Simulation-based Inference

Inverting Simulators

Talk by Stefan Wezel

mlcolab @ Tübingen University Cluster of Excellence

May 3, 2021

Overview

- The problem setting
- Traditional approaches and their issues
- Using Neural Nets to alleviate them
- Applying SBI

- Inverting simulators?
- What is a simulator?
  - Forward, generative model with parameters and stochasticity
  - Produces observations
  - In context of this talk computer program, but can be electrical circuit (Hodgkin–Huxley model)
- Used by scientists to model empirical observed data
- Used in particle physics, population genetics, epidemiology
- Encode knowledge about systems gathered over centuries

What and Why?

An Example

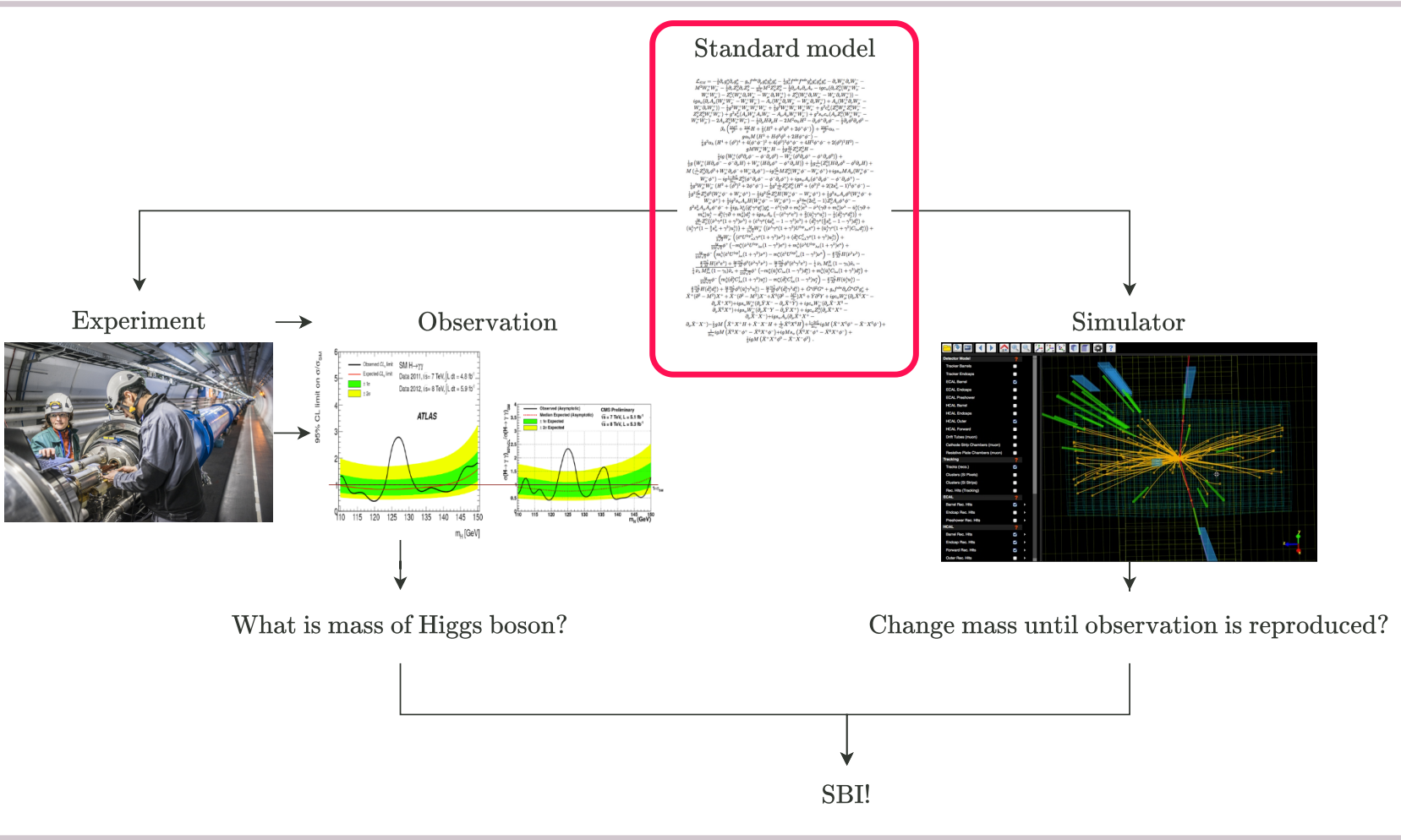


Figure derived from cern.ch, Achintya Rao and Tom McCauley, sciencealert.com

What and Why?

An Example

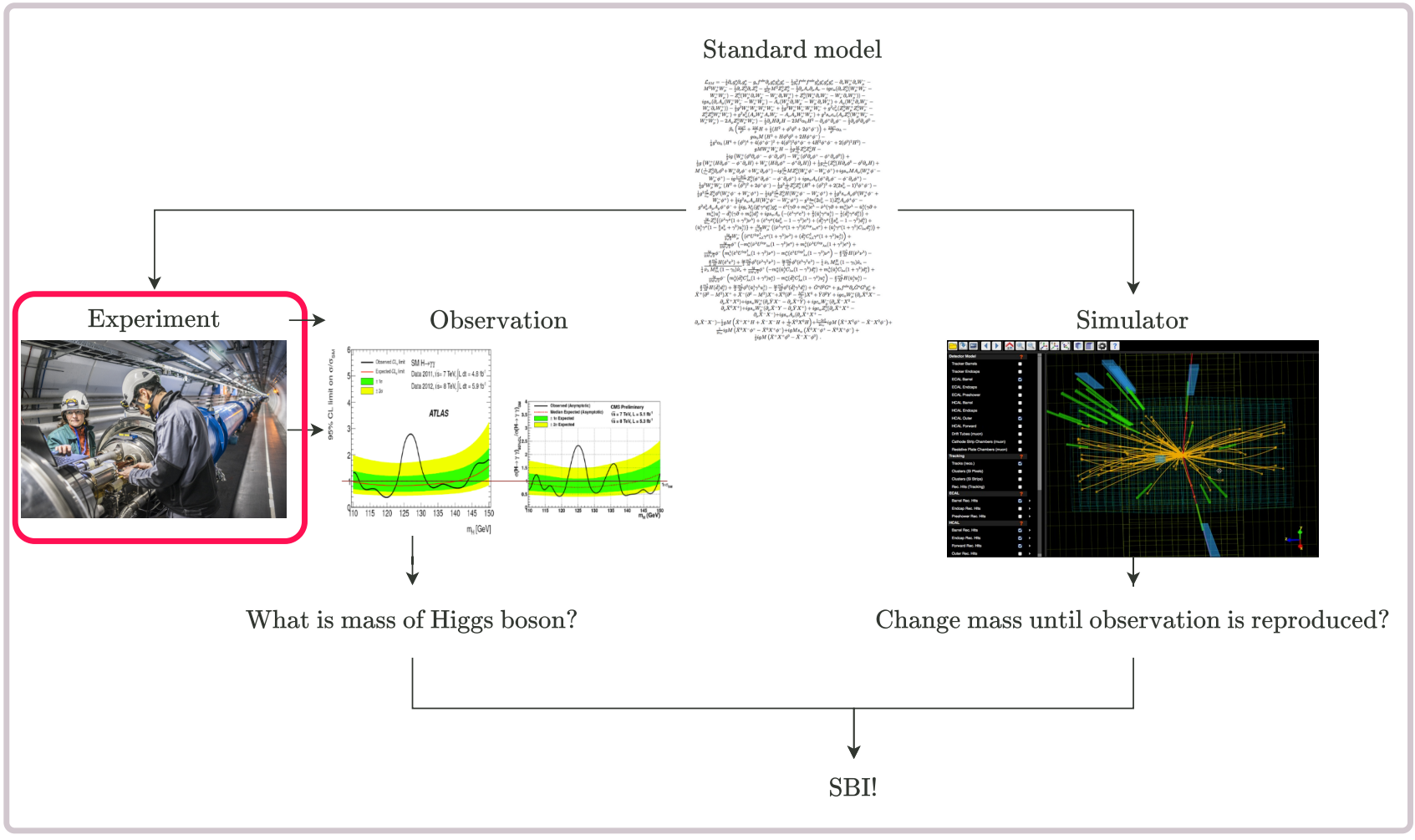


Figure derived from cern.ch, Achintya Rao and Tom McCauley, sciencealert.com

What and Why?

An Example

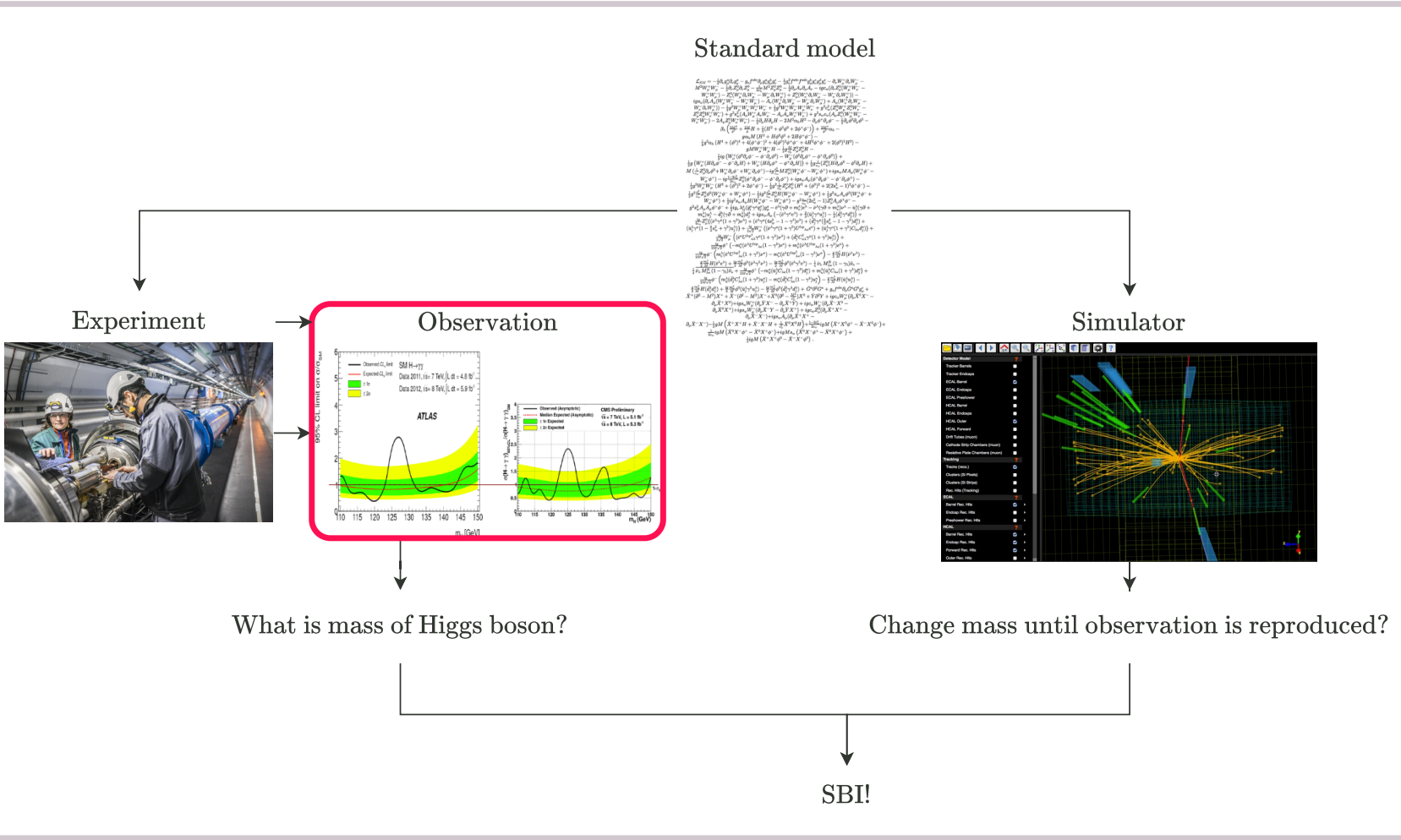


Figure derived from cern.ch, Achintya Rao and Tom McCauley, sciencealert.com

What and Why?

An Example

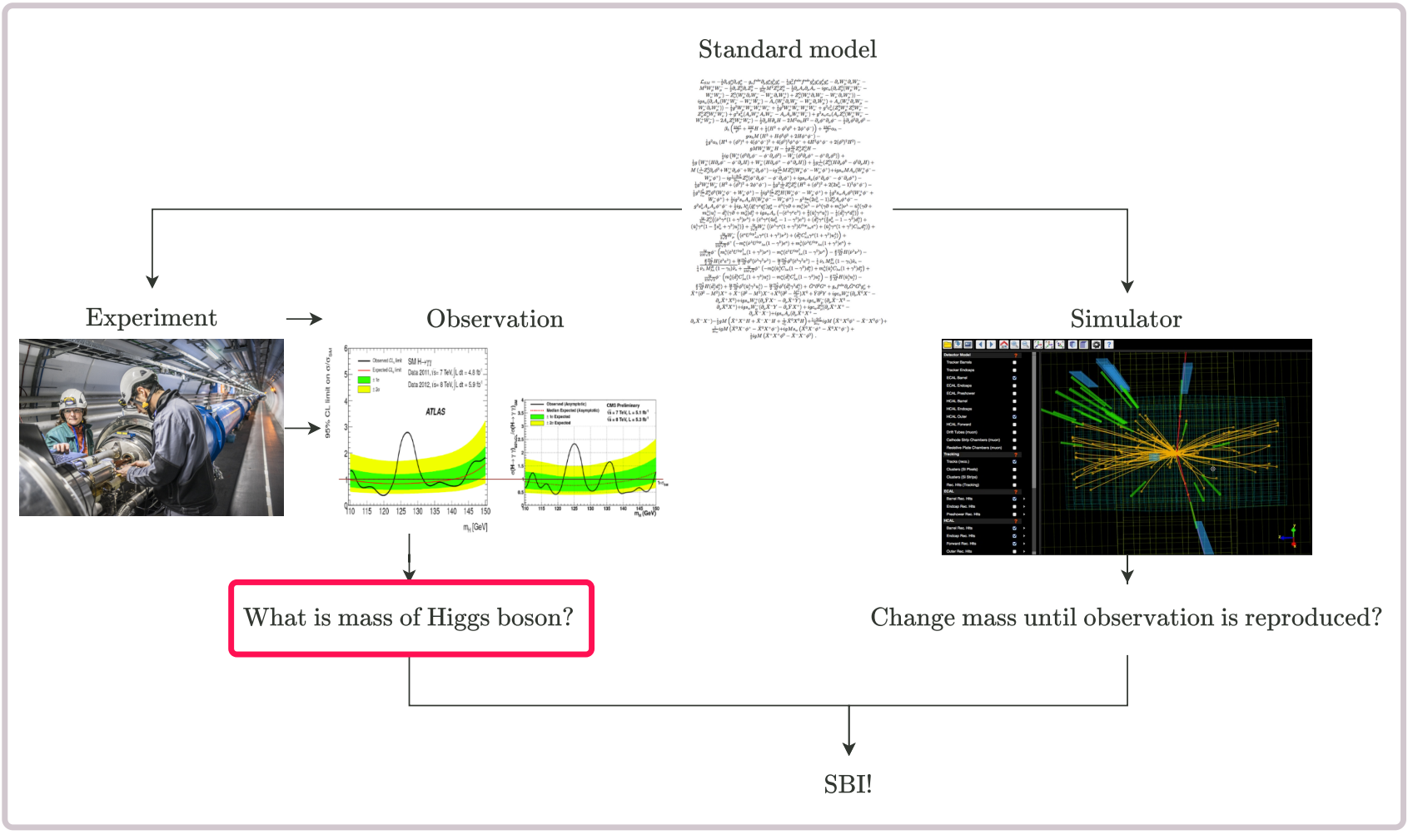


Figure derived from cern.ch, Achintya Rao and Tom McCauley, sciencealert.com

What and Why?

An Example

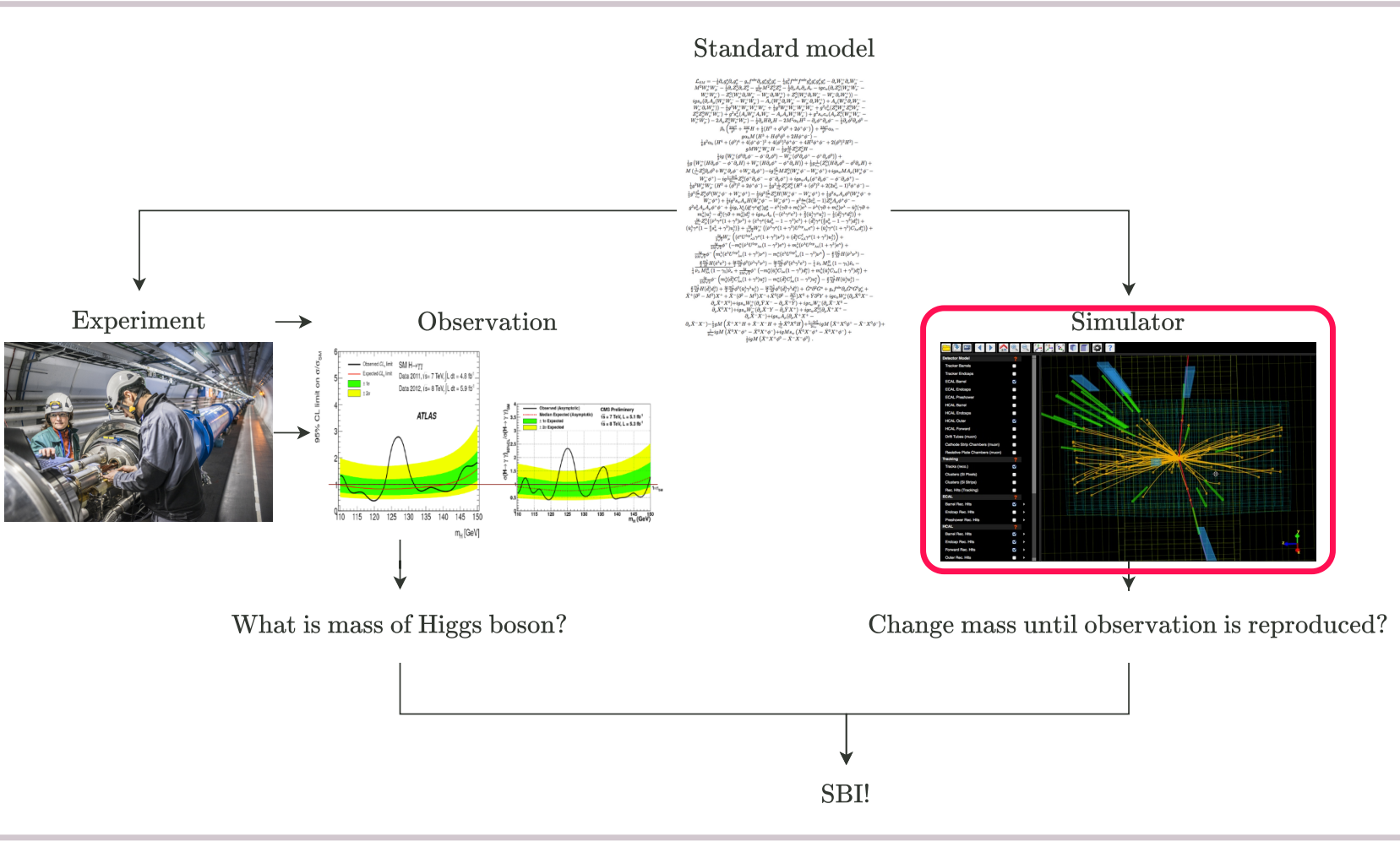


Figure derived from cern.ch, Achintya Rao and Tom McCauley, sciencealert.com

What and Why?

An Example

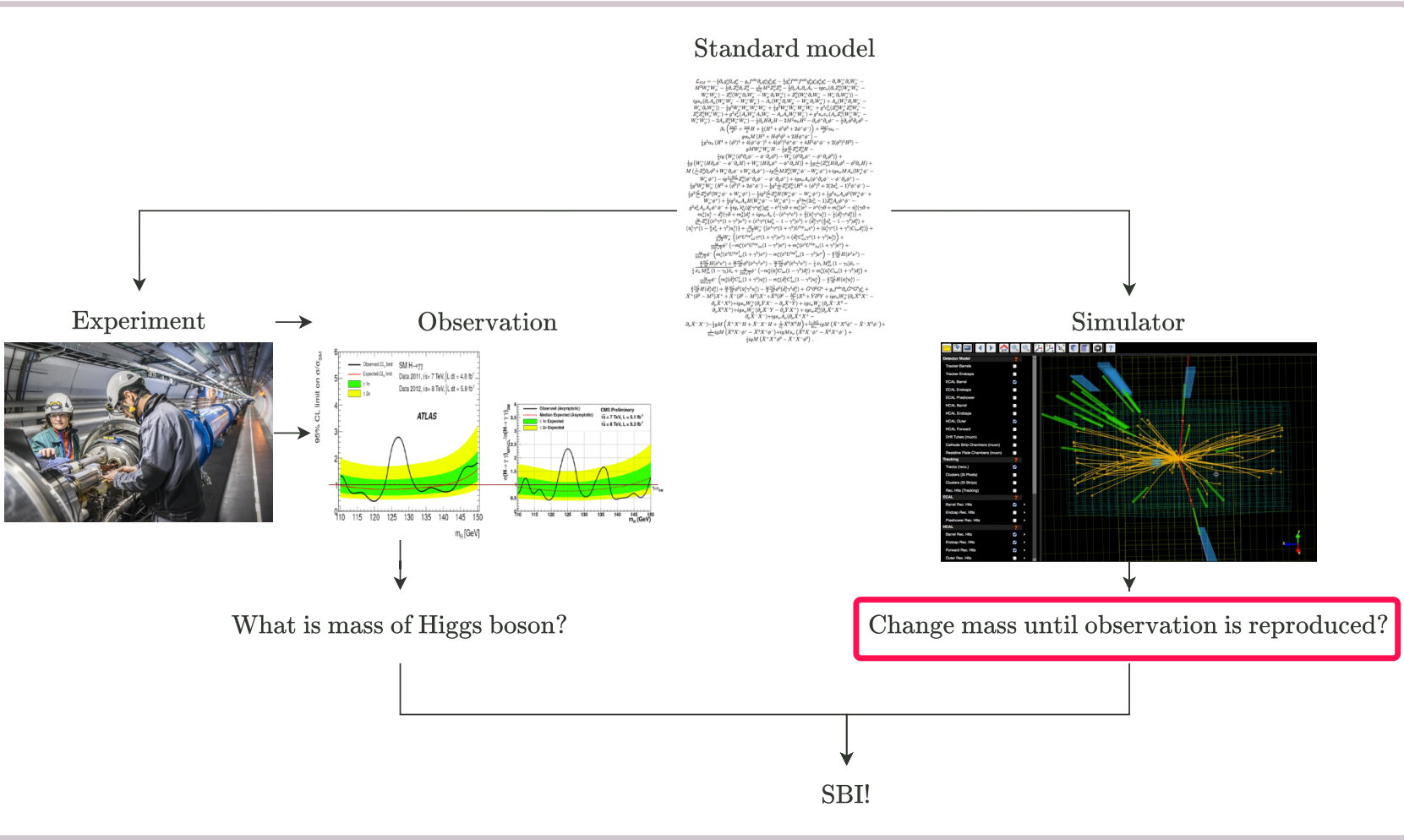


Figure derived from cern.ch, Achintya Rao and Tom McCauley, sciencealert.com

What and Why?

An Example

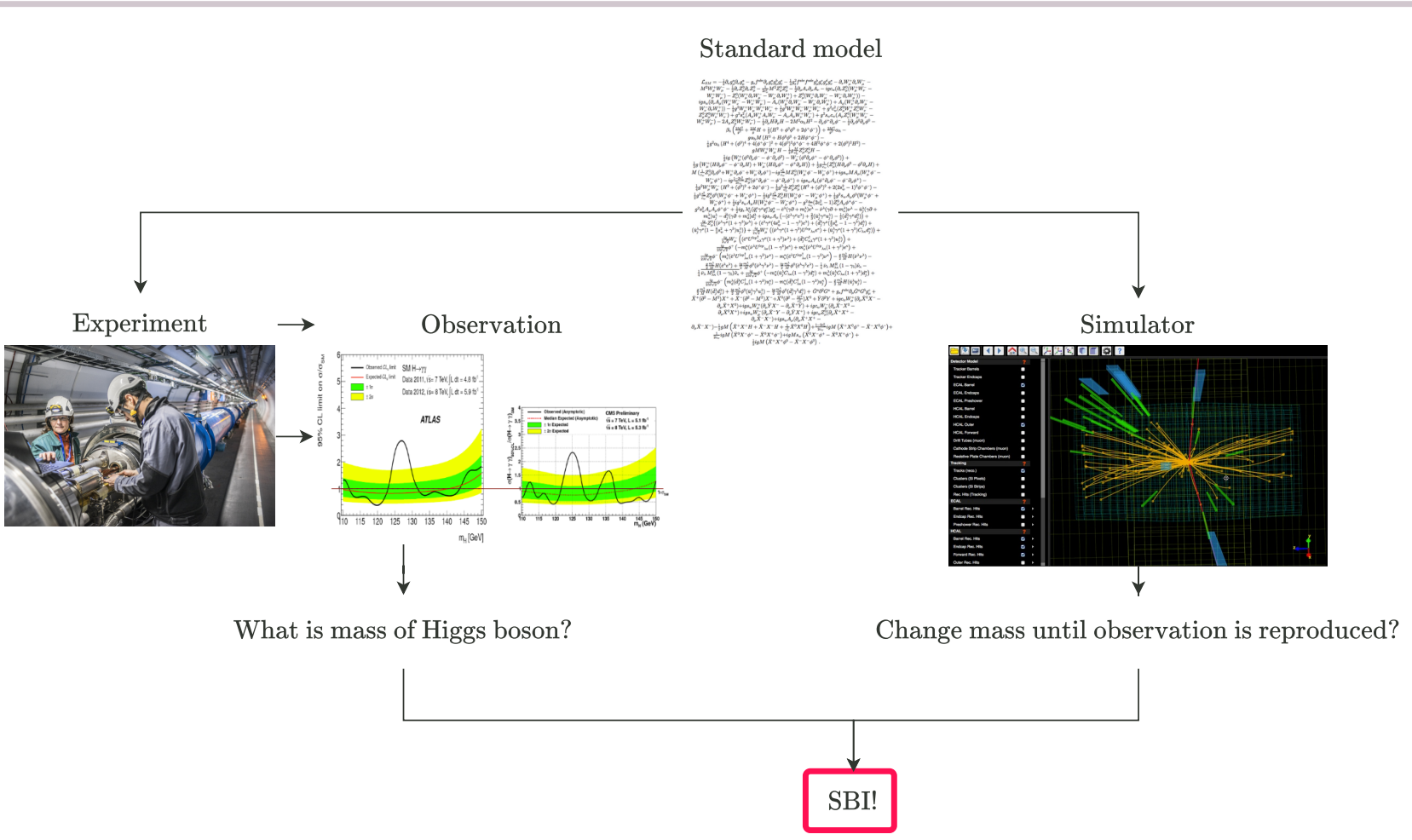


Figure derived from cern.ch, Achintya Rao and Tom McCauley, sciencealert.com

- Before we go into SBI, let's take a minute to formalize our setting
- Simulator has parameters θ, nuisance parameter z and produces data $\hat{x}$
  - It implicitly defines $p(\hat{x}, z|\theta)$ (it yields samples from this distribution)
- Given a (real) observation x_0, we would be interested in the parameters that produced it

$$p(\theta|x = x_0) = \frac{p(x|\theta)p(\theta)}{p(x)} = \frac{\overbrace{\int p(x, z|\theta)dz}^{intractable}\, p(\theta)}{\underbrace{\int p(x|\theta)p(\theta)d\theta}_{intractable}}, \tag{1}$$

What and Why?

Simulation-based Inference to the rescue

- This is exactly a problem setting that scientists often deal with
  - They have a sophistic model (the simulator) that encapsulates a lot of prior knowledge
  - But inference in this setting often boils down to: What were parameters that produced this observation
- Simulation-based Inference inference tries to solve this problem
- by inverting the simulator

- Broader term: Approximate Bayesian Computation (ABC)
- Rejection ABC
  - Change θ until something ϵ-close to x_0 is produced
  - Approximates $p(\theta|\ ||\ x - x_0\ ||< \epsilon)$
  - Inefficient for small ϵ
  - No actual posterior but only point estimate
- Some improvements
  - Sampling ABC (perturb matched parameters) [6]
  - Sequential ABC (simulate sequence of slowly-changing distribution with importance sampling) [1, 2]
- Point-estimate to posterior over parameters for ϵ-ball around observation

- Papamakarios and Murray [7] learn approximation to actual posterior
- Use simulator to create training set
  - $\{x_j, \theta_j\}_{j \in N}$
- Train neural network to predict parameters of posterior over θs

Simulation-based Inference

Predicting Parameter Posterior

for $n = 1..N$ **do**
  sample $\theta_n \sim p(\theta)$
  sample $x_n \sim p(x|\theta_n)$
end
train $q_\phi(\theta|x)$ on $\{x_n, \theta_n\}$
sample $\theta \sim q(\theta|x = x_0)$

Pseudocode derived from [7]

for $n = 1..N$ **do**
  sample $\theta_n \sim p(\theta)$
  sample $x_n \sim p(x|\theta_n)$
end
train $q_\phi(\theta|x)$ on $\{x_n, \theta_n\}$ (More in a second)
sample $\theta \sim q(\theta|x = x_0)$

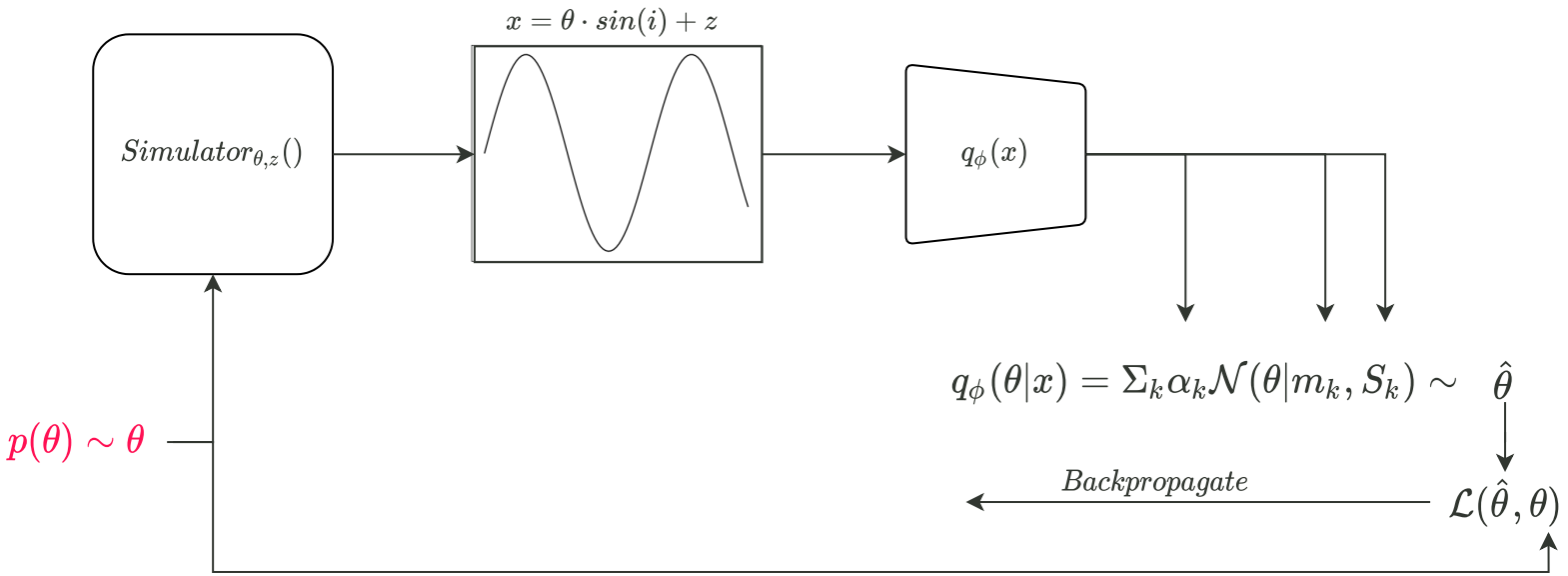
Pseudocode derived from [7]

for $n = 1..N$ do
  sample $\theta_n \sim p(\theta)$
  sample $x_n \sim p(x|\theta_n)$
end
train $q_\phi(\theta|x)$ on $\{x_n, \theta_n\}$
sample $\theta \sim q(\theta|x = x_0)$

Pseudocode derived from [7]

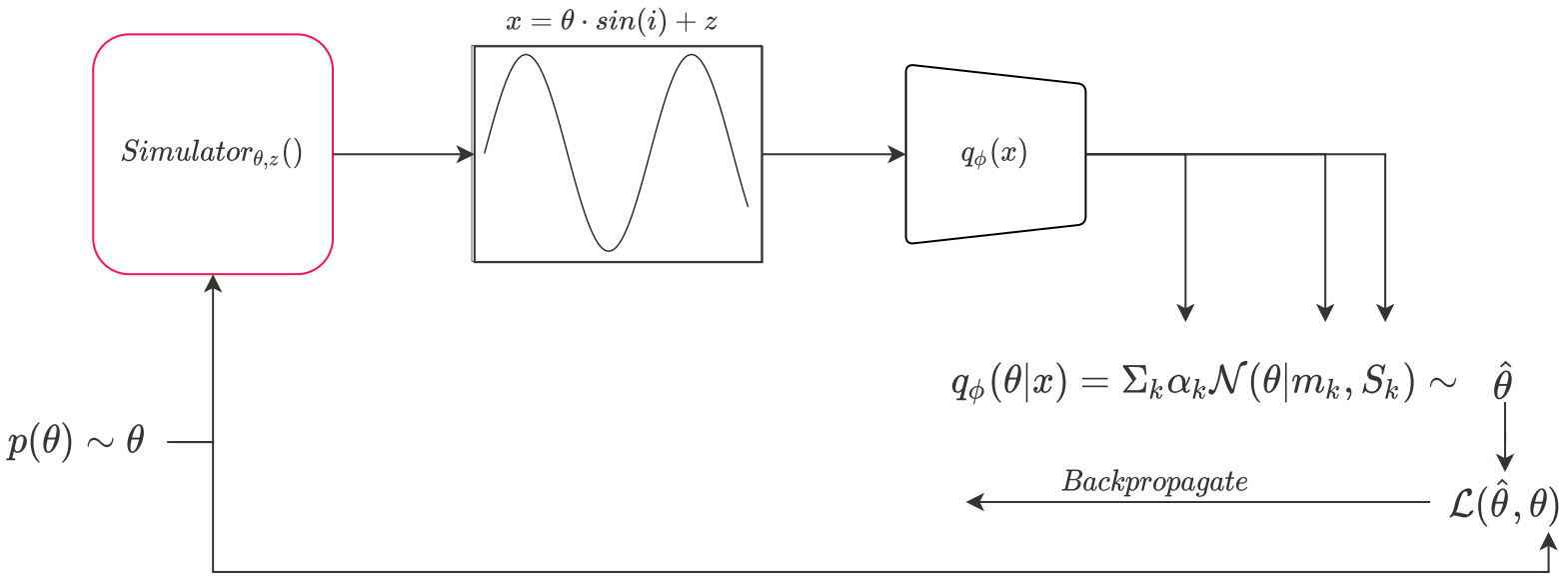
Learning a Posterior

from Simulations



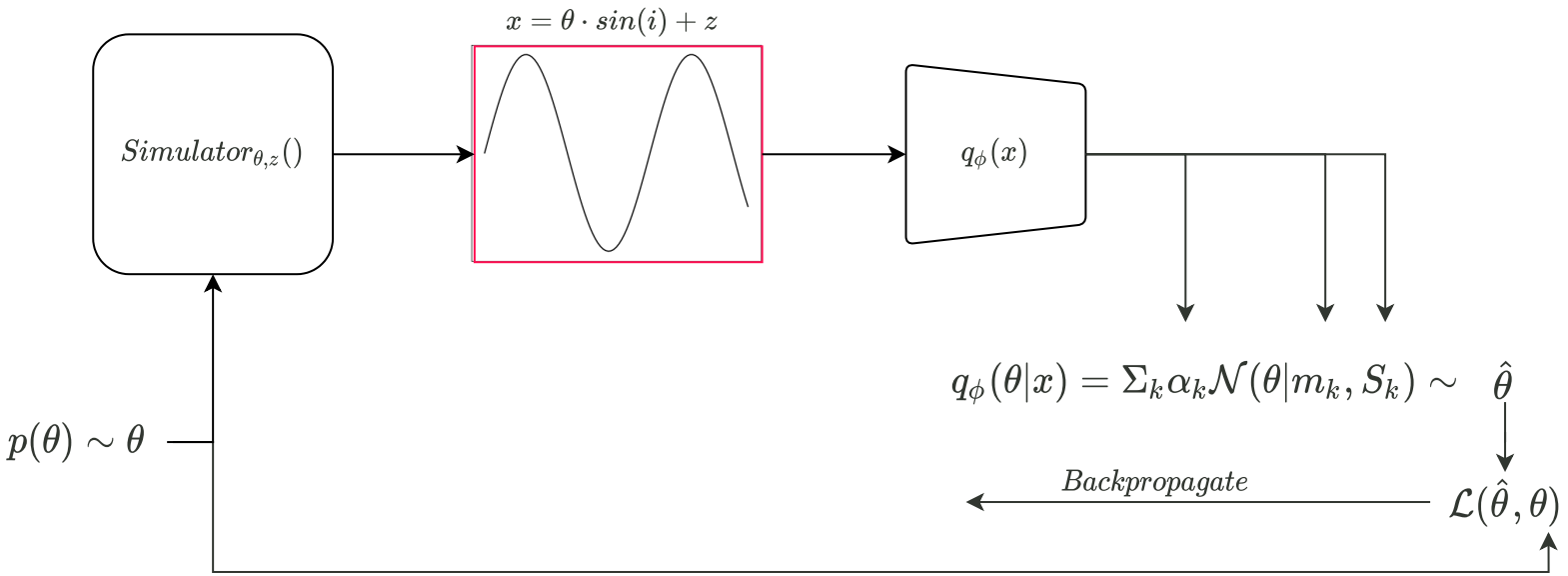
Learning a Posterior

from Simulations



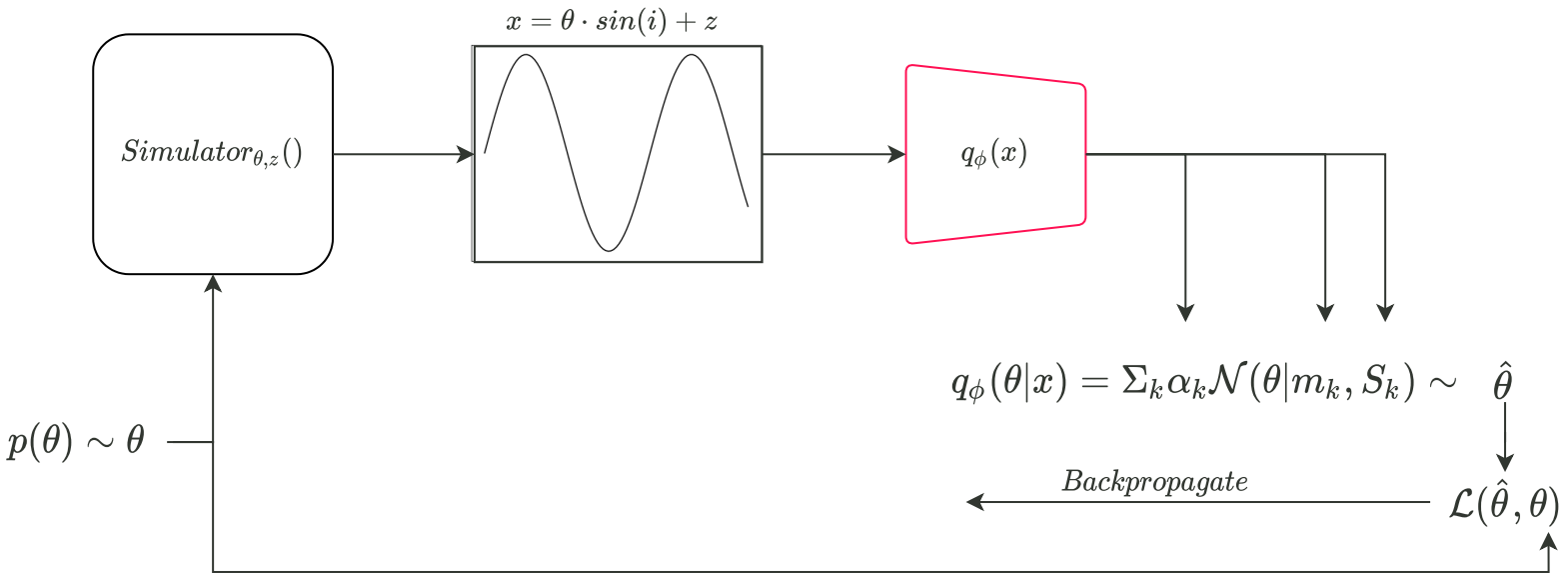
Learning a Posterior

from Simulations



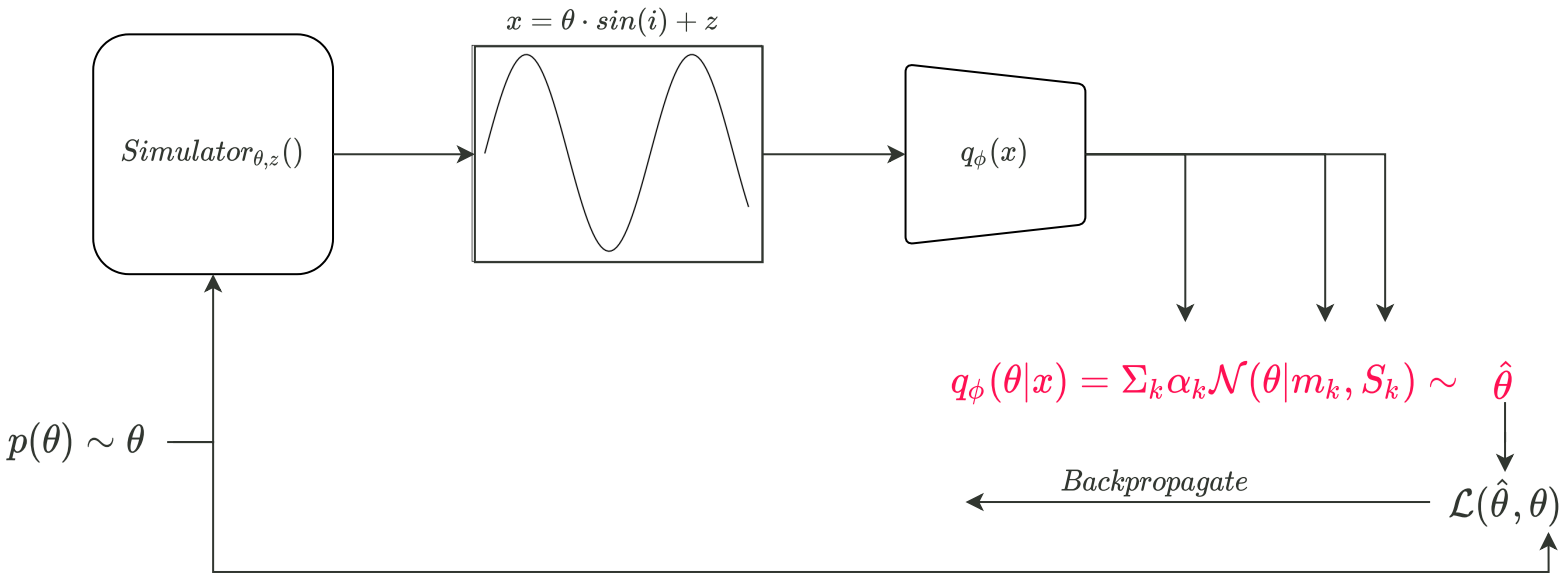
Learning a Posterior

from Simulations



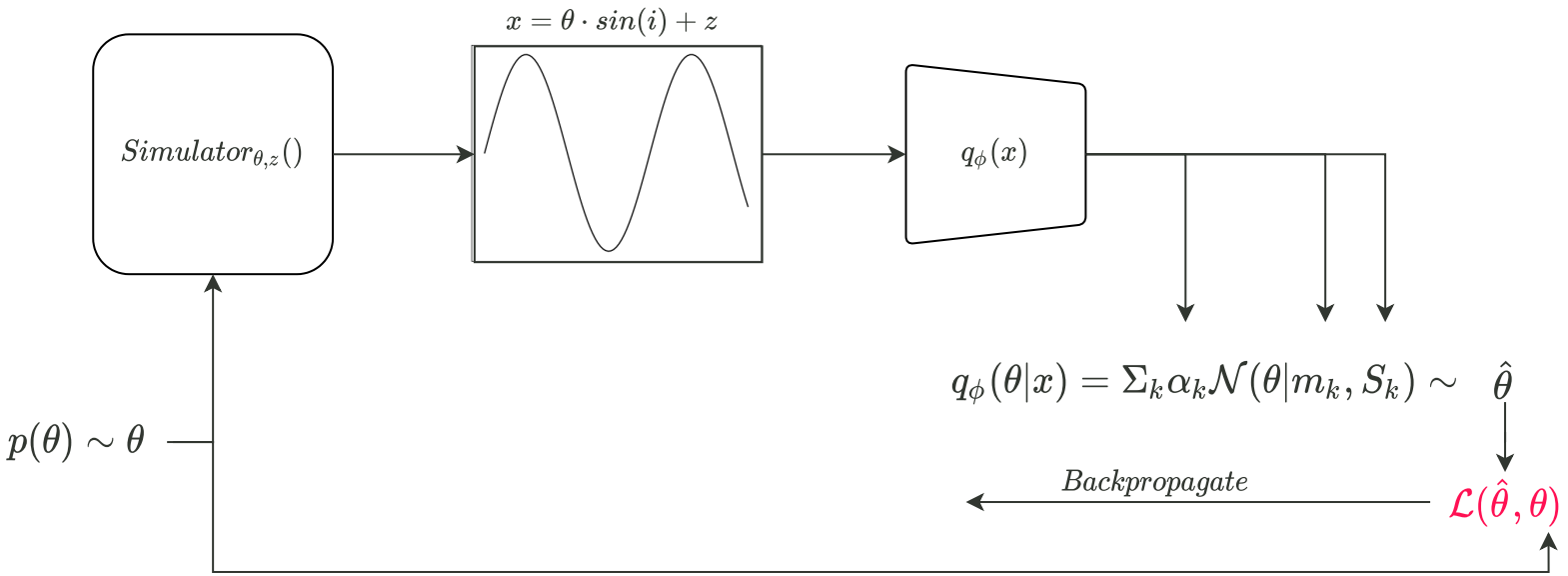
Learning a Posterior

from Simulations



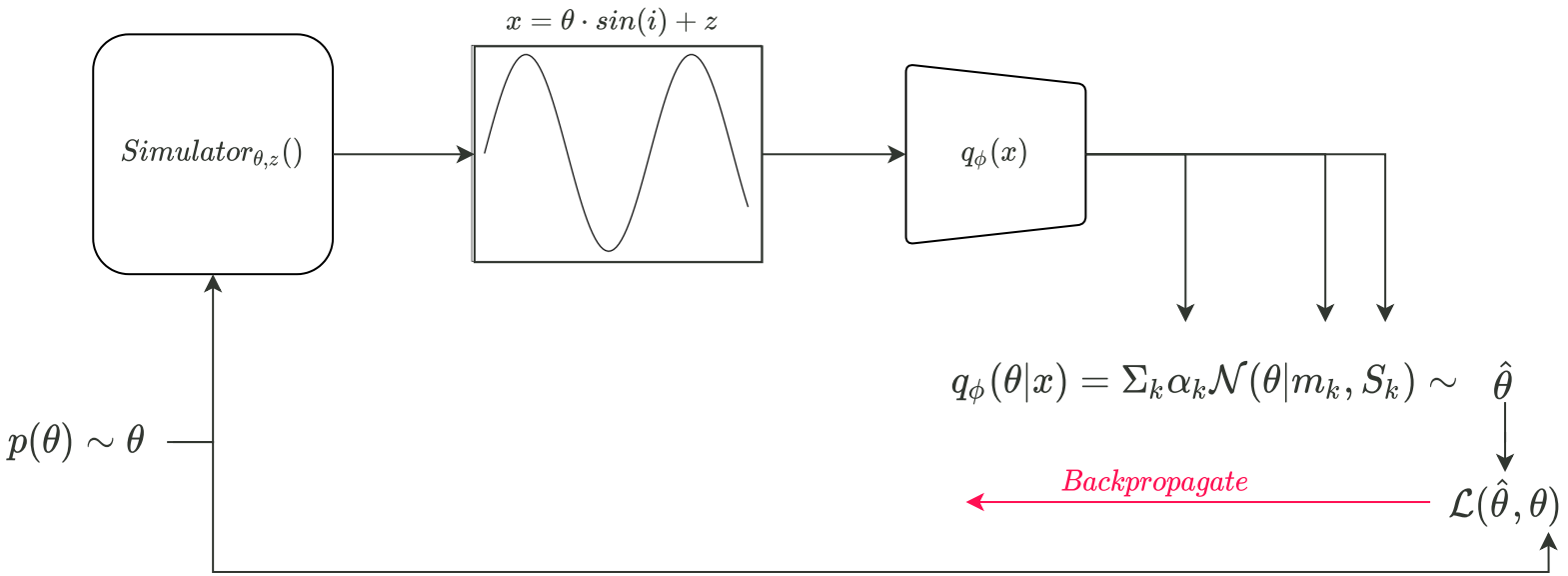
Learning a Posterior

from Simulations



Learning a Posterior

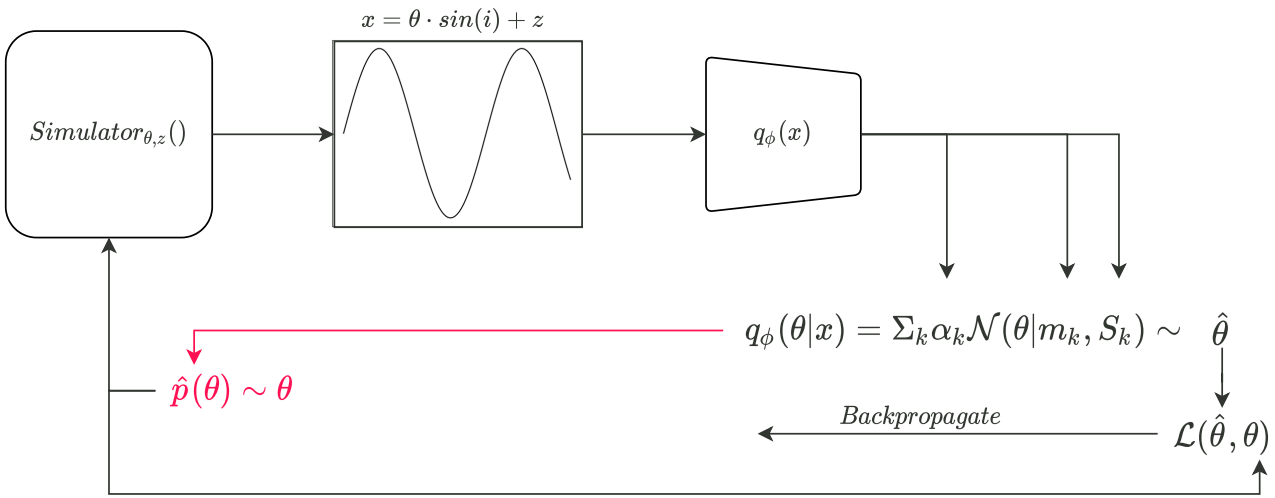
from Simulations



Updating the Prior

With the Posterior

- What if we only care about single observation?
- Iteratively replace prior with posterior
- Improves convergence speed



Updating the Prior

With the Posterior

$\tilde{p}(\theta) \leftarrow p(\theta)$

repeat

  for $n = 1..N$ **do**

    sample $\theta_n \sim \tilde{p}(\theta)$

    sample $x_n \sim p(x|\theta_n)$

  end

  train $q_\phi(\theta|x)$ on $\{x_n, \theta_n\}$

  $\tilde{p}(\theta) \leftarrow q_\phi(\theta|x)$

until $\tilde{p}(\theta)$ has converged;

Pseudocode derived from [7]

Updating the Prior

With the Posterior

$\tilde{p}(\theta) \leftarrow p(\theta)$

repeat

  for $n = 1..N$ **do**

    sample $\theta_n \sim \tilde{p}(\theta)$

    sample $x_n \sim p(x|\theta_n)$

  end

  train $q_\phi(\theta|x)$ on $\{x_n, \theta_n\}$

  $\tilde{p}(\theta) \leftarrow q_\phi(\theta|x)$

until $\tilde{p}(\theta)$ has converged;

Pseudocode derived from [7]

Updating the Prior

With the Posterior

$\tilde{p}(\theta) \leftarrow p(\theta)$

repeat

  for $n = 1..N$ **do**

    sample $\theta_n \sim \tilde{p}(\theta)$

    sample $x_n \sim p(x|\theta_n)$

  end

  train $q_\phi(\theta|x)$ on $\{x_n, \theta_n\}$

  $\tilde{p}(\theta) \leftarrow q_\phi(\theta|x)$

until $\tilde{p}(\theta)$ has converged;

Pseudocode derived from [7]

Updating the Prior

With the Posterior

$\tilde{p}(\theta) \leftarrow p(\theta)$

repeat

  for $n = 1..N$ **do**

    sample $\theta_n \sim \tilde{p}(\theta)$

    sample $x_n \sim p(x|\theta_n)$

  end

  train $q_\phi(\theta|x)$ on $\{x_n, \theta_n\}$

  $\tilde{p}(\theta) \leftarrow q_\phi(\theta|x)$

until $\tilde{p}(\theta)$ has converged;

Pseudocode derived from [7]

Applying SBI

to Hodgkin–Huxley

- Lueckmann et al. [5] apply SBI to neuro-science setting
  - Diverging simulations
  - Corrupt data
  - Summary statistics challenging
- They propose
  - Posterior over network weights
  - Train classifier to predict whether parameters fail
  - Extract Features with RNN
- Apply Hodgkin-Huxley model [4]
  - Dynamics of neuron's membrane
  - Parameters (concentration of sodium, potassium, ...)
  - Numerical simulators available (NEURON software [3])

Applying SBI

Results

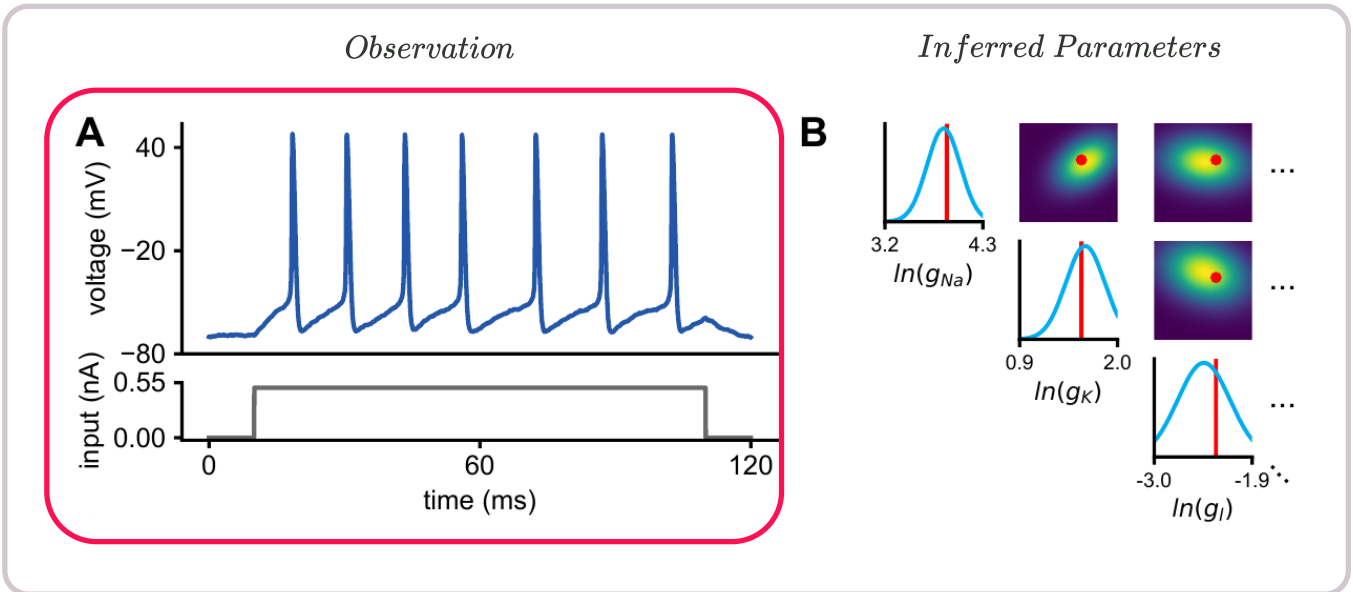


Figure derived from [5]

Applying SBI

Results

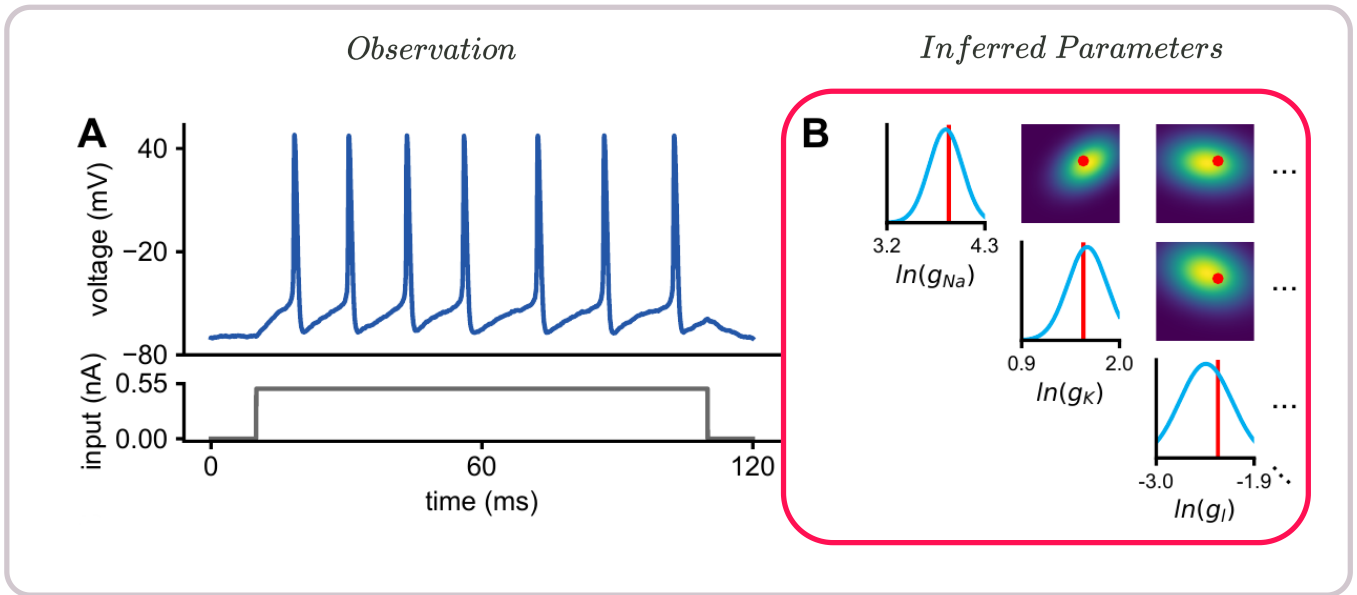


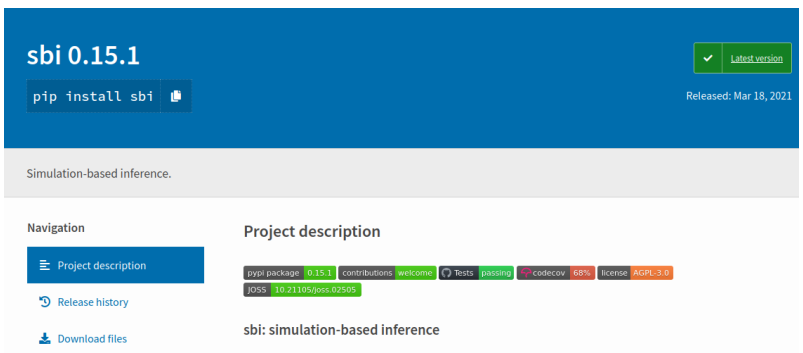
Figure derived from [5]

- Example

How to use SBI?

Some advertisement

- mlcolab and mackelab is developing a sbi python library
- If you are interested: try it out and give us your feedback
- `colab.research.google.com/drive/1L1T45hruoScyu3hN4WkDW6f0De4UemjI?usp=sharing`
- `github.com/mackelab/sbi`
- `pip install sbi`



References

[1] M. A. Beaumont, J.-M. Cornuet, J.-M. Marin, and C. P. Robert. Adaptive approximate bayesian computation. Biometrika, 96(4):983–990, 2009.

[2] F. V. Bonassi, M. West, et al. Sequential monte carlo with adaptive weights for approximate bayesian computation. Bayesian Analysis, 10(1):171–187, 2015.

[3] N. T. Carnevale and M. L. Hines. The NEURON book. Cambridge University Press, 2006.

[4] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. The Journal of physiology, 117(4):500–544, 1952.

[5] J.-M. Lueckmann, P. J. Goncalves, G. Bassetto, K. Öcal, M. Nonnenmacher, and J. H. Macke. Flexible statistical inference for mechanistic models of neural dynamics. arXiv preprint arXiv:1711.01861, 2017.

[6] P. Marjoram, J. Molitor, V. Plagnol, and S. Tavaré. Markov chain monte carlo without likelihoods. Proceedings of the National Academy of Sciences, 100(26):15324–15328, 2003.

[7] G. Papamakarios and I. Murray. Fast epsilon-free inference of simulation models with bayesian conditional density estimation. arXiv preprint arXiv:1605.06376, 2016.