Lesson 5: Case Study - Measles in England and Wales

Aaron A. King Edward L. Ionides Translated in pypomp by Kunyang He

2025-12-24

Table of contents I

1. Introduction
   - Objectives
   - Measles Revisited
2. Setup
   - Import Packages
   - Load Data
3. The Model
   - SEIR with Cohort Effect and Seasonality
   - Constructing the POMP Object
4. Particle Filtering
   - Running the Particle Filter

Table of contents II

5 Iterated Filtering (IF2)
- Setting Up IF2

6 Simulation
- Simulating from the Fitted Model

7 Profile Likelihoods
- Why Profile Likelihoods?

8 Model Diagnostics
- ARMA Benchmark

9 Interpretation of Results
- Parameter Estimates

10 Exercises

Table of contents III

- Exercise 5.1: Model Diagnostics
- Exercise 5.2: Alternative Seasonality
- Exercise 5.3: Extra-Demographic Stochasticity
- Exercise 5.4: Cohort Effect Identifiability
- Exercise 5.5: Fixed Reporting Rate

11 Summary

- Key Takeaways

This lesson presents an in-depth case study of measles in England and Wales, demonstrating the application of POMP methods with **pypomp** to long time series with complex dynamics.

Learning Objectives

1. To display a published case study using plug-and-play methods with non-trivial model complexities.
2. To show how extra-demographic stochasticity can be modeled.
3. To demonstrate the use of covariates in **pypomp**.
4. To demonstrate the use of profile likelihood in scientific inference.
5. To discuss the interpretation of parameter estimates.
6. To emphasize the potential need for extra sources of stochasticity in modeling.

Why Measles? I

Measles is the paradigm for a nonlinear ecological system that can be well described by low-dimensional nonlinear dynamics.

A tradition of careful modeling studies have proposed and found evidence for a number of specific mechanisms, including:

- Seasonal variation in transmission (school terms)
- Birth cohort effects (school entry)
- Response to changing birth rates
- Coupling between cities via human movement
- Extra-demographic stochasticity (environmental or behavioral noise)

Why Measles? II

Long, detailed time series are available for many cities. Measles has well-understood natural history and strong prior knowledge about key parameters.

Our goals are to:

1. Demonstrate building a reasonably complex model in **pypomp**
2. Fit it to data
3. Explore the use of profile likelihoods for inference

The He et al. (2010) Study I

We'll implement the model from He et al. (2010), which fits measles data from 20 cities in England and Wales.

Key model features:

1. **SEIR structure** with births and deaths
2. **Seasonal transmission**: high during school terms, low during holidays
3. **Cohort effect**: fraction of birth cohort enters susceptible pool at school start
4. **Extra-demographic stochasticity**: multiplicative gamma white noise in force of infection
5. **Demographic stochasticity**: Euler-multinomial transitions

Import Packages

```
import jax
import jax.numpy as jnp
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Import pypomp components
from pypomp import Pomp, RWSigma, ParTrans
from pypomp.util import logmeanexp, logmeanexp_se

# Import the built-in UK Measles module
from pypomp.measles.measlesPomp import UKMeasles
import pypomp.measles.model_001b as m001b

# For reproducibility
np.random.seed(594709947)
```

The Data I

The **pypomp** package includes the UK measles data from He et al. (2010). The `UKMeasles` class provides convenient access to the data and model construction.

```
# Load data for London
data = UKMeasles.subset(units=["London"])

# View the measles case data
measles = data["measles"]
print(f"Measles data shape: {measles.shape}")
print(f"Date range: {measles['date'].min()} to {measles['date'].max()}")
print(measles.head())
```

The Data II

```
Measles data shape: (1108, 3)
Date range: 1944-01-07 00:00:00 to 1965-03-26 00:00:00
        date    unit  cases
0 1944-01-07  London     82
1 1944-01-14  London     98
2 1944-01-21  London    118
3 1944-01-28  London    153
4 1944-02-04  London    206
```

The Data III

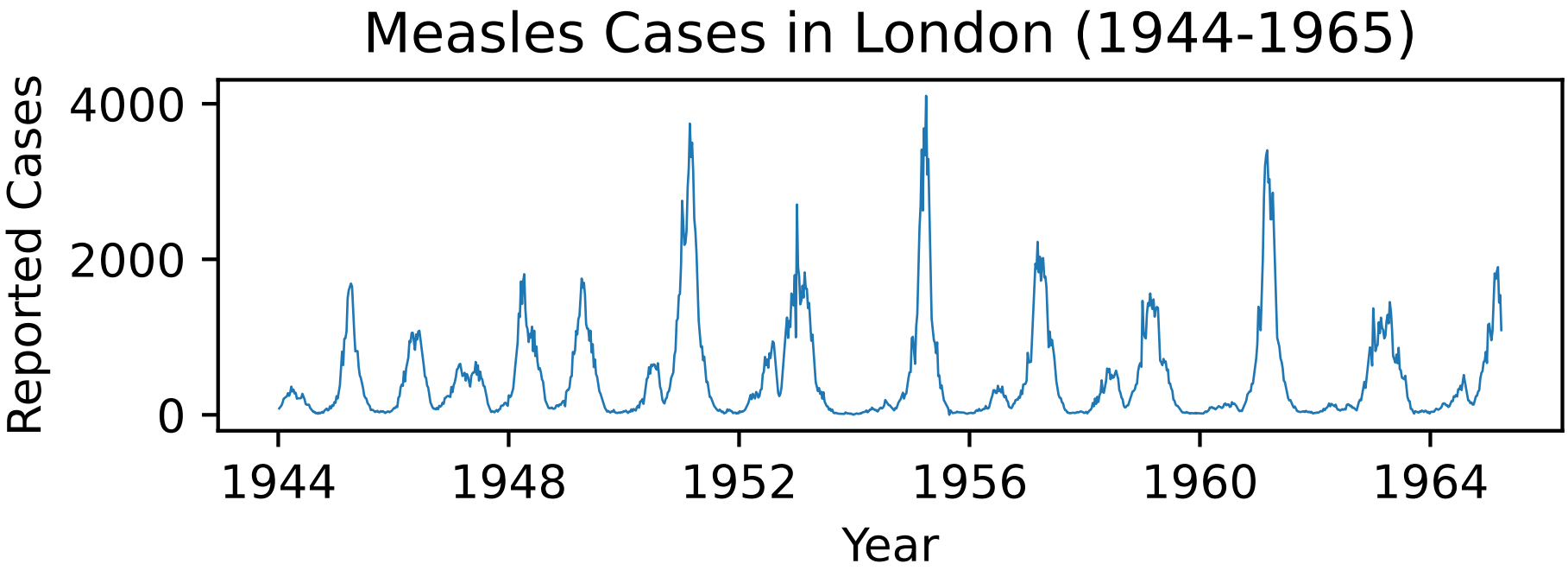
```python
# View demographic data
demog = data["demog"]
print(f"Demographic data shape: {demog.shape}")
print(demog.head())
```

```
Demographic data shape: (21, 4)
   year    unit      pop  births
0  1944  London  2462500   44851
1  1945  London  2601370   45840
2  1946  London  3109240   66023
3  1947  London  3245000   70685
4  1948  London  3339100   60805
```

Visualizing the Data I

```
fig, ax = plt.subplots(figsize=(5, 2))
ax.plot(measles['date'], measles['cases'], linewidth=0.5)
ax.set_xlabel('Year')
ax.set_ylabel('Reported Cases')
ax.set_title('Measles Cases in London (1944-1965)')
plt.tight_layout()
plt.show()
```

Visualizing the Data II



Model Structure I

The state variables are:

- S: Susceptible individuals
- E: Exposed (latent) individuals
- I: Infectious individuals
- R: Recovered individuals (computed as $N - S - E - I$)
- W: Cumulative white noise (for diagnostics)
- C: Cumulative cases (accumulator variable, reset each observation interval)

Model Structure II

The model incorporates several important features:

1. **The birth-cohort effect**: A specified fraction (`cohort`) of each annual birth cohort enters the susceptible pool all at once, on the first day of the school year.
2. **Seasonality in the transmission rate**: During school terms, the transmission rate is higher than during holidays.
3. **Extra-demographic stochasticity**: A Gamma white-noise term acts multiplicatively on the force of infection.
4. **Demographic stochasticity**: Implemented using Euler-multinomial distributions.

Model Parameters I

The model parameters (from `model_001b`) are:

Parameter	Description
R_0	Basic reproduction number
σ	Rate of transition from E to I (latent rate)
γ	Rate of transition from I to R (recovery rate)
ι	Imported cases
ρ	Reporting probability
σ_{SE}	Extra-demographic stochasticity intensity
ψ	Reporting overdispersion
cohort	Fraction of births entering at school start
amplitude	Seasonality amplitude
S_0, E_0, I_0, R_0	Initial state proportions

Model Parameters II

Note: The mortality rate $\mu = 0.02$ is fixed in the model.

Seasonality Function I

The model implements term-time forcing based on UK school schedules. The transmission rate varies between school terms and holidays:

$$\beta(t) = R_0 \cdot \text{seas}(t) \cdot (1 - e^{-(\gamma+\mu)\,dt})/dt$$

where:

$$\text{seas}(t) = \begin{cases} 1 + \text{amplitude} \cdot \frac{0.2411}{0.7589} & \text{during school term} \\ 1 - \text{amplitude} & \text{during holidays} \end{cases}$$

School terms run approximately:

- Jan 7 to Apr 10 (days 7-100)
- Apr 25 to Jul 18 (days 115-199)
- Sep 9 to Oct 27 (days 252-300)

Seasonality Function II

- Nov 4 to Dec 22 (days 308-356)

Cohort Effect I

The cohort effect captures the fact that susceptible children enter the population continuously through births, but enter school (and the school-aged mixing group) at discrete times.

At the start of the school year (day 251), a fraction `cohort` of the year's births enter the susceptible pool all at once:

$$\text{br}(t) = \begin{cases} \frac{\text{cohort}\cdot\text{birthrate}}{dt} + (1 - \text{cohort}) \cdot \text{birthrate} & |t - 251/365| < dt/2 \\ (1 - \text{cohort}) \cdot \text{birthrate} & \text{otherwise} \end{cases}$$

Extra-Demographic Stochasticity I

The force of infection includes multiplicative gamma white noise:

$$\text{foi} = \frac{\beta(I + \iota)}{N} \cdot \frac{dW}{dt}$$

where $dW \sim \text{Gamma}\left(\frac{dt}{\sigma_{SE}^2}, \sigma_{SE}^2\right)$ has:

- Mean: dt
- Variance: $\sigma_{SE}^2 \cdot dt$

This adds "environmental" or "behavioral" stochasticity beyond what demographic stochasticity provides.

Using UKMeasles.Pomp() I

The **pypomp** package provides a convenient factory function to construct the measles POMP object:

```
# Load MLEs from He et al. (2010)
mles = UKMeasles.AK_mles()
print("MLE parameters for London:")
print(mles["London"])
```

```
MLE parameters for London:
R0           56.800000
sigma        28.900000
gamma        30.400000
iota          2.900000
rho           0.488000
sigmaSE       0.087800
psi           0.116000
cohort        0.557000
```

Using UKMeasles.Pomp() II

```
amplitude      0.554000
S_0            0.029700
E_0            0.000052
I_0            0.000051
R_0            0.970000
Name: London, dtype: float64
```

Using UKMeasles.Pomp() III

```
# Extract London MLE as a dictionary
theta_mle = mles["London"].to_dict()
print("\nKey parameter values:")
print(f"  R0 = {theta_mle['R0']:.1f}")
print(f"  sigma = {theta_mle['sigma']:.1f} per year")
print(f"  gamma = {theta_mle['gamma']:.1f} per year")
print(f"  rho = {theta_mle['rho']:.3f}")
print(f"  sigmaSE = {theta_mle['sigmaSE']:.4f}")
```

```
Key parameter values:
  R0 = 56.8
  sigma = 28.9 per year
  gamma = 30.4 per year
  rho = 0.488
  sigmaSE = 0.0878
```

Using UKMeasles.Pomp() IV

```
# Construct the POMP object for London
measles_pomp = UKMeasles.Pomp(
    unit=["London"],
    theta=theta_mle,
    model="001b",
    interp_method="shifted_splines",
    first_year=1950,
    last_year=1963,
    dt=1/365.25,
    clean=True  # Remove suspicious data points
)

print(f"Time range: {measles_pomp.ys.index[0]:.2f} to {measles_pomp.ys.index[-1]:.2
print(f"Number of observations: {len(measles_pomp.ys)}")
print(f"State names: {measles_pomp.statenames}")
```

```
Time range: 1950.01 to 1963.98
Number of observations: 730
State names: ['S', 'E', 'I', 'R', 'W', 'C']
```

Model Components I

The model components are defined in `pypomp.measles.model_001b`:

```
# View available model components
print("Parameter names:", m001b.param_names)
print("\nState names:", m001b.statenames)
```

```
Parameter names: ('R0', 'sigma', 'gamma', 'iota', 'rho', 'sign

State names: ['S', 'E', 'I', 'R', 'W', 'C']
```

The model provides:

- `rinit`: Initial state simulator
- `rproc`: Process model simulator
- `dmeas`: Measurement model density
- `rmeas`: Measurement model simulator
- `to_est`/`from_est`: Parameter transformations

Evaluating the Likelihood I

Evaluating the Likelihood II

```
# Run particle filter to evaluate likelihood at MLE
key = jax.random.key(998468235)

# Multiple replications to assess Monte Carlo error
reps = 10
J = 5000

# IMPORTANT: Pass CLL=True to compute conditional log-likelihoods
measles_pomp.pfilter(
    J=J,
    key=key,
    reps=reps,
    thresh=0,
    CLL=True  # Enable conditional log-likelihood computation
)

# Access results from results_history
pf_result = measles_pomp.results_history[-1]
logliks = pf_result.logLiks.values.flatten()

ll_est = logmeanexp(logliks)
ll_se = logmeanexp_se(logliks)

print(f"Log-likelihood estimate: {ll_est:.2f}")
```

Diagnostic Plots I

Diagnostic Plots II

```
# Plot conditional log-likelihoods
# CORRECTED: Use pf_result.CLL (not cond_logLiks)
# CLL has shape: (n_theta, reps, n_times) with dims ('theta', 'replicate', 'time')
times = measles_pomp.ys.index.values

fig, ax = plt.subplots(figsize=(5, 2))

if pf_result.CLL is not None:
    cond_logliks = pf_result.CLL.values  # Shape: (n_theta, reps, n_times)
    n_reps = cond_logliks.shape[1]
    for i in range(min(5, n_reps)):
        # Index: [theta_idx, replicate_idx, time_idx]
        ax.plot(times, cond_logliks[0, i, :], alpha=0.5, linewidth=0.5)
    ax.set_xlabel('Year')
    ax.set_ylabel('Conditional Log-Likelihood')
    ax.set_title('Particle Filter Conditional Log-Likelihoods')
else:
    ax.text(0.5, 0.5, 'CLL not computed (pass CLL=True to pfilter)',
            ha='center', va='center', transform=ax.transAxes)
    ax.set_title('No CLL data available')

plt.tight_layout()
plt.show()
```

Random Walk Standard Deviations I

The IF2 algorithm requires specifying random walk standard deviations for each parameter being estimated.

Random Walk Standard Deviations II

```
# Parameters to estimate
est_params = ["R0", "sigma", "gamma", "sigmaSE", "psi",
              "cohort", "amplitude", "S_0", "E_0", "I_0", "R_0"]

# Random walk standard deviations (on transformed scale)
rw_sd = RWSigma(
    sigmas={
        "R0": 0.02,
        "sigma": 0.02,
        "gamma": 0.02,
        "iota": 0.0,       # Fixed
        "rho": 0.0,        # Fixed (can be estimated if desired)
        "sigmaSE": 0.02,
        "psi": 0.02,
        "cohort": 0.02,
        "amplitude": 0.02,
        "S_0": 0.02,
        "E_0": 0.02,
        "I_0": 0.02,
        "R_0": 0.02
    },
    init_names=["S_0", "E_0", "I_0", "R_0"]
)
```

Running IF2 I

```
key = jax.random.key(1234567)

# Run iterated filtering
measles_pomp.mif(
    J=2000,       # Number of particles
    M=50,         # Number of iterations
    a=0.95,       # Cooling rate
    rw_sd=rw_sd,
    key=key,
    thresh=0
)

# Access results
mif_result = measles_pomp.results_history[-1]
print(f"IF2 completed with {mif_result.M} iterations")
```

Convergence Diagnostics I

Convergence Diagnostics II

```
# CORRECTED: Use traces_da xarray DataArray
# traces_da has dims: [replicate, iteration, variable]
traces = mif_result.traces_da

# Plot likelihood trace
fig, ax = plt.subplots(figsize=(5, 2))
# Access logLik from the traces DataArray
loglik_trace = traces.sel(variable="logLik").values.flatten()
ax.plot(loglik_trace)
ax.set_xlabel('Iteration')
ax.set_ylabel('Log-Likelihood')
ax.set_title('IF2 Likelihood Trace')
plt.tight_layout()
plt.show()

# Plot parameter traces
params_to_plot = ["R0", "sigma", "gamma", "sigmaSE",
                  "cohort", "amplitude", "S_0", "E_0", "I_0"]
fig, axes = plt.subplots(3, 3, figsize=(5, 4))
for ax, param in zip(axes.flat, params_to_plot):
    # Select the variable from traces_da
    param_vals = traces.sel(variable=param).values.flatten()
    ax.plot(param_vals)
    ax.set_title(param)
```

Model Simulation I

```
# Simulate from the model at MLE
key = jax.random.key(42)

# CORRECTED: simulate() returns DataFrames, not arrays
# Returns (X_sims_df, Y_sims_df) with columns:
#   replicate, sim, time, state_0/obs_0, state_1/obs_1, ...
X_sims, Y_sims = measles_pomp.simulate(key=key, nsim=20)

print(f"State simulations shape: {X_sims.shape}")
print(f"Observation simulations shape: {Y_sims.shape}")
print("\nY_sims columns:", Y_sims.columns.tolist())
print("\nY_sims head:")
print(Y_sims.head())
```

Model Simulation II

```
State simulations shape: (14620, 9)
Observation simulations shape: (14600, 4)

Y_sims columns: ['replicate', 'sim', 'time', 'obs_0']

Y_sims head:
   replicate  sim          time  obs_0
0          0    0  1950.013672   60.0
1          0    0  1950.032837   24.0
2          0    0  1950.052002   39.0
3          0    0  1950.071167   46.0
4          0    0  1950.090332   39.0
```

Model Simulation III

```
# Plot simulations vs data
fig, axes = plt.subplots(2, 1, figsize=(5, 3))

times = measles_pomp.ys.index.values
data_cases = measles_pomp.ys["cases"].values

# CORRECTED: Access DataFrame columns properly
# Y_sims has columns: replicate, sim, time, obs_0
n_sims = Y_sims['sim'].nunique()

# Panel 1: All simulations
ax = axes[0]
for sim_idx in range(n_sims):
    # Filter for this simulation (replicate=0, sim=sim_idx)
    sim_data = Y_sims[(Y_sims['replicate'] == 0) & (Y_sims['sim'] == sim_idx)]
    sim_data = sim_data.sort_values('time')
    ax.plot(sim_data['time'].values, sim_data['obs_0'].values,
            alpha=0.2, color='blue', linewidth=0.5)
ax.plot(times, data_cases,
        color='red', linewidth=1.5, label='Data')
ax.set_xlabel('Year')
ax.set_ylabel('Cases')
ax.set_title('Model Simulations (blue) vs Data (red)')
ax.legend()
```

Scientific Inference I

Profile likelihoods are essential for:

1. **Valid confidence intervals**: Profile-based CIs are valid even when the likelihood surface is non-quadratic.
2. **Parameter correlations**: Profile traces reveal how parameters compensate for each other.
3. **Identifiability diagnostics**: Flat profiles indicate weak identifiability.
4. **Hypothesis testing**: Likelihood ratio tests can be constructed.

Profile Likelihood Theory I

For a focal parameter ϕ, the profile likelihood is:

$$\ell_{\text{profile}}(\phi) = \max_{\theta:\theta_1=\phi} \ell(\theta)$$

A $(1-\alpha)$ confidence interval is:

$$\{\phi : \ell_{\text{profile}}(\phi) > \max \ell - \chi^2_1(1-\alpha)/2\}$$

For 95% confidence, the cutoff is $1.92 = \chi^2_1(0.95)/2$.

Computing Profiles in pypomp I

See the supplementary document `lesson5_profile.qmd` for detailed profile computation code.

The general approach:

1. Fix the focal parameter at a grid of values
2. At each grid point, optimize remaining parameters using IF2
3. Evaluate likelihood with particle filter
4. Plot profile and extract confidence intervals

```
from pypomp import mcap

# Example: using mcap for profile analysis (with results)
# mcap_result = mcap(
#     parameter=param_grid,
#     loglik=profile_logliks,
#     level=0.95,
#     span=0.75
# )
```

Comparing to Statistical Models I

A useful diagnostic is to compare the mechanistic model's likelihood to that of a purely statistical model like ARMA.

```
from statsmodels.tsa.arima.model import ARIMA

# Fit ARMA(2,0,2) to log(cases+1)
cases = measles_pomp.ys["cases"].values
log_cases = np.log(np.maximum(cases, 1))

model = ARIMA(log_cases, order=(2, 0, 2))
fit = model.fit()

# Adjust for Jacobian
arma_loglik = fit.llf - np.sum(log_cases)

print(f"ARMA(2,0,2) log-likelihood: {arma_loglik:.2f}")
print(f"Mechanistic model log-likelihood: {ll_est:.2f}")
```

The mechanistic model should fit at least as well as ARMA if it captures the dynamics correctly.

Understanding the Estimates I

Key findings from He et al. (2010):

1. $R_0 \approx 40 - 60$: Much higher than the commonly cited value of 15-20. This reflects the fact that the susceptible fraction is much smaller than in the "virgin population" used for classical estimates.
2. $\sigma_{SE} > 0$: Extra-demographic stochasticity is essential. Without it, the model fits poorly.
3. **Cohort effect**: A substantial fraction of births enter the susceptible pool at school start, contributing to seasonal dynamics.
4. **Reporting rate** $\rho \approx 0.5$: About half of cases are reported.

Scientific Questions I

1. What does it mean that parameter estimates from fitting disagree with estimates from other data?
2. How can one interpret the correlation between infectious period and city size in the parameter estimates?
3. How do we interpret the need for extra-demographic stochasticity in this model?

Exercise 5.1: Model Diagnostics

Simulate from the fitted model and compare simulations to data.

- Do simulations capture the qualitative dynamics?
- Are the seasonal patterns reproduced?
- What aspects of the data are not well captured?

Exercise 5.2: Alternative Seasonality

Modify the seasonality function to use a sinusoidal approximation instead of term-time forcing.

- How does this affect the fit (log-likelihood)?
- What are the implications for interpretation?

Exercise 5.3: Extra-Demographic Stochasticity

Set $\sigma_{SE} = 0$ and compare with the full model.

- How does the likelihood compare?
- How do other parameters change?
- Use simulations to diagnose differences.

Exercise 5.4: Cohort Effect Identifiability

Compute a profile likelihood over the `cohort` parameter.

- Is this parameter well-identified?
- What do profile traces reveal about parameter correlations?

Exercise 5.5: Fixed Reporting Rate

Fix $\rho = 0.6$ and maximize likelihood over remaining parameters.

- How do other estimates change?
- Is the model consistent with this constraint?

Summary I

1. **Complex models are tractable** with pypomp's built-in UKMeasles module
2. **Covariates** are handled via DataFrame interface with automatic interpolation
3. **Extra-demographic stochasticity** is essential for fitting real epidemic data
4. **Profile likelihoods** provide valid inference for challenging models
5. **Model diagnostics** through simulation and benchmarking are essential

pypomp API Summary I

pypomp API Summary II

```
# Construct model
pomp_obj = UKMeasles.Pomp(
    unit=["London"], theta=theta, model="001b"
)

# Particle filter (with CLL diagnostic)
pomp_obj.pfilter(J=5000, key=key, reps=10, CLL=True)

# Access conditional log-likelihoods
pf_result = pomp_obj.results_history[-1]
cond_logliks = pf_result.CLL  # xarray DataArray, shape (n_theta, reps, n_times)
# Access replicate i: cond_logliks.values[0, i, :]

# Iterated filtering
pomp_obj.mif(J=2000, M=50, a=0.95, rw_sd=rw_sd, key=key)

# Access IF2 traces
mif_result = pomp_obj.results_history[-1]
traces = mif_result.traces_da  # xarray with dims [replicate, iteration, variable]
loglik_trace = traces.sel(variable="logLik").values

# Simulation (returns DataFrames)
X_sims, Y_sims = pomp_obj.simulate(key=key, nsim=20)
# Y_sims columns: replicate, sim, time, obs_0, obs_1, ...
```

References I

He, D., Ionides, E. L., and King, A. A. (2010). Plug-and-play inference for disease dynamics: measles in large and small populations as a case study. *J R Soc Interface*, 7:271–283.