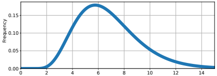
Latent period? Pre-Symptomatic and symptomatic, or asymptomatic infection?

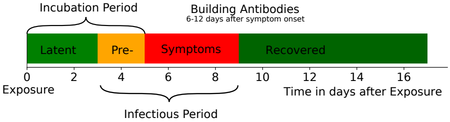


$\rightarrow$ infection course is age-dependent

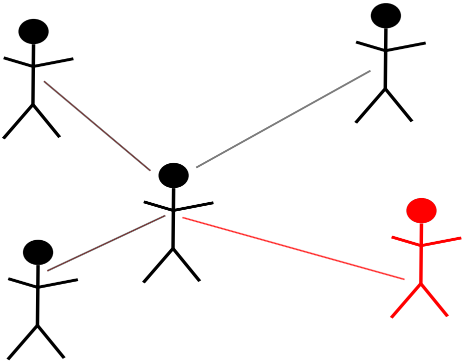
Model sketch

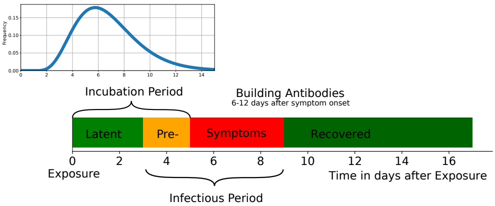






Model sketch



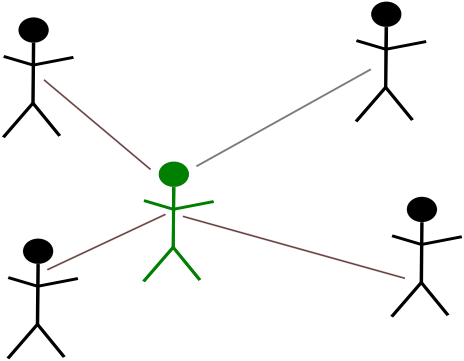


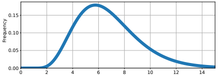
simple contagion

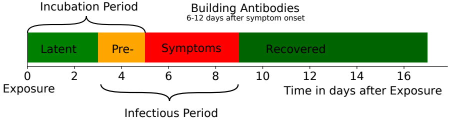
vs.

complex contagion

Model sketch





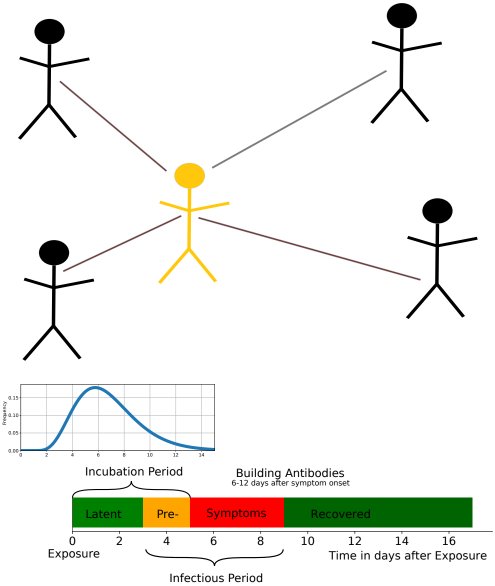


simple contagion

vs.

complex contagion

Model sketch

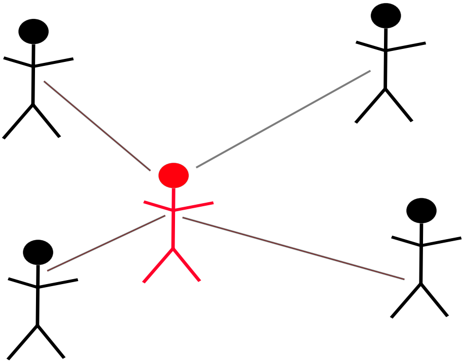


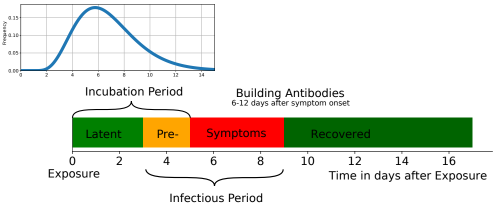
simple contagion

vs.

complex contagion

Model sketch



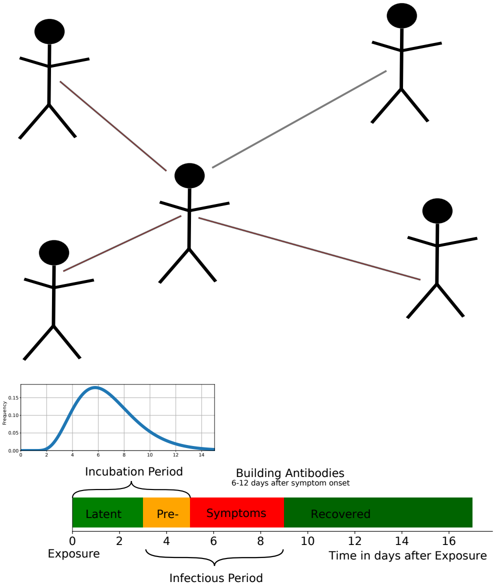


simple contagion

vs.

complex contagion

Model sketch



simple contagion

vs.

complex contagion

Agent and initialisation

```
class Agent:
    pass
```

An empty class.

Agent and initialisation

```
n_agents = 1000



class Agent:
      pass

def initialise():
      global agents
      agents = []

      for i in range(n_agents):
            ag = Agent()
            ag.id = i
            ag.health_state = "susceptible"
            ag.age = ???
            agents.append(ag)
```

Create all agents.

We want to have 20 % children, 50 % adults/low-risk, 30 % elderly/high-risk agents!

Concept 1: Drawing from discrete distributions

Concept 1: Discrete distributions – a simple example

Example

Problem: create a population with an attribute 'sex'

Concept 1: Discrete distributions – a simple example

Example

Problem: create a population with an attribute 'sex'

✓ For each agent, assign sex to male with 50 % and female else

```
for ag in agents:
    ag.sex = "male" if np.random.random() < 0.5 else "female"
```

Concept 1: Discrete distributions – a simple example

Example

Problem: create a population with an attribute 'sex'

✓ For each agent, assign sex to male with 50 % and female else
✓ For each agent, randomly draw one of two sexes (male/female with each 50 % probability)
→ *np.random.choice(options, size, replace, probability)*
i.e. choose *size* samples of *options* with given *probabilities*.

```
for ag in agents:
    ag.sex = "male" if np.random.random() < 0.5 else "female"
```

```
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,          # nr of draws (default=1)
            replace = True     # sample with/without replacement (default=True)
          )
```

Agent and initialisation

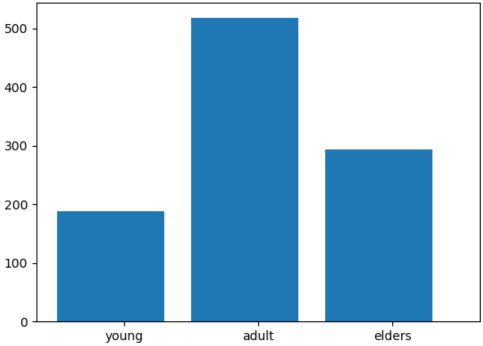
```python
n_agents = 1000
age_groups = ["child", "adult", "elderly"]
fraction_age_groups = [0.2, 0.5, 0.3]

class Agent:
        pass

def initialise():
        global agents
        agents = []

        for i in range(n_agents):
                ag = Agent()
                ag.id = i
                ag.health_state = "susceptible"
                ag.age = np.random.choice(age_groups, p=fraction_age_groups)
                agents.append(ag)
```

We want to have 20 % children, 50 % adults/low-risk, 30 % elderly/high-risk agents!

✓ Draw from discrete probability distribution for each agent

Agent and initialisation

```
n_agents = 1000
age_groups = ["child", "adult", "elder
fraction_age_groups = [0.2, 0.5, 0.3]

class Agent:
      pass

def initialise():
      global agents
      agents = []

      for i in range(n_agents):
            ag = Agent()
            ag.id = i
            ag.health_state = "suscept
            ag.age = np.random.choice
            agents.append(ag)
```



We want to have 20 % children, 50 % adults/low-risk, 30 % elderly/high-risk agents!

✓ Draw from discrete probability distribution for each agent

Agent and initialisation

```python

...
n_infected_init = 2  # Number of exposed agents at t=0.

class Agent:
      pass

def initialise():
      global agents
      agents = []

      for i in range(n_agents):
            ag = Agent()
            ag.id = i
            ag.health_state = "susceptible"
            ag.age = np.random.choice(age_groups, p=fraction_age_groups)
            agents.append(ag)

      symptomatic_agents = np.random.choice(agents, size=n_infected_init)
      for ag in symptomatic_agents:
            catch_virus(ag, t=0)

      return
```

Infect a few randomly selected agents
(we will define the function *catch_virus* in the next slides)

Concept 2: Drawing from continuous distributions

Concept 2: Continuous distributions

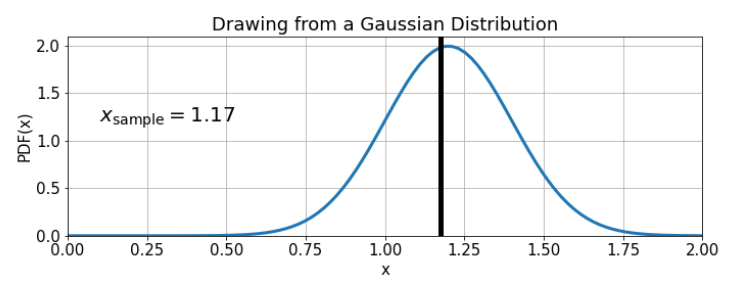
Random variable

- Random variable x
- Probability density function PDF: $p(x)$
- Needs to integrate to one: $\int_{-\infty}^{\infty} p(x)\ dx = 1$
- Now, we draw samples from this distribution

Concept 2: Continuous distributions

Random variable

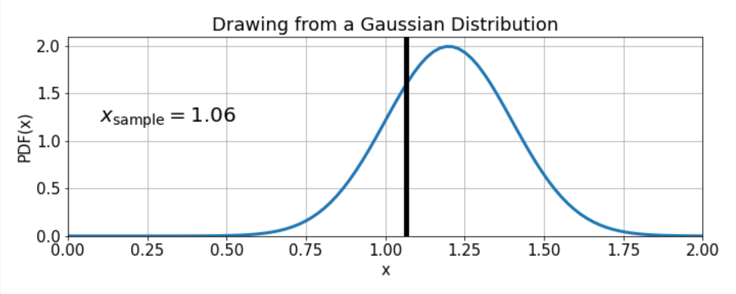
- Random variable x
- Probability density function PDF: $p(x)$
- Needs to integrate to one: $\int_{-\infty}^{\infty} p(x)\, dx = 1$
- Now, we draw samples from this distribution



Concept 2: Continuous distributions

Random variable

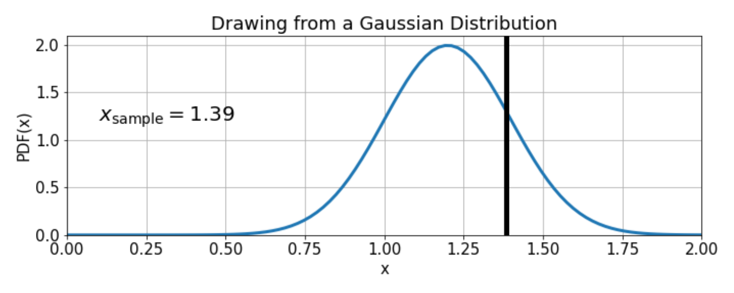
- Random variable x
- Probability density function PDF: $p(x)$
- Needs to integrate to one: $\int_{-\infty}^{\infty} p(x)\, dx = 1$
- Now, we draw samples from this distribution



Concept 2: Continuous distributions

Random variable

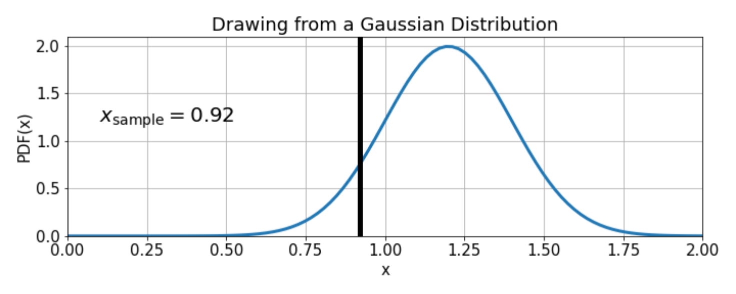
- Random variable x
- Probability density function PDF: $p(x)$
- Needs to integrate to one: $\int_{-\infty}^{\infty} p(x)\, dx = 1$
- Now, we draw samples from this distribution



Concept 2: Continuous distributions

Random variable

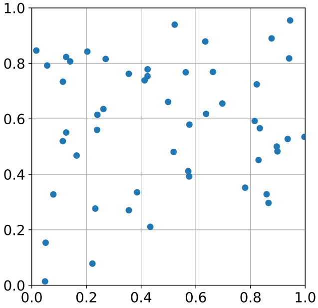
- Random variable x
- Probability density function PDF: $p(x)$
- Needs to integrate to one: $\int_{-\infty}^{\infty} p(x)\, dx = 1$
- Now, we draw samples from this distribution



Concept 2: Repetition

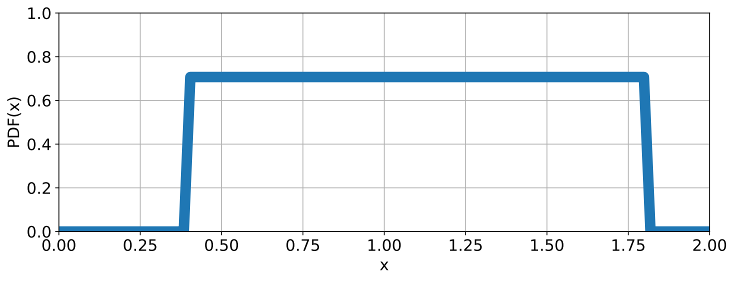
We have already applied 'Concept 2: Drawing from continuous distributions' in both previous ABMs in Lectures 9 and 10:

Agents (e.g. foxes and rabbits) were randomly spawned on a 2D space (x, y) with $x, y \in [0, 1]$.
Notation: *Uniform distribution* between 0 and 1.



Concept 2: Common continuous distributions

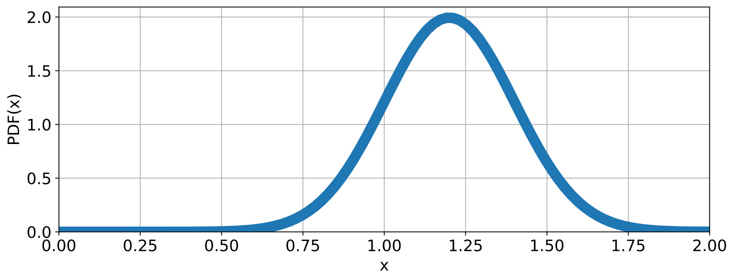
Uniform distribution



What	continuous, bounded range (max and min are known)
PDF	$\mathcal{U}(x_{\min}, x_{\max}) = \frac{1}{x_{\max} - x_{\min}}$
Usage	Uninformative. Use when we have no clue about the random variable.

Concept 2: Common continuous distributions

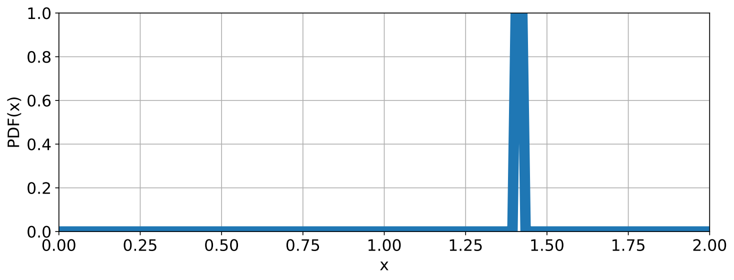
Gaussian or Normal distribution



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

Concept 2: Common continuous distributions

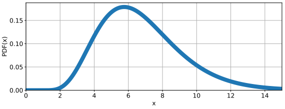
Delta distribution



What	continuous, inifinte/bounded range
PDF	$\delta(x-\tilde{x}) =: \delta_{\tilde{x}} = \begin{cases} \infty \text{ if } x = \tilde{x} \\ 0 \text{ else} \end{cases}$ $\quad \int \delta(x-\tilde{x}) \mathrm{d}x := 1$
Usage	We are absolutely certain about the parameter x, e.g. $g = 9.81\, m/s^2$! Typically, we simply fix $x = \tilde{x}$

Concept 2: Common continuous distributions

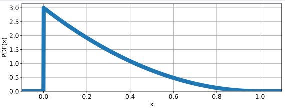
Gamma distribution



$\Gamma(5.9807,\ 0.948)$

- 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 $[0, 1]$
- often used for parameters that represent uncertain probabilities

$\rightarrow$ There are soo many distributions 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:

```
import scipy.stats as stats

mu = 1.2
sigma = 0.2
some_normal_dist = stats.norm(mu, sigma)
```

- For other distributions, simply replace 'norm' with e.g. 'beta' and look up what parameters you need to specify!
- (as always in python, documentation is your friend 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)
- PDF via . pdf(x)
-

PDF:

```
x = np.linspace(0,2)
plt.plot(x, some_normal_dist.pdf(x))
plt.title(f'PDF of a normal distribution with mu={mu}, sigma={sigma}')
```

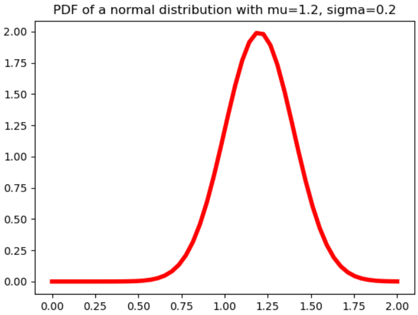
Concept 2: Python Package 'scipy.stats'

'scipy.stats'

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

PDF:

```
x = np.linspace(0,2)
plt.plot(x, some_normal_dist.pdf(x))
plt.title(f'PDF of a normal distribution
```



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:

```
samples = some_normal_dist.rvs(100)  # Argument = Nr of samples
plt.hist(samples)
plt.xlabel("x")
plt.ylabel("frequency")
plt.title("Histogram of samples")
plt.show()
```

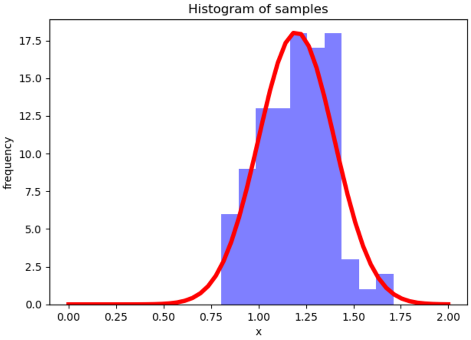
Concept 2: Python Package 'scipy.stats'

'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 distri

```
samples = some_normal_dist.rvs(100)  # Ar
plt.hist(samples)
plt.xlabel("x")
plt.ylabel("frequency")
plt.title("Histogram of samples")
plt.show()
```



ABM base units

```python
class agent:
    ...

def initialise():
    ...

def initialise_network():
    ...

def update():
    # (1) agents update health_status
    # (2) interactions using catch_virus, infect_others,
    ...

def catch_virus(ag, t_exposure):
    ...

def infect_others(ag, t_exposure):
    ...

# Run
initialise()
for t in range(T):
    update()
observe()
```

‘Catch the virus and go through the stages of the infection’

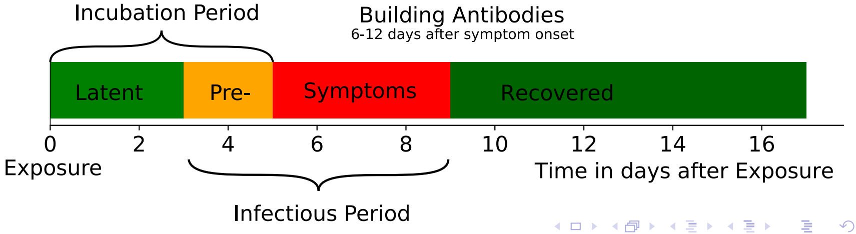
We predefine the course of an infection at the time of exposure:

1. Is the infection symptomatic or asymptomatic?

‘Catch the virus and go through the stages of the infection’

We predefine the course of an infection at the time of exposure:

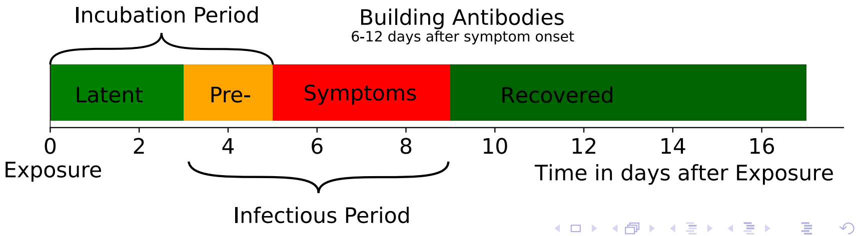
1. Is the infection symptomatic or asymptomatic?
2. How long is the incubation period? (i.e. when do symptoms start?)
   $\rightarrow$ drawn from Gamma Distribution



‘Catch the virus and go through the stages of the infection’

We predefine the course of an infection at the time of exposure:

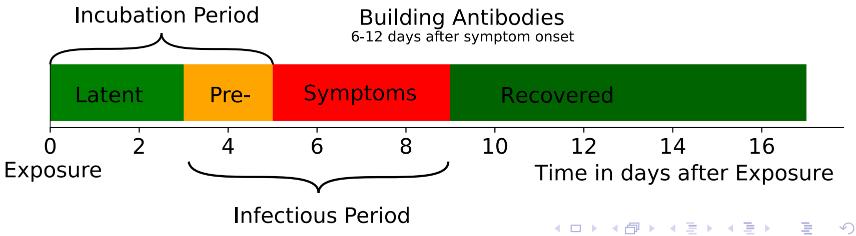
1. Is the infection symptomatic or asymptomatic?
2. How long is the incubation period? (i.e. when do symptoms start?)
   → drawn from Gamma Distribution
3. When is the agent infectious?
   → (age-dependent) rate of asymptomatic cases



‘Catch the virus and go through the stages of the infection’

We predefine the course of an infection at the time of exposure:

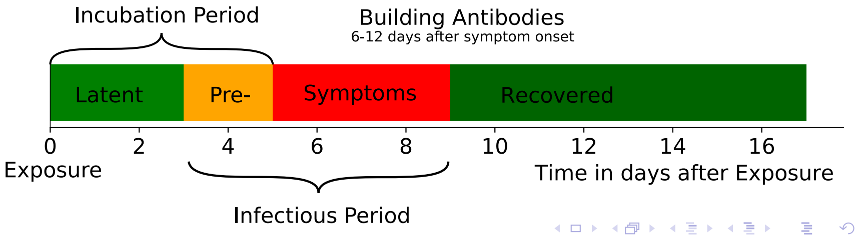
1. Is the infection symptomatic or asymptomatic?
2. How long is the incubation period? (i.e. when do symptoms start?)
   $\rightarrow$ drawn from Gamma Distribution
3. When is the agent infectious?
   $\rightarrow$ (age-dependent) rate of asymptomatic cases
4. Will the agent die from the infection?
   $\rightarrow$ (age-dependent) case fatality rate



'Catch the virus and go through the stages of the infection'

We predefine the course of an infection at the time of exposure:

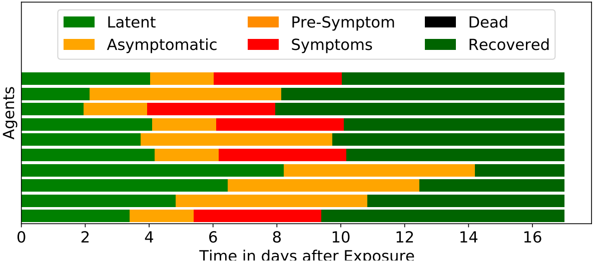
1. Is the infection symptomatic or asymptomatic?
2. How long is the incubation period? (i.e. when do symptoms start?)
   → drawn from Gamma Distribution
3. When is the agent infectious?
   → (age-dependent) rate of asymptomatic cases
4. Will the agent die from the infection?
   → (age-dependent) case fatality rate
5. How infectious is the agent?
   → drawn from Beta and multiplied by a factor depending on the health state.



Code of ‘catch_virus’

```
p_symptomatic = {"child": 0.1, "adult": 0.5, "elderly": 0.8}
incubation_period_dist = stats.gamma(5.807, 0.948)
time_infectious_preSymptomstart = 2
time_infectious_postSymptomstart = 4
case_fatality_ratio = {"child": 0.00001, "adult": 0.005, "elderly": 0.05}
base_infectiousness_dist = stats.beta(1, 3)

def catch_virus(ag, t_exposure):
    ag.health_state = "latent"
    ag.t_e = t_exposure

    p_s = p_symptomatic[ag.group]
    ag.symptomatic = True if np.random.random() < p_s else False

    incubation_period = incubation_period_dist.rvs()

    ag.infectious_period = [
        ag.t_e + incubation_period - time_infectious_preSymptomstart,
        ag.t_e + incubation_period + time_infectious_postSymptomstart
    ]

    if ag.symptomatic:
        ag.t_onset_symptoms = ag.t_e + incubation_period
        p_d = case_fatality_rate[ag.group]
        ag.fatal_outcome = True if np.random.random() < p_d else False
    else:
        ag.t_onset_symptoms = np.nan
        ag.fatal_outcome = False

    ag.base_infectiousness = base_infectiousness_dist.rvs()

      return
```

A few examples of typical infection courses



Concept 1 and 2 Summary

Drawing from distributions

- We can create heterogeneous agents (or heterogeneous infection dynamics) by drawing independent parameters or properties from probability distributions (which are inferred from data) whenever we initialise an agent (or an 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)
```

Design an ABM - II

Let's create an ABM (Slide 17/47 from Lecture 9 on ABM)

1. Design the data structure to store the attributes of the agents.
   class *agent()* with attributes *age*, *health_state*
2. ~~Design the data structure to store the states of the environment.~~
3. ~~Describe the rules for how the environment behaves on its own.~~
4. ~~Describe the rules for how agents interact with the environment.~~
5. Describe the rules for how agents behave on their own.
   when healthy: stay healthy
   when exposed: become infected
   when infected: go through the stages of Covid-19 following stochastic, age-dependent patterns.
6. Describe the rules for how agents interact with each other.
   agents are connected through a network. They meet 'physically' with their network neighbours, potentially infecting each other.

What we will cover today.

- Agent-Based Model that simulates the spread of Covid-19 in a small society consisting of three age groups and realistic, stochastic infection dynamics. Understand how a model like this can be used to design policies.
- *Concept 1+2:* Parametrisation of ABMs: draw heterogeneous agent features from (discrete/continuous) distributions
- *Concept 3:* Interaction in ABMs: network of agents
- *Concept 4:* Event scheduling in ABMs

What we will cover today.

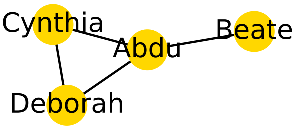
- Agent-Based Model that simulates the spread of Covid-19 in a small society consisting of three age groups and realistic, stochastic infection dynamics. Understand how a model like this can be used to design policies.
- *Concept 1+2:* Parametrisation of ABMs: draw heterogeneous agent features from (discrete/continuous) distributions
- *Concept 3:* Interaction in ABMs: network of agents
- *Concept 4:* Event scheduling in ABMs

Concept 3: Networks

Social network – Basics

Who interacts with who? $\rightarrow$ Social network
Here: who may get infected by who? $\rightarrow$ Social network

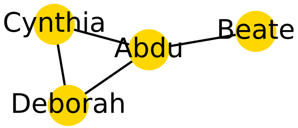
- Each node represents one agent
- Link/Edge between nodes means that agents can be in 'physical contact'



Social network – Basics

Who interacts with who? $\rightarrow$ Social network
Here: who may get infected by who? $\rightarrow$ Social network

- Each node represents one agent
- Link/Edge between nodes means that agents can be in 'physical contact'
- (Average) node degree = (avg) number of links from agents



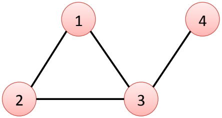
Social network – Basics

Who interacts with who? $\rightarrow$ Social network
Here: who may get infected by who? $\rightarrow$ Social network

- Each node represents one agent
- Link/Edge between nodes means that agents can be in 'physical contact'

- (Average) node degree = (avg) number of links from agents
- Adjacency matrix A, where $A_{ij} = 1$ denotes that a link connects nodes i and j

Network



Adjacency matrix

$$
\begin{array}{c|c}
{}_{i}\ {}^{j} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\
\begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}
\end{array}
$$

Adjacency list

i

1 $\longrightarrow$ {2, 3}

2 $\longrightarrow$ {1, 3}

3 $\longrightarrow$ {1, 2, 4}

4 $\longrightarrow$ {3}

Social network - Topology

- Here, we use 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.

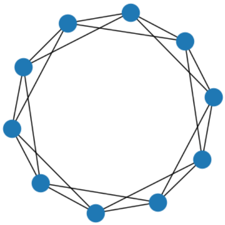
Social network - Topology

- Here, we use 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

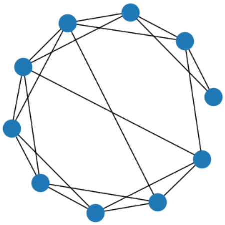
- Here, we use a 'Watts-Strogatz network' – often also referred to as 'small-world network'.
  - All n nodes/agents are aligned in a ring.
  - They are connected to their k nearest neighbours (left/right)



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

Social network - Topology

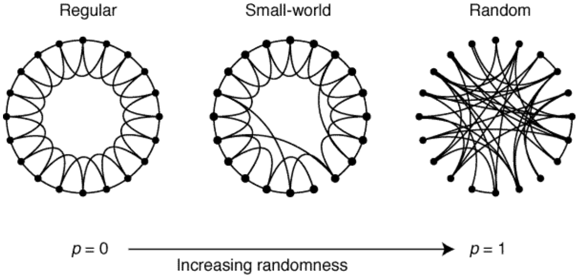
- Here, we use a 'Watts-Strogatz network' – often also referred to as 'small-world network'.
  - All n nodes/agents are aligned in a ring.
  - They are connected to their k nearest neighbours (left/right)
  - We loop through each agent and through each link to the right of that agent. With probability p, the 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

- Here, we use 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)
  - We loop through each agent and through each link to the right of that agent. With probability p, the link is capped and re-drawn to a random node/agent anywhere on the ring.



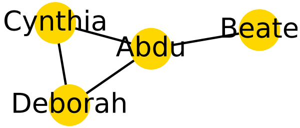
D. J. Watts & S. H. Strogatz, Collective dynamics of 'small-world' networks, *Nature*, 393:440–442, 1998.

Social network – Perspective

- Network theory is one of the hottest topics in science
- The method can be applied to various systems, topics, problems in ANY discipline
  neural networks, social media, climate tipping points, collapse of stock markets, ...
- More on network theory:
  - Directed and weighted links
  - Topologies of different networks:
    - Scale-free network $\rightarrow$ e.g. Barabási-Albert-Model
    - Random graph
  - Clustering
  - Adaptive networks, i.e. networks that change over time depending on the state of the system/agents.

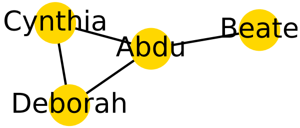
The *networkx* package in python

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



The *networkx* package in python

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

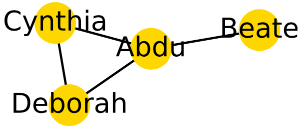


What's the adjacency Matrix?

$$\begin{pmatrix} \textit{Abdu} & A \\ \textit{Beate} & B \\ \textit{Cynthia} & C \\ \textit{Deborah} & D \end{pmatrix} \Rightarrow \begin{array}{c} \begin{pmatrix} A & B & C & D \end{pmatrix} \\ \Downarrow \\ \begin{pmatrix} \quad & \quad & \quad & \quad \\ & & & \\ & & & \\ & & & \end{pmatrix} \end{array}$$

The *networkx* package in python

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



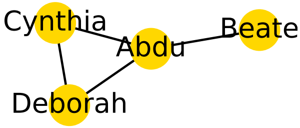
What's the adjacency Matrix?

$$\begin{array}{c} \\ \end{array} \qquad \begin{array}{c} (A \quad B \quad C \quad D) \\ \Downarrow \end{array}$$

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

The *networkx* package in python

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

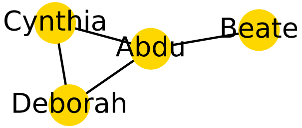


What's the adjacency Matrix?

$$\begin{array}{c} \\ \\ \begin{pmatrix} Abdu\ A \\ Beate\ B \\ Cynthia\ C \\ Deborah\ D \end{pmatrix} \end{array} \Rightarrow \begin{array}{c} (A \quad B \quad C \quad D) \\ \Downarrow \\ \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & & \\ 1 & 0 & & \end{pmatrix} \end{array}$$

The *networkx* package in python

```python
import networkx as nx
G = nx.Graph()
G.add_node("Abdu")
...
G.add_edge("Abdu", "Cynthia")
...
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?

$$\begin{array}{c} \\ \\ \begin{pmatrix} \textit{Abdu}\ A \\ \textit{Beate}\ B \\ \textit{Cynthia}\ C \\ \textit{Deborah}\ D \end{pmatrix} \end{array} \Rightarrow \begin{array}{c} \begin{pmatrix} A & B & C & D \end{pmatrix} \\ \Downarrow \\ \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

The *networkx* package in python

```
import networkx as nx
G = nx.Graph()
G.add_node("Abdu")
...
G.add_edge("Abdu", "Cynthia")
...
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:

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

pos = nx.circular_layout(network)
nx.draw(network, pos, with_labels=True)

print("The adjacency matrix is: ", nx.adjacency_matrix(network))
print("The adjacency list for agent ‘ag’ is: ", network.adj[ag.id])
```

Concept 3 Summary

Social Network

- Social networks can be used to represent communication or (physical) interaction between agents
- Topology of the network matters (especially clustering and distribution of node degrees)
- The python package *networkx* is wonderful and simple to use:

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

What we will cover today.

- Agent-Based Model that simulates the spread of Covid-19 in a small society consisting of three age groups and realistic, stochastic infection dynamics. Understand how a model like this can be used to design policies.
- *Concept 1+2:* Parametrisation of ABMs: draw heterogeneous agent features from (discrete/continuous) distributions
- *Concept 3:* Interaction in ABMs: network of agents
- *Concept 4:* Event scheduling in ABMs

Design an ABM - II

Let's create an ABM (Slide 17/47 from Lecture 9 on ABM)

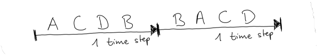
1. Design the data structure to store the attributes of the agents.
   class *agent()* with attributes *age*, *health_state*
2. ~~Design the data structure to store the states of the environment.~~
3. ~~Describe the rules for how the environment behaves on its own.~~
4. ~~Describe the rules for how agents interact with the environment.~~
5. Describe the rules for how agents behave on their own.
   when healthy: stay healthy
   when exposed: become infected
   when infected: go through the stages of Covid-19 following stochastic, age-dependent patterns.
6. Describe the rules for how agents interact with each other.
   agents are connected through a network. They meet 'physically' with their network neighbours, potentially infecting each other.

Concept 4: Event scheduling in ABM

Event scheduling decisions

	queue	sampling
asynch.		
synch.		

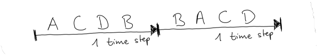
Event scheduling decisions

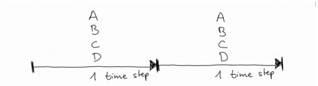
	queue	sampling
asynch.		
synch.		

Asynchronous queue

- What? each agent updates at the same frequency but after each other. Agents potentially observe what others did right before them.
- Examples: well moderated panel discussion, stock market $\rightarrow$ herding

Event scheduling decisions

<table>
<tr><th></th><th>queue</th><th>sampling</th></tr>
<tr><td>asynch.</td><td>


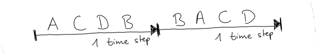
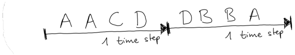
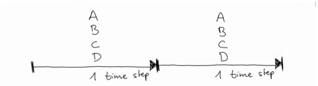
</td><td></td></tr>
<tr><td>synch.</td><td>


</td><td></td></tr>
</table>

Synchronous queue

- What? all agents update at the same time (without knowing what the others do at this point in time)
- Example: election, quiz night

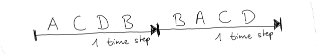
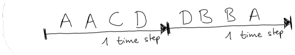
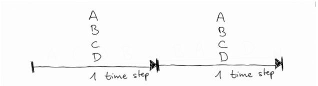
Event scheduling decisions

	queue	sampling
asynch.		
synch.		

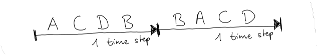
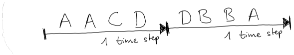
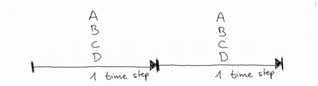
Asynchronous sampling

- What? agents update at different frequencies and after each other. Agents potentially observe what others did right before them.
- Example: social media posting, harvesting/hunting (fox-rabbit, Schelling)

Event scheduling decisions

	queue	sampling
asynch.		
synch.		—

Event scheduling decisions

	queue	sampling
asynch.		
synch.		—

For Covid-19 model: **asynchronous queue**.

Concept 4 Summary

Event scheduling

- Synchronous vs. asynchronous updating:
  (1) Do agents act simultaneously or after each other?
  (2) What do they know when they are updated?
- Queue vs. sampling:
  Do agents update at the same frequency or different frequencies?
- Sometimes, event scheduling can make a huge difference ($\rightarrow$ herding), most often it is irrelevant.

Concept 4 Summary

Event scheduling

- Synchronous vs. asynchronous updating:
  (1) Do agents act simultaneously or after each other?
  (2) What do they know when they are updated?
- Queue vs. sampling:
  Do agents update at the same frequency or different frequencies?
- Sometimes, event scheduling can make a huge difference ($\rightarrow$ herding), most often it is irrelevant.

- Attention: When we have different types of agents (e.g. fox/rabbit), think about relative update frequencies.
- Note: There are much more options (e.g. adaptive time: foxes will try to harvest more often when they are unsuccessful)
- Note: when agents die, the number of updates per time step change.

Update Function and Run Function

ABM base units

```
class agent:
    ...

def initialise():
    ...

def initialise_network():
    ...

def update():
    # (1) agents update health_status
    # (2) interactions using catch_virus, infect_others,
    ...

def catch_virus(ag, t_exposure):
    ...

def infect_others(ag, t_exposure):
    ...

# Run
initialise()
for t in range(T):
    update()
observe()
```

Update Function

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

```python


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

Update Function

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

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

Update Function

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

```


def update(t_now):
      queue = np.random.choice(agents, size=n_agents, replace=False)
      for ag in queue:
            if ag.health_state == "susceptible":
                  pass  # Do nothing
```

Update Function

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

```


def update(t_now):
    queue = np.random.choice(agents, size=n_agents, replace=False)
    for ag in queue:
        if ag.health_state == "susceptible":
            pass  # Do nothing
        if ag.health_state == "latent":
            # latent --> infectious?
            # do the symptoms start?
                # if symptomatic, does the agent die?
```

- Two days before the incubation time, switch from latent to infectious (pre-symptomatic/asympotmatic).
- Later, switch from pre-symptomatic to symptomatic when incubation period is over.
- After the infectious period, either recover or die.

Update Function

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

```python


def update(t_now):
    queue = np.random.choice(agents, size=n_agents, replace=False)
    for ag in queue:
        ...
        if "infectious" in ag.health_state:
            infect_others(ag, t_now)
```

```python
relative_infectiousness  = {"infectious_preSymptom": 0.5, "infectious_symptomatic":
    1, "infectious_asymptomatic": 0.2}

def infect_others(ag, t):
    p_i = ag.base_infectiousness * relative_infectiousness[ag.health_state]
    # Loop through contacts and potentially infect them
    linked_contacts = list(network.adj[ag.id])  # Indices of neighbours
    for c in linked_contacts:
        contact_person = agents[c]
        if contact_person.health_state == "susceptible"
            if np.random.random() < p_i:
                catch_virus(contact_person, t)
```

Run and Observe Function

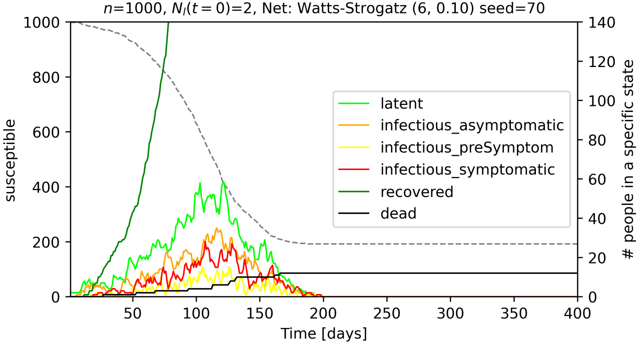
Goal: Perform simulation and track how many agents are in state "susceptible", "latent", ... over time.

```
initialise()
network = initialise_network(agents, k_ws=6, p_ws=0.1)

T_ARRAY = np.linspace(0, 400, 0.5)

results = np.empty([len(T_ARRAY), len(states)])

for n, t in enumerate(T_ARRAY):
      update(t)
      results[n, :] = observe(agents)
```

```

def observe(agents):
      states_of_agents = [ag.state for ag in agents]
      N_s = states_of_agents.count("susceptible")
      N_l = states_of_agents.count("latent")
      ...
      return np.array([N_s, N_l, N_ia, ...])
```

Run and Observe Function

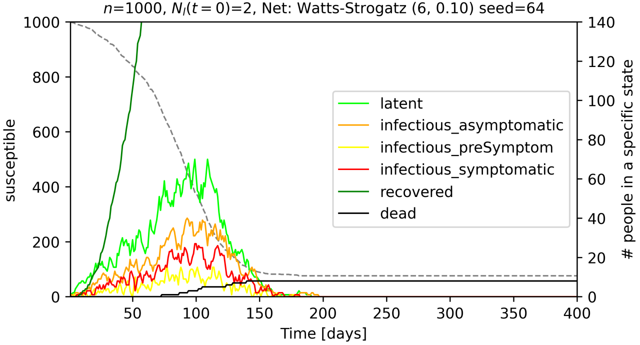
Goal: Perform simulation and track how many agents are in state "susceptible", "latent", ... over time.



Outbreak in a network with $k = 6$, $p = 0.2$, $n = 1000$ agents of which two are exposed at $t = 0$.

Run and Observe Function

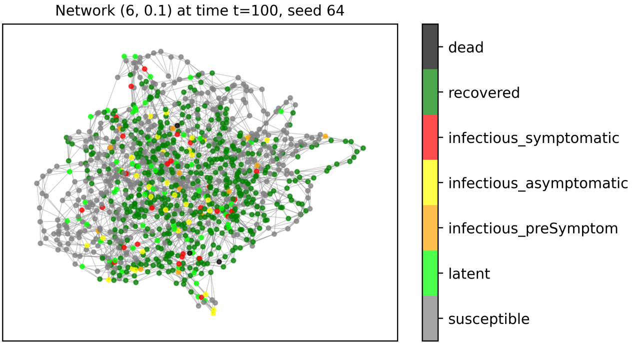
Goal: Perform simulation and track how many agents are in state "susceptible", "latent", ... over time.



Another outbreak in a network with $k = 6$, $p = 0.2$, $n = 1000$ agents of which two are exposed at $t = 0$.

Run and Observe Function

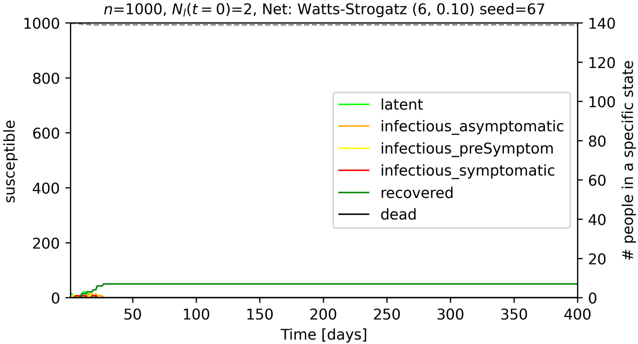
Goal: Perform simulation and track how many agents are in state "susceptible", "latent", ... over time.



Another outbreak in a network with $k = 6$, $p = 0.2$, $n = 1000$ agents of which two are exposed at $t = 0$.

Run and Observe Function

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



Same model configuration, but no outbreak! The model is stochastic!

The simulation depends on the occurrence of microscopic events.

Design an ABM - I

Let's think of an ABM (Slide 16/47 from Lecture 9 on ABM)

1. Specific problem to be solved by the ABM.
   How do a few infected agents affect a small, interconnected, simple society, which consists of agents in three age groups? 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' behaviour
5. Design of agent mutual interactions.
6. Availability of data.
   Data informs the parametrisation
7. Method of model validation.
   Compare with model and empirical reproductive number R_0

Validation

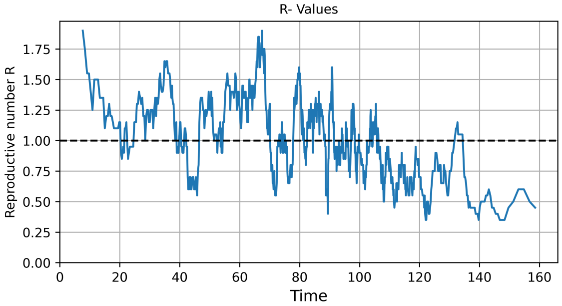
Steps to verify the results

1. Perform **ensemble runs**: Do many different stochastic simulation runs.
2. Get **aggregate indicators**: e.g. R_0 in the first 100 time steps, or rate of infected.
3. **Compare** model indicators with data: e.g. R_0 in the data vs. R_0 averaged over ensemble runs of the model.

Validation

Steps to verify the results

1. Perform **ensemble runs**: Do many different stochastic simulation runs.
2. Get **aggregate indicators**: e.g. R_0 in the first 100 time steps, or rate of infected.
3. **Compare** model indicators with data: e.g. R_0 in the data vs. R_0 averaged over ensemble runs of the model.



Policies

Policies

The major idea of our model was to design and test the impact of local policies.

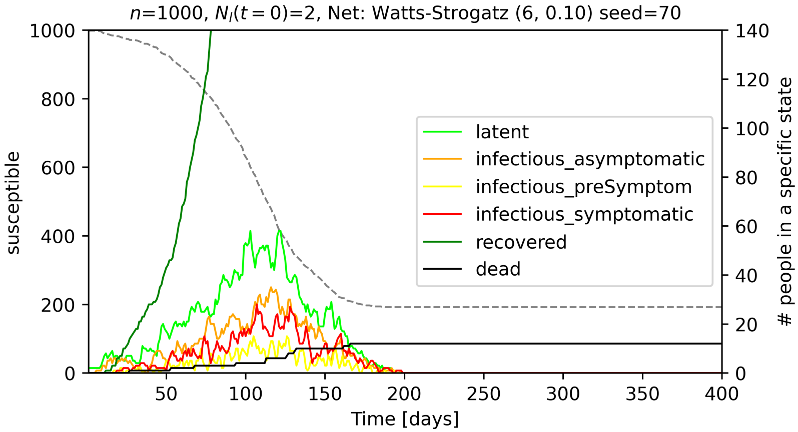
What happens, for example,

- if people reduce their contacts (due to social distancing policies)?
- if people keep their contacts within confined clusters (households, neighbours)?
- if people reduce their infectiousness by wearing a mask?

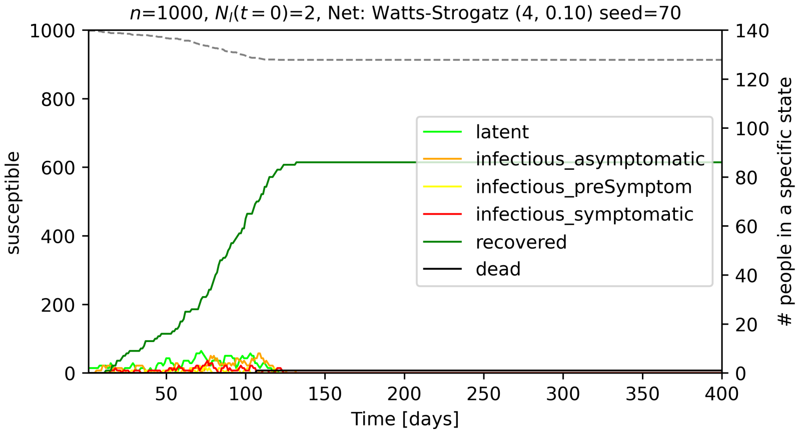
Parameters of the Model

Parameter	Description	current value
fraction_age_groups	Percentage of each age group	[0.2, 0.5, 0.3]
base_infectiousness_dist	Distribution of base infectiousness of the agents	beta(1, 3)
p_symptomatic	Probability to develop symptoms	[0.1, 0.5, 0.8]
incubation_period_dist	Distribution of the incubation time	gamma(5.807, 0.948)
case_fatality_rate	Case fatality ratio for each age group	[0.0001, 0.005, 0.05]
relative_infectiousness	rel. strength of infectiousness for a (pre-)symptomatic and asymptomatic infections	[0.5,1,0.2]
time_infectious_preSymptomstart	infectious days before symptom onset	2
time_infectious_postSymptomstart	infectious days after symptom onset	4
k_ws	Number of (nearest) neighbours in networks for nodes	6
p_ws	Probability of each link to be rewired	0.1
n_agents	Nr of agents	1000
n_infected_init	Nr of agents set to "exposed" at t=0	2

Policy: Decrease network node degree $k = 6 \rightarrow k = 4$



Policy: Decrease network node degree $k = 6 \to k = 4$



What we will cover today.

- Agent-Based Model that simulates the spread of Covid-19 in a small society consisting of three age groups and realistic, stochastic infection dynamics. Understand how a model like this can be used to design policies.
- *Concept 1+2:* Parametrisation of ABMs: draw heterogeneous agent features from (discrete/continuous) distributions
- *Concept 3:* Interaction in ABMs: network of agents
- *Concept 4:* Event scheduling in ABMs

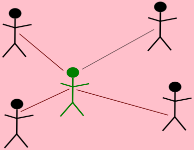
Potential Project?

- Basics:
  - Change the network topology and properties. What policy could this correspond to?
  - Try an entirely different network
  - Change distributions for *incubation period* or *base infectiousness* (e.g. decrease the incubation period (Omikron?) or make all agents equally infectious).
- Policies
  - Implement a soft isolation policy: when an agent becomes symptomatic, she/he quarantines and strongly reduces his/her contacts.
  - Such a policy may not take effect immediately. Policy makers implement it once they notice the local outbreak. Implement such a delay. How does the delay change the effectiveness of a policy?
  - Implement your own (time-dependent) policy strategy. This could include, e.g. (1) dynamic/adaptive changes in the network or (2) different/adaptive behaviour of each age-group.
  - Assume that a few agents are defectors. They do not adhere to your policy. What fraction of defectors makes your policy useless?

What we will cover today.

- Agent-Based Model that simulates the spread of Covid-19 in a small society consisting of three age groups and realistic, stochastic infection dynamics. Understand how a model like this can be used to design policies.
- *Concept 1+2:* Parametrisation of ABMs: draw heterogeneous agent features from (discrete/continuous) distributions
- *Concept 3:* Interaction in ABMs: network of agents
- *Concept 4:* Event scheduling in ABMs

ABM → Understanding → Recommendation



References

Bookstaber, R. (May 2017). *The End of Theory: Financial Crises, the Failure of Economics, and the Sweep of Human Interaction*. 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*. Report. DOI: 10.25561/77482. (Visited on 10/28/2020).

He, X. et al. (May 2020). "Temporal Dynamics in Viral Shedding and Transmissibility of COVID-19". In: *Nature Medicine* 26.5, pp. 672–675. ISSN: 1546-170X. DOI: 10.1038/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". In: *medRxiv*, p. 2020.10.13.20211813. ISSN: 2021-1813. DOI: 10.1101/2020.10.13.20211813. (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. (Visited on 10/28/2020).

Reiner, R. C. et al. (Oct. 2020). "Modeling COVID-19 Scenarios for the United States". In: *Nature Medicine*, pp. 1–12. ISSN: 1546-170X. DOI: 10.1038/s41591-020-1132-9. (Visited on 10/28/2020).