Lesson 2: Simulation of Stochastic Dynamic Models

Aaron A. King Edward L. Ionides Kunyang He

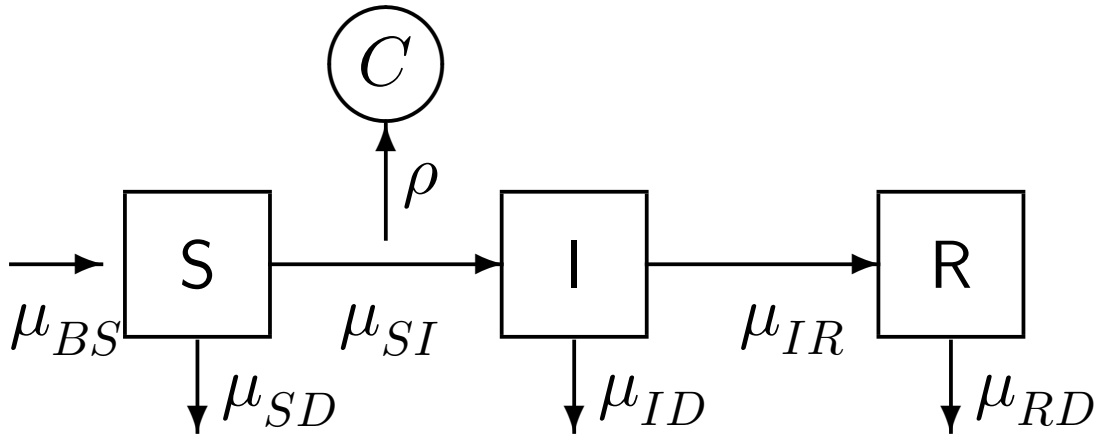
Monday, March 16, 2026

This tutorial develops some classes of dynamic models relevant to biological systems, especially for epidemiology.

1. Dynamic systems can often be represented in terms of **flows** between **compartments**.
2. We define a **compartmental model** for which we specify **rates** for the flows between compartments.
3. We show how deterministic and stochastic versions of a compartmental model are derived and related.
4. We introduce Euler's method to simulate from dynamic models.
5. We specify deterministic and stochastic compartmental models in `pypomp` using Euler-method simulation.

A basic compartment model: The SIR model

- We develop deterministic and stochastic representations of a susceptible-infected-recovered (SIR) system, a fundamental class of models for disease transmission dynamics.
- We set up notation applicable to general compartment models (Bretó et al., 2009). \begin{center}



S : susceptible
I : infected and infectious
R : recovered and/or removed
C : reported cases

- We suppose that each arrow has an associated rate, so here there is a rate $\mu_{SI}(t)$ at which individuals in S transition to I, and μ_{IR} at which individuals in I transition to R.
- To account for demography (births/deaths/migration) we include source and sink compartments, which may not be represented on the flow diagram. We write μ_{BS} for a rate of births into S, and denote mortality rates by μ_{SD}, μ_{ID}, μ_{RD}.

- The rates may be either constant or varying. For a simple SIR model, the recovery rate μ_{IR} is a constant but the infection rate has the time-varying form

$$\mu_{SI}(t) = \beta\, I(t),$$

with β being the *transmission rate*. For the simplest SIR model, ignoring demography, we set

$$\mu_{BS} = \mu_{SD} = \mu_{ID} = \mu_{RD} = 0.$$

General notation for compartment models

To develop a systematic notation, it is convenient to keep track of the flows between compartments as well as the number of individuals in each compartment. Let

$$N_{SI}(t)$$

count the number of individuals who have transitioned from S to I by time t. We say that $N_{SI}(t)$ is a **counting process**. A similarly constructed process

$$N_{IR}(t)$$

counts individuals transitioning from I to R.

To include demography, we could keep track of birth and death events with the counting processes $N_{BS}(t)$, $N_{SD}(t)$, $N_{ID}(t)$ and $N_{RD}(t)$.

- For **discrete-population** compartment models, the flow counting processes are non-decreasing and integer-valued.
- For **continuous-population** compartment models, the flow counting processes are non-decreasing and real-valued.

Compartment processes from counting processes

- The numbers of people in each compartment can be computed via these counting processes. Ignoring demography, we have:

$$
\begin{aligned}
S(t) &= S(0) - N_{SI}(t) \\
I(t) &= I(0) + N_{SI}(t) - N_{IR}(t) \\
R(t) &= R(0) + N_{IR}(t)
\end{aligned}
$$

- These equations represent **conservation of individuals**: what goes in must come out.

Ordinary differential equation interpretation

Together with initial conditions specifying $S(0)$, $I(0)$ and $R(0)$, we just need to write down ordinary differential equations (ODEs) for the flow counting processes. These are:

$$\frac{dN_{SI}}{dt} = \mu_{SI}(t)\, S(t)$$
$$\frac{dN_{IR}}{dt} = \mu_{IR}\, I(t)$$

Continuous-time Markov chain interpretation I

- Continuous-time Markov chains are the basic tool for building **discrete population epidemic models**.
- The Markov property lets us specify a model by the transition probabilities on small intervals (together with the initial conditions). For the SIR model, we have

$$\begin{aligned}
\Pr\left[N_{SI}(t+\delta) = N_{SI}(t) + 1\right] &= \mu_{SI}(t)\, S(t)\, \delta + o(\delta) \\
\Pr\left[N_{SI}(t+\delta) = N_{SI}(t)\right] &= 1 - \mu_{SI}(t)\, S(t)\, \delta + o(\delta) \\
\Pr\left[N_{IR}(t+\delta) = N_{IR}(t) + 1\right] &= \mu_{IR}\, I(t)\, \delta + o(\delta) \\
\Pr\left[N_{IR}(t+\delta) = N_{IR}(t)\right] &= 1 - \mu_{IR}(t)\, I(t)\, \delta + o(\delta)
\end{aligned}$$

- Here, we are using *little-o notation*.

  We write $h(\delta) = o(\delta)$ to mean $\lim_{\delta \to 0} \frac{h(\delta)}{\delta} = 0$.

Exercise 2.1

What is the link between little o notation and the derivative? Explain why

$$f(x+\delta) = f(x) + \delta\, g(x) + o(\delta)$$

is the same statement as

$$\frac{df}{dx} = g(x).$$

What considerations might help you choose which of these notations to use?

Simple counting processes

- A **simple counting process** is one which cannot count more than one event at a time.
- By this definition, the SIR Markov-chain model constructed above is simple.
- One may want to model the extra randomness resulting from multiple simultaneous events: someone sneezing in a bus; large gatherings at football matches; etc. This extra randomness may even be critical to match the variability in data.
- Later in the course, we may see situations where this extra randomness plays an important role. Setting up the model using counting processes, as we have done here, turns out to be useful for this.

Euler's method for ordinary differential equations

- Euler (1707 – 1783) wanted a numeric solution of an ordinary differential equation (ODE) $dx/dt = h(x)$ with an initial condition $x(0)$.
- He supposed this ODE has some true solution $x(t)$ which could not be worked out analytically. He wanted an approximation $\tilde{x}(t)$ of $x(t)$.
- He initialized the numerical solution at the known starting value,

$$\tilde{x}(0) = x(0).$$

- For $k = 1, 2, ...$, he supposed that the gradient dx/dt is approximately constant over the small time interval $k\delta \leq t \leq (k+1)\delta$. Therefore, he defined

$$\tilde{x}\big((k+1)\delta\big) = \tilde{x}(k\delta) + \delta\, h\big(\tilde{x}(k\delta)\big).$$

- This only defines $\tilde{x}(t)$ when t is a multiple of δ, but suppose $\tilde{x}(t)$ is constant between these discrete times.
- We now have a numerical scheme, stepping forwards in time increments of size δ, that can be readily evaluated by computer.

Euler's method versus other numerical methods

- Mathematical analysis of Euler's method says that, as long as the function $h(x)$ is not too exotic, then $x(t)$ is well approximated by $\tilde{x}(t)$ when the discretization time-step δ is sufficiently small.
- Euler's method is not the only numerical scheme to solve ODEs. More advanced schemes have better convergence properties, meaning that the numerical approximation is closer to $x(t)$. However, there are three reasons we choose to lean heavily on Euler's method:
  1. Euler's method is the simplest (cf. the KISS principle).
  2. Euler's method extends naturally to stochastic models, both continuous-time Markov-chain models and stochastic differential equation (SDE) models.
  3. Close approximation of the numerical solutions to a continuous-time model is less important than it may at first appear—a topic to be discussed.

Continuous-time models and discretized approximations

- In some physical and engineering situations, a system follows an ODE model closely. For example, Newton's laws provide a very good approximation to the motions of celestial bodies.
- In many biological situations, ODE models only become close mathematical approximations to reality at reasonably large scale. On small temporal scales, models cannot usually capture the full scope of biological variation and complexity.
- If we are going to expect substantial error in using $x(t)$ to model a biological system, maybe the numerical solution $\tilde{x}(t)$ represents the system being modeled just as well as $x(t)$ does.

- If our model fitting, model investigation, and final conclusions are all based on our numerical solution $\tilde{x}(t)$ (i.e. we are sticking entirely to simulation-based methods) then we are most immediately concerned with how well $\tilde{x}(t)$ describes the system of interest. In that sense, $\tilde{x}(t)$ becomes more important than the original model $x(t)$.

Numerical solutions as scientific models

- It is important that a scientist fully describe the numerical model $\tilde{x}(t)$. Arguably, the main purpose of the original model $x(t)$ is to give a succinct description of how $\tilde{x}(t)$ was constructed.
- All numerical methods are, ultimately, discretizations. Epidemi- ologically, setting δ to be a day or an hour can be quite different from setting δ to be two weeks or a month. For continuous-time modeling, we still require that δ is small compared to the timescale of the process being modeled, so the choice of δ should not play an explicit role in the interpretation of the model.
- Putting more emphasis on the scientific role of the numerical solution itself reminds you that the numerical solution has to do more than approximate a target model in some asymptotic sense: the numerical solution should be a sensible model in its own right.

Euler's method for a discrete SIR model I

- Recall the simple continuous-time Markov-chain interpretation of the SIR model without demography:

$$\Pr\big[N_{SI}(t+\delta) = N_{SI}(t) + 1\big] = \mu_{SI}(t)\, S(t)\, \delta + o(\delta),$$
$$\Pr\big[N_{IR}(t+\delta) = N_{IR}(t) + 1\big] = \mu_{IR}\, I(t)\, \delta + o(\delta).$$

- We want a numerical solution with state variables $\tilde{S}(k\delta)$, $\tilde{I}(k\delta)$, $\tilde{R}(k\delta)$.

Euler's method for a discrete SIR model II

- The counting processes for the flows between compartments are $\tilde{N}_{SI}(t)$ and $\tilde{N}_{IR}(t)$. They relate to the numbers of individuals in the compartments via

$$\begin{aligned}\tilde{S}(k\delta) &= S(0) - \tilde{N}_{SI}(k\delta),\\ \tilde{I}(k\delta) &= I(0) + \tilde{N}_{SI}(k\delta) - \tilde{N}_{IR}(k\delta),\\ \tilde{R}(k\delta) &= R(0) + \tilde{N}_{IR}(k\delta).\end{aligned}$$

- We focus on a numerical solution to $N_{SI}(t)$, since the same methods can be applied to $N_{IR}(t)$.

Three different stochastic Euler solutions I

1. Poisson approximation

$$\tilde{N}_{SI}(t+\delta) = \tilde{N}_{SI}(t) + \text{Poisson}[\mu_{SI}(\tilde{I}(t))\,\tilde{S}(t)\,\delta],$$

where $\text{Poisson}(\mu)$ is a Poisson random variable with mean μ and

$$\mu_{SI}(\tilde{I}(t)) = \beta\,\tilde{I}(t).$$

2. Binomial approximation

$$\tilde{N}_{SI}(t+\delta) = \tilde{N}_{SI}(t) + \text{Binomial}[\tilde{S}(t), \mu_{SI}(\tilde{I}(t))\,\delta],$$

where $\text{Binomial}(n,p)$ has mean np and variance $np(1-p)$, with $p = \mu_{SI}(\tilde{I}(t))\,\delta$.

Three different stochastic Euler solutions II

3. Binomial approximation with exponential transition probability

$$\tilde{N}_{SI}(t+\delta) = \tilde{N}_{SI}(t) + \text{Binomial}[\tilde{S}(t), 1 - \exp\{-\mu_{SI}(\tilde{I}(t))\,\delta\}].$$

Analytically, it is usually easiest to reason using **(1)** or **(2)**.
Practically, it is often preferable to work with **(3)**.

Compartment models as stochastic differential equations I

- The Euler method extends naturally to stochastic differential equations (SDEs).
- A natural way to add stochastic variation to an ODE $dx/dt = h(x)$ is

$$\frac{dX}{dt} = h(X) + \sigma \frac{dB}{dt},$$

where $\{B(t)\}$ is Brownian motion, so dB/dt is Brownian noise.

Compartment models as stochastic differential equations II

- An Euler approximation $\tilde{X}(t)$ is

$$\tilde{X}((k+1)\delta) = \tilde{X}(k\delta) + \delta\, h(\tilde{X}(k\delta)) + \sigma\sqrt{\delta}\, Z_k,$$

  where $Z_1, Z_2, \ldots$ are independent standard normal variables ($Z_k \sim \mathcal{N}(0,1)$).

- Although SDEs are often considered advanced, the Euler approximation itself requires little more than familiarity with the normal distribution.

Exercise 2.2. Euler's method vs Gillespie's algorithm

A widely used exact simulation method for continuous-time Markov chains is Gillespie's algorithm.
We do not emphasize it here. Why? When would you prefer an implementation of Gillespie's algorithm over an Euler solution?

Numerically, Gillespie's algorithm is often approximated using tau-leaping methods, which are closely related to Euler's approach; in this context Euler's method is sometimes called *tau-leaping*.

Compartment models in pypomp: Measles in Consett

As an example, we investigate an outbreak of measles that occurred in the small town of Consett in England in 1948. The town had a population of 38 820, with 737 births over the course of the year.

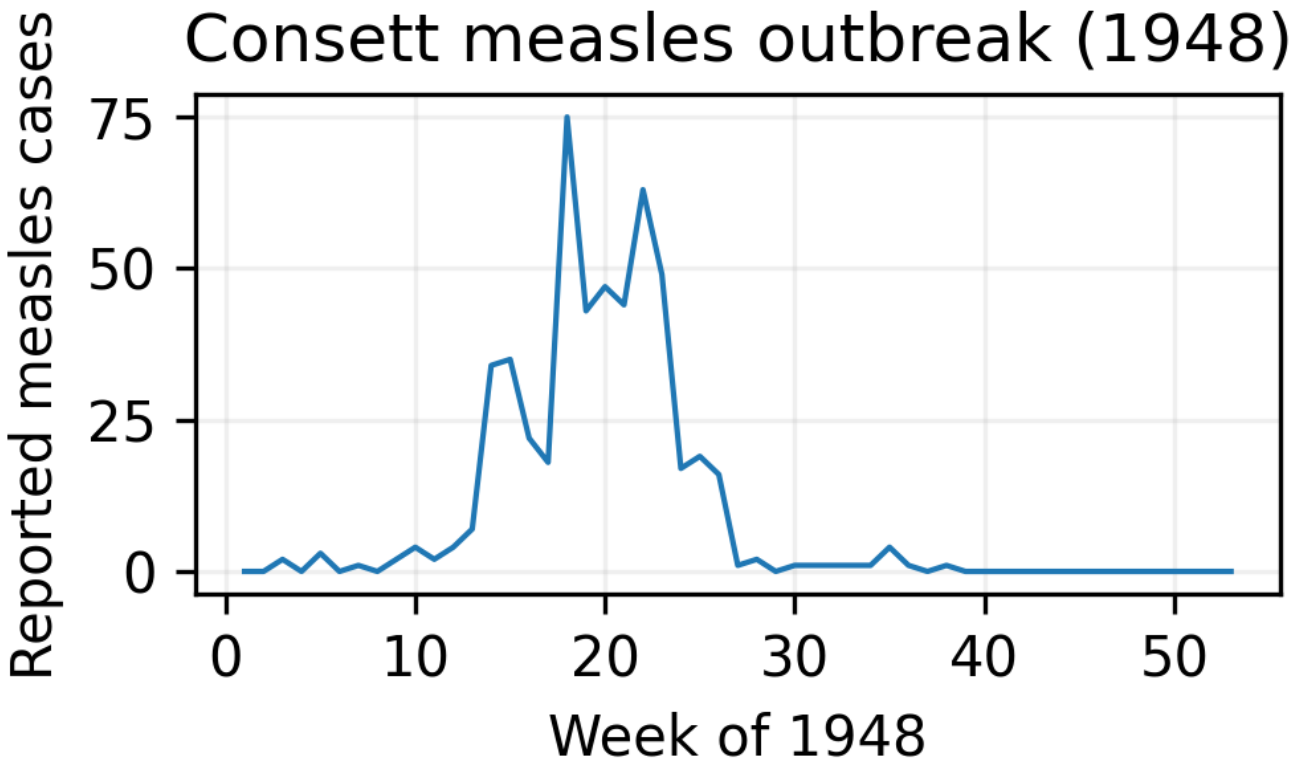


Figure 1: Consett measles outbreak (1948)

A simple POMP model for measles

- These are incidence data: the `reports` variable counts the number of new measles cases each week.
- We will model the outbreak using the simple **SIR** model.
- Tasks: (i) estimate the parameters of the SIR; (ii) decide whether SIR adequately describes these data.
- The rate at which individuals move from S to I is the *force of infection* $\mu_{SI} = \beta\, I/N$, while that at which individuals leave I for R is μ_{IR}.

Framing the SIR as a POMP model

- **Latent state variables**: numbers of individuals $S(t)$, $I(t)$, $R(t)$ in the S, I, R compartments.
- Treat population size $N = S + I + R$ as fixed at the known value 38,000.
- The actual numbers moving between compartments over any interval are modeled as **stochastic processes**.
- We assume the stochasticity is **purely demographic**: every individual in a compartment faces the same exit risk at any time.
- *Demographic stochasticity* is the unavoidable randomness arising from chance events in a discrete, finite population.

Implementing the SIR model in pypomp I

- To implement the model in pypomp, we first need a **stochastic simulator** for the latent process.
- Following method 3 (binomial with exponential transition), the number moving S $\to$ I over Δt is

$$\Delta N_{SI} \sim \text{Binomial}\Big(S,\ 1 - e^{-\beta \frac{I}{N} \Delta t}\Big),$$

and the number moving I $\to$ R is

$$\Delta N_{IR} \sim \text{Binomial}\Big(I,\ 1 - e^{-\mu_{IR} \Delta t}\Big).$$

Implementing the SIR model in pypomp II

In pypomp, model components are defined as Python functions with specific argument signatures. The initial state simulator `rinit` must have arguments `theta_`, `key`, `covars`, and `t0` (in that order), and return a dictionary mapping state names to their initial values:

```
def rinit(theta_, key, covars, t0):
    """Initial state simulator for SIR model."""
    N = theta_["N"]
    eta = theta_["eta"]
    S0 = jnp.round(N * eta)
    I0 = 1.0
    R0 = jnp.round(N * (1 - eta)) - 1.0
    H0 = 0.0  # Accumulator for true incidence
    return {"S": S0, "I": I0, "R": R0, "H": H0}
```

Implementing the SIR model in pypomp III

- Now assume the case reports result from a process by which new infections are diagnosed and reported with probability ρ, which we can think of as the probability that a child's parents take the child to the doctor, who recognises measles and reports it to the authorities.
- Measles symptoms tend to be quite recognisable, and children with measles tend to be confined to bed. Therefore diagnosed cases have, presumably, a much lower transmission rate. Accordingly, let's treat each week's `reports` as being related to the number of individuals who have moved from I to R over the course of that week.
- We need a variable to track these **daily** counts. We modify our model to add a variable H to tally the true incidence.

Implementing the SIR model in pypomp IV

- The process simulator `rproc` must have the following arguments, in order:

`X_`

`theta_`

`key`

`covars`

`t`

`dt`

- It must return a dictionary of the updated state

```
def rproc(X_, theta_, key, covars, t, dt):
    """Process simulator for SIR model
    using the Euler-binomial scheme."""
    S, I, R, H = X_["S"], X_["I"], X_["R"], X_["H"]
    Beta = theta_["Beta"]
    mu_IR = theta_["mu_IR"]
    N = theta_["N"]
    p_SI = 1.0 - jnp.exp(-Beta * I / N * dt)
    p_IR = 1.0 - jnp.exp(-mu_IR * dt)
    key_SI, key_IR = jax.random.split(key)
    dN_SI = jax.random.binomial(key_SI, n=jnp.int32(S), p=p_SI)
    dN_IR = jax.random.binomial(key_IR, n=jnp.int32(I), p=p_IR)
    S_new = S - dN_SI
    I_new = I + dN_SI - dN_IR
    R_new = R + dN_IR
    H_new = H + dN_IR
    return {"S": S_new, "I": I_new, "R": R_new, "H": H_new}
```

- Now, we'll model the data, conditional on the latent state, by a negative-binomial variable:

$$\text{reports}_t \sim \text{NegBin}\big(\rho\, H(t),\, k\big),$$

  with mean $\rho\, H(t)$ and variance $\rho\, H(t) + \big(\rho\, H(t)\big)^2/k$. The binomial distribution does not have a separate variance parameter.

- To include the observations in the model, we must write either a `dmeas` or an `rmeas` component, or both.

First, we define helper functions for the negative binomial distribution:

```
def nbinom_logpmf(x, k, mu):
    """Log PMF of NegBin(k, mu) that is robust when mu == 0."""
    x = jnp.asarray(x)
    k = jnp.asarray(k)
    mu = jnp.asarray(mu)
    logp_zero = jnp.where(x == 0, 0.0, -jnp.inf)
    safe_mu = jnp.where(mu == 0.0, 1.0, mu)
    core = (jax.scipy.special.gammaln(k + x)
            - jax.scipy.special.gammaln(k)
            - jax.scipy.special.gammaln(x + 1)
            + k * jnp.log(k / (k + safe_mu))
            + x * jnp.log(safe_mu / (k + safe_mu)))
    return jnp.where(mu == 0.0, logp_zero, core)

def rnbinom(key, k, mu):
    """Sample from NegBin(k, mu) via Gamma-Poisson mixture."""
    key_g, key_p = jax.random.split(key)
    lam = jax.random.gamma(key_g, k) * (mu / k)
    return jax.random.poisson(key_p, lam)
```

The measurement density `dmeas` and simulator `rmeas`:

```
def dmeas(Y_, X_, theta_, covars, t):
    """Measurement density: log P(reports | H, rho, k)."""
    rho = theta_["rho"]
    k = theta_["k"]
    H = X_["H"]
    mu = rho * H
    return nbinom_logpmf(Y_["reports"], k, mu)
```

```
def rmeas(X_, theta_, key, covars, t):
    """Measurement simulator: sample reports ~ NegBin(k, rho*H)."""
    rho = theta_["rho"]
    k = theta_["k"]
    H = X_["H"]
    mu = rho * H
    reports = rnbinom(key, k, mu)
    return jnp.array([reports])
```

A call to `Pomp` assembles the model components and the data:

```
theta = {
    "Beta": 7.5,
    "mu_IR": 0.5,
    "N": 38000.0,
    "eta": 0.03,
    "rho": 0.5,
    "k": 10.0
}
statenames = ["S", "I", "R", "H"]
sir_obj = pp.Pomp(
    rinit=rinit, rproc=rproc,
    dmeas=dmeas, rmeas=rmeas,
    ys=ys, theta=theta, statenames=statenames,
    t0=0.0, nstep=7, accumvars=("H",),
    ydim=1, covars=None
)
```

Guessing plausible parameter values I

- To check that the code is working properly, we will **simulate** the model. This requires plausible parameter values, which we can obtain with a few back-of-the-envelope estimates.

Basic epidemiological theory defines $\mathcal{R}_0$ to be the expected number of secondary infections caused by one primary infection introduced into a fully susceptible population.

For an SIR infection we have $\mathcal{R}_0 \approx \frac{L}{A}$, where L is host lifespan and A is mean age of infection.

- Age-stratified serology indicates $A \approx 4$–5 yr (Anderson and May, 1991). Assuming $L \approx 60$–70 yr gives $\mathcal{R}_0 \approx 15$.

Guessing plausible parameter values II

- The final-size equation for an SIR epidemic is

$$\mathcal{R}_0 = -\frac{\log(1-f)}{f},$$

  where f is the fraction of initial susceptibles who ultimately become infected. For $\mathcal{R}_0 > 5$, this implies $f > 0.99$.

- The data contain 521 reported infections. Assuming a 50 % reporting rate, we have $S_0 \approx 1042$, so that $\eta = \frac{S_0}{N} \approx 0.027$.

- If the infectious period is roughly 2 weeks, then $1/\mu_{IR} \approx 2$ wk and $\beta = \mu_{IR}\,\mathcal{R}_0 \approx 7.5$ wk^{-1}.

Let's now simulate the model with these parameter values.

```
n_sims = 20
key = jax.random.key(42)
X_sims, Y_sims = sir_obj.simulate(key=key, nsim=n_sims)
sim_df = Y_sims.pivot_table(
    index="time", columns="sim", values="obs_0"
)
sim_df.columns = [f"sim_{i+1}" for i in range(n_sims)]
```

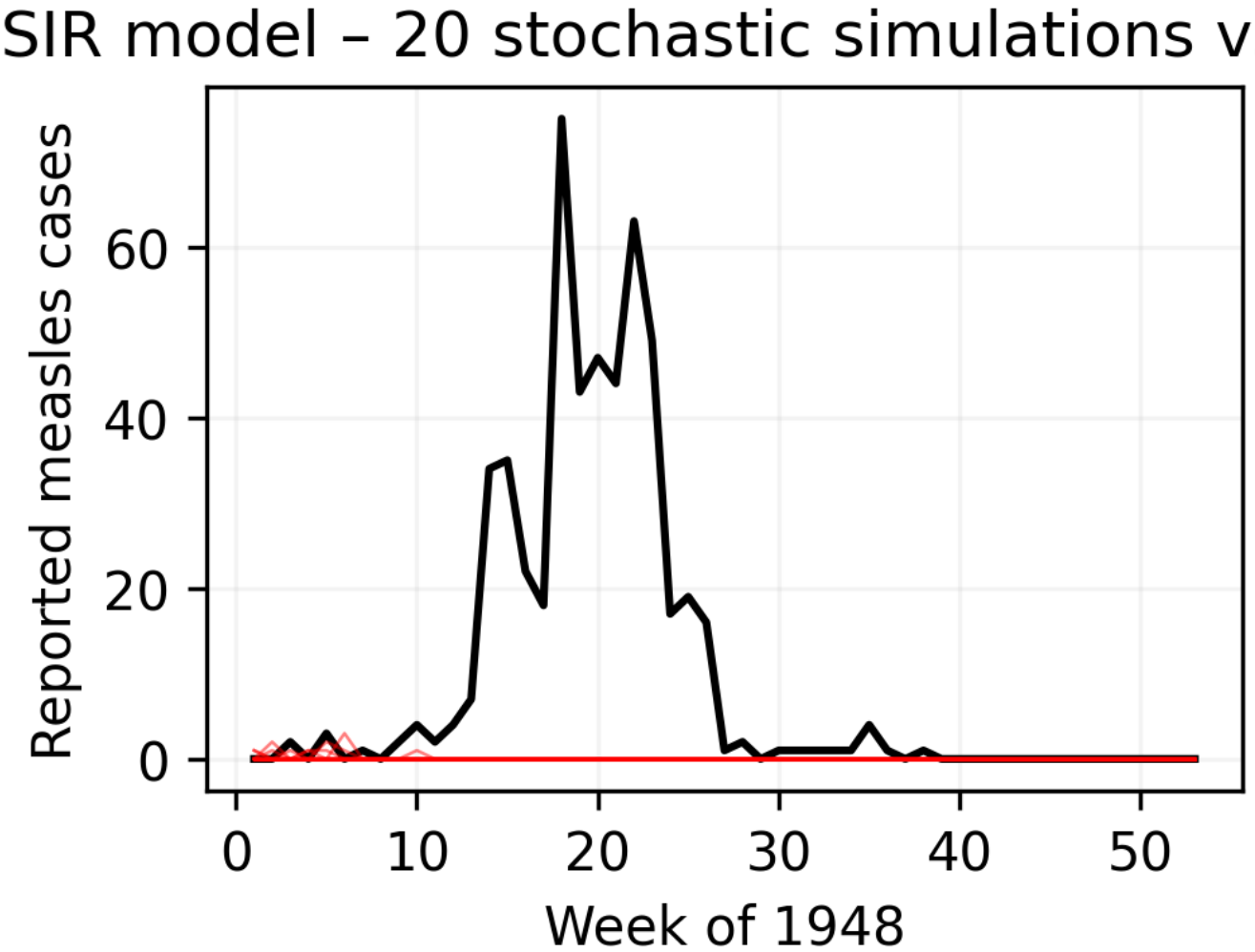


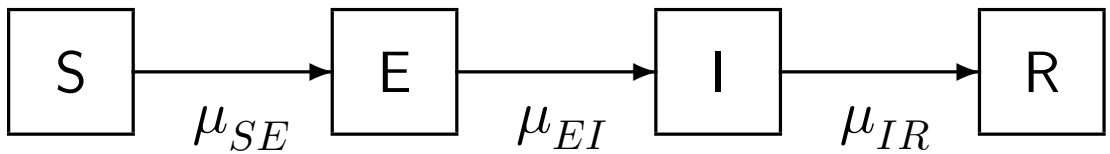
Figure 2: SIR simulation (red) and 1948 Consett measles data (black). Clearly, this leaves something to be desired. In the exercises, you'll see if this model can do better.

Exercise 2.3. Explore the SIR model.

Fiddle with the parameters to see if you can't find a model for which the data are a more plausible realization.

Exercise 2.4. The SEIR model

Below is a diagram of the so-called SEIR model. This differs from the SIR model in that infected individuals must pass a period of latency before becoming infectious.



Modify the codes above to construct a `Pomp` object containing the Consett measles data and an SEIR model. Perform simulations as above and adjust parameters to get a sense of whether improvement is possible by including a latent period.

Anderson, R. M. and May, R. M. (1991). *Infectious Diseases of Humans*. Oxford Univesity Press, Oxford.

Bretó, C., He, D., Ionides, E. L., and King, A. A. (2009). Time series analysis via mechanistic models. *Ann Appl Stat*, 3(1):319–348.