Beyond Likelihoods: Bayesian Parameter Inference for Black-Box Simulators with sbi

A Hands-On Introduction to Simulation-Based Inference

EuroSciPy 2025 | Kraków, Poland | 90 minutes

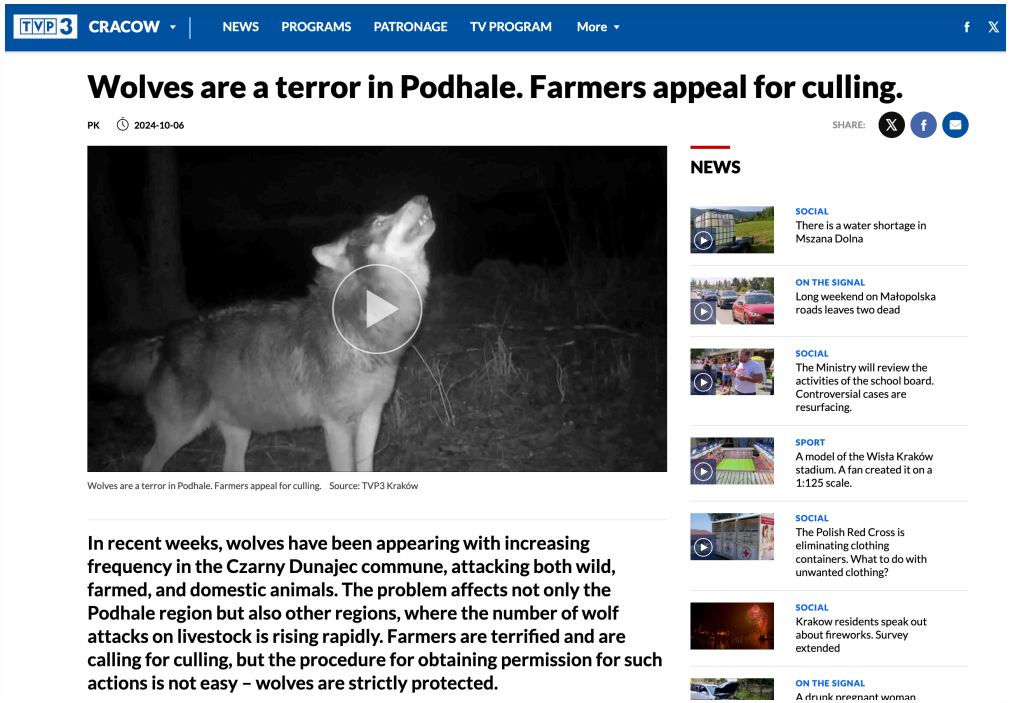
Case Study: Ecological Monitoring with Limited Data

Jan Teusen (Boelts) | TransferLab, appliedAI Institute for Europe

 **Materials:** `github.com/janfb/euroscipy-2025-sbi-tutorial`

A Real Conservation Crisis in Poland

October 2024: Headlines from Southern Poland



TVP3 CRACOW | NEWS PROGRAMS PATRONAGE TV PROGRAM More

Wolves are a terror in Podhale. Farmers appeal for culling.

PK 2024-10-06 SHARE:

Wolves are a terror in Podhale. Farmers appeal for culling. Source: TVP3 Kraków

In recent weeks, wolves have been appearing with increasing frequency in the Czarny Dunajec commune, attacking both wild, farmed, and domestic animals. The problem affects not only the Podhale region but also other regions, where the number of wolf attacks on livestock is rising rapidly. Farmers are terrified and are calling for culling, but the procedure for obtaining permission for such actions is not easy – wolves are strictly protected.

NEWS

SOCIAL
There is a water shortage in Mszana Dolna

ON THE SIGNAL
Long weekend on Małopolska roads leaves two dead

SOCIAL
The Ministry will review the activities of the school board. Controversial cases are resurfacing.

SPORT
A model of the Wisła Kraków stadium. A fan created it on a 1:125 scale.

SOCIAL
The Polish Red Cross is eliminating clothing containers. What to do with unwanted clothing?

SOCIAL
Krakow residents speak out about fireworks. Survey extended

ON THE SIGNAL
A drunk pregnant woman

TVP Kraków Reports:

"Wolves are the terror of Podhale. Farmers are calling for a cull"

The Crisis

- **Wolf attacks increasing** south Poland
- Targeting livestock and domestic animals
- **Farmers demanding action**
- **Wolves strictly protected** by law

Research Confirms the Growing Problem

ANIMAL SCIENCE AND GENETICS

ANIMAL SCIENCE AND GENETICS
Published by the Polish Society of Animal Production
vol. 20 (2024), no 4, 49-65
DOI: 10.5604/01.3001.0055.0385
Research Article *Open Access*

Preliminary report on wolf attacks on flocks of sheep of native breeds in Poland

Marta Pasternak#, Michał Puchała, Aldona Kawęcka

Department of Sheep Breeding, National Research Institute of Animal Production, 32-083 Balice n. Kraków, Poland

SUMMARY

The primary food source of wolves is wild ungulates, mainly deer, but farm animals, including sheep, may also fall prey to these predators. The aim of this study was to characterize the scale of wolf attacks on flocks of sheep of native breeds in Poland in 2015–2020, with particular emphasis on mountainous regions, considered to be the grazing areas most at risk of attacks by wolves. The study included an analysis of documentation submitted by breeders

Pasternak et al. (March 2025):

"Preliminary report on wolf attacks on flocks of sheep"

Key Findings (2015-2020)

- **76.9% of attacks** in southern Poland
- **Peak season:** July-August
- **Trend:** Increasing year over year
- **Most affected:** Podhale Zackel sheep

> *"Methods of protecting flocks should be improved"*

🎯 Your Mission: Inform Policy Decisions

You're consulting for the State Environmental Agency

The Dilemma

- **Conservation success:** Wolves recovering after near-extinction
- **Economic impact:** Farmers losing livestock
- **Policy question:** How much culling?

Your Task

- Model wolf-deer dynamics
- Infer population parameters
- **Provide uncertainty estimates**

Available Data

```
# Summary statistics from monitoring
observations = {
    "deer_mean": 45.2,
    "wolf_mean": 8.7,
    "deer_std": 12.1,
    "wolf_std": 2.4,
    "max_counts": [78, 15],
    ...
}
```

Challenge: From limited data, infer ecosystem dynamics to guide policy

🔬 Our Tool: The Lotka-Volterra Model

Classic predator-prey dynamics

The Equations

$$\frac{dx}{dt} = \alpha x - \beta xy \ , \quad \frac{dy}{dt} = \delta xy - \gamma y$$

- x = deers, y = wolves
- α = deer birth rate
- β = predation rate
- δ = wolf efficiency
- γ = wolf death rate

Why This Model?

- **Well-understood** ecological dynamics
- **Captures oscillations** seen in nature
- **Parameters map** to real processes
- **Fast to simulate** (enables SBI)

> ***Next challenge:*** *How do we infer these parameters from observations?*

The Traditional Approach: Optimization

Finding the "best" parameters

```
# Grid search or optimization
best_params = optimize(
    simulator,
    observed_data
)
```

✅ Gives an answer

❌ No uncertainty

❌ Misses alternatives

The result: A single point

```
α* = 0.52  # Birth rate
β* = 0.024 # Predation
δ* = 0.011 # Efficiency
γ* = 0.48  # Death rate
```

But how confident are we?

🎯 The Hidden Problem

Many parameters can explain your data!

Three different parameter sets, similar observations:

Parameters	α	β	δ	γ	Result
Set 1	0.52	0.024	0.011	0.48	✓ Matches
Set 2	0.48	0.026	0.009	0.51	✓ Matches
Set 3	0.55	0.022	0.012	0.45	✓ Matches

> ***Which one is correct?*** 🤔
> ***What about future predictions?*** 📈

What We Really Want: Distributions

❌ Point Estimate

- Single "best" value
- No uncertainty
- False confidence
- Poor predictions

✅ Posterior Distribution

- **Range of plausible values**
- **Quantified uncertainty**
- **Parameter correlations**
- **Robust predictions**

> ***Goal:*** `p(parameters | observation)`
>
> *The probability distribution of parameters given what we observed*

The Likelihood Problem

Why can't we just use Bayes' rule?

Bayes' Rule:

$$p(\theta|x) \propto p(x|\theta) \times p(\theta)$$

For complex simulators:

- 📦 **Black-box:** No analytical likelihood `p(x|θ)`
- 🐌 **Slow:** Likelihood evaluations infeasible

Examples: Climate models, neural circuits, epidemics, cosmology...

🚀 Enter: Simulation-Based Inference

Let neural networks learn from simulations!

```
# The SBI workflow
1. parameters ~ prior()              # Sample parameters
2. data = simulator(parameters)      # Run simulation
3. train neural_network on (parameters, data) pairs
4. posterior = neural_network(observed_data)  # Inference!
```

Key insight: Turn inference into supervised learning!

- No likelihood needed ✓
- Works with any simulator ✓
- Learns from examples ✓

What You'll Learn Today

Three hands-on exercises, progressive difficulty

📓 Exercise 1: First inference

15 minutes

- Load Lotka-Volterra simulator
- Run NPE in 5 lines
- Visualize posterior
- See uncertainty!

🔍 Exercise 2: SBI Diagnostics

20 minutes

- Posterior predictive checks
- Coverage diagnostics
- Warning signs
- "Can I trust this?"

🚀 Exercise 3: Your SBI Problem

20 minutes

- Adapt template to your simulator, OR use provided examples

Part 2: Core Intuition

Two Approaches to SBI

Classical vs Modern SBI

📚 Classical: Rejection Sampling

- Simple and intuitive
- No neural networks
- Inefficient in high-D
- Good for understanding

🧠 Modern: Density Estimation

- Efficient and scalable
- Amortized inference
- Handles high-D
- Powers the `sbi` package

> *We'll see both for intuition, then use the modern approach*

Rejection Sampling in 5 Lines

```
# The simplest SBI algorithm
accepted_params = []

for _ in range(n_simulations):
    θ = prior.sample()                    # 1. Sample parameters
    x_sim = simulator(θ)                  # 2. Simulate data
    if distance(x_sim, x_obs) < ε:        # 3. Accept if close
        accepted_params.append(θ)         # 4. Store accepted

posterior_samples = accepted_params       # 5. These approximate p(θ|x)
```

Intuition: Keep parameters that produce data similar to observations

The Curse of Dimensionality

Acceptance rate drops exponentially! 📉

Dimensions	Acceptance Rate	Simulations for 1000 samples
2D	10%	10,000 ✅
5D	0.1%	1,000,000 😐
10D	0.00001%	10,000,000,000 ❌

Problem: In high dimensions, almost nothing is "close" to your observation

> ***Solution:*** *Learn the relationship instead of rejecting!*

Neural Posterior Estimation (NPE)

Learning to predict *distributions* given data

The Network

Input: Observed data `x`

Output: Distribution `p(θ|x)`

```
# Training
for θ, x in training_data:
    loss = -log q(θ|x)
    optimize(loss)

# Inference (instant!)
posterior = q(θ|x_observed)
```

Key Innovation

Transform inference into **supervised learning**

1. Generate training pairs
2. Train neural density estimator
   - Gaussian: learn mean and std
   - Mixture of Gaussians
   - Normalizing flows
3. Instant posterior for new data!

How NPE Training Works

Three simple steps:

1️⃣ Generate Training Data

```
for i in range(n_simulations):
    θ[i] ~ prior()
    x[i] = simulator(θ[i])
```

2️⃣ Train Neural Network

```
neural_net = NeuralPosterior()
neural_net.train(parameters=θ, observations=x)
```

3️⃣ Get Posterior (instant!)

```
posterior = neural_net(x_observed)
samples = posterior.sample(10000)  # Milliseconds!
```

The Power of Amortization

Train once, infer many times! ⚡

Method	New observation	Computational Cost
MCMC	Re-run everything	Hours ⏰
Rejection	Re-run everything	Hours ⏰
NPE	Forward pass	**Milliseconds!** ⚡

Perfect for:

- Real-time applications
- Interactive exploration
- Multiple observations

🚀 Let's Code!

Three exercises, increasing complexity

📓 **Exercise 1:** First Inference (15 min)

👨‍🎓 **SBI Diagnostics:** Recap and Input (5 min)

🔍 **Exercise 2:** Diagnostics (20 min)

🎯 **Exercise 3:** Your Problem (20 min)

> ***Setup check:*** *Can everyone run this?*

```
import sbi
import torch
print("Ready for SBI! 🚀")
```

Exercise 1: Your First Inference

Wolf-Deer Dynamics from Summary Statistics!

```
# The entire SBI workflow
from sbi.inference import NPE

# 1. Setup: simulator outputs summary stats
θ = prior.sample((10_000))
x = lambda θ: compute_summary_stats(lotka_volterra(θ))

# 2. Train neural network on summary statistics
npe = NPE(prior)
npe.append_simulations(θ, x).train()

# 3. Infer parameters from observed summaries
posterior = npe.build_posterior()

# 4. Sample & visualize uncertainty!
samples = posterior.sample((1000,), x=observed_stats)
```

 **Open notebook:** `01_first_inference.ipynb`

🔍 Diagnostics for SBI

Building Trust in Neural Posteriors

What We Just Did: Recap

Neural Posterior Estimation (NPE)

1️⃣ Observe and simulate data

- Observed data as summary stats
- Choose Lotka-Volterra and prior
- Generate parameters and data

2️⃣ Trained NPE on simulations

- Neural network learned p(θ|x)
- Amortized for instant inference

3️⃣ Got posterior distributions

- Not just point estimates!
- Full uncertainty quantification
- Parameter correlations revealed

4️⃣ Made predictions

```
# Sample full trajectories
θ_post ~ posterior
x_pred = simulator(θ_post)
```

But wait... *How do we know we can trust these results?* 🤔

Why Diagnostics are Critical

SBI is approximate inference - verification essential!

⚠️ Three sources of error:

1. **Neural approximation:** Is the network accurate? Did we use enough data?
2. **Summary statistics:** Did we lose critical information?
3. **Prior specification:** Does it cover the true parameters?

Without diagnostics, you risk:

- ❌ **Overconfident conclusions** (too narrow posteriors)
- ❌ **Missing the truth** (biased inference)
- ❌ **Policy disasters** (remember the wolves!)

The Four Essential Diagnostics

Your trust-building workflow

1. Prior Predictive Check

Question: Can my prior generate realistic data?

How: Sample prior → simulate → compare to observed

2. Training Convergence

Question: Did the neural network learn properly?

How: Check loss curves, validation metrics

3. Posterior Predictive Check

Question: Can the posterior recreate observations?

How: Sample posterior → simulate → compare to observed

4. Calibration Check

Question: Are uncertainties calibrated?

How: Test if 90% CI contains truth 90% of time

Diagnostic 1: Prior Predictive Check

Start before training! 🎲

```
# Sample from prior and simulate
for _ in range(100):
    θ ~ prior()
    x = simulator(θ)
    plot(x)  # Should look reasonable!
```

✅ Good Prior

- Generates diverse, realistic data
- Covers observed range
- Includes edge cases

❌ Bad Prior

- Creates impossible scenarios
- Too narrow/wide
- Misses observed data

Example failure: Prior allows negative birth rates → Populations go extinct instantly!

Diagnostic 3: Posterior Predictive Check

Can we recreate what we observed? 🔄

```
θ_samples = posterior.sample((1000,))
for θ in θ_samples:
    x_pred = simulator(θ)
    summary_pred = compute_summaries(x_pred)

compare(summary_pred, summary_observed)
```

What to look for:

- ✅ Predicted summaries match observed
- ✅ Reasonable variation
- ✅ No systematic bias

Red flags:

- ❌ Can't recreate observations
- ❌ Too narrow/wide predictions
- ❌ Missing key features

> ***If this fails:*** *Your summary statistics likely lost critical information!*

Diagnostic 4: Simulation-Based Calibration

Are your uncertainties honest? 📏

The test:

1. Sample "true" parameters from prior
2. Simulate data and infer posterior
3. Check: Is truth in the credible interval?
4. Repeat 100+ times

```
coverage_test = []
for _ in range(100):
    θ_true ~ prior()
    x = simulator(θ_true)
    posterior = infer(x)

    # Check if truth in 90% CI
    in_ci = θ_true in posterior.confidence_interval(0.9)
    coverage_test.append(in_ci)

coverage = mean(coverage_test)  # Should be ~0.9!
```

Expected: 90% CI contains truth 90% of time

Overconfident: Coverage < 0.9 (CIs too narrow)

Underconfident: Coverage > 0.9 (CIs too wide)

Exercise 2: Trust but Verify

Critical with Summary Statistics! 🔍

Why extra important? SBI is approximate → Needs validation!

Four key diagnostics:

1. Prior Predictive Check

- Can prior generate observations?
- Catch bad prior specification

2. Training Diagnostics

- Did neural network converge?
- Check for overfitting

3. Posterior Predictive Check

- Can posterior recreate data?
- Validates summary statistics choice

4. Simulation-Based Calibration

- Are credible intervals calibrated?
- 90% CI contains truth 90% of time?

📝 **Open notebook:** `02_diagnostics.ipynb`

🔍 Recap: Diagnostics for SBI

Building Trust in Neural Posteriors

Exercise 3: Your Own Problem

Three options:

 Option A: Your Simulator

If you brought one, we'll adapt it!

 Option B: Ball Throw Physics

Simple projectile motion with air resistance

 Option C: SIR Epidemic Model

Disease spread dynamics

**  Open notebook:** `03_your_problem.ipynb`

Part 4: Next Steps

Where to go from here

Beyond NPE: The Full SBI Toolbox

Method	What it learns	Best for	Key advantage
NPE	$p(\theta\|x)$	Fast amortized inference	Instant posteriors
NLE	$p(x\|\theta)$	MCMC sampling	Exact inference
NRE	$p(\theta,x)/p(\theta)p(x)$	Model comparison	Hypothesis testing
Sequential	Iteratively	Sample efficiency	10x fewer simulations

All available in the `sbi` package with the same interface!

Advanced Topics

Where to dive deeper 🏊

Methods

- NLE+ `pyro` (**Talk Wed, 11:40, 1.38**)
- Multi-round inference (sequential)
- Flow matching, diffusion models
- Tabular Foundation Models for NPE

Applications

- Hierarchical Bayesian inference
- Expensive simulators
- High-dimensional problems
- Training-free SBI

> 📚 ***Resources:*** *Papers, tutorials, and examples at sbi.readthedocs.io*

🌍 Real-World Applications

SBI in the wild

Science

- 🧠 **Neuroscience:** Neural circuits
- 🦠 **Epidemiology:** COVID-19 models
- 🌍 **Climate:** Weather prediction
- 🔬 **Physics:** Gravitational waves
- 🧬 **Biology:** Gene regulation

Engineering

- 🚗 **Automotive:** Safety testing
- 📞 **Telecomm.:** Radio propagation

Webapp with overview of SBI applications

Join the SBI Community!

SBI Hackathon 2025, Tübingen - Join us next time!

📦 The Package

- GitHub: github.com/sbi-dev/sbi
- 700+ stars, 82+ contributors
- Active development

📚 Resources

- SBI Documentation
- New paper out **today**:
  "SBI: a practical guide"

💬 Get Help & Connect

- GitHub Discussions
- Discord Server
- 🦋 Bluesky

🤝 Contribute!

- Join the next hackathon
- Use the package, raise issues
- Help others get started

Thank You! 🙏

Questions?

- GitHub Discussions
- Discord Server
- Let's talk after the session

📱 Feedback Form

https://forms.gle/vf6rHA5DcAt2ird98

📱 Materials

github.com/janfb/euroscipy-2025-sbi-tutorial`

What will you infer? 🚀

References & Acknowledgments

🙏 Thanks To

- **Funding**: appliedAI Institute for Europe
  - 🚀 **We're hiring!** AI Research Engineer @ TransferLab, Apply here

- **Communities**: SBI community & EuroSciPy community

🛠️ Tools Used

- **Marp**: Markdown presentation ecosystem
- **Claude + Serena MCP**: AI-assisted drafting & refactoring

See also the references file

Backup Slides

Mathematical Details: NPE Loss

Training objective

The neural posterior estimator minimizes:

$$\mathcal{L} = -\mathbb{E}_{p(\theta,x)}[\log q_\phi(\theta|x)]$$

Where:

- $q_\phi(\theta|x)$ is the neural network approximation
- ϕ are the network parameters
- Expectation over joint distribution of parameters and data

Implementation: Normalizing flows for flexible distributions