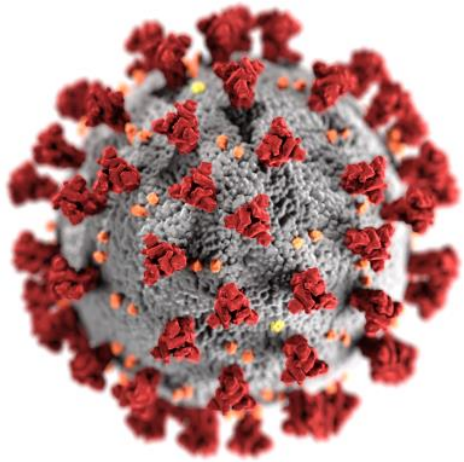
Agent-Based Model of Covid-19

Lecture 11

Peter Steiglechner

06 December 2022



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 appears in Wuhan, China (Dec 19)
- Spreads mainly airborne (aerosols, ...)
- Spreads fast in our societies:
  basic reproductive number R_0: 'expected number of cases directly generated by one case in a population where all individuals are susceptible to infection'
- Virus $\rightarrow$ disease Covid-19
- WHO declares pandemic on 31st Jan 2020
- Cases across the world within three months

Coronavirus and Covid-19

- SARS-CoV-2 appears in Wuhan, China (Dec 19)
- Spreads mainly airborne (aerosols, ...)
- Spreads fast in our societies:
  basic reproductive number R_0: 'expected number of cases directly generated by one case in a population where all individuals are susceptible to infection'
- Virus $\rightarrow$ disease Covid-19
- WHO declares pandemic on 31st Jan 2020
- Cases across the world within three months

- Early 2020: "How dangerous is Covid-19?"
  - High death rates (especially March 2020)
  - Fast spreading

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

$\rightarrow$ 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?

$\rightarrow$ A model to the rescue!?

Role of Modeling

- Initially: basic models to understand epidemiology
  e.g. project cases based on reproductive number R_0
  $\rightarrow$ 'flatten the curve'.

Role of Modeling

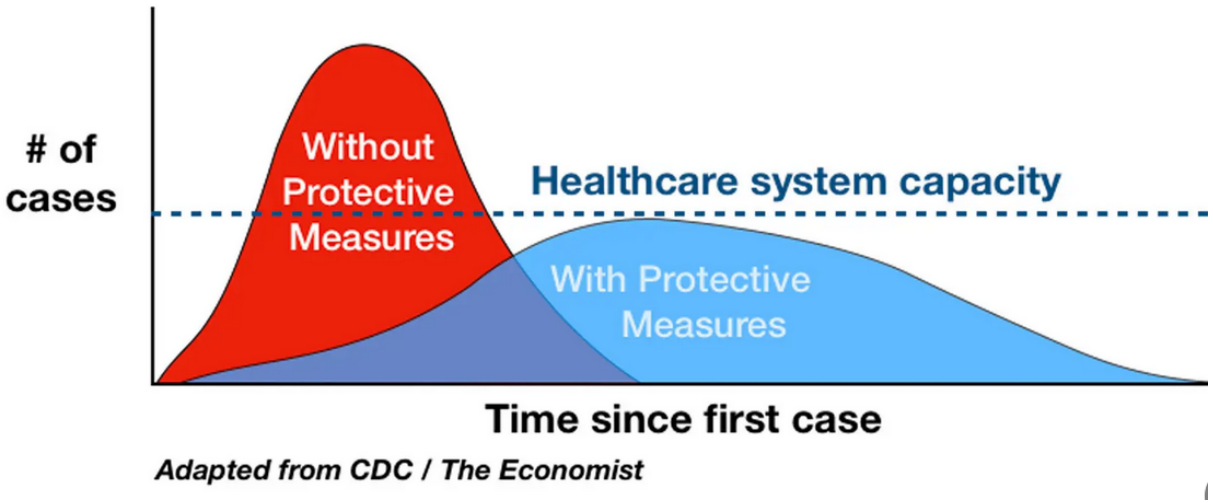
- Initially: basic models to understand epidemiology
  e.g. project cases based on reproductive number R_0
  $\rightarrow$ 'flatten the curve'.

NORK The New York Times

What's Going On in This Graph? | Flatten the Curve

How can protective measures like social distancing affect how the coronavirus outbreak impacts our society?

Give this article 1.2K



Adapted from CDC / The Economist

Role of Modeling

- Initially: basic models to understand epidemiology
  e.g. project cases based on reproductive number R_0
  $\rightarrow$ 'flatten the curve'.

- Later: more complex, realistic models
  e.g. age-structured, local resolution, realistic human behaviour
  $\rightarrow$ what measures work best to reduce pressure on the health system?

Role of Modeling

- Initially: basic models to understand epidemiology
  e.g. project cases based on reproductive number R_0
  $\rightarrow$ 'flatten the curve'.

- Later: more complex, realistic models
  e.g. age-structured, local resolution, realistic human behaviour
  $\rightarrow$ what measures work best to reduce pressure on the health system?

- Models to design and assess policies:
  - Suggest 'new' policies that are predicted to be (more) effective (Keeling et al., 2020)
  - Hindsight analysis: which policy strategy was efficient? (e.g. Reiner et al., 2020) (models answer what-if questions)

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


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. number of friends, degree of compliance with policies
- The modeled world is discrete and we need to simulate the actual sequence of events.
  If we had two entirely segregated societies, an outbreak in society A would not impact society B at all.
- Single, random and 'microscopic' events are crucial! Reality is path-dependent!
  E.g. carnival in Heinsberg (Germany). The pandemic wouldn't exist if "patient 0" in Wuhan had somehow avoided all contacts while infectious
- Uncertainty is crucial:
  We cannot fully predict the dynamics.
- The course of small-scale events/decisions changes the 'rules of the game' (non-ergodic system)
  Wuhan citizens react differently to a 2nd wave than Bremen citizens.

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 split into 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 split into 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: *health_state* changes over time 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: *health_state* changes over time following stochastic, age-dependent patterns.
6. Describe the rules for how agents interact with each other.
   agents are connected through a social network. They meet 'physically' with their neighbours and may infect 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

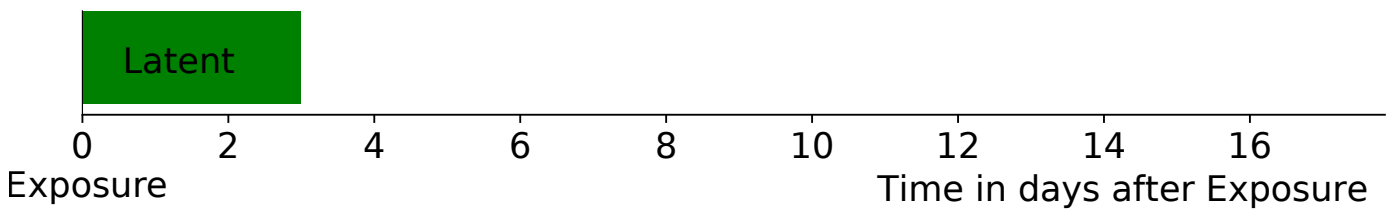
- Understand individual infection course
  what happens with infected people
- Understand transmissions
  how do people infect each other
- Understand social structure
  who infects who

Covid-19: Course of an infection (2020)

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

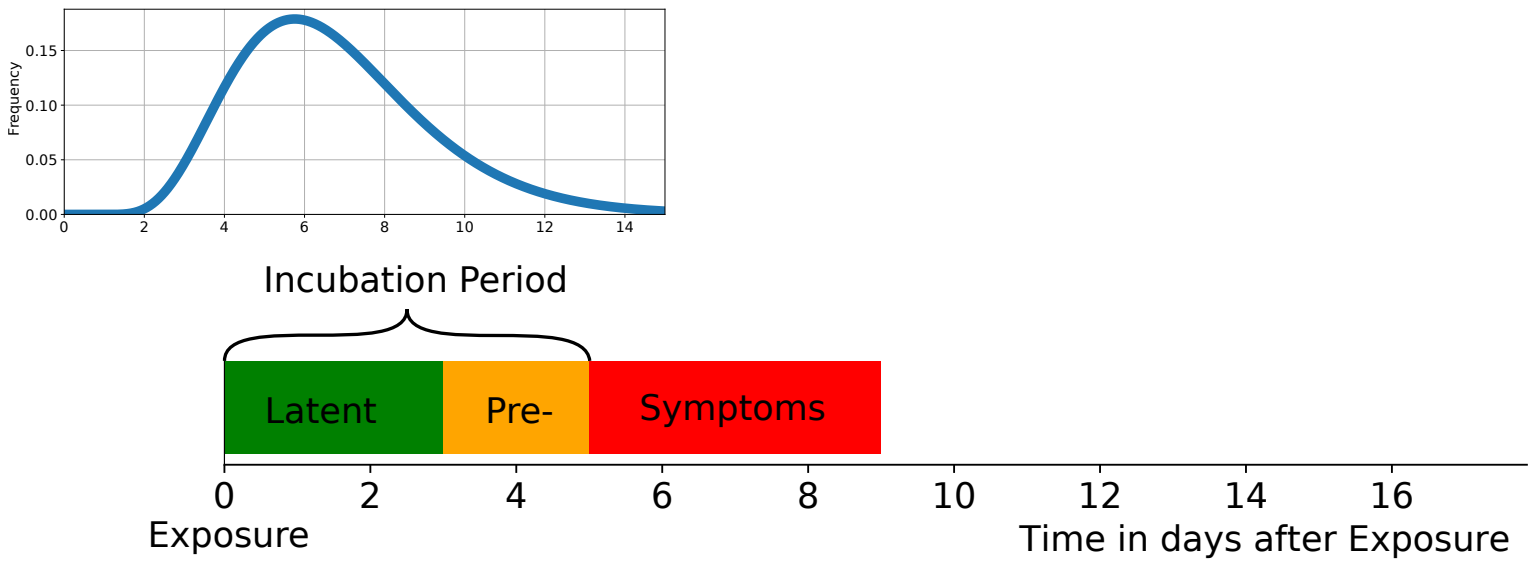
Covid-19: Course of an infection (2020)

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



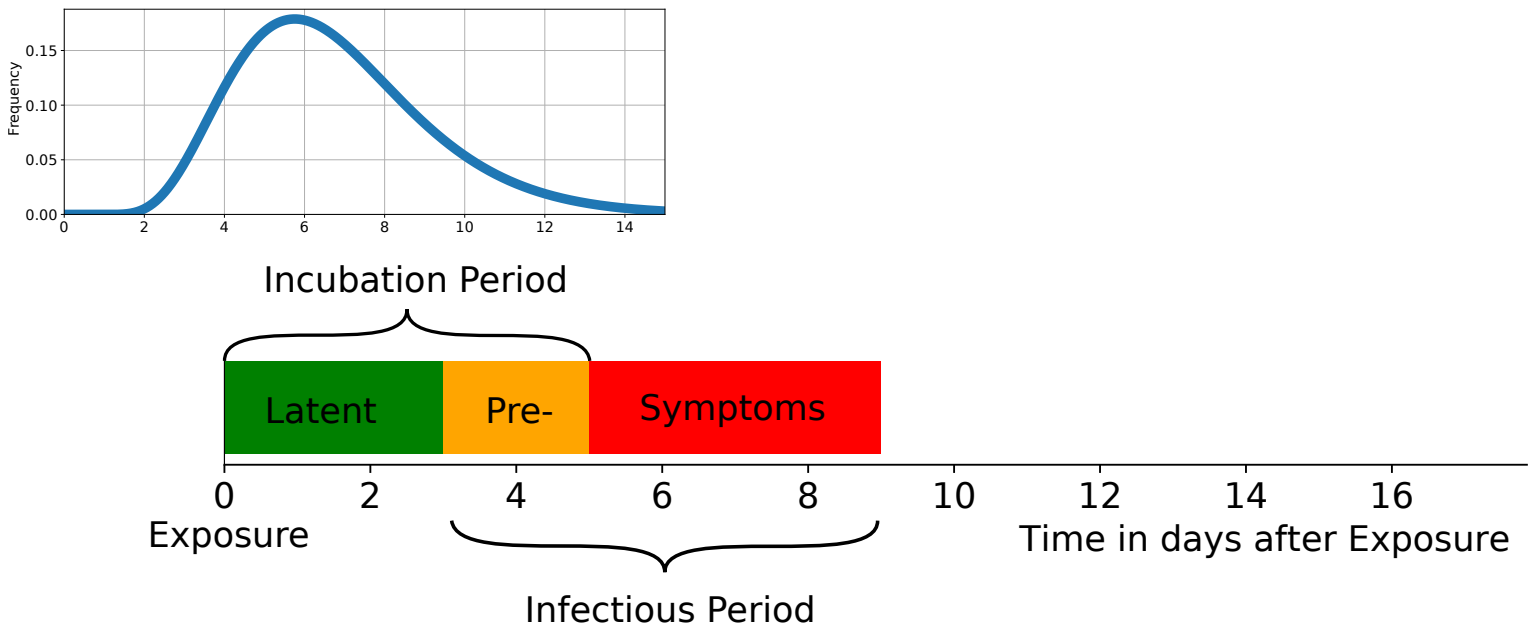
Covid-19: Course of an infection (2020)

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



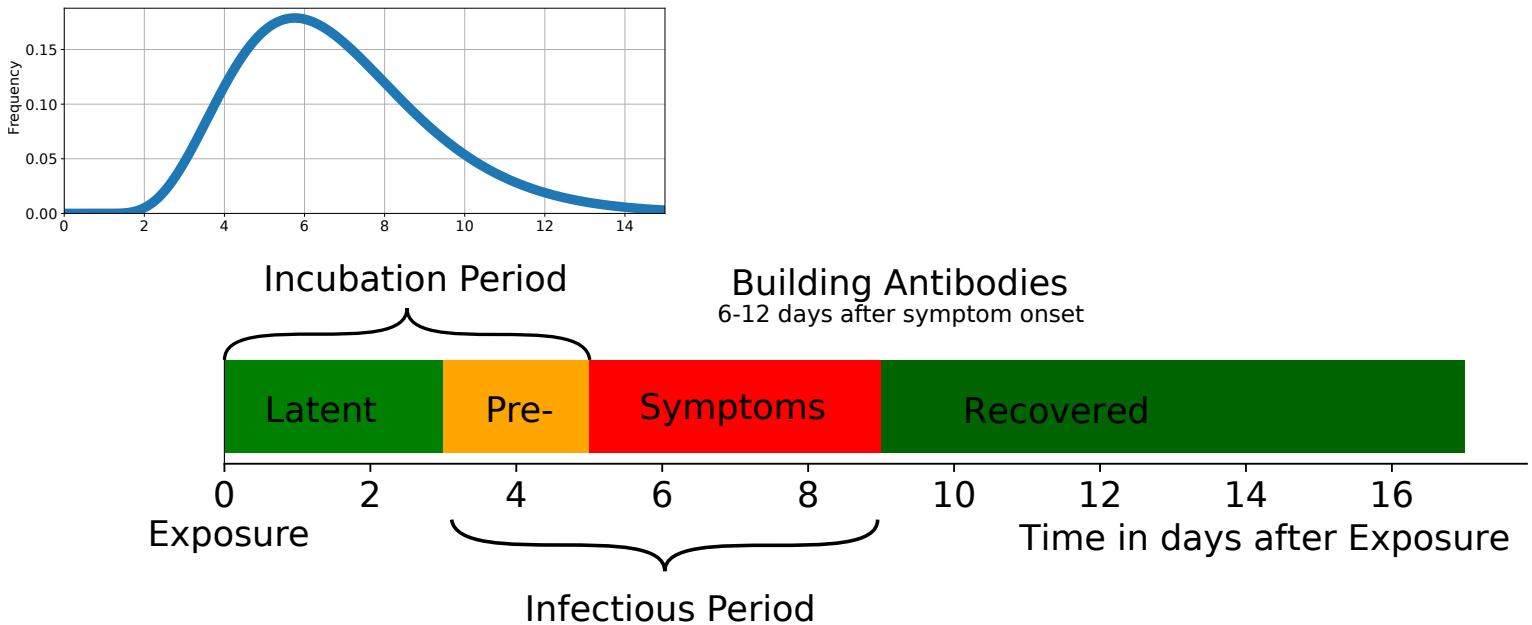
Covid-19: Course of an infection (2020)

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



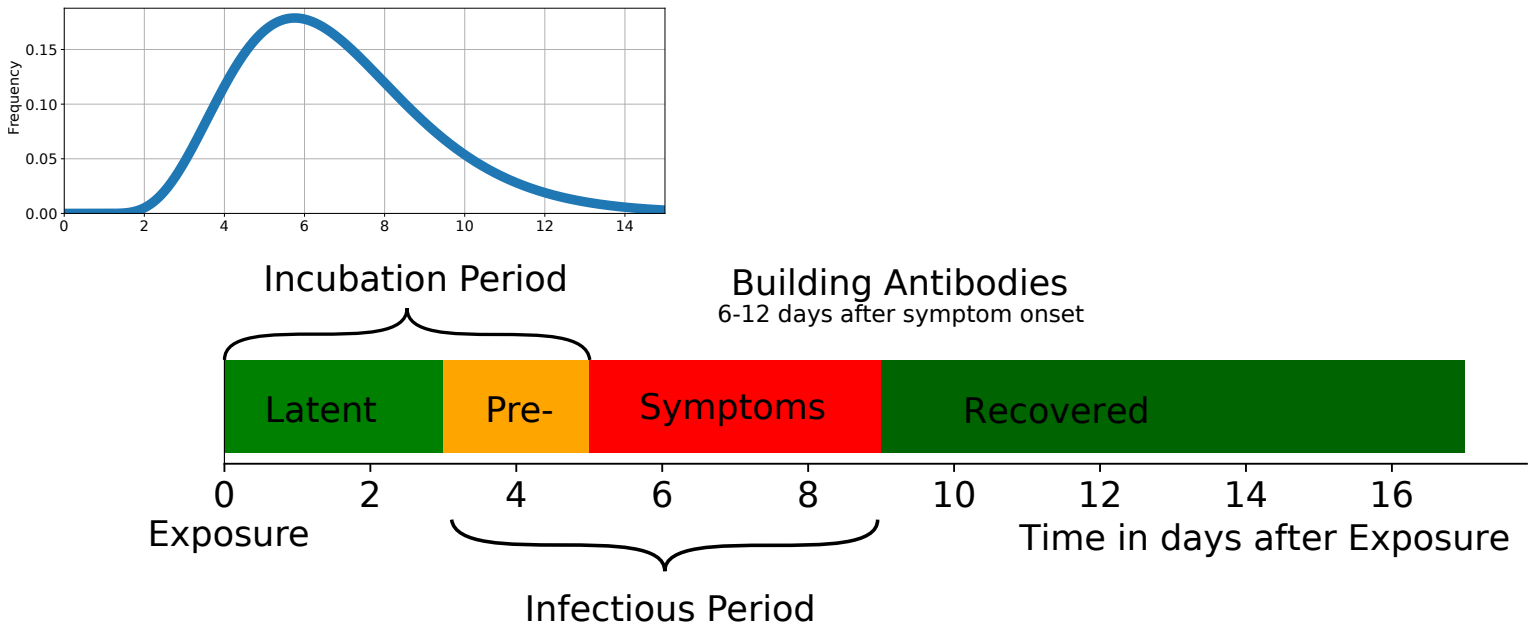
Covid-19: Course of an infection (2020)

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



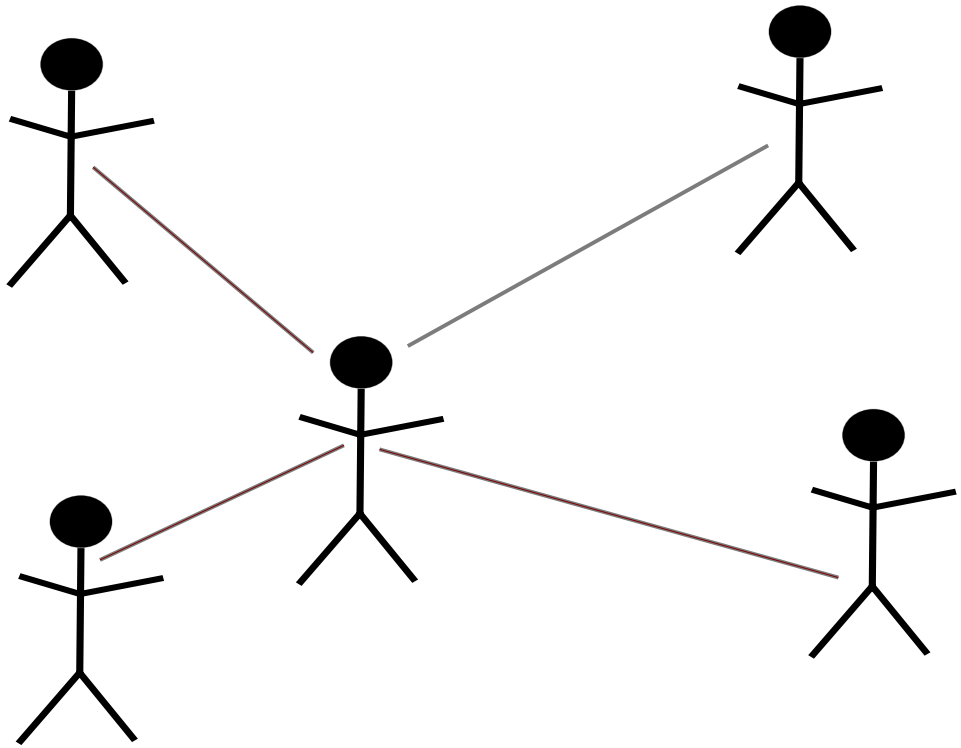
Covid-19: Course of an infection (2020)

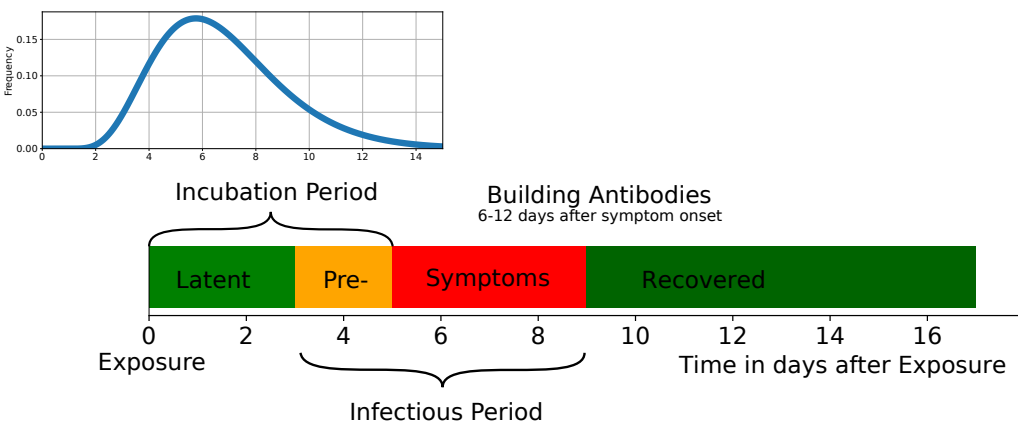
Latent period? Pre-Symptomatic and symptomatic, or asymptomatic infection?



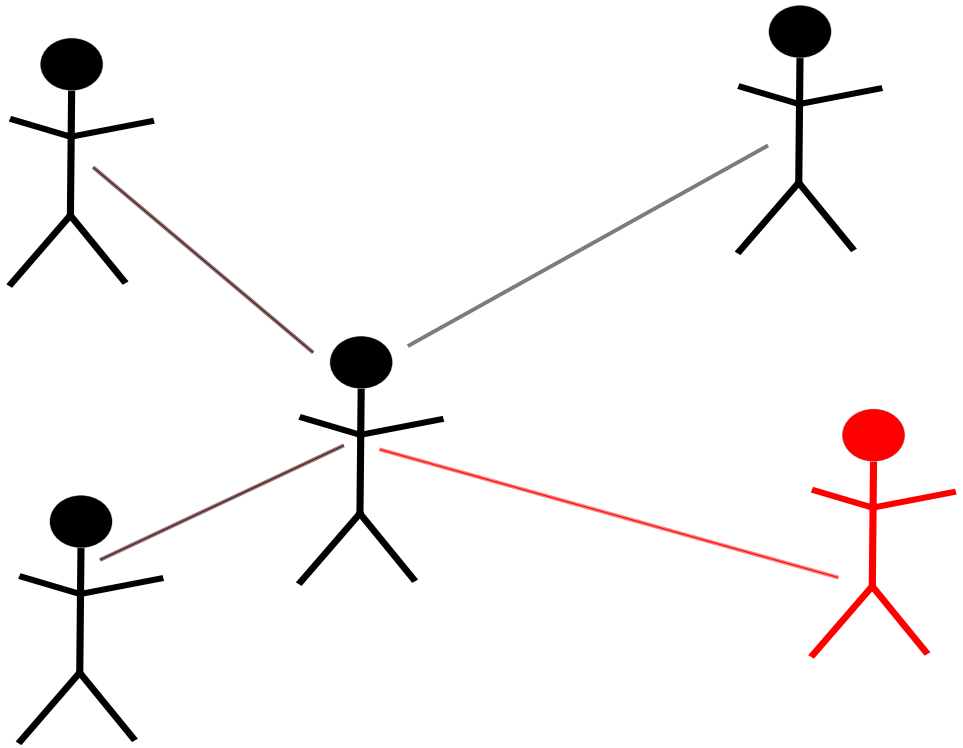
$\rightarrow$ infection course is age-dependent

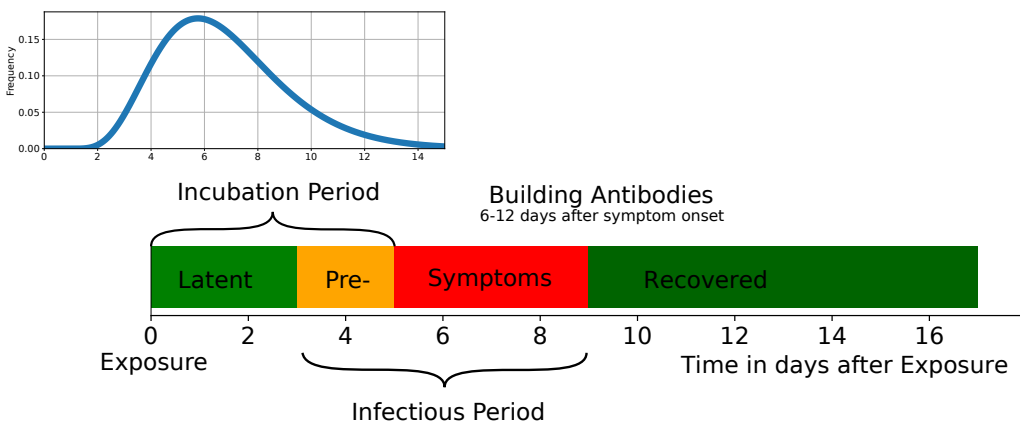
Model sketch





Model sketch



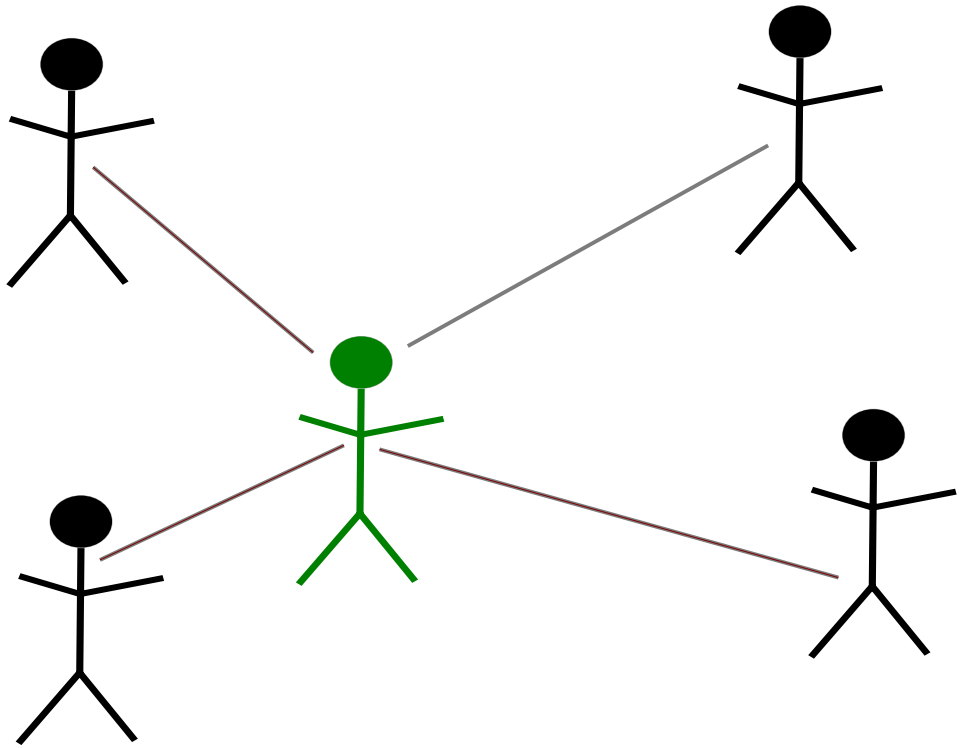


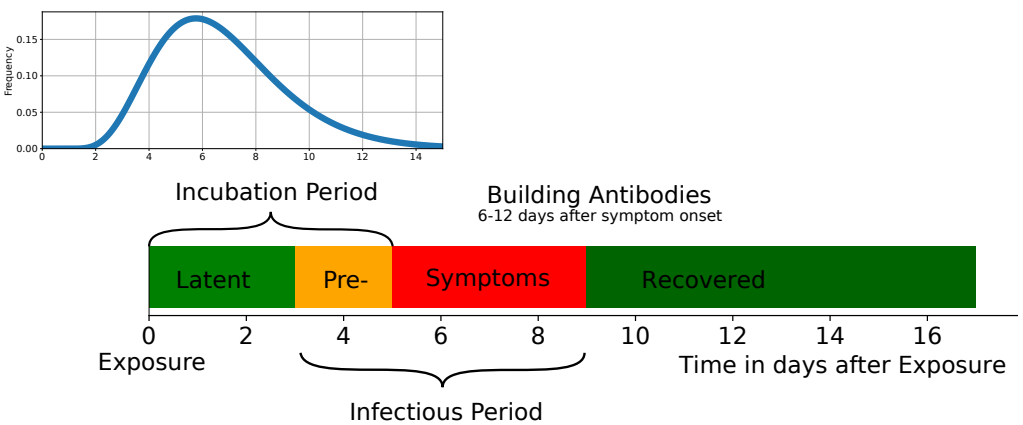
simple contagion

vs.

complex contagion

Model sketch



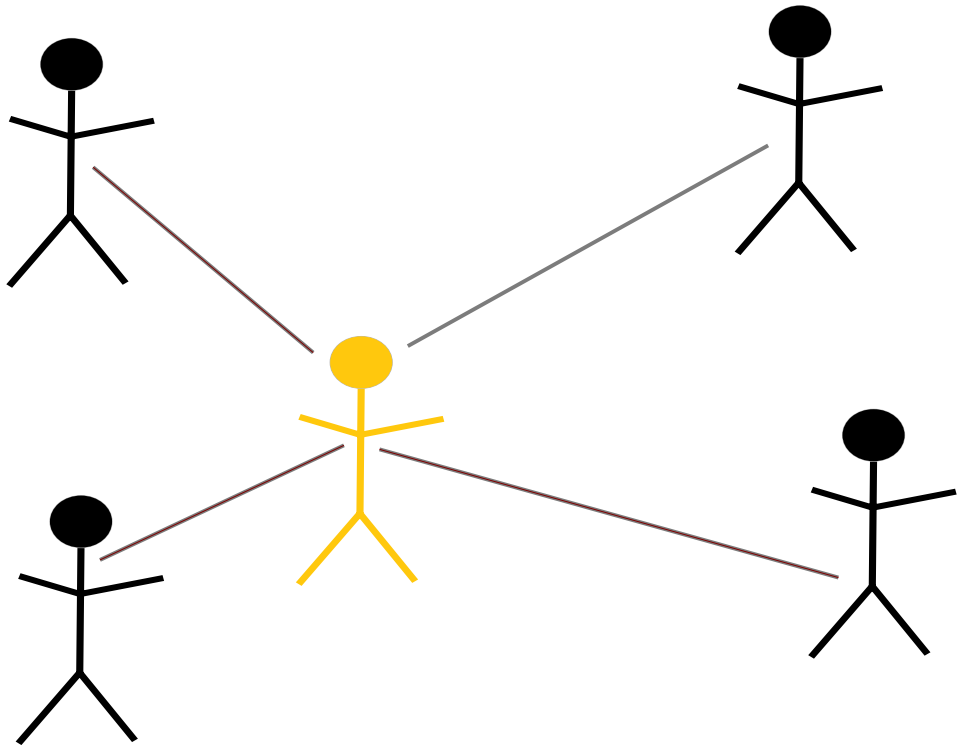


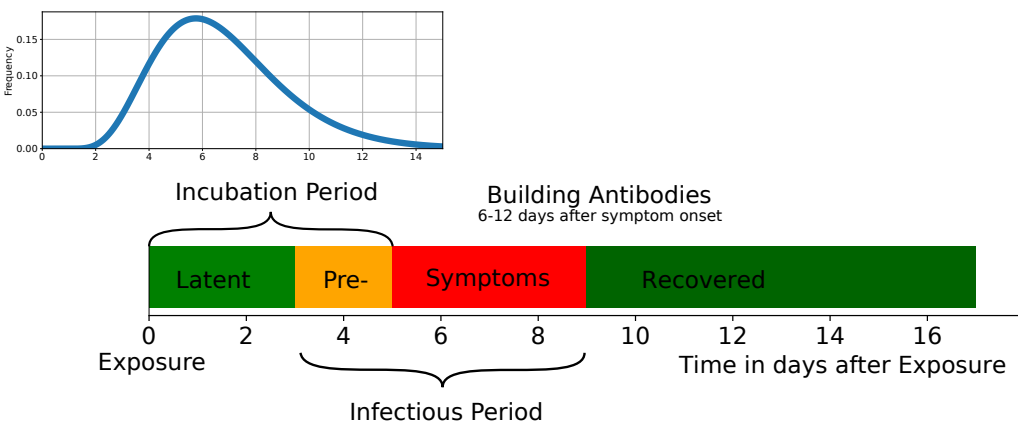
simple contagion

vs.

complex contagion

Model sketch



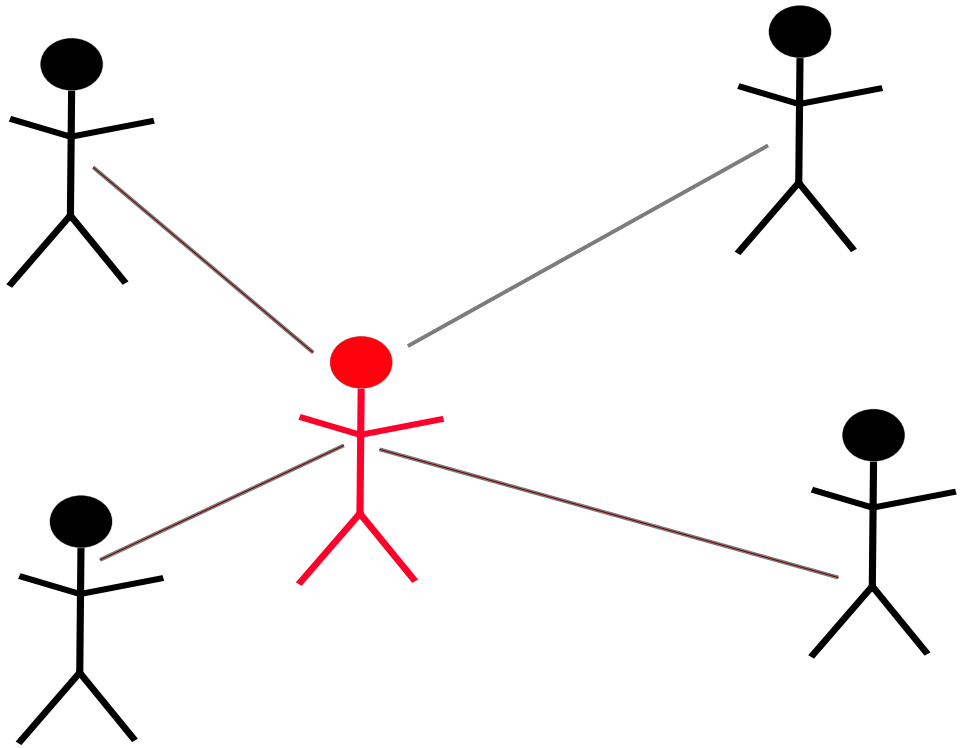


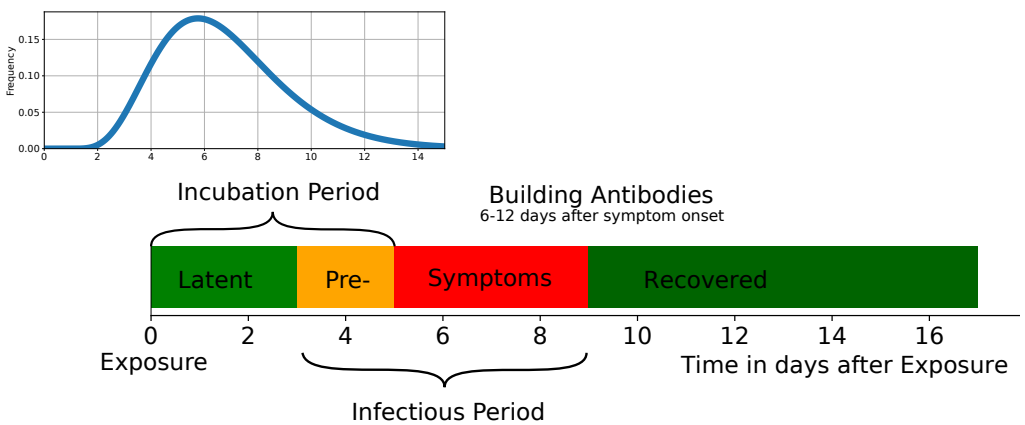
simple contagion

vs.

complex contagion

Model sketch



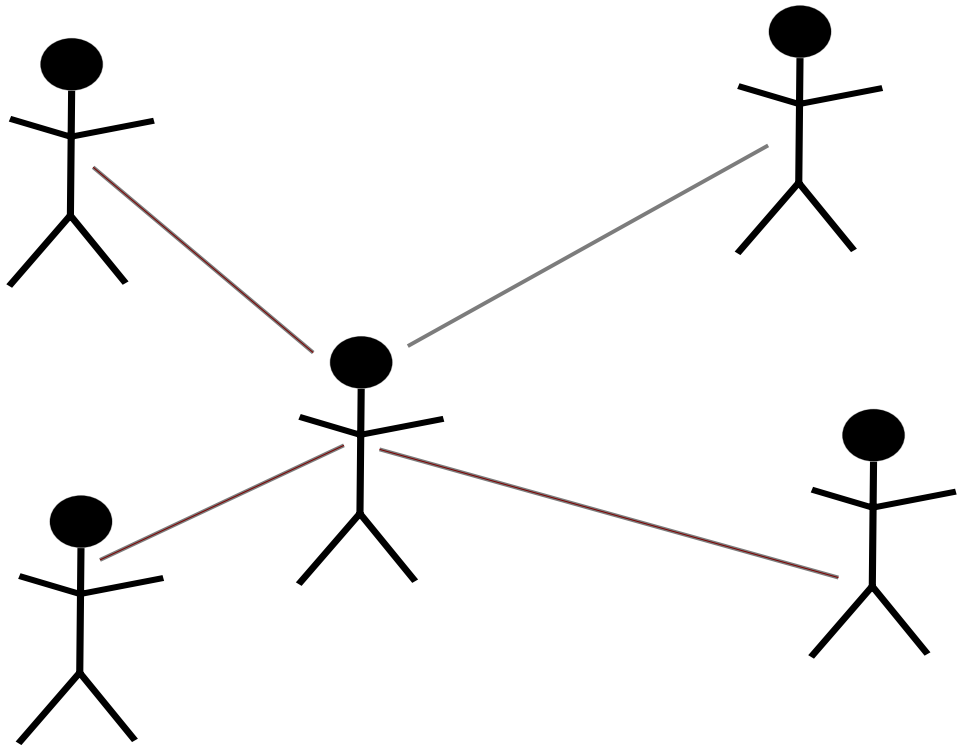


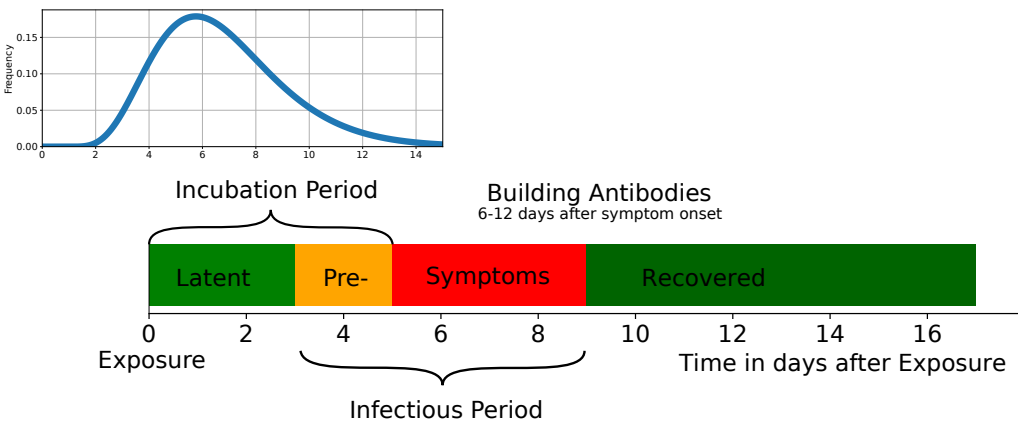
simple contagion

vs.

complex contagion

Model sketch





simple contagion

vs.

complex contagion

Agent and initialisation

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

An empty class.

Agent and initialisation

```
1  N_AGENTS = 1000
2
3
4
5  class Agent:
6        pass
7
8  def initialise():
9        global agents
10       agents = []
11
12       for i in range(N_AGENTS):
13             ag = Agent()
14             ag.id = i
15             ag.health_state = "susceptible"
16             ag.age = ???
17             agents.append(ag)
```

Create all agents.

Problem: want 20 % children, 50 % adults/low-risk, 30 % elderly/high-risk

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

```
1 for ag in agents:
2     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

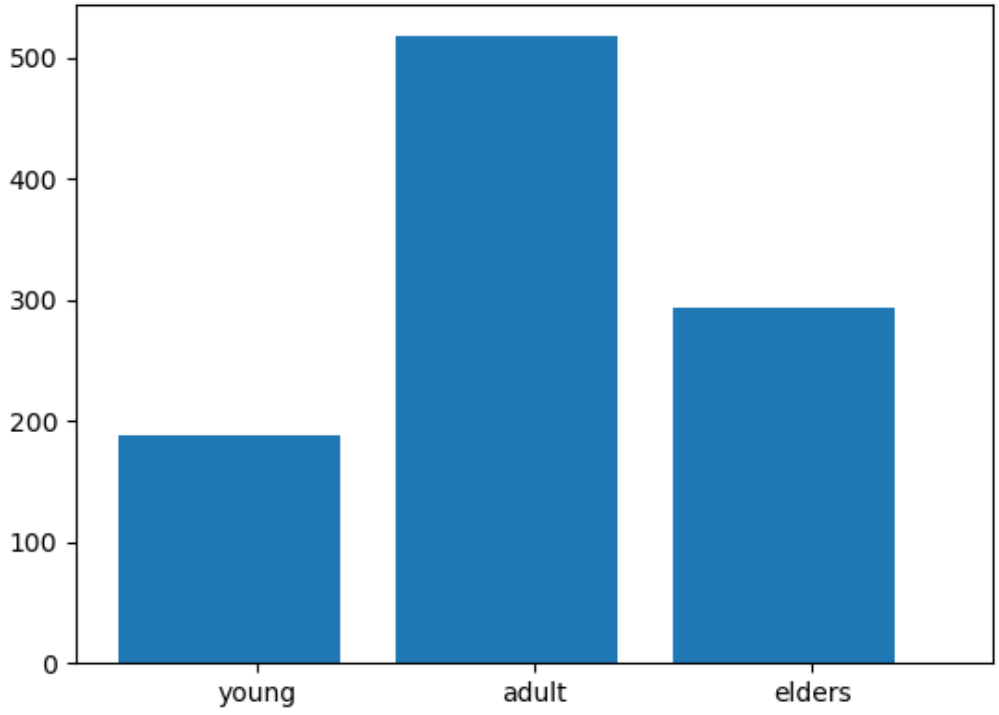
```
1  N_AGENTS = 1000
2  AGE_GROUPS = ["child", "adult", "elderly"]
3  FRACTION_AGE_GROUPS = [0.2, 0.5, 0.3]
4
5  class Agent:
6        pass
7
8  def initialise():
9        global agents
10       agents = []
11
12       for i in range(N_AGENTS):
13             ag = Agent()
14             ag.id = i
15             ag.health_state = "susceptible"
16             ag.age = np.random.choice(AGE_GROUPS, p=FRACTION_AGE_GROUPS)
17             agents.append(ag)
```

Problem: want 20 % children, 50 % adults/low-risk, 30 % elderly/high-risk

✓ Draw from discrete probability distribution for each agent

Agent and initialisation

```
1  N_AGENTS = 1000
2  AGE_GROUPS = ["child", "adult", "elde
3  FRACTION_AGE_GROUPS = [0.2, 0.5, 0.3]
4
5  class Agent:
6        pass
7
8  def initialise():
9        global agents
10       agents = []
11
12       for i in range(N_AGENTS):
13             ag = Agent()
14             ag.id = i
15             ag.health_state = "suscep
16             ag.age = np.random.choice
17             agents.append(ag)
```



Problem: want 20 % children, 50 % adults/low-risk, 30 % elderly/high-risk

✓ Draw from discrete probability distribution for each agent

Agent and initialisation

```
1
2  ...
3  N_INFECTED_INIT = 2  # Number of agents in state "exposed" at t=0.
4
5  class Agent:
6        pass
7
8  def initialise():
9        global agents
10       agents = []
11
12       for i in range(N_AGENTS):
13             ag = Agent()
14             ag.id = i
15             ag.health_state = "susceptible"
16             ag.age = np.random.choice(AGE_GROUPS, FRAC_AGE_GROUPS)
17             agents.append(ag)
18
19       symptomatic_agents = np.random.choice(agents, size=N_INFECTED_INIT)
20       for ag in symptomatic_agents:
21             catch_virus(ag, t=0)
22
23       return
```

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

Concept 2: Drawing from continuous distributions

Concept 2: Continuous distributions

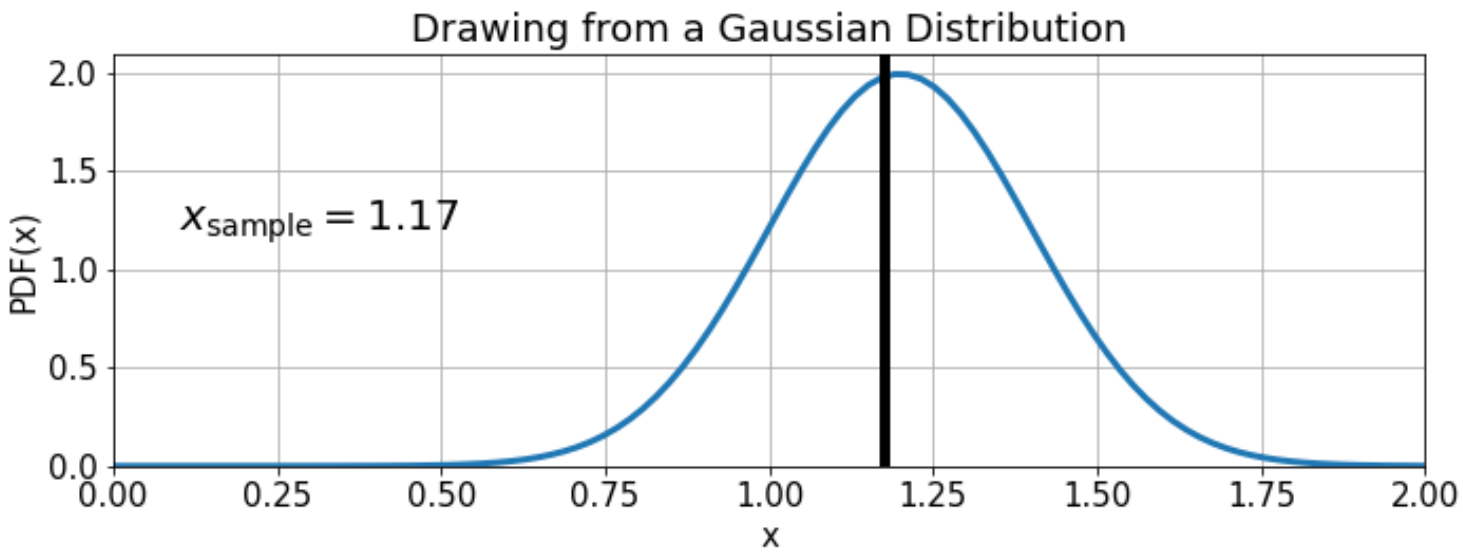
Random variable

- 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

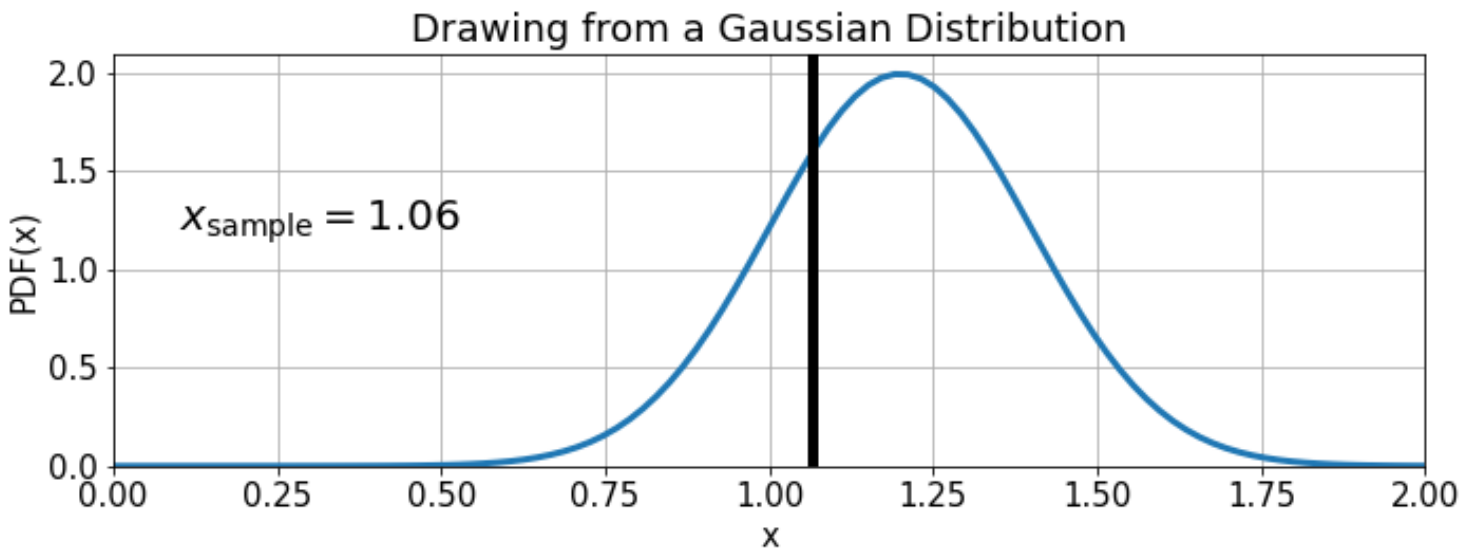
- Random variable x
- Probability density function PDF: $p(x)$
- Needs to integrate to one: $\int_{-\infty}^{\infty} p(x)\, dx = 1$
- Now, we draw samples from this distribution



Concept 2: Continuous distributions

Random variable

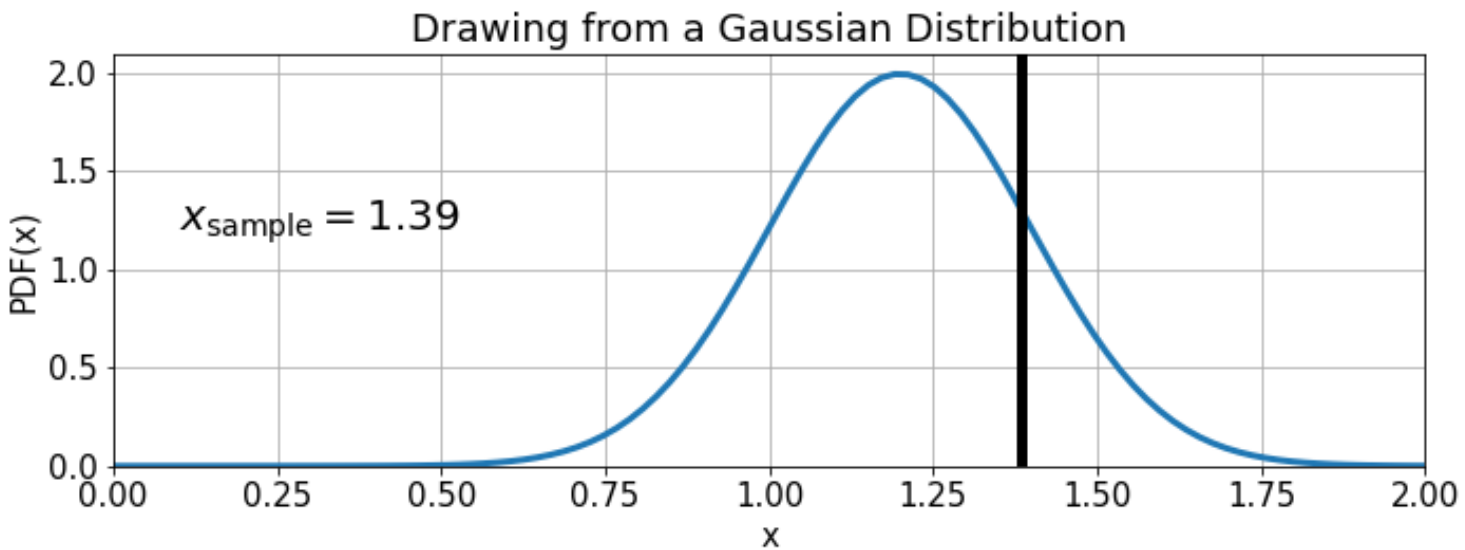
- Random variable x
- Probability density function PDF: $p(x)$
- Needs to integrate to one: $\int_{-\infty}^{\infty} p(x)\, dx = 1$
- Now, we draw samples from this distribution



Concept 2: Continuous distributions

Random variable

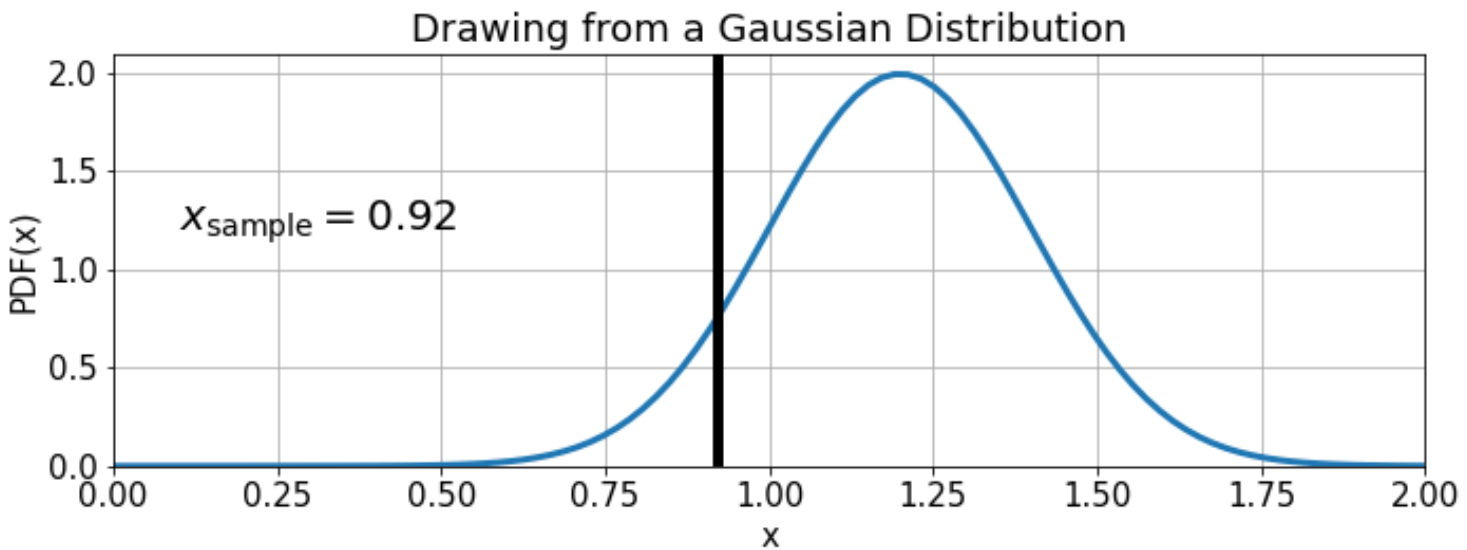
- Random variable x
- Probability density function PDF: $p(x)$
- Needs to integrate to one: $\int_{-\infty}^{\infty} p(x)\, dx = 1$
- Now, we draw samples from this distribution



Concept 2: Continuous distributions

Random variable

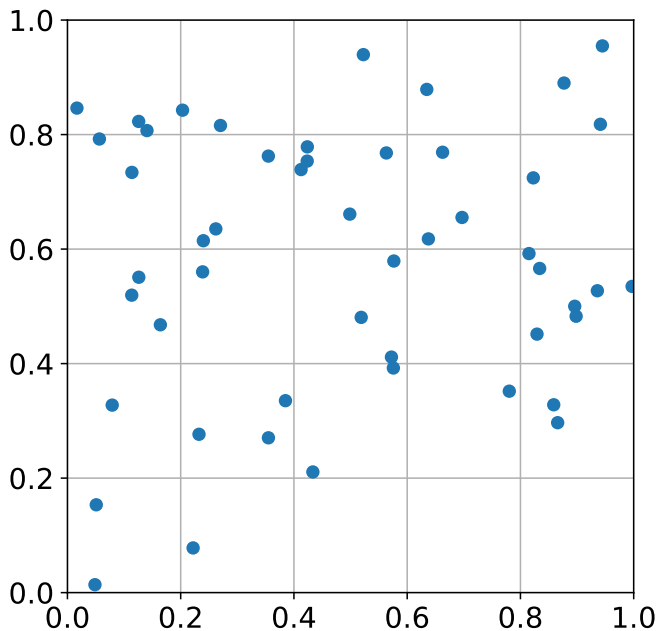
- Random variable x
- Probability density function PDF: $p(x)$
- Needs to integrate to one: $\int_{-\infty}^{\infty} p(x)\, dx = 1$
- Now, we draw samples from this distribution



Concept 2: Repetition

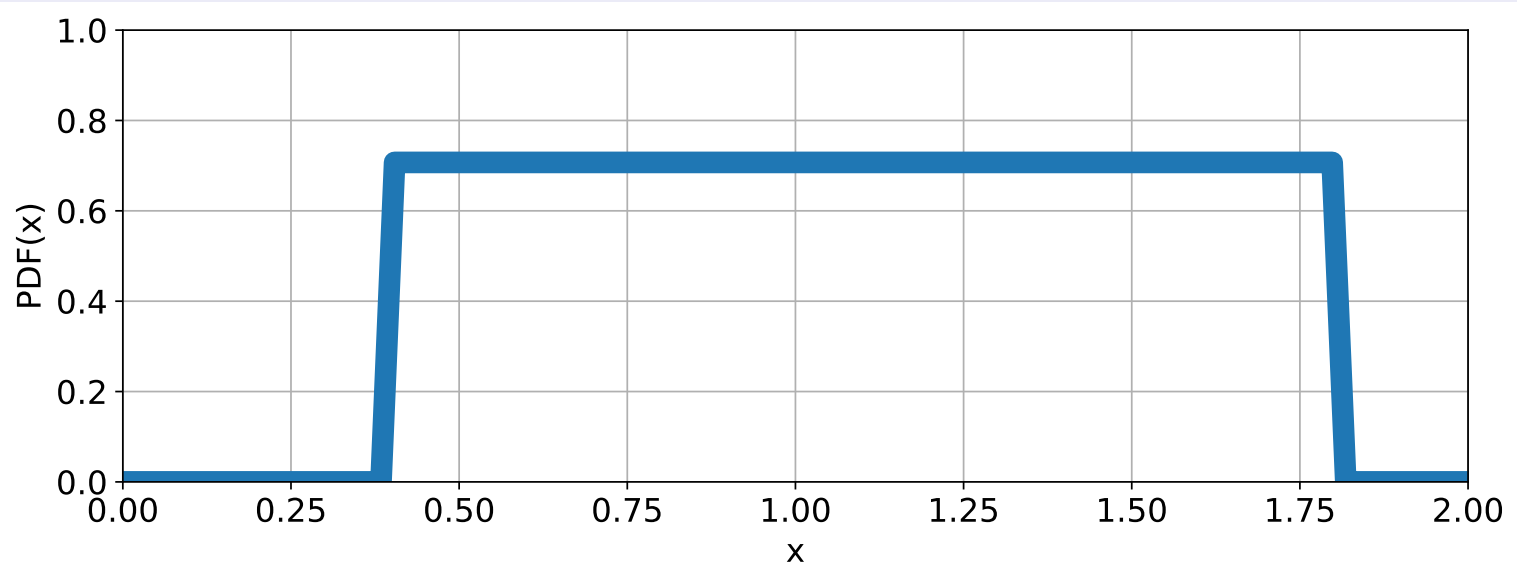
We have already applied 'Concept 2: Drawing from continuous distributions' in both previous ABMs in Lectures 9 and 10:

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

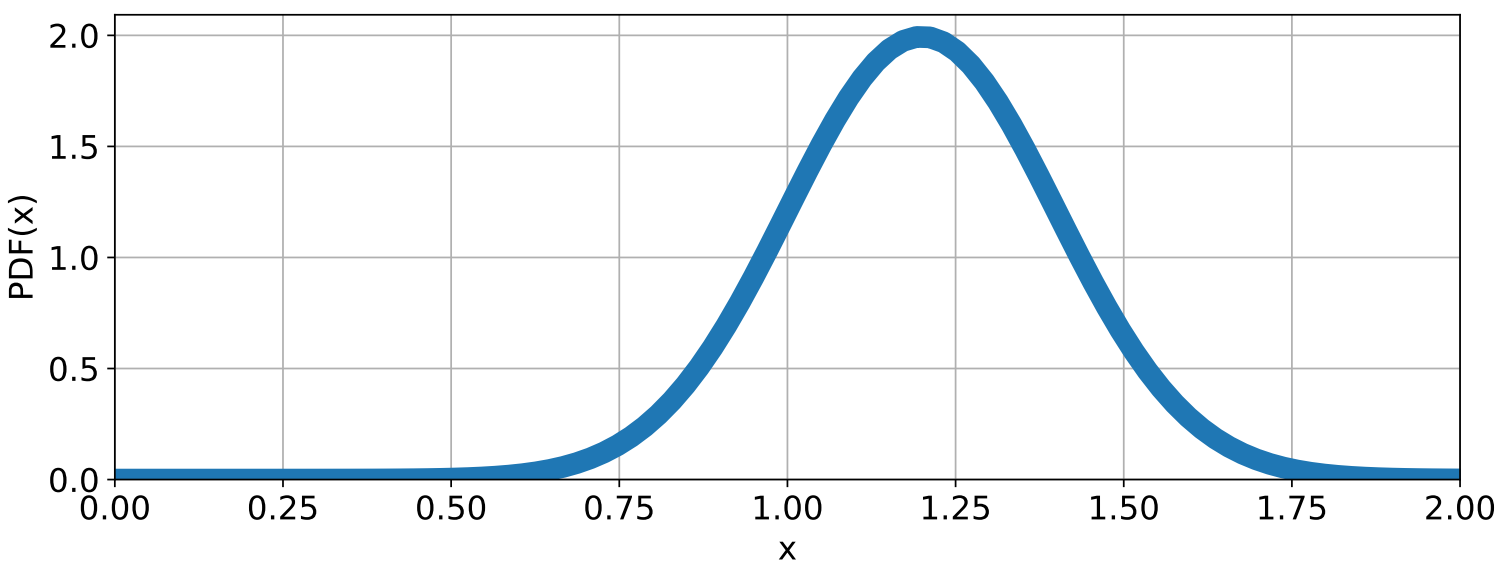
Uniform distribution



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

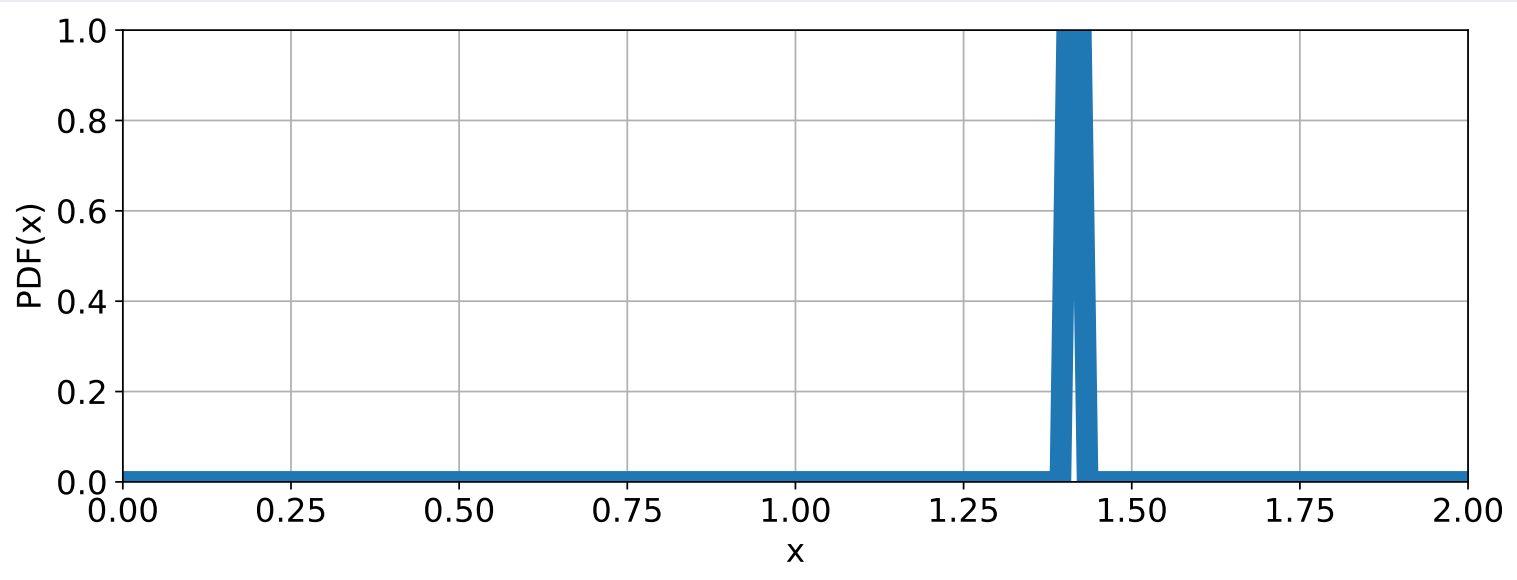
Gaussian or Normal distribution



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

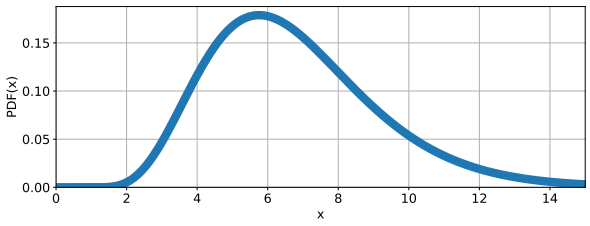
Delta distribution



What	continuous, inifinte/bounded range
PDF	$\delta(x-\tilde{x}) =: \delta_{\tilde{x}} = \begin{cases} \infty \text{ if } x = \tilde{x} \\ 0 \text{ else} \end{cases}$ $\int \delta(x-\tilde{x})\mathrm{d}x := 1$
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

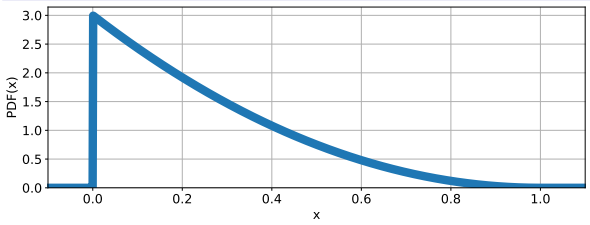
Gamma distribution



$\Gamma(5.9807,\ 0.948)$

- 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:

```
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' 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:

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

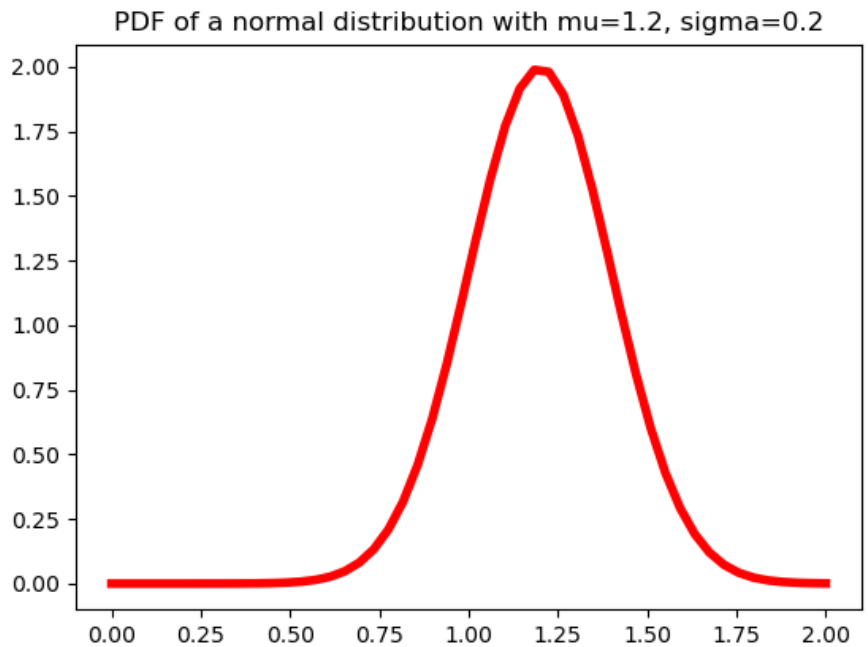
Concept 2: Python Package 'scipy.stats'

'scipy.stats'

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

PDF:

```
1 x = np.linspace(0,2)
2 plt.plot(x, some_normal_dist.pdf(x))
3 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:

```
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()
```

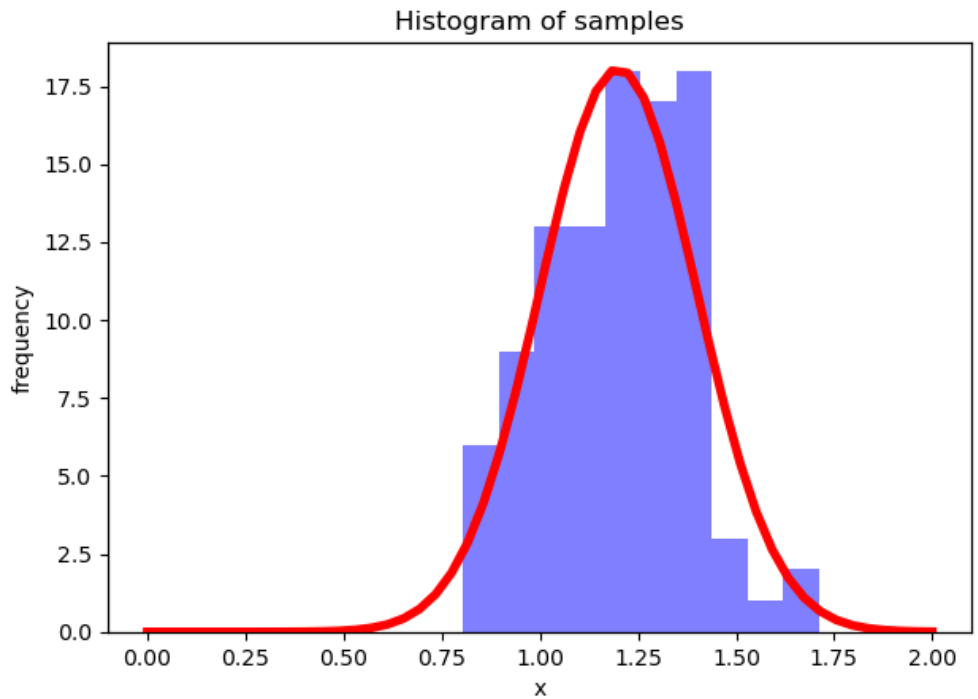
Concept 2: Python Package 'scipy.stats'

'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

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



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()
```

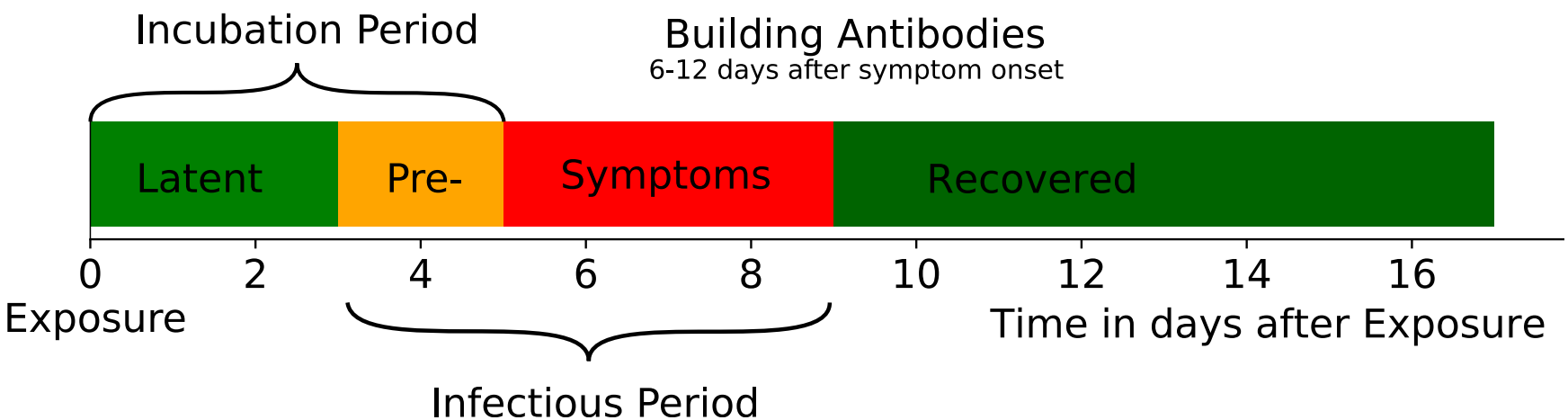
Back to the agent

So far: **general properties** for an agent.
Now: **infection-specific properties**.

Back to the agent

So far: **general properties** for an agent.
Now: **infection-specific properties**.

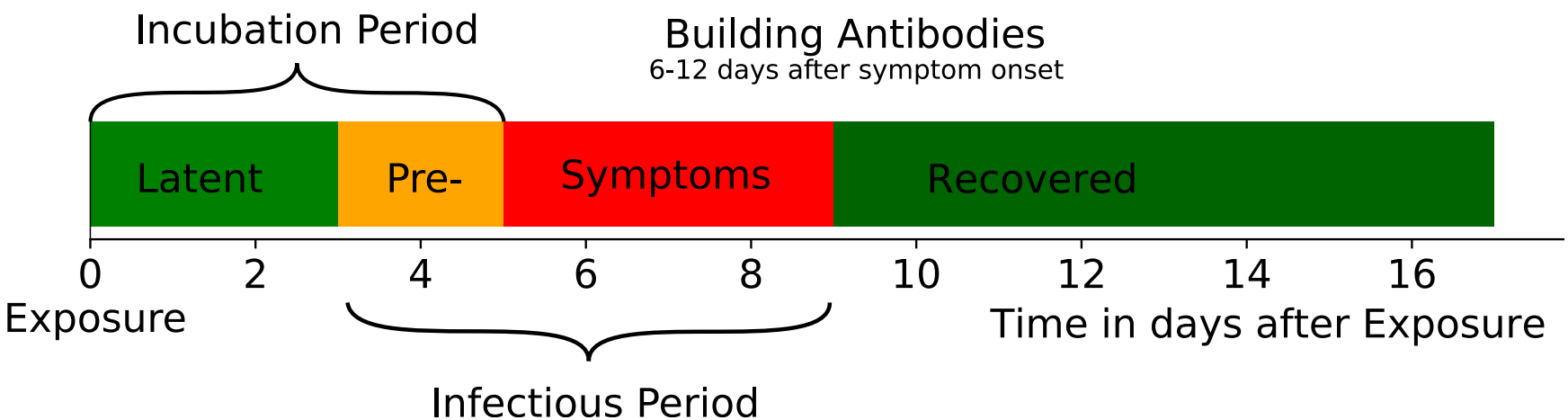
1. Time of exposure



Back to the agent

So far: **general properties** for an agent.
Now: **infection-specific properties**.

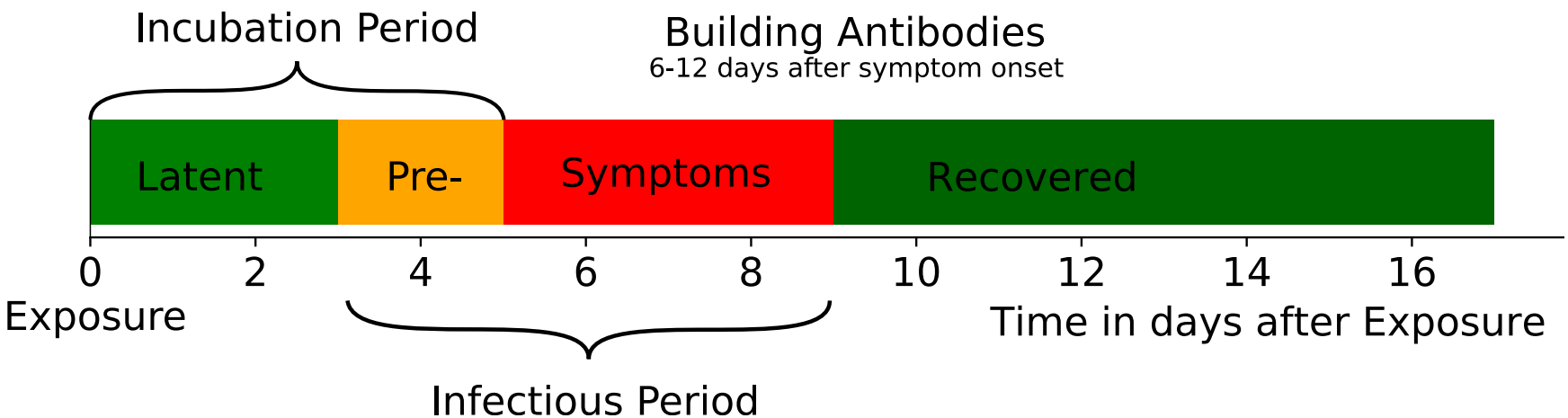
1. Time of exposure
2. Symptomatic or asymptomatic?



Back to the agent

So far: **general properties** for an agent.
Now: **infection-specific properties**.

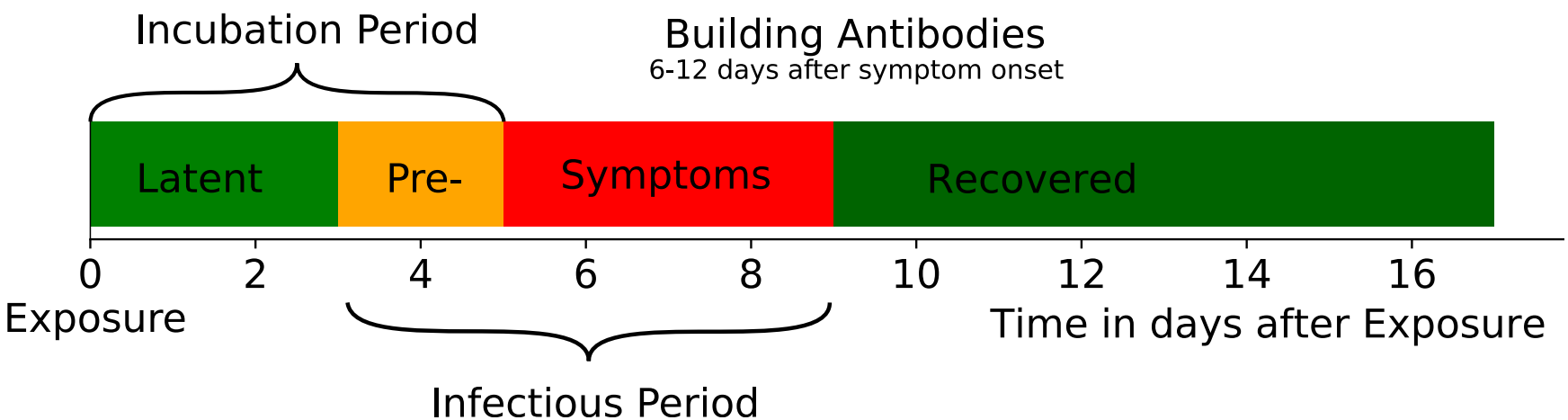
1. Time of exposure
2. Symptomatic or asymptomatic?
3. How long is the incubation period? (when do symptoms start?)



Back to the agent

So far: **general properties** for an agent.
Now: **infection-specific properties**.

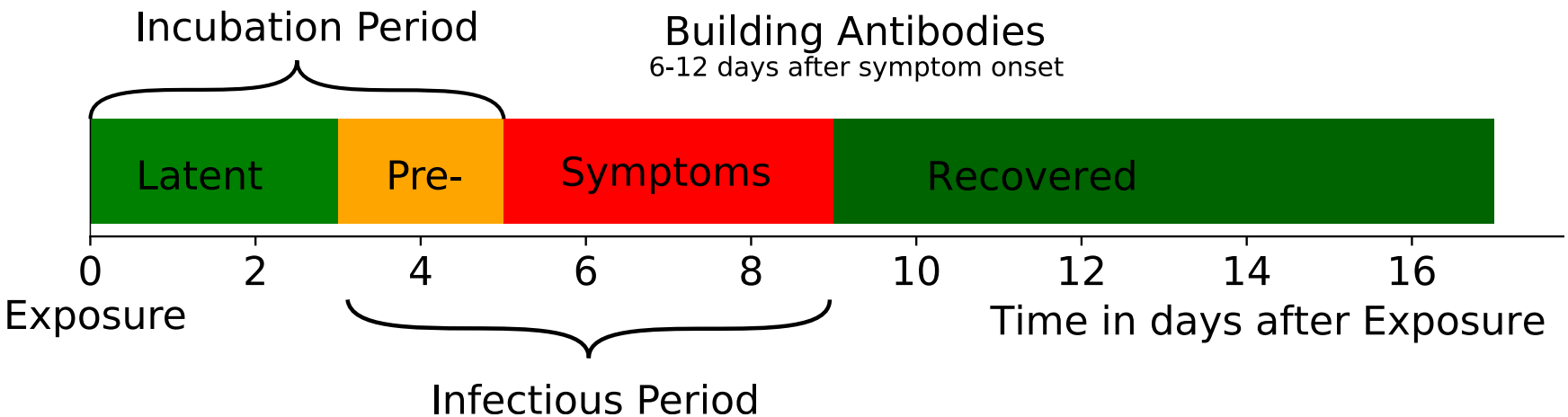
1. Time of exposure
2. Symptomatic or asymptomatic?
3. How long is the incubation period? (when do symptoms start?)
4. From when to when is the agent infectious?



Back to the agent

So far: **general properties** for an agent.
Now: **infection-specific properties**.

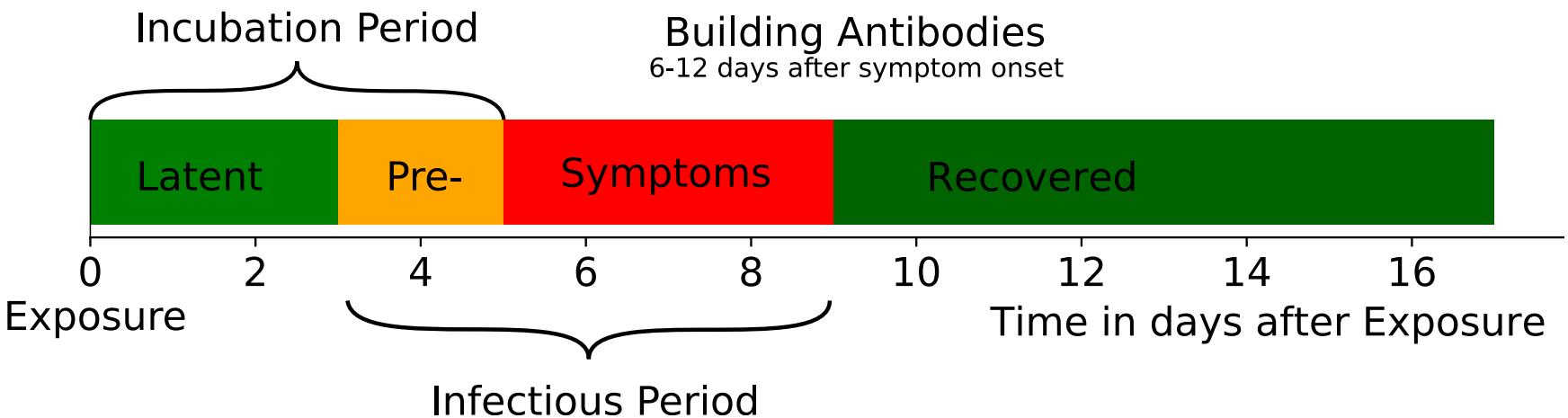
1. Time of exposure
2. Symptomatic or asymptomatic?
3. How long is the incubation period? (when do symptoms start?)
4. From when to when is the agent infectious?
5. Will the agent die from the infection?



Back to the agent

So far: **general properties** for an agent.
Now: **infection-specific properties**.

1. Time of exposure
2. Symptomatic or asymptomatic?
3. How long is the incubation period? (when do symptoms start?)
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.health_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 *health_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 = {"child": 0.1, "adult": 0.5, "elderly": 0.8}
2
3
4 def catch_virus(ag, t_exposure):
5     ...
6     p_s = P_SYMPTOMATIC[ag.age]
7     ag.symptomatic = True if np.random.random() < p_s else False
```

2:

determine whether the infection is

- symptomatic (*ag.symptomatic* = *True*) with probabiltiy *p_s* or
- asymptomatic (*ag.symptomatic* = *False*)

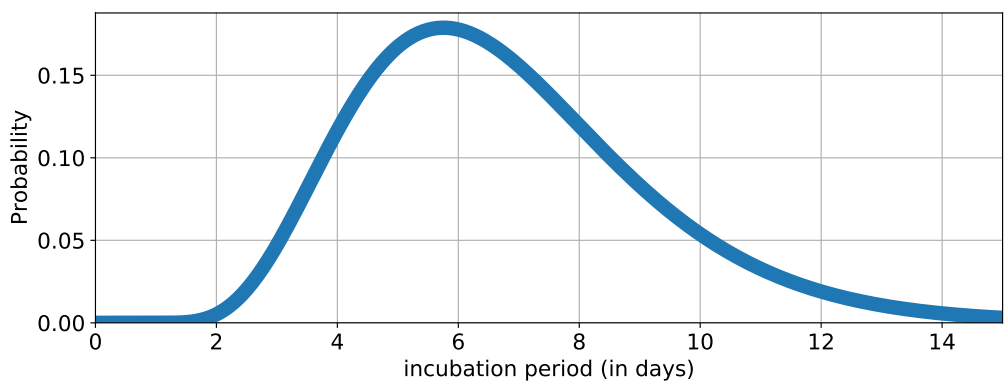
(☞ sample from two options with some probability).
The probability depends on age (vulnerability).

Catch the virus – Determine course of infection

```
1 INCUBATION_PERIOD_DIST = stats.gamma(5.807, 0.948)
2
3
4 def catch_virus(ag, t_exposure):
5     ...
6     incubation_period = INCUBATION_PERIOD.rvs()
```

3:

determine incubation period by drawing a sample from a gamma distribution inferred from 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 determine 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:

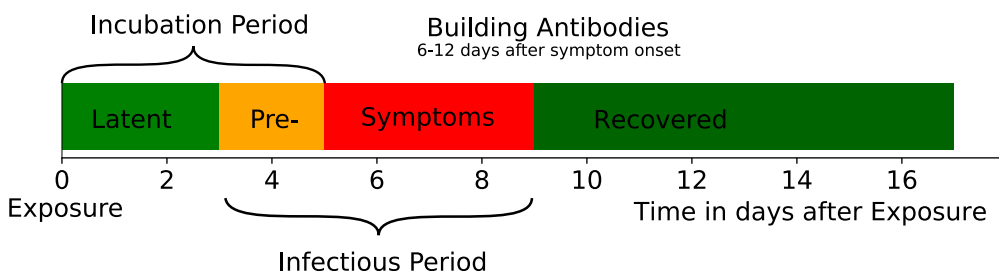
symptoms start after the incubation period, if *ag*'s infection is symptomatic. If *ag* is asymptomatic, simply ignore.

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 two days before and four days after the onset of symptoms (or, for asymptomatic cases, the theoretical onset).



Catch the virus – Determine course of infection

```
1  CFR = {"child": 0.00001, "adult": 0.005, "elderly": 0.05}
2
3
4  def catch_virus(ag, t_exposure):
5        ...
6        if ag.symptomatic:
7              # Symptomatic Case, might die
8              p_d = CFR[ag.age]
9              ag.fatal_outcome = True if np.random.random() < p_d else False
10       else:
11             # Asymptomatic Case, can not die
12             ag.fatal_outcome = False
```

5:

if symptomatic, agent *ag* might die (*ag.fatal_outcome* = *True*).
Here, asymptomatic agents do not die.
The probability of dying depends strongly on age.

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()      # max probability to infect others
```

6:

each agent has a *ag.base_infectiousness* ($\in [0, 1]$), a scale-factor determining how likely the agent will infect others when they interact.

beta distribution: a few agents are highly infectious, most are barely infectious.

Later: infectiousness depends on stage/type of infection (a- or pre-symptomatic people tend to be less infectious than symptomatic people) $\rightarrow$ defined later

Catch the virus – Determine course of infection

Summary

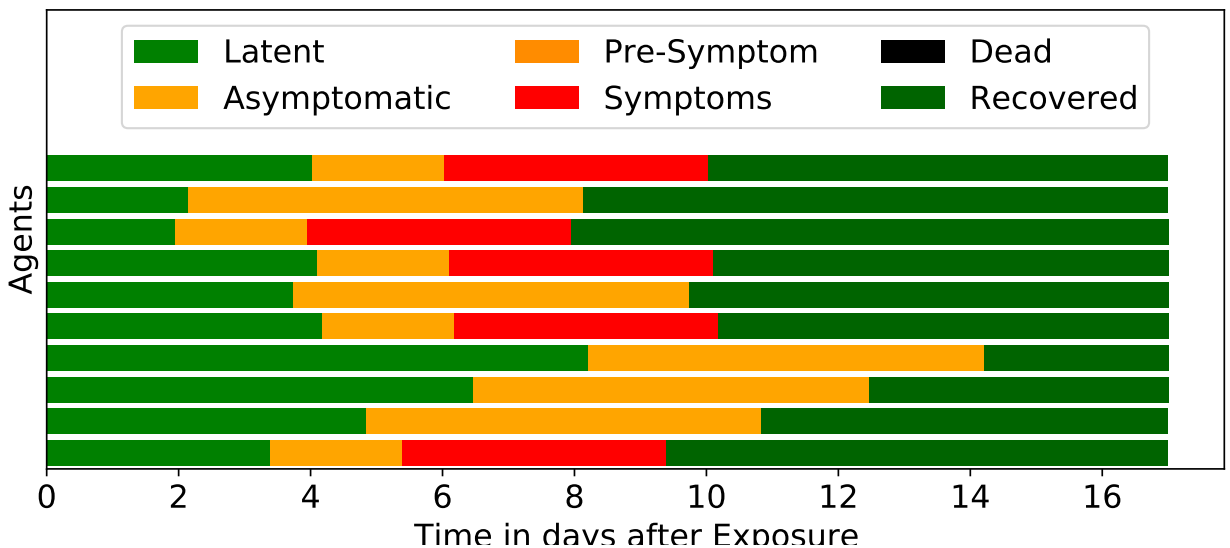
```
def catch_virus(ag, t_exposure):
    ag.health_state = "exposed"
    ag.t_e = t_exposure

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

    incubation_period = INCUBATION_PERIOD.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_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_I.rvs()   # * FACTOR_INFECTIOUSNESS

      return
```

A few examples of infection courses [Assignment]

1. Create an agent
2. The agent catches the virus (i.e. call *catch_virus*)
3. Plot the course of the infection.
4. Repeat 1-3

A few examples of infection courses [Assignment]

1. Create an agent
2. The agent catches the virus (i.e. call *catch_virus*)
3. Plot the course of the infection.
4. Repeat 1-3



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: *health_state* changes over time following stochastic, age-dependent patterns.
6. Describe the rules for how agents interact with each other.
   agents are connected through a social network. They meet 'physically' with their neighbours and may infect 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.

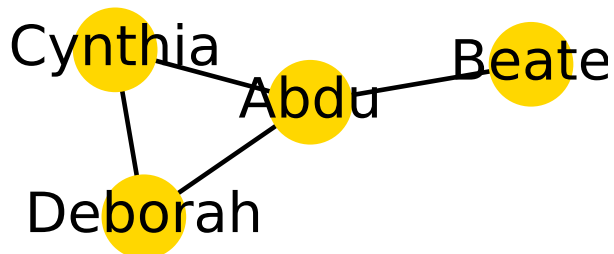
- Agent-Based Model that simulates the spread of Covid-19 in a small society consisting of three age groups and realistic, stochastic infection dynamics. Understand how a model like this can be used to design policies.
- *Concept 1+2:* Parametrisation of ABMs: draw heterogeneous agent features from (discrete/continuous)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

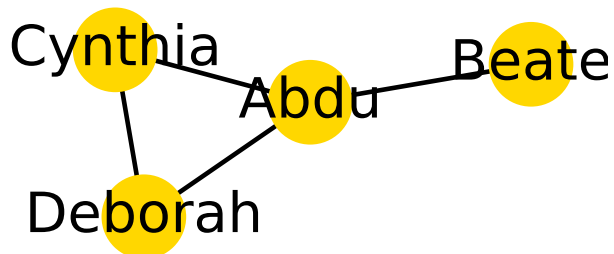
- Each node represents one agent
- Link/Edge between nodes means that agents can be in 'physical contact'



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

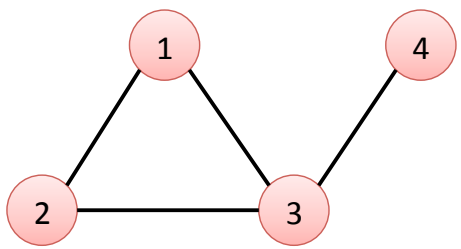


Social network – Basics

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

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

- 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.

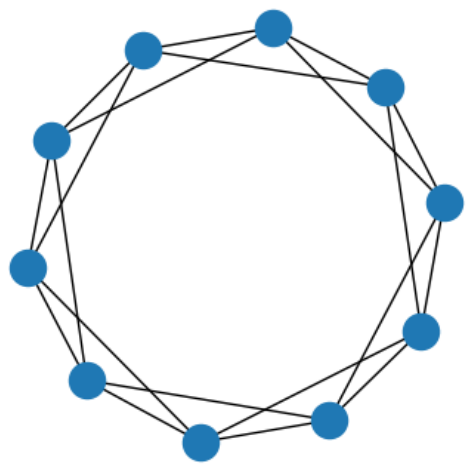
Social network - Topology

- 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

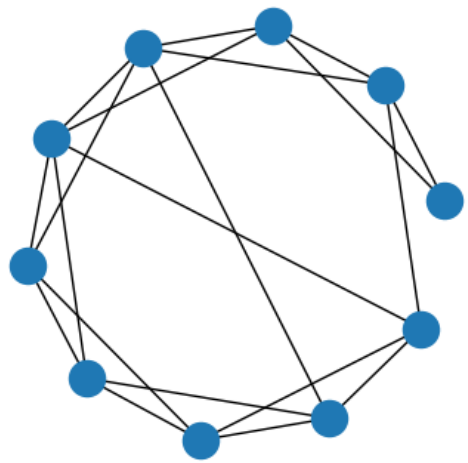
- Here, we use a 'Watts-Strogatz network' – often also referred to as 'small-world network'.
  - All n nodes/agents are aligned in a ring.
  - They are connected to their k nearest neighbours (left/right)



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

Social network - Topology

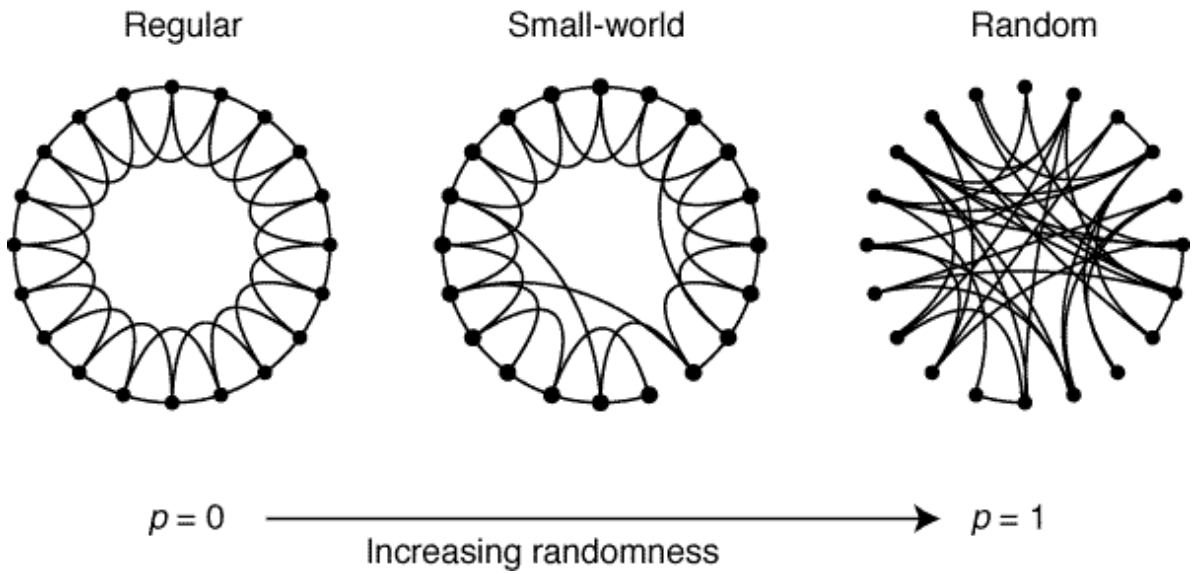
- Here, we use a 'Watts-Strogatz network' – often also referred to as 'small-world network'.
  - All n nodes/agents are aligned in a ring.
  - They are connected to their k nearest neighbours (left/right)
  - We loop through each agent and through each link to the right of that agent. With probability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

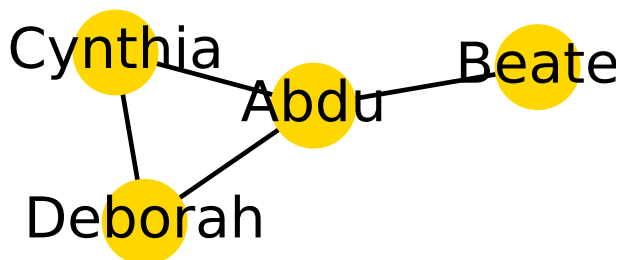
Social network – Perspective

- 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.

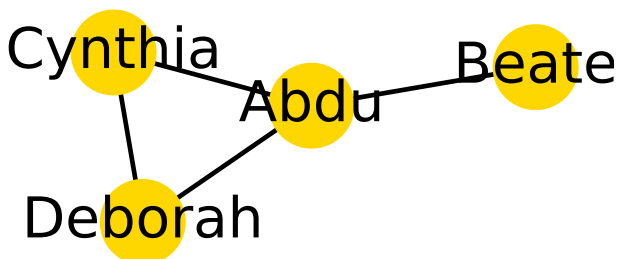
The *networkx* package in python

```
1 import networkx as nx
2 G = nx.Graph()
3 G.add_node("Abdu")
4 ...
5 G.add_edge("Abdu", "Cynthia")
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

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



What's the adjacency Matrix?

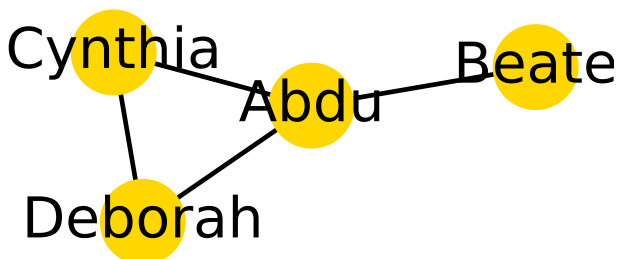
$$(A \quad B \quad C \quad D)$$

$$\downarrow$$

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

The *networkx* package in python

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



What's the adjacency Matrix?

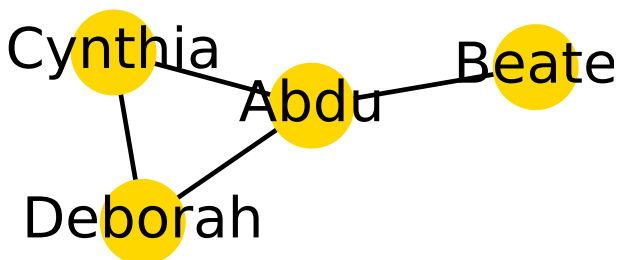
$$(A \quad B \quad C \quad D)$$

$$\downarrow$$

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

The *networkx* package in python

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



What's the adjacency Matrix?

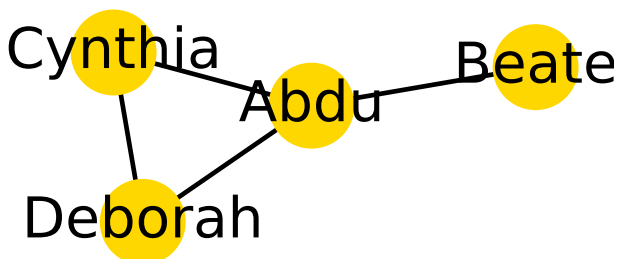
$$(A \quad B \quad C \quad D)$$

$$\downarrow$$

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

The *networkx* package in python

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



What's the adjacency Matrix?

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

$$\downarrow$$

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

The *networkx* package in python

```
import networkx as nx
G = nx.Graph()
G.add_node("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:

```
1 import networkx as nx
2 G = nx.watts_strogatz_graph(n=100, k=4, p=0.1)
3 nx.draw(G)
4 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: *health_state* changes over time following stochastic, age-dependent patterns.
6. Describe the rules for how agents interact with each other.
   agents are connected through a social network. They meet 'physically' with their neighbours and may infect each other.

Concept 4: Event scheduling in ABM

Event scheduling decisions

	queue	sampling
asynchronous		
synchronous		

Event scheduling decisions

	queue	sampling
asynchronous	all agents are updated once per time step after each other	
synchronous		

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

	queue	sampling
asynchronous	all agents are updated once per time step after each other	
synchronous	all agents are updated simultaneously	

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
asynchronous	all agents are updated once per time step after each other	n agents (including duplicates) are updated in one time step after each other
synchronous	all agents are updated simultaneously	

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
asynchronous	all agents are updated once per time step after each other	n agents (including duplicates) are updated in one time step after each other
synchronous	all agents are updated simultaneously	~~n agents (including duplicates) are updated simultaneously~~

Event scheduling decisions

	queue	sampling
asynchronous	all agents are updated once per time step after each other	n agents (including duplicates) are updated in one time step after each other
synchronous	all agents are updated simultaneously	~~n agents (including duplicates) are updated simultaneously~~

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*).

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

Update Function

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

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

Update Function

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

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

Update Function

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

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

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

Update Function

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

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

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 update) health state $\rightarrow$ state-dependent action
  - Potentially infect others in network with certain probability

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

If agent in *health_state ="infectious_..."* 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 update) health state $\rightarrow$ state-dependent action
  - Potentially infect others in network with certain probability

```
1
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 "infectious" in ag.health_state:
8                   infect_others(ag, t_now)
```

```
1 RELATIVE_INFECTIOUSNESS = {"infectious_presymptomatic": 0.5, "
      infectious_symptomatic": 1, "infectious_asymptomatic": 0.2}
2
3 def infect_others(ag, t):
4       p_i = ag.base_infectiousness * RELATIVE_INFECTIOUSNESS[ag.health_state]
5       # Loop through contacts and potentially infect them
6       linked_contacts = list(network.adj[ag.id])  # Indices of neighbours
7       for c in linked_contacts:
8             contact_person = agents[c]
9             if contact_person.health_state == "susceptible"
10                  if np.random.random() < p_i:
11                        catch_virus(contact_person, t)
```

Run and Observe Function

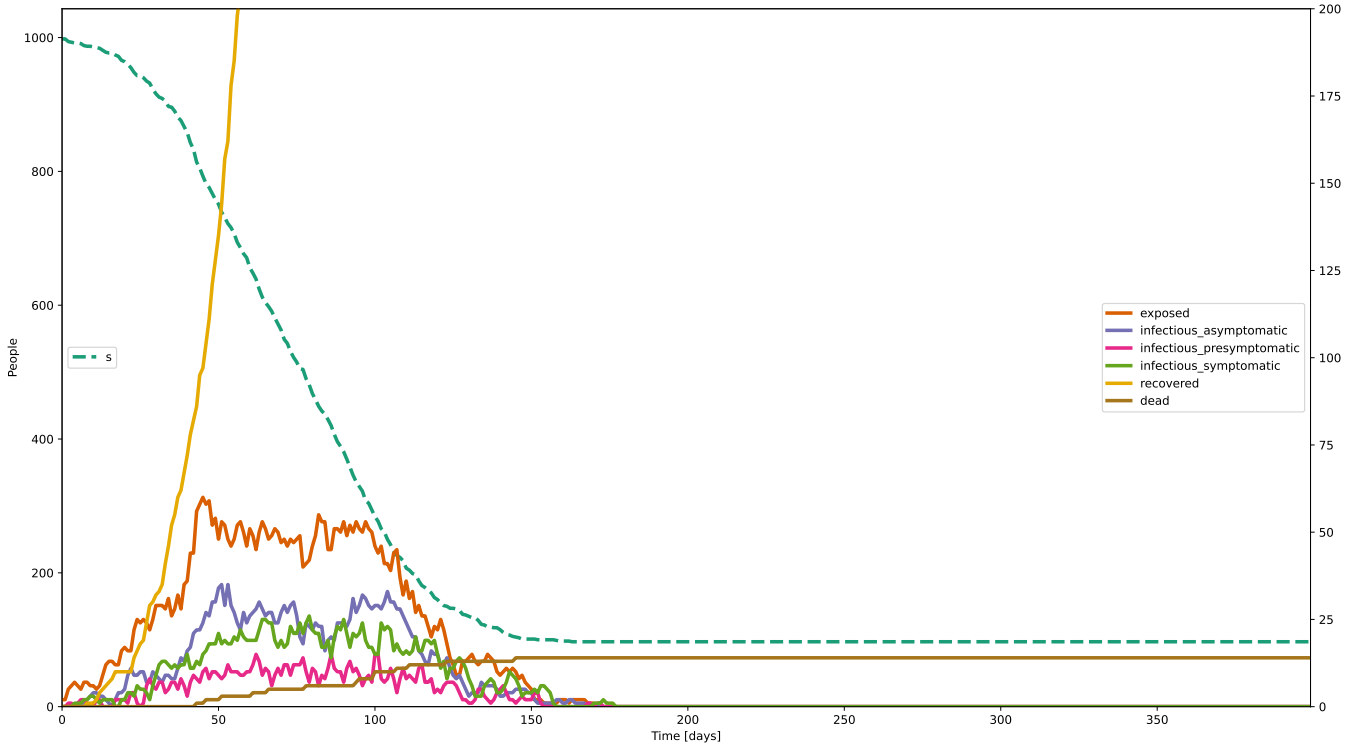
Goal: Perform simulation and track how many agents are in state "susceptible", "exposed", ... over time.

```
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_e = states_of_agents.count("exposed")
      ...
      return np.array([N_s, N_e, N_ia, ...])
```

Run and Observe Function

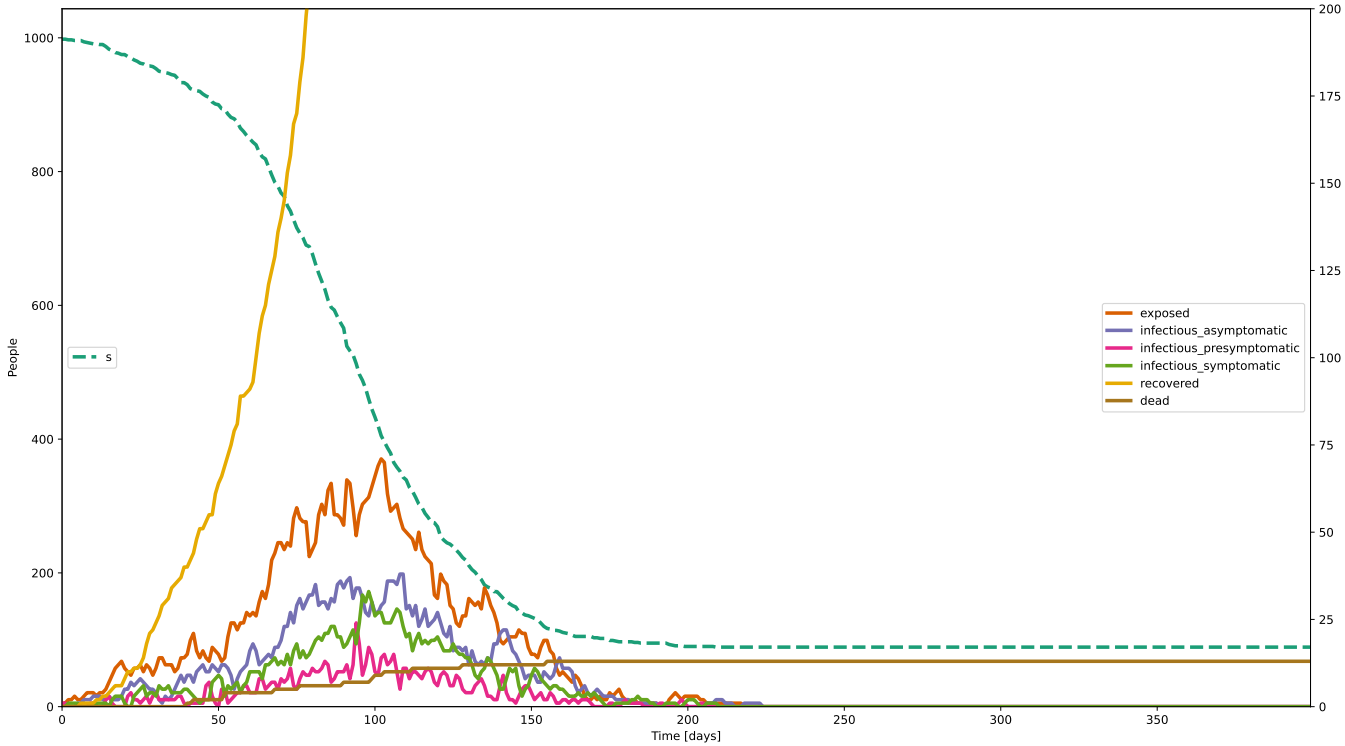
Goal: Perform simulation and track how many agents are in state "susceptible", "exposed", ... over time.



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

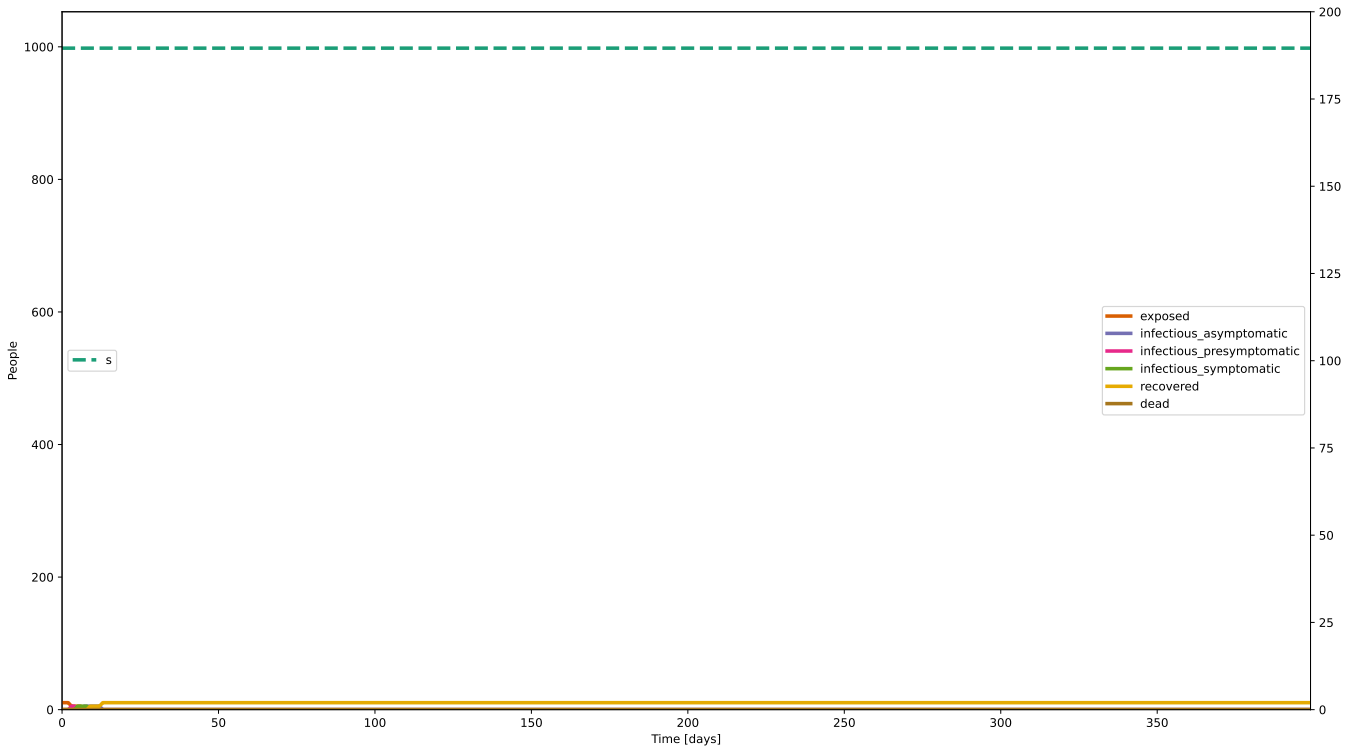
Goal: Perform simulation and track how many agents are in state "susceptible", "exposed", ... over time.



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", "exposed", ... over time.



The model is stochastic!

Same model configuration $\rightarrow$ might not lead to an outbreak at all.

Model output 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 split into 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

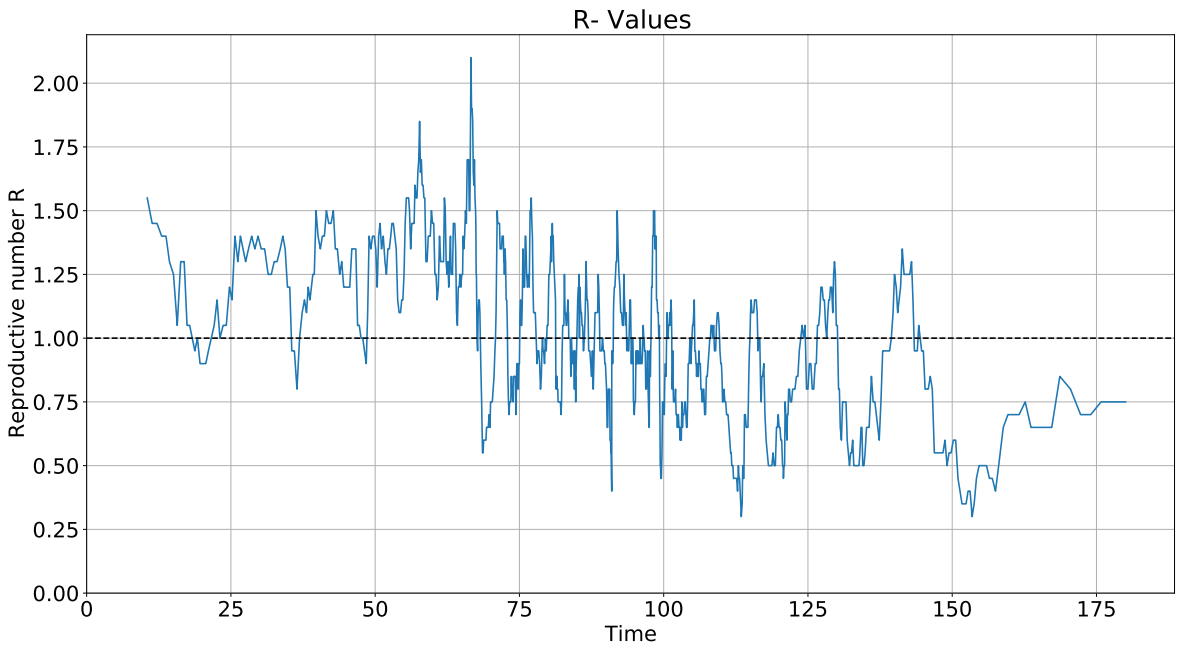
How can we verify if these results make sense? E.g.

1. **Ensemble runs**: Do many different stochastic simulation runs.
2. **Compare with data**: For ABM, typically aggregate indicators. For example, R_0 in the data vs. mean R_0 in ensemble runs of the model.

Validation

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

1. **Ensemble runs**: Do many different stochastic simulation runs.
2. **Compare with data**: For ABM, typically aggregate indicators. For example, R_0 in the data vs. mean R_0 in ensemble runs of the model.



Policies

Policies

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

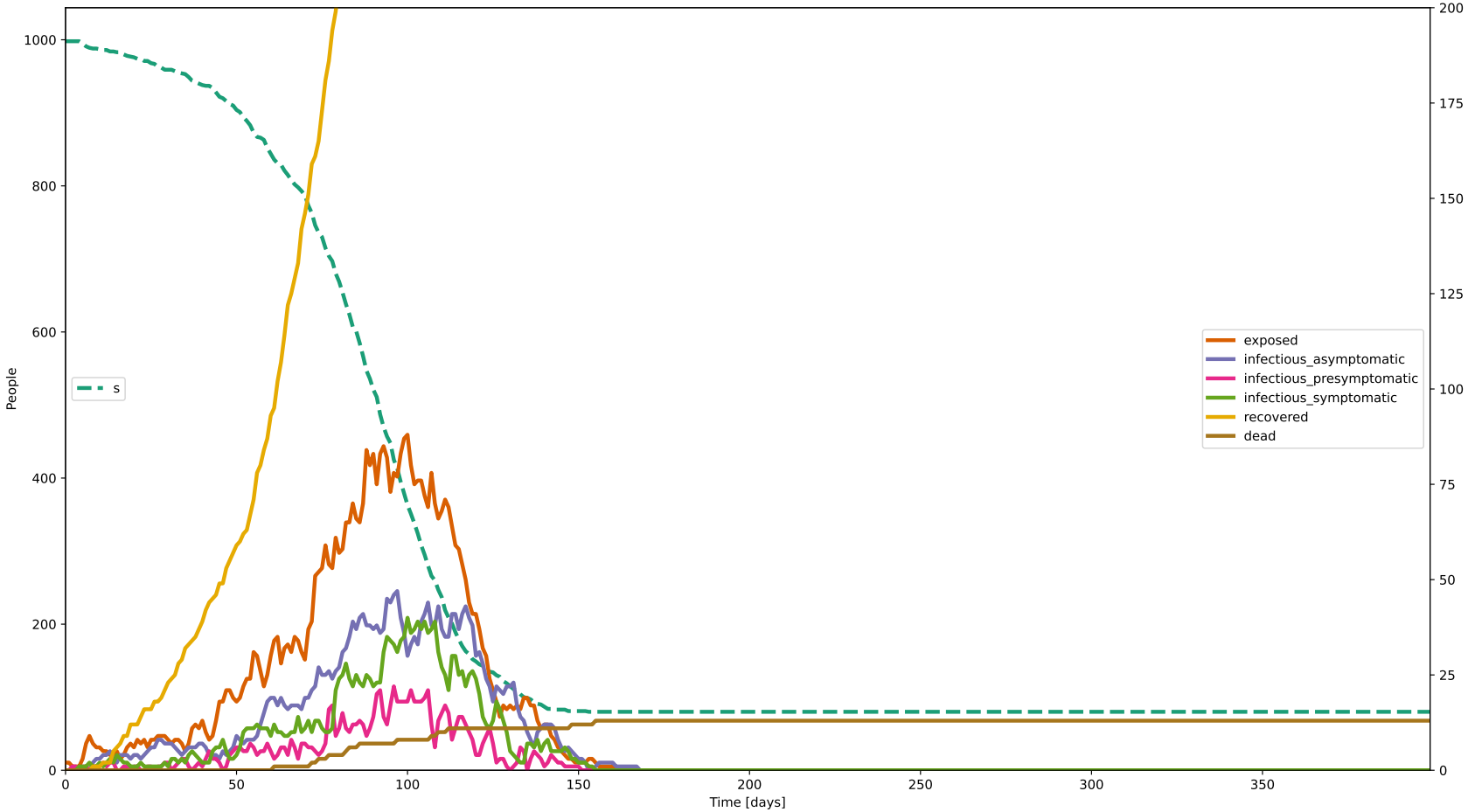
What happens, for example,

- 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?

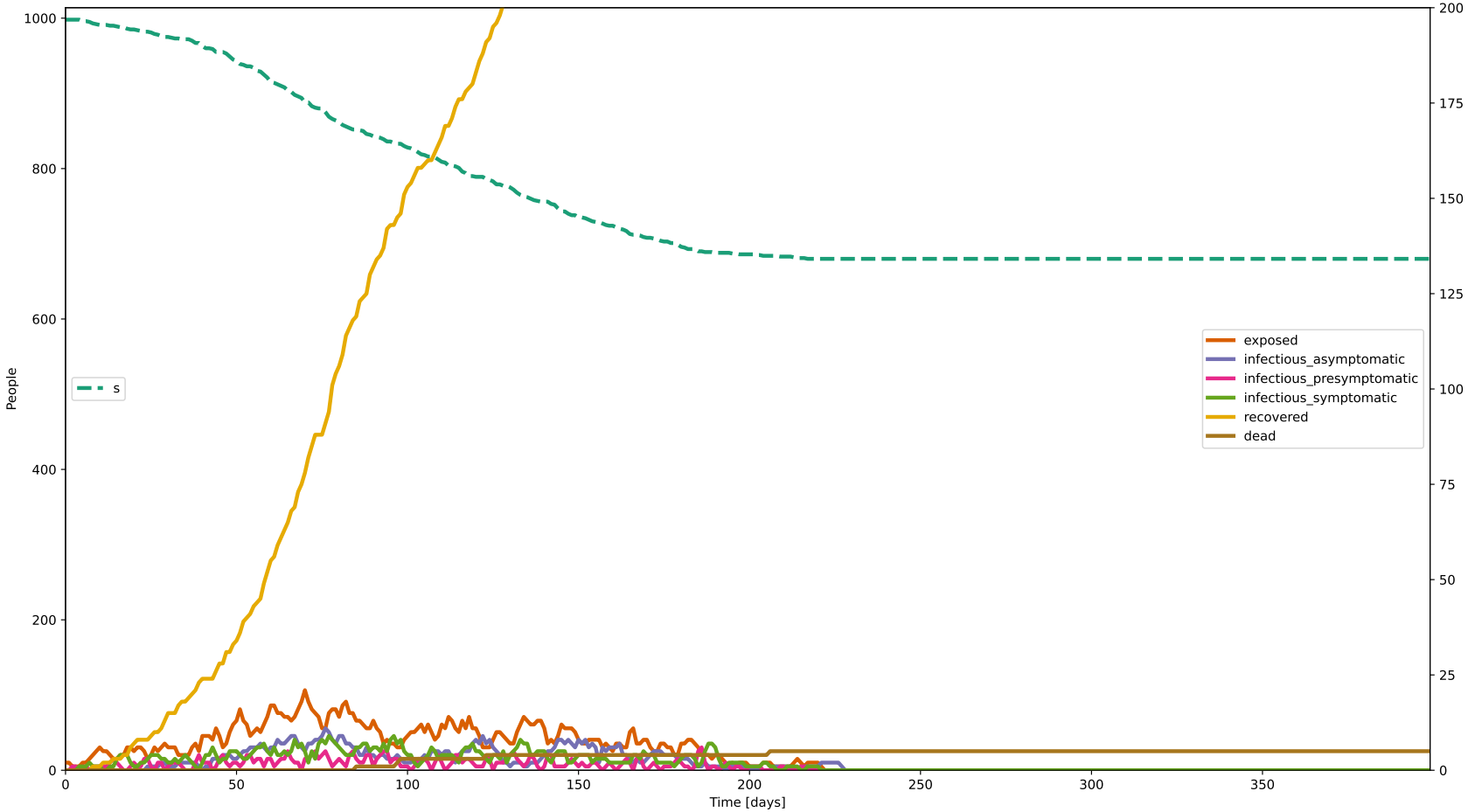
Parameters of the Model

Parameter	Description	current value
FRAC_AGE_GROUPS	Percentage of each age 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]
RELATIVE_INFECTIOUSNESS	rel. strength of infectiousness for a (pre-)symptomatic and asymptomatic infections	[0.5,1,0.2]
TIME_I_PRESYMPT	infectious days before symptom onset	2
TIME_I_POSTSYMPT	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 $k = 6$ of network to $k = 4$



Policy: Decrease $k = 6$ of network 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 turns symptomatic, she/he will quarantine and strongly reduce contacts.
  - This policy may not apply immediately, but only after the outbreak has been noticed by policy makers. Implement a delay mechanism. How does the delay impact the effectiveness of a policy?
  - Implement your own (time-dependent) policy strategy. This could include (1) dynamic changes in the network or (2) different 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?

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