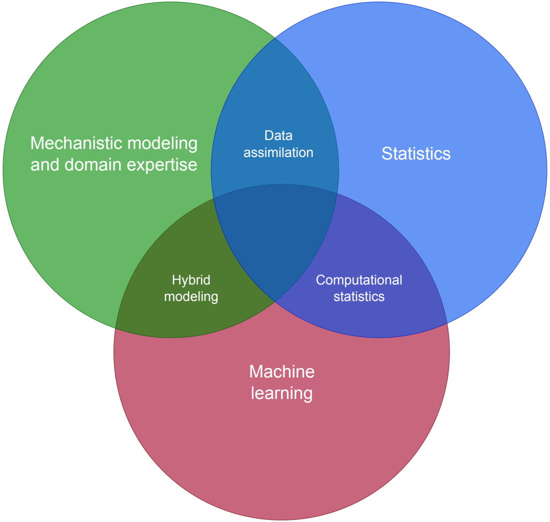
SimulationBasedInference.jl: A flexible toolkit for Bayesian inference with process-based models

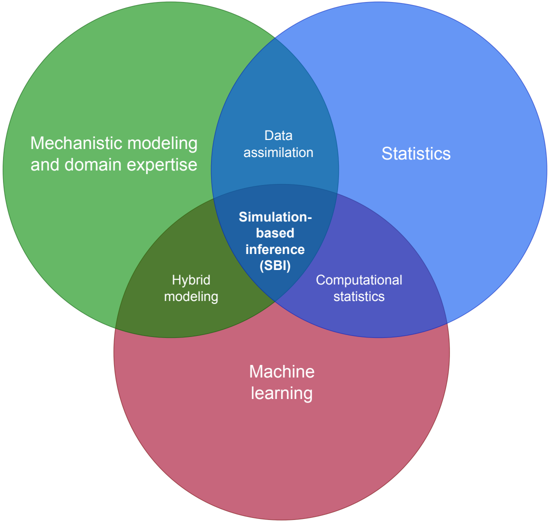
Brian Groenke[1,2,3] Kristoffer Aalstad[4] Jakob Zscheischler[1,5] Guillermo Gallego[3] Julia Boike[2,6]

[1] UFZ Leipzig, [2]AWI Potsdam, [3]TU Berlin, [4]University of Oslo, [5]TU Dresden, [6]Humboldt Universität Berlin

What is simulation-based inference?



What is simulation-based inference?

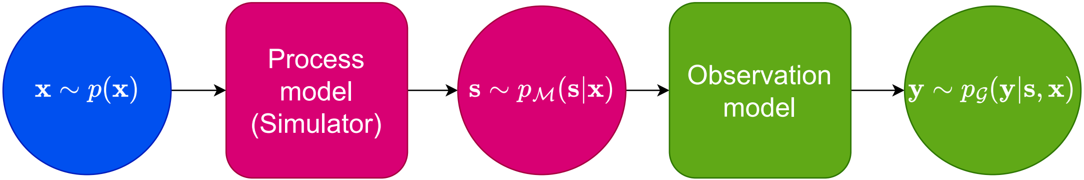


Bayesian inverse modeling

Let $\mathbf{s} = \mathcal{M}(\mathbf{x}, \mathbf{s}_0)$ represent a forward model (simulator) $\mathcal{M}$ with latent states $\mathbf{s}$, unknown or partially known inputs $\mathbf{x}$, and observation operator $\mathbf{y} = \mathcal{G}(\mathbf{s})$.

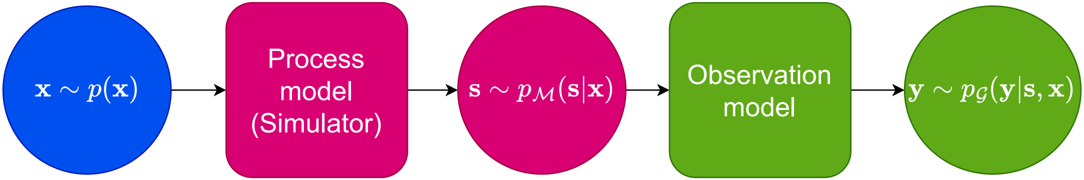
Bayesian inverse modeling

Let $\mathbf{s} = \mathcal{M}(\mathbf{x}, \mathbf{s}_0)$ represent a forward model (simulator) $\mathcal{M}$ with latent states $\mathbf{s}$, unknown or partially known inputs $\mathbf{x}$, and observation operator $\mathbf{y} = \mathcal{G}(\mathbf{s})$.



Bayesian inverse modeling

Let $\mathbf{s} = \mathcal{M}(\mathbf{x}, \mathbf{s}_0)$ represent a forward model (simulator) $\mathcal{M}$ with latent states $\mathbf{s}$, unknown or partially known inputs $\mathbf{x}$, and observation operator $\mathbf{y} = \mathcal{G}(\mathbf{s})$.



The Bayesian inverse problem given observations $\mathbf{y}$ is then:

$$p(\mathbf{s}, \mathbf{x}|\mathbf{y}) \propto p_{\mathcal{G}}(\mathbf{y}|\mathbf{s}, \mathbf{x}) p_{\mathcal{M}}(\mathbf{s}|\mathbf{x}) p(\mathbf{x}) \tag{1}$$

Machine learning in SBI

There are several ways ML can be applied within the framework of SBI:

- **Low-dimensional embedding** of high-dimensional input and output spaces

Machine learning in SBI

There are several ways ML can be applied within the framework of SBI:

- **Low-dimensional embedding** of high-dimensional input and output spaces

- **Emulation** of the simulator using (possibly physics-informed) ML

Machine learning in SBI

There are several ways ML can be applied within the framework of SBI:

- **Low-dimensional embedding** of high-dimensional input and output spaces
- **Emulation** of the simulator using (possibly physics-informed) ML
- **Data-driven estimation** of the observation noise/error model

Machine learning in SBI

There are several ways ML can be applied within the framework of SBI:

- **Low-dimensional embedding** of high-dimensional input and output spaces
- **Emulation** of the simulator using (possibly physics-informed) ML
- **Data-driven estimation** of the observation noise/error model
- **Amortized inference** via neural density estimators (NDEs)

SimulationBasedInference.jl

SimulationBasedInference.jl is a software package in the **Julia programming language** that aims to:

- Bridge the **gap between data-driven and simulation-based** (Bayesian) statistical inference

SimulationBasedInference.jl

SimulationBasedInference.jl is a software package in the **Julia programming language** that aims to:

- Bridge the **gap between data-driven and simulation-based** (Bayesian) statistical inference
- Provide a **unified interface** for embedding simulators into a Bayesian modeling framework

SimulationBasedInference.jl

SimulationBasedInference.jl is a software package in the **Julia programming language** that aims to:

- Bridge the **gap between data-driven and simulation-based** (Bayesian) statistical inference
- Provide a **unified interface** for embedding simulators into a Bayesian modeling framework
- Facilitate **rapid prototyping** and development of custom inference algorithms and **hybrid modeling** workflows

SimulationBasedInference.jl

SimulationBasedInference.jl is a software package in the **Julia programming language** that aims to:

- Bridge the **gap between data-driven and simulation-based** (Bayesian) statistical inference
- Provide a **unified interface** for embedding simulators into a Bayesian modeling framework
- Facilitate **rapid prototyping** and development of custom inference algorithms and **hybrid modeling** workflows
- Integrate with state-of-the-art software for **probabilistic** and **differentiable** programming

Why Julia?

- Julia (mostly) solves the so-called “two language problem” through just-in-time (JIT) compilation.

Why Julia?

- Julia (mostly) solves the so-called "two language problem" through just-in-time (JIT) compilation.

- Basic syntax similar to MATLAB and python with performance often close to or better than C/C++ and Fortran

Why Julia?

- Julia (mostly) solves the so-called “two language problem” through just-in-time (JIT) compilation.

- Basic syntax similar to MATLAB and python with performance often close to or better than C/C++ and Fortran

- Highly flexible and robust type system that facilitates extensive synergy between packages

Why Julia?

- Julia (mostly) solves the so-called “two language problem” through just-in-time (JIT) compilation.

- Basic syntax similar to MATLAB and python with performance often close to or better than C/C++ and Fortran

- Highly flexible and robust type system that facilitates extensive synergy between packages

- Excellent package and dependency management

Why Julia?

- Julia (mostly) solves the so-called “two language problem” through just-in-time (JIT) compilation.
- Basic syntax similar to MATLAB and python with performance often close to or better than C/C++ and Fortran
- Highly flexible and robust type system that facilitates extensive synergy between packages
- Excellent package and dependency management
- 100% free and open source

What if I don't know Julia?

- That's OK! Minimal Julia familiarity is required to use the package at a basic level.

[1] https://sbi-dev.github.io/sbi/
[2] https://bayesflow.org/

What if I don't know Julia?

- That's OK! Minimal Julia familiarity is required to use the package at a basic level.
- It is possible to define a simulator that wraps code in other languages like python, C, or Fortran.

[1] https://sbi-dev.github.io/sbi/
[2] https://bayesflow.org/

What if I don't know Julia?

- That's OK! Minimal Julia familiarity is required to use the package at a basic level.
- It is possible to define a simulator that wraps code in other languages like python, C, or Fortran.
- You can also consider similar recently developed **python** frameworks like **sbi**[1] and **bayesflow**[2].

[1] https://sbi-dev.github.io/sbi/
[2] https://bayesflow.org/

Example: Linear ODE

```julia
using SimulationBasedInference, OrdinaryDiffEq

# define dynamics
f(u,p,t) = -p[1]*u;
# define "true" parameters
p = [0.2];
# define simulation time span
tspan = (0.0,10.0);
# initial state
u0 = [1.0]
# define ODE problem
ode_prob = ODEProblem(f, u0, tspan, p)
```

Example: Linear ODE

```
# define the observable
t_save = 0.1:0.1:10.0
observable = ODEObservable(
    :y, ode_prob, t_save, samplerate=0.01
)

# define the "forward problem"
forward_prob = SimulatorForwardProblem(
    ode_prob,
    observable,
    # can add more observables here...
)
```

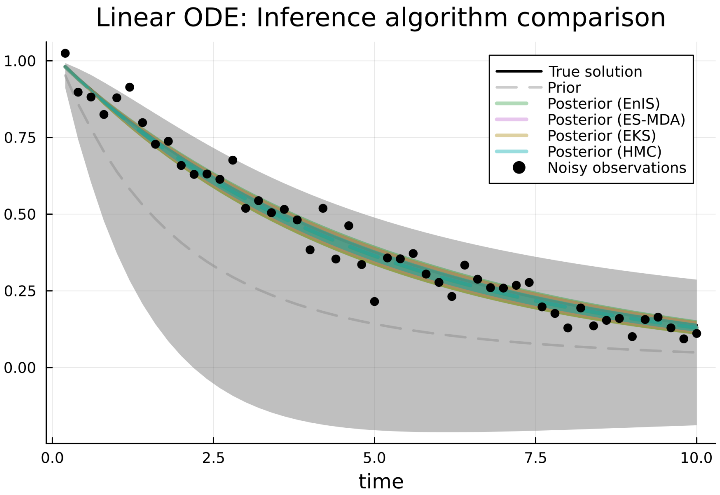
Example: Linear ODE

```
# define prior and likelihood (omitted for brevity)
simulator_prior = ...
likelihood = ...

# define inference problem
inference_prob = SimulatorInferenceProblem(
    forward_prob,
    forward_solver,
    simulator_prior,
    likelihood,
);
```

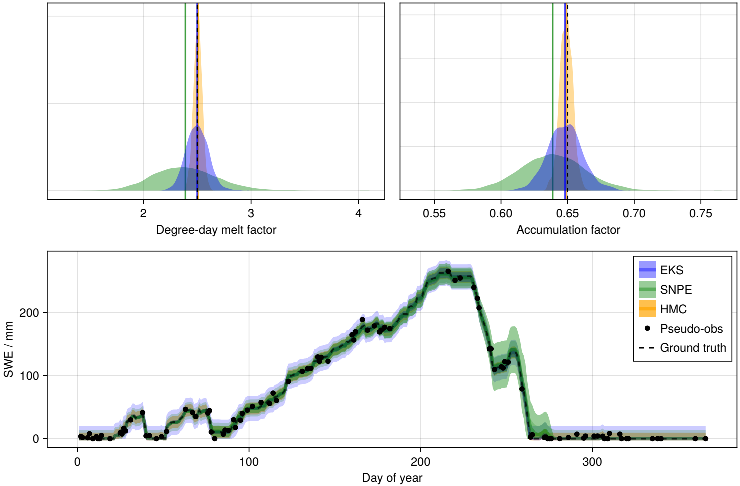
Example: Linear ODE

```
# solve with ensemble importance sampling
enis_sol = solve(inference_prob, EnIS());

# solve with ensemble smoother
esmda_sol = solve(inference_prob, ESMDA());

# solve with ensemble Kalman sampling
eks_sol = solve(inference_prob, EKS());

# solve with Hamiltonian Monte Carlo (HMC)
hmc_sol = solve(inference_prob, MCMC(NUTS()));
```

Example: Linear ODE

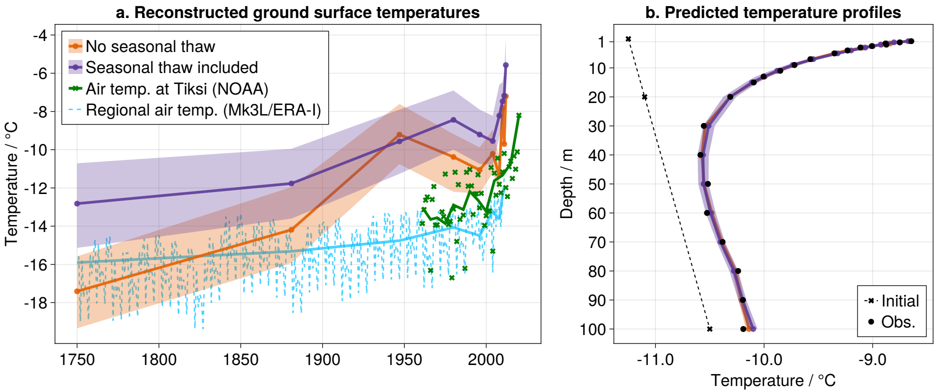


Example: Degree-day snow modeling



Calibration of degree-day snow melt model from synthetic pseudo-observations

Example: Surface temperature inversion with EKS



Groenke et al. 2024. *Robust reconstruction of historical climate change from permafrost boreholes*. JGR: Earth Surface. In review.

Conclusions

- Simulation-based inference (SBI) provides a flexible framework for combining data with physics-based models.

Conclusions

- Simulation-based inference (SBI) provides a flexible framework for combining data with physics-based models.

- There are numerous avenues for the application of ML in improving the tractability of SBI in scientific workflows.

Conclusions

- Simulation-based inference (SBI) provides a flexible framework for combining data with physics-based models.
- There are numerous avenues for the application of ML in improving the tractability of SBI in scientific workflows.
- SimulationBasedInference.jl provides a flexible and user-friendly framework for applying SBI to scientific models both big and small.

Thank you!

brian.groenke@awi.de