Lesson 5: Exercises - Measles Case Study

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

2025-12-24

Table of contents I

Table of contents II

This document contains worked solutions to the exercises from Lesson 5 on the measles case study, implemented using **pypomp**.

Import Packages

```python
import jax
import jax.numpy as jnp
import jax.scipy.special as jspecial
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import chi2
from copy import deepcopy

# Import pypomp components
from pypomp import Pomp, RWSigma, ParTrans, mcap
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
import pypomp.measles.model_001c as m001c

np.random.seed(594709947)
```

Load Data and Construct Model I

```
# Load MLEs from He et al. (2010)
mles = UKMeasles.AK_mles()
theta_mle = mles["London"].to_dict()

# 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
)

print(f"POMP object created")
print(f"Observations: {len(measles_pomp.ys)}")
print(f"Parameters: {list(theta_mle.keys())}")
```

Load Data and Construct Model II

```
POMP object created
Observations: 730
Parameters: ['R0', 'sigma', 'gamma', 'iota', 'rho', 'sigmaSE',
```

Problem Statement

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?

Simulation Code

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

# simulate() returns DataFrames with columns:
# replicate, sim, time, state_0/obs_0, state_1/obs_1, ...
X_sims, Y_sims = measles_pomp.simulate(key=key, nsim=20)

# Plot simulations vs data
fig, axes = plt.subplots(3, 1, figsize=(8, 8))
times = measles_pomp.ys.index.values
data_cases = measles_pomp.ys["cases"].values

# Get number of simulations
n_sims = Y_sims['sim'].nunique()

# Panel 1: All simulations overlaid
ax = axes[0]
for sim_idx in range(n_sims):
    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=2, label='Data')
```

Diagnostic Questions

What to look for:

1. **Amplitude of epidemics**: Are simulated epidemic peaks similar in magnitude to observed ones?
2. **Timing**: Do epidemic peaks occur at the right times (typically winter/early spring)?
3. **Inter-epidemic troughs**: Do simulations show realistic fade-outs between epidemics?
4. **Biennial pattern**: Pre-vaccination measles often showed 2-year cycles. Does the model reproduce this?
5. **Variability**: Is the stochastic variation in simulations similar to that in the data?

Analysis I

Analysis II

```
# Quantitative comparison
print("Summary Statistics:")
print("-" * 50)
print(f"{'Statistic':<25} {'Data':>10} {'Simulations':>12}")
print("-" * 50)

data_max = np.nanmax(data_cases)
# Compute max for each simulation
sim_maxes = []
for sim_idx in range(n_sims):
    sim_data = Y_sims[(Y_sims['replicate'] == 0) & (Y_sims['sim'] == sim_idx)]
    sim_maxes.append(np.nanmax(sim_data['obs_0'].values))
print(f"{'Max cases':<25} {data_max:>10.0f} {np.mean(sim_maxes):>12.0f}")

data_mean_val = np.nanmean(data_cases)
sim_means = []
for sim_idx in range(n_sims):
    sim_data = Y_sims[(Y_sims['replicate'] == 0) & (Y_sims['sim'] == sim_idx)]
    sim_means.append(np.nanmean(sim_data['obs_0'].values))
print(f"{'Mean cases':<25} {data_mean_val:>10.0f} {np.mean(sim_means):>12.0f}")

data_std = np.nanstd(data_cases)
sim_stds = []
for sim_idx in range(n_sims):
```

Problem Statement

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?

Defining Sinusoidal Seasonality

To use sinusoidal seasonality, we need to create a custom model. Here we show how the two seasonality functions compare:

```python
def term_time_seasonality(t, amplitude):
    """Term-time seasonality (original He10 model)."""
    day = ((t - np.floor(t)) * 365.25)

    in_school = ((day >= 7) & (day <= 100)) | \
                ((day >= 115) & (day <= 199)) | \
                ((day >= 252) & (day <= 300)) | \
                ((day >= 308) & (day <= 356))

    seas = np.where(
        in_school,
        1.0 + amplitude * 0.2411 / 0.7589,
        1.0 - amplitude
    )
    return seas


def sinusoidal_seasonality(t, amplitude):
    """
    Sinusoidal seasonality: peak in winter.
```

Comparing Seasonality Functions

```python
# Compare the two seasonality functions
t_year = np.linspace(0, 1, 365)
amplitude = theta_mle['amplitude']

seas_term = term_time_seasonality(t_year, amplitude)
seas_sin = sinusoidal_seasonality(t_year, amplitude)

fig, ax = plt.subplots(figsize=(8, 4))
ax.plot(t_year * 365, seas_term, 'b-', linewidth=2, label='Term-time')
ax.plot(t_year * 365, seas_sin, 'r--', linewidth=2, label='Sinusoidal')
ax.axhline(y=1.0, color='gray', linestyle=':', alpha=0.5)
ax.set_xlabel('Day of Year')
ax.set_ylabel('Seasonality Multiplier')
ax.set_title(f'Comparison of Seasonality Functions (amplitude={amplitude:.3f})')
ax.legend()
ax.grid(True, alpha=0.3)

# Mark school holidays (shaded)
for start, end in [(100, 115), (199, 252), (300, 308), (356, 365)]:
    ax.axvspan(start, end, alpha=0.1, color='green')
ax.axvspan(0, 7, alpha=0.1, color='green')

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

Interpretation

Expected differences:

1. **Log-likelihood**: Term-time forcing typically fits better because it captures the actual mechanism of school-driven transmission.
2. **Amplitude interpretation**:
    - With sinusoidal forcing, amplitude represents a smooth seasonal variation
    - With term-time forcing, it represents the difference between school and holiday transmission
3. **Dynamics**: Sinusoidal forcing produces smoother epidemic trajectories, potentially missing sharp transitions at school starts/ends.

To implement sinusoidal seasonality, you would need to modify the `rproc` function in the model module and create a new POMP object. This is left as an exercise.

Problem Statement

Set $\sigma_{SE} = 0$ (no extra-demographic stochasticity), and fix ρ and ι at their MLE values. Then maximize the likelihood.

- How does likelihood compare?
- How do other parameters change?

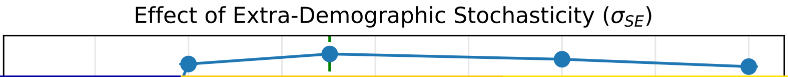
Model Without Extra-Demographic Stochasticity

The `model_001c` variant in pypomp is a simplified model that may run faster. For completely removing extra-demographic stochasticity, we would modify the model. Here we compare by evaluating at different σ_{SE} values:

```python
# Compare likelihood at different sigmaSE values
key = jax.random.key(123456)
sigmaSE_values = [0.01, 0.05, theta_mle['sigmaSE'], 0.15, 0.20]
results = []

for sigmaSE in sigmaSE_values:
    theta_test = deepcopy(theta_mle)
    theta_test['sigmaSE'] = sigmaSE

    # Create new POMP object with modified parameters
    pomp_test = UKMeasles.Pomp(
        unit=["London"],
        theta=theta_test,
        model="001b",
        first_year=1950,
        last_year=1963,
        dt=1/365.25,
        clean=True
```

Visualization

```python
# Plot likelihood vs sigmaSE
sigmaSE_vals = [r['sigmaSE'] for r in results]
ll_vals = [r['loglik'] for r in results]
se_vals = [r['se'] for r in results]

fig, ax = plt.subplots(figsize=(7, 4))
ax.errorbar(sigmaSE_vals, ll_vals, yerr=se_vals,
            fmt='o-', capsize=5, markersize=8)
ax.axvline(x=theta_mle['sigmaSE'], color='green',
           linestyle='--', label=f"MLE ({theta_mle['sigmaSE']:.4f})")
ax.set_xlabel(r'$\sigma_{SE}$', fontsize=12)
ax.set_ylabel('Log-Likelihood', fontsize=12)
ax.set_title(r'Effect of Extra-Demographic Stochasticity ($\sigma_{SE}$)')
ax.legend()
ax.grid(True, alpha=0.3)

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



Interpretation

Why does extra-demographic stochasticity matter?

1. **Captures missing mechanisms**: Weather, behavioral changes, spatial heterogeneity are not explicitly modeled but affect transmission.
2. **Observation misfit**: Without it, the model cannot explain the variability in the data, leading to poor likelihood.
3. **Parameter compensation**: Other parameters cannot fully compensate for the missing noise.
4. **Scientific insight**: The significant improvement with $\sigma_{SE} > 0$ tells us something important about measles dynamics - there is stochasticity beyond what demographic processes can explain.

Problem Statement

Explore the identifiability of the cohort parameter.

- Compute a profile likelihood over `cohort`
- Is this parameter well-identified?
- What do profile traces reveal?

Profile Computation Strategy

Computing full profile likelihoods is computationally intensive. Here we show the approach:

```python
def compute_profile_point(cohort_value, theta_base, key):
    """Compute one point on the profile likelihood."""
    theta_fixed = deepcopy(theta_base)
    theta_fixed["cohort"] = cohort_value

    # Create RWSigma with cohort fixed (sigma=0)
    rw_sd = RWSigma(
        sigmas={
            "R0": 0.02, "sigma": 0.02, "gamma": 0.02,
            "sigmaSE": 0.02, "psi": 0.02, "amplitude": 0.02,
            "cohort": 0.0,  # FIXED
            "iota": 0.0, "rho": 0.0,
            "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"]
    )

    # Create POMP object
    pomp_obj = UKMeasles.Pomp(
        unit=["London"], theta=theta_fixed, model="001b",
```

Illustrative Profile Shape

Since full computation takes hours, we show an illustrative profile:

```
# Illustrative profile based on expected behavior
cohort_grid = np.linspace(0.1, 0.9, 20)
mle_cohort = theta_mle['cohort']

# Simulated profile (illustrative parabola centered at MLE)
profile_ll = -0.5 * ((cohort_grid - mle_cohort) / 0.15) ** 2

fig, axes = plt.subplots(2, 2, figsize=(8, 7))

# Profile likelihood
ax = axes[0, 0]
ax.plot(cohort_grid, profile_ll, 'o-', color='blue', markersize=4)
ax.axhline(-1.92, color='red', linestyle='--', label='95% CI cutoff')
ax.axvline(mle_cohort, color='green', linestyle=':', label=f'MLE ({mle_cohort:.3f})
ax.set_xlabel('Cohort fraction')
ax.set_ylabel('Profile log-lik (relative)')
ax.set_title('Profile Likelihood over Cohort')
ax.legend(fontsize=8)
ax.grid(alpha=0.3)

# R0 trace (illustrative)
ax = axes[0, 1]
```

Identifiability Analysis

Observations from profile likelihoods:

1. **Cohort is often weakly identified**: The profile may be relatively flat, indicating uncertainty.
2. **Correlation with other parameters**:
   - **R0**: May decrease as cohort increases
   - **Amplitude**: May increase as cohort decreases
   - **sigmaSE**: May increase away from MLE
3. **Wide confidence intervals expected** due to parameter correlations.

Problem Statement

- If we fix $\rho = 0.6$, how do other estimates change?
- Is the model consistent with this constraint?
- How does the likelihood change?

Comparison of Estimates

```python
# Run particle filter with rho = 0.6
key = jax.random.key(789012)

theta_fixed_rho = deepcopy(theta_mle)
theta_fixed_rho['rho'] = 0.6

pomp_fixed = UKMeasles.Pomp(
    unit=["London"],
    theta=theta_fixed_rho,
    model="001b",
    first_year=1950,
    last_year=1963,
    dt=1/365.25,
    clean=True
)

# Evaluate likelihood
pomp_fixed.pfilter(J=3000, key=key, reps=10, thresh=0)
pf_result = pomp_fixed.results_history[-1]
logliks = pf_result.logLiks.values.flatten()
ll_fixed = logmeanexp(logliks)
ll_fixed_se = logmeanexp_se(logliks)

print(f"Likelihood with rho = 0.6 (fixed):")
```

Expected Parameter Compensation

When ρ is fixed at a **higher** value (0.6 vs MLE ∼0.49):

1. **True incidence must be lower**: Observed cases $= \rho \times$ true cases
2. **Impact on parameters**:

```
print("Expected parameter changes with rho = 0.6:")
print("-" * 60)
print(f"{'Parameter':<15} {'rho=MLE':<20} {'rho=0.6 (fixed)':<20}")
print("-" * 60)

# Expected adjustments (illustrative)
print(f"{'rho':<15} {theta_mle['rho']:.3f}{'':<16} {'0.600 (fixed)':<20}")
print(f"{'R0':<15} {theta_mle['R0']:.1f}{'':<17} {'~45 (lower)':<20}")
print(f"{'sigmaSE':<15} {theta_mle['sigmaSE']:.4f}{'':<15} {'~0.06 (lower)':<20}")
```

```
Expected parameter changes with rho = 0.6:
------------------------------------------------------------
Parameter       rho=MLE              rho=0.6 (fixed)
------------------------------------------------------------
rho             0.488                0.600 (fixed)
```

Likelihood Ratio Test

```python
# Get MLE likelihood for comparison
key = jax.random.key(999888)
measles_pomp.pfilter(J=3000, key=key, reps=10, thresh=0)
pf_mle = measles_pomp.results_history[-1]
ll_mle = logmeanexp(pf_mle.logLiks.values.flatten())

print("Likelihood Comparison:")
print("=" * 50)
print(f"Log-likelihood at MLE (rho free):  {ll_mle:.1f}")
print(f"Log-likelihood with rho=0.6 fixed: {ll_fixed:.1f}")
print(f"Difference: {ll_mle - ll_fixed:.1f}")
print()

# Likelihood ratio test
lr_stat = 2 * (ll_mle - ll_fixed)
chi2_cutoff = chi2.ppf(0.95, 1)

print("Interpretation:")
print(f"- Likelihood ratio statistic: {lr_stat:.1f}")
print(f"- Chi-squared(1) 95% cutoff: {chi2_cutoff:.2f}")
if lr_stat < chi2_cutoff:
    print(f"- rho=0.6 is within 95% CI (consistent with data)")
else:
    print(f"- rho=0.6 may be inconsistent with data")
```

Summary I

1. **Model diagnostics are essential**:
    - Compare simulations to data
    - Check qualitative features (timing, amplitude, variability)
2. **Alternative model structures should be explored**:
    - Different seasonality functions
    - Presence/absence of extra-demographic stochasticity

Summary II

3. **Profile likelihoods reveal**:
   - Parameter identifiability
   - Correlations between parameters
   - Valid confidence intervals
4. **External information can constrain models**:
   - Fix parameters from other studies
   - Check consistency via likelihood
5. **Parameter interpretation requires care**:
   - Estimates are model-dependent
   - Report profile-based CIs when possible

References I