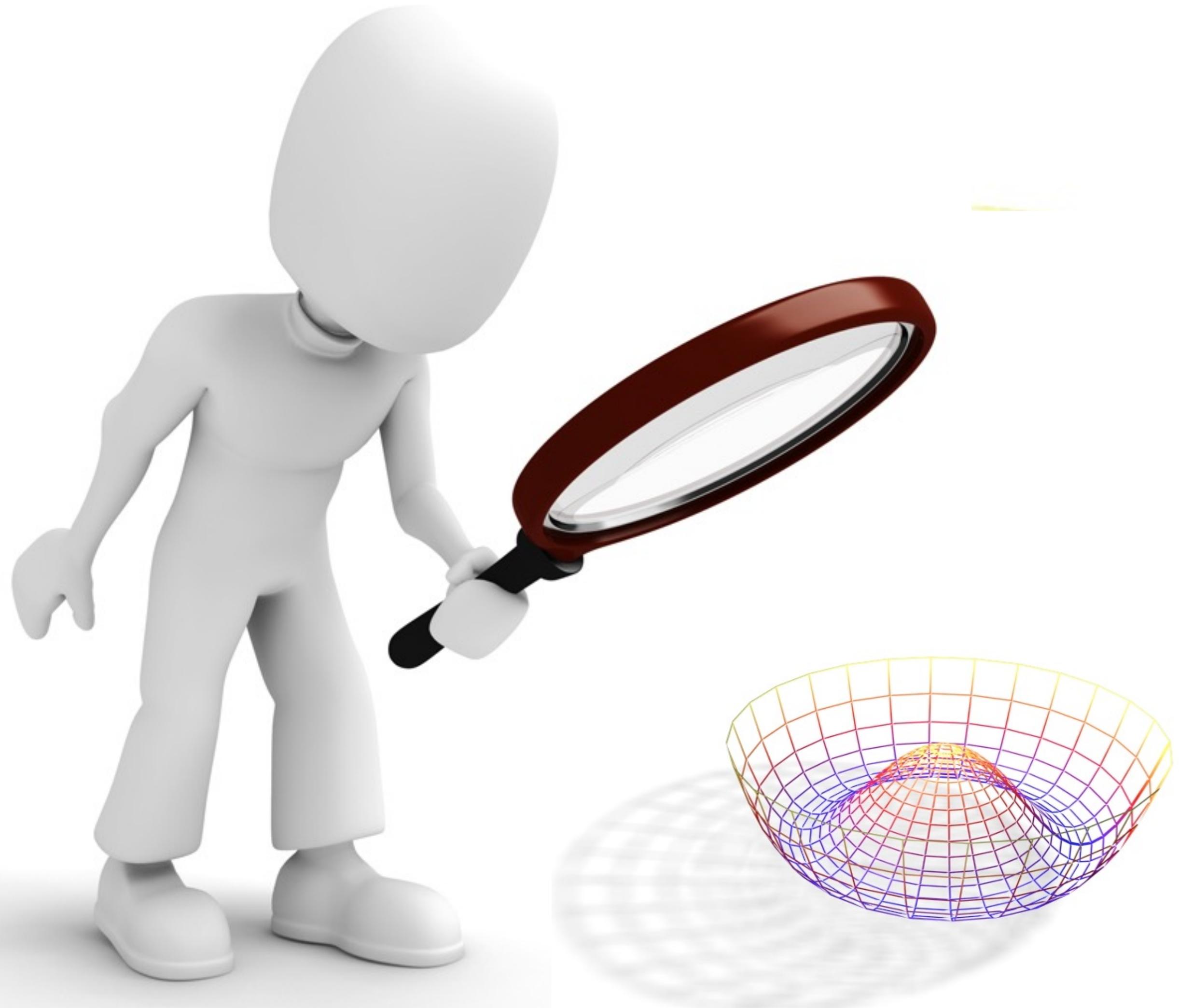




MADMINER TUTORIAL

+ TOOLS & INFRASTRUCTURE



@KyleCranmer

New York University

Department of Physics

Center for Data Science

CILVR Lab

Acknowledgements



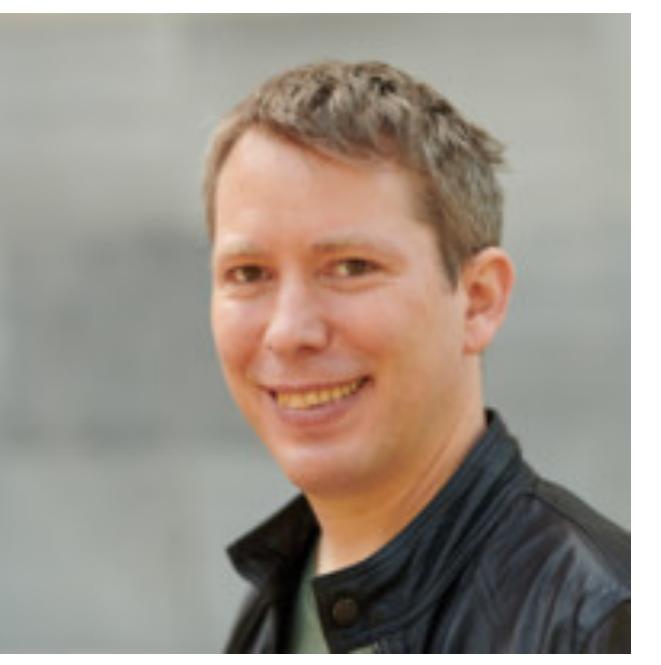
Johann Brehmer



Gilles Louppe



Juan Pavez



Markus Stoye



Felix Kling



Irina Espejo

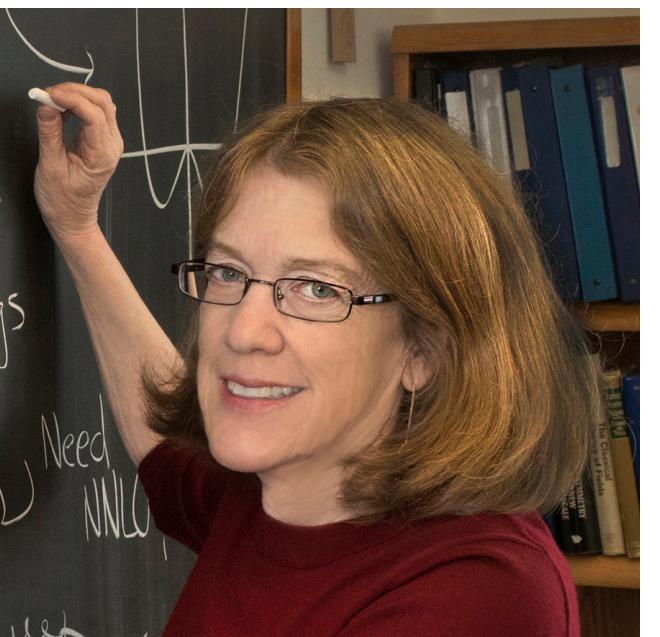


Sinclert Perez

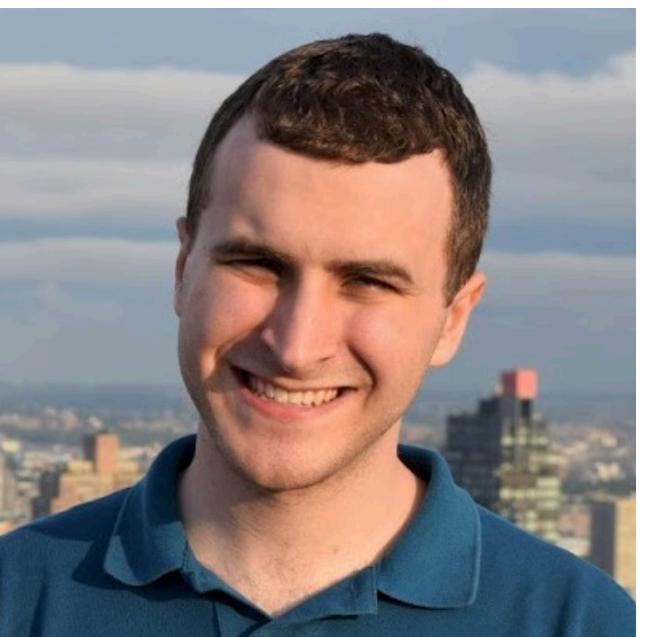
Special thanks
to Johann
for slides I
borrowed



Tilman Plehn



Sally Dawson



Sam Homiller



Giordon Stark



Matthew Feickert

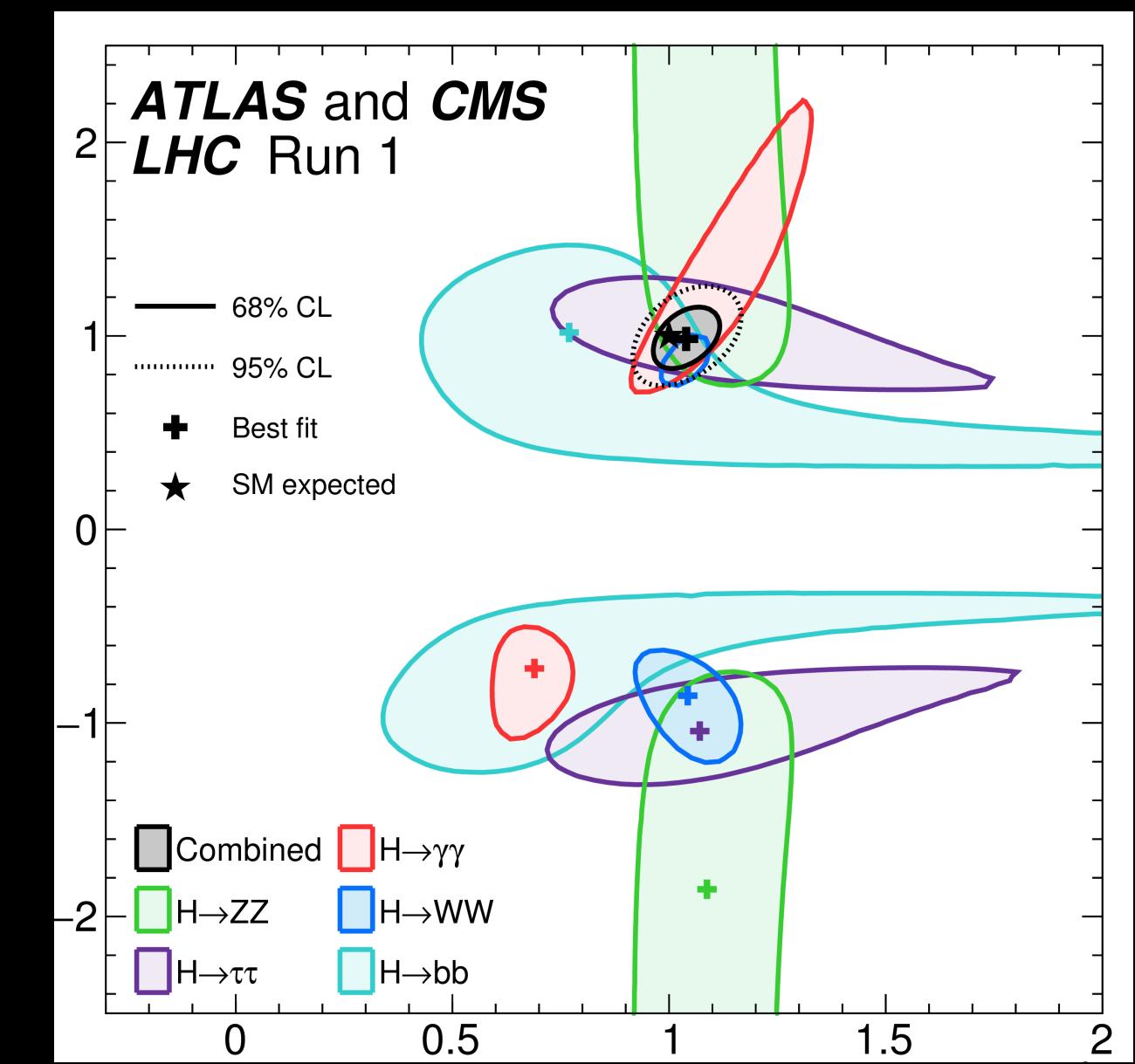
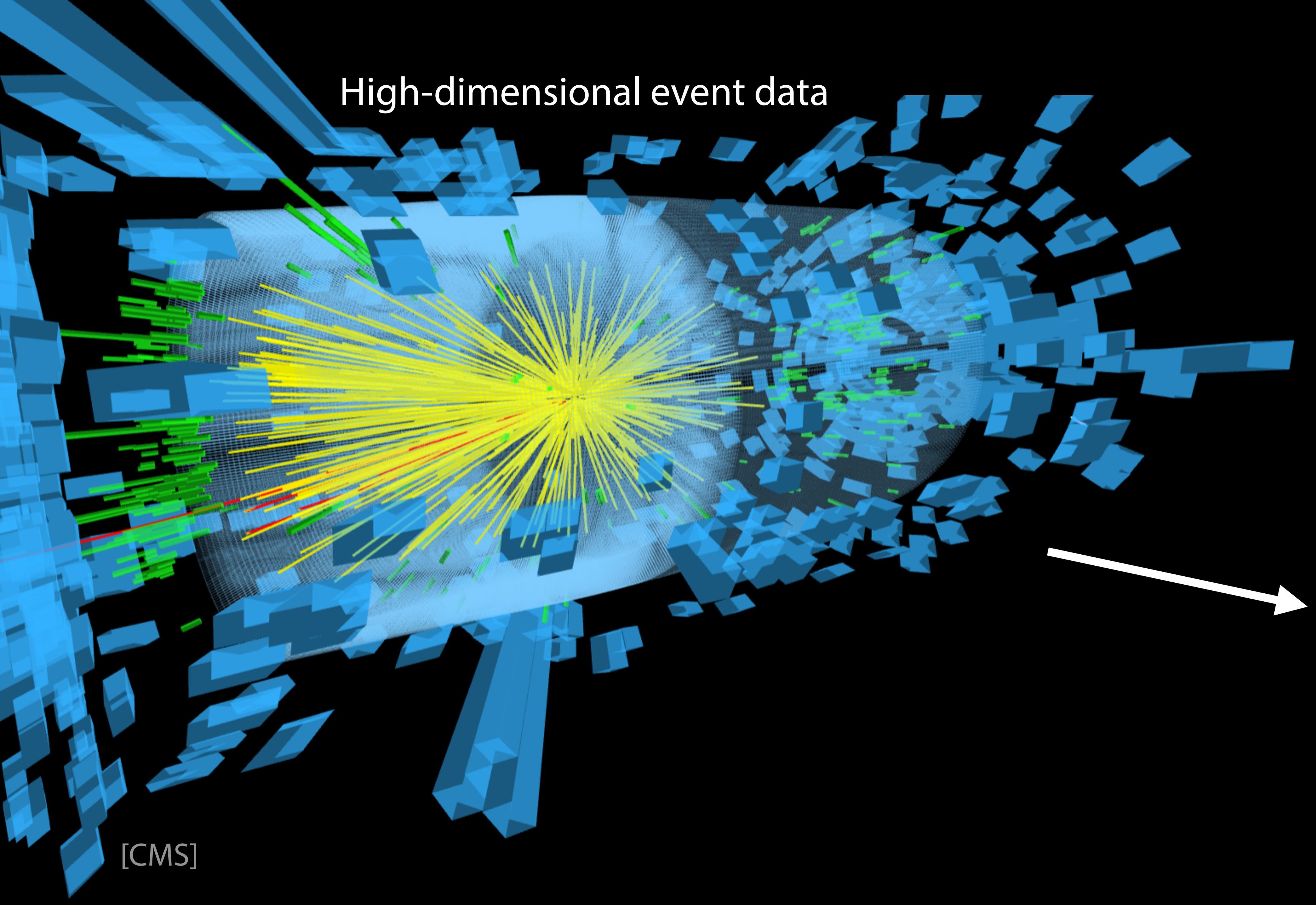


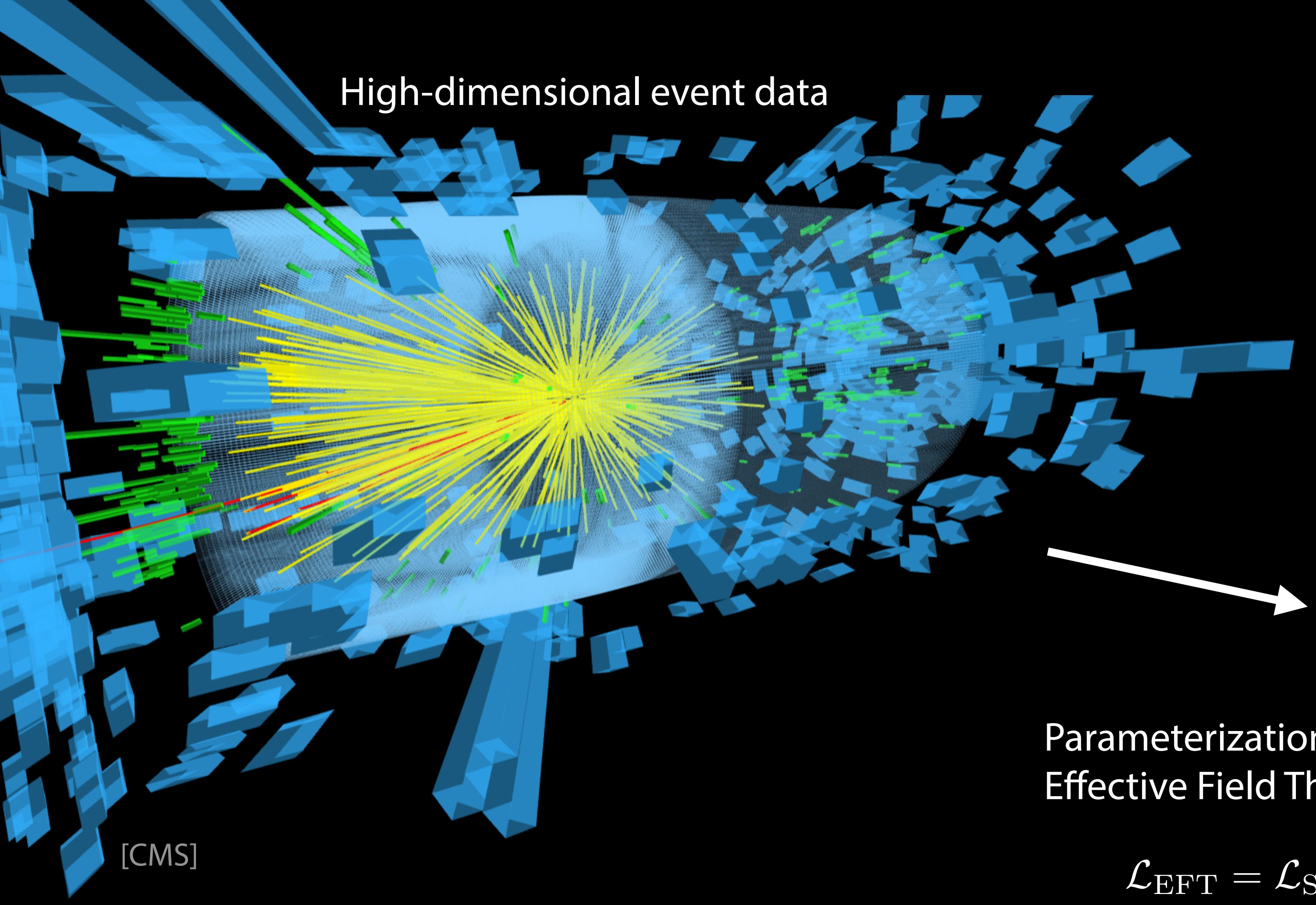
Lukas Heinrich



The SCAILFIN Project
scailfin.github.io



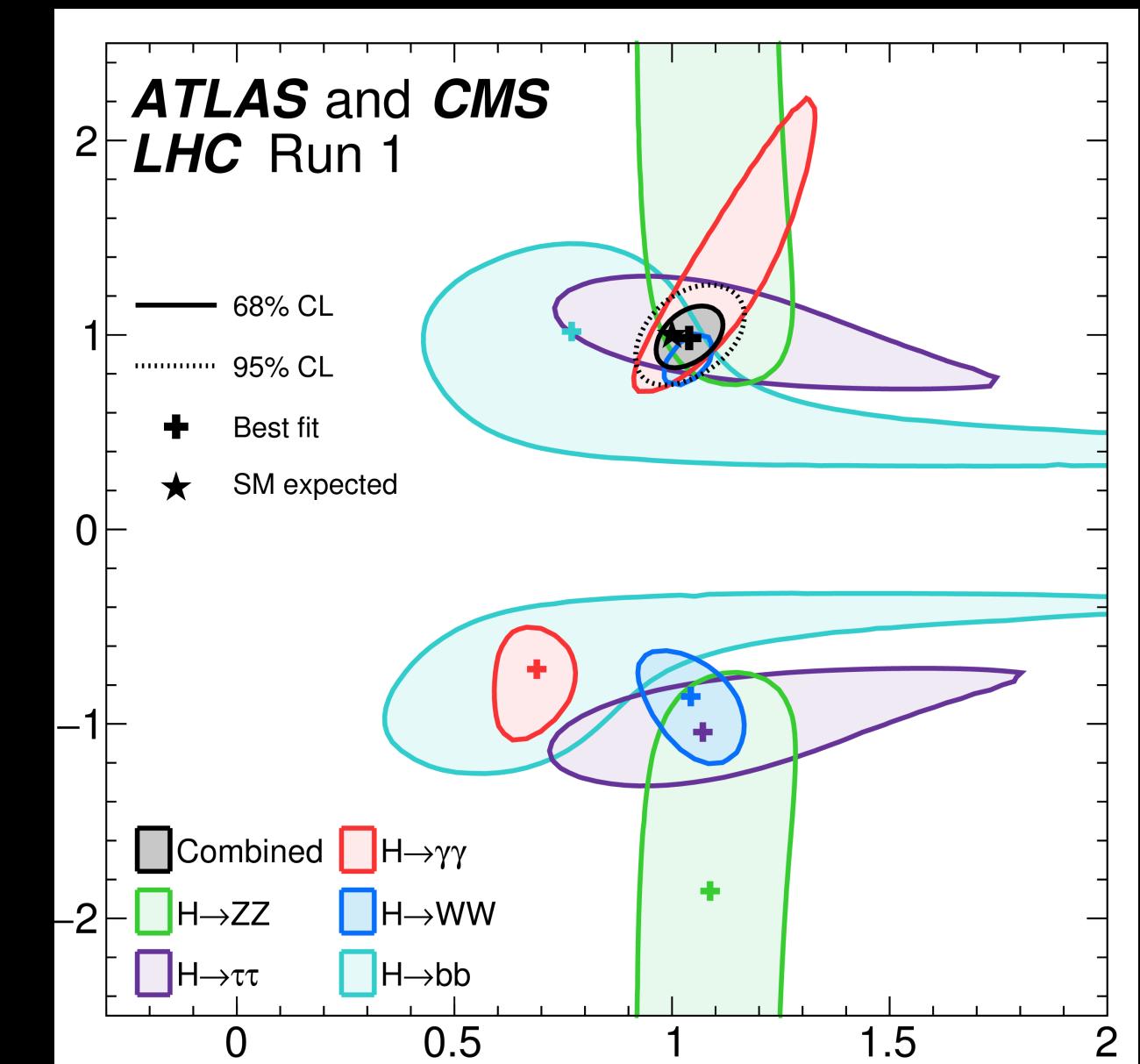




Parameterization e.g. in
Effective Field Theory:

$$\mathcal{L}_{\text{EFT}} = \mathcal{L}_{\text{SM}} + \sum_i \frac{f_i}{\Lambda^2} \mathcal{O}_i + \dots$$

10s to 100s “universal”
parameters to measure

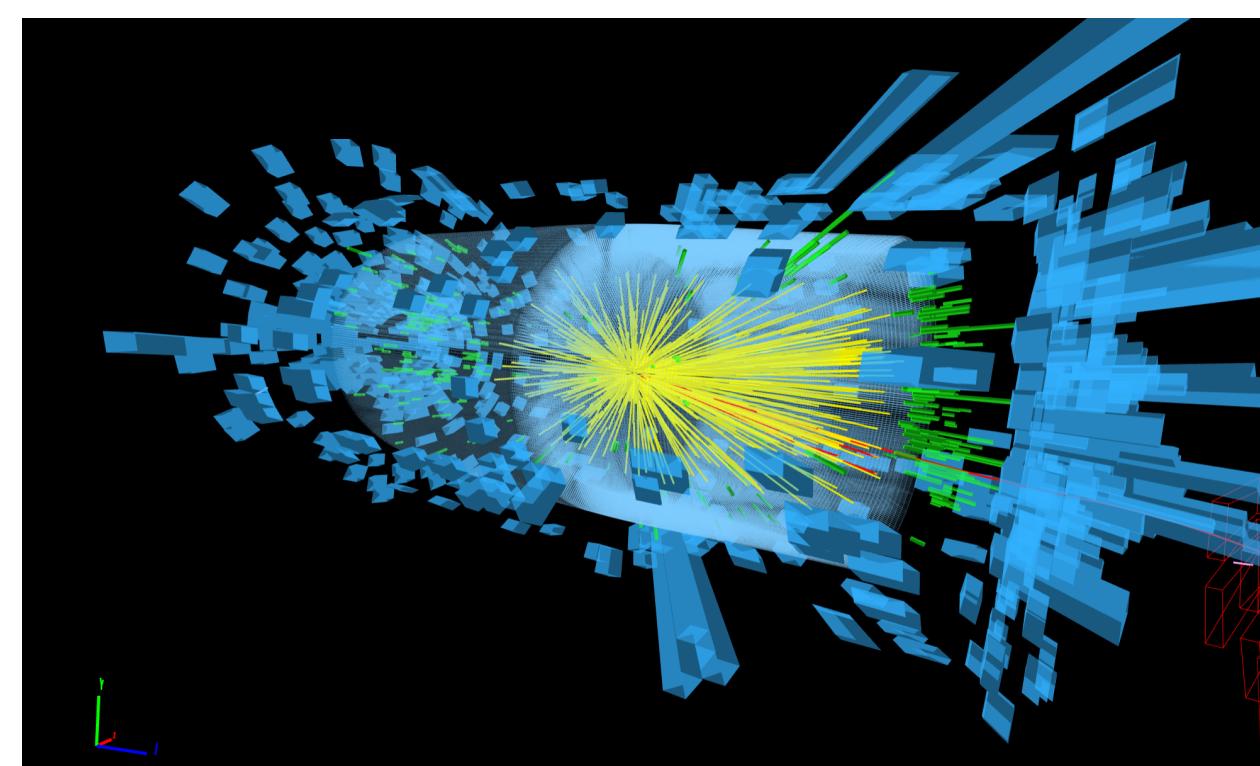


Precision constraints on
new physics

systematic expansion of
new physics around
Standard Model

The likelihood is a key object

Let θ denote the coefficients of higher dimensional operators in the Lagrangian, x be high-dimensional data associated to an event, and $p(x | \theta) = \frac{1}{\sigma(\theta)} \frac{d\sigma}{d\theta}$ be the distribution for the data



High-dimensional
event data x

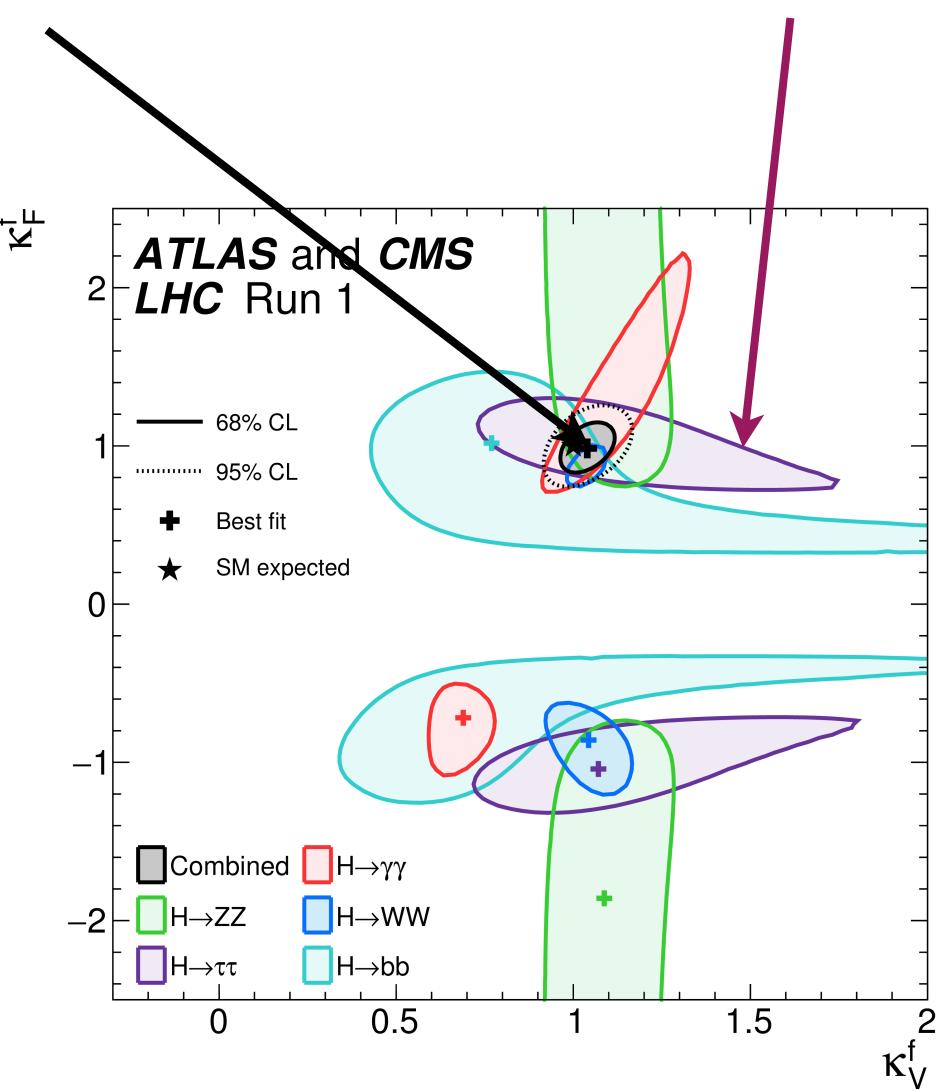


Likelihood function
 $p(x|\theta)$

Maximum-likelihood
estimator



Confidence limits based
on likelihood ratio tests



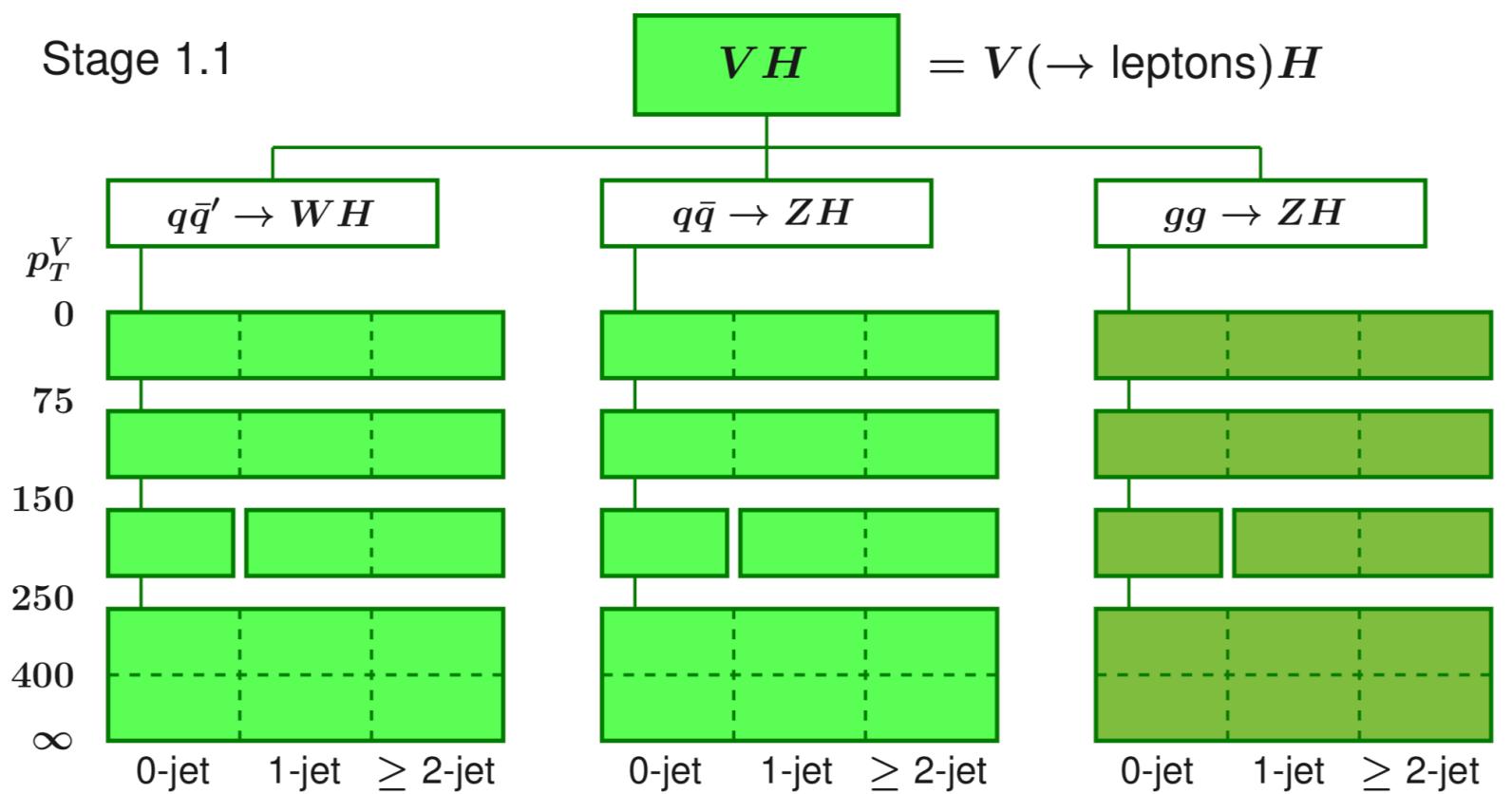
Constraints on
parameters θ

Benchmarking STXS in fully differential in WH

[JB, S. Dawson, S. Homiller, F. Kling, T. Plehn 1908.06980]

- Simplified Template Cross-Sections (STXS) define observable bins that are supposed to capture as much information on NP as possible

[N. Berger et al. 1906.02754; HXSWG YR4]



- Let's check! How much information on

$$\tilde{\mathcal{O}}_{HD} = \mathcal{O}_{H\square} - \frac{\mathcal{O}_{HD}}{4} = (\phi^\dagger \phi) \square (\phi^\dagger \phi) - \frac{1}{4} (\phi^\dagger D^\mu \phi)^* (\phi^\dagger D_\mu \phi)$$

$$\mathcal{O}_{HW} = \phi^\dagger \phi W_{\mu\nu}^a W^{\mu\nu a}$$

$$\mathcal{O}_{Hq}^{(3)} = (\phi^\dagger i \overleftrightarrow{D}_\mu^a \phi) (\bar{Q}_L \sigma^a \gamma^\mu Q_L),$$

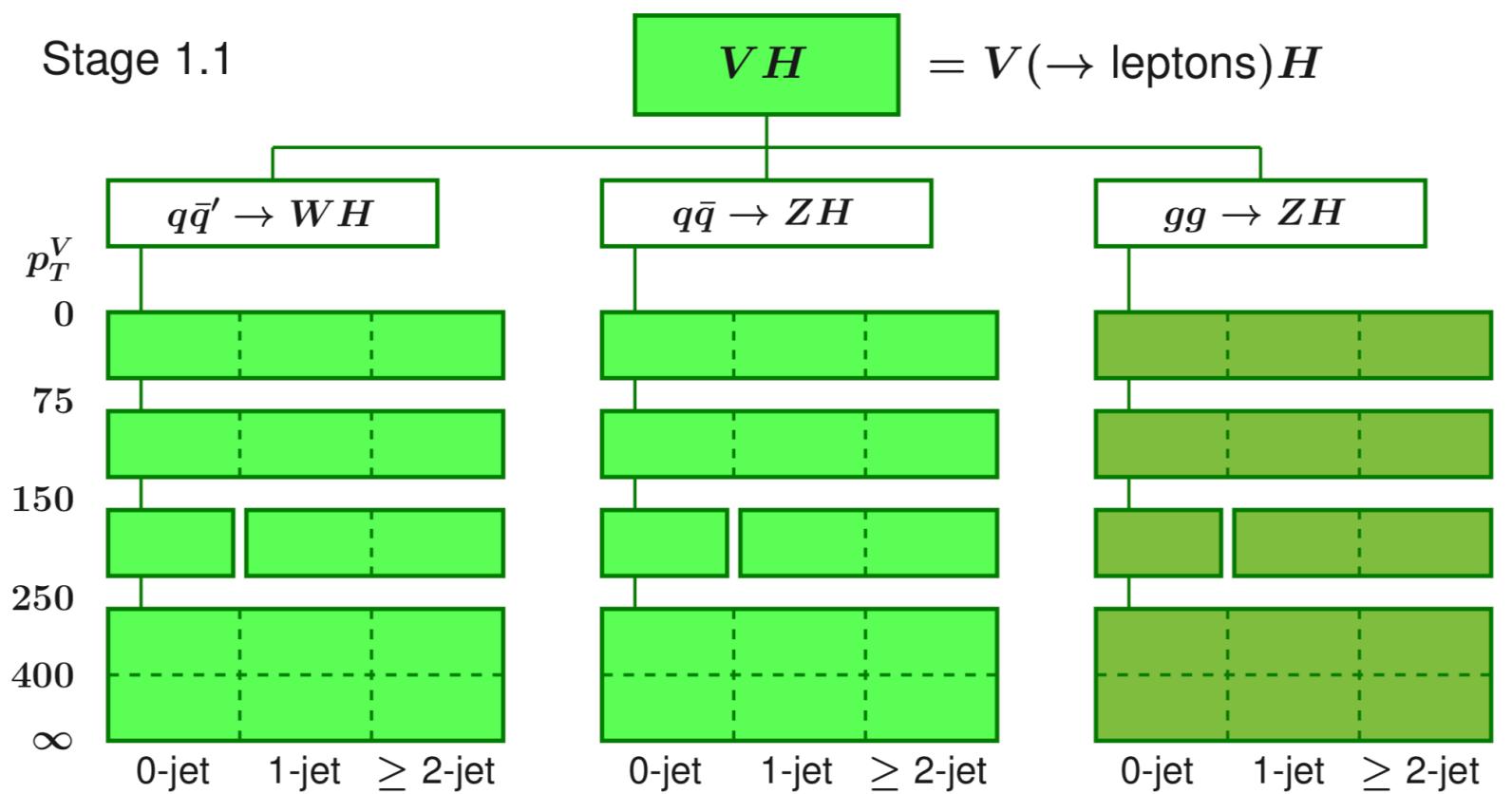
can we extract from $pp \rightarrow WH \rightarrow \ell\nu b\bar{b}$?

Benchmarking STXS in fully differential in WH

[JB, S. Dawson, S. Homiller, F. Kling, T. Plehn 1908.06980]

- Simplified Template Cross-Sections (STXS) define observable bins that are supposed to capture as much information on NP as possible

[N. Berger et al. 1906.02754; HXSWG YR4]



- Let's check! How much information on

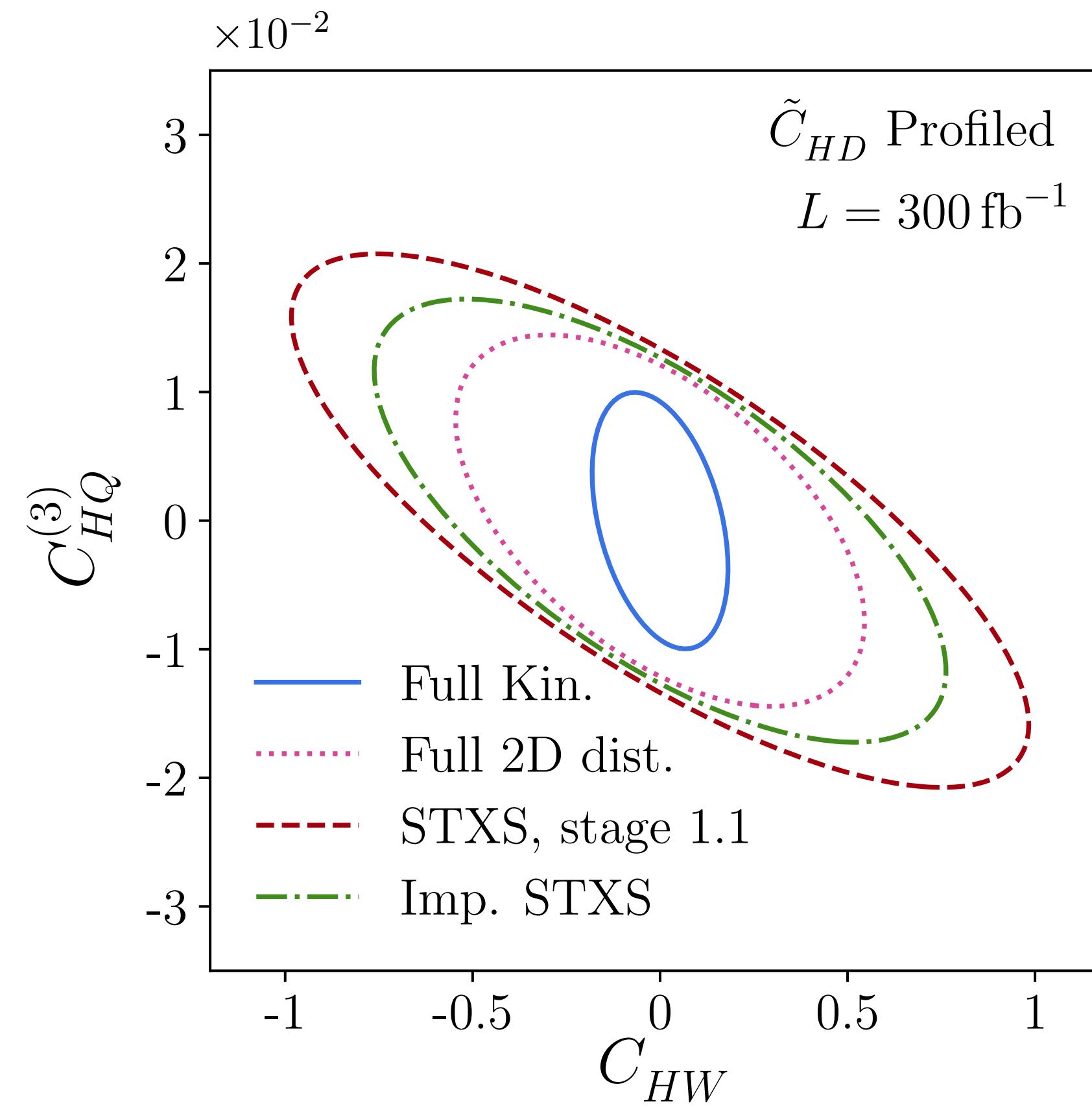
$$\tilde{\mathcal{O}}_{HD} = \mathcal{O}_{H\square} - \frac{\mathcal{O}_{HD}}{4} = (\phi^\dagger \phi) \square (\phi^\dagger \phi) - \frac{1}{4} (\phi^\dagger D^\mu \phi)^* (\phi^\dagger D_\mu \phi)$$

$$\mathcal{O}_{HW} = \phi^\dagger \phi W_{\mu\nu}^a W^{\mu\nu a}$$

$$\mathcal{O}_{Hq}^{(3)} = (\phi^\dagger i \overleftrightarrow{D}_\mu^a \phi) (\bar{Q}_L \sigma^a \gamma^\mu Q_L),$$

can we extract from $pp \rightarrow WH \rightarrow \ell\nu b\bar{b}$?

- Results: STXS are indeed sensitive to operators, adding a few more bins improve them, but a multivariate analysis is much stronger

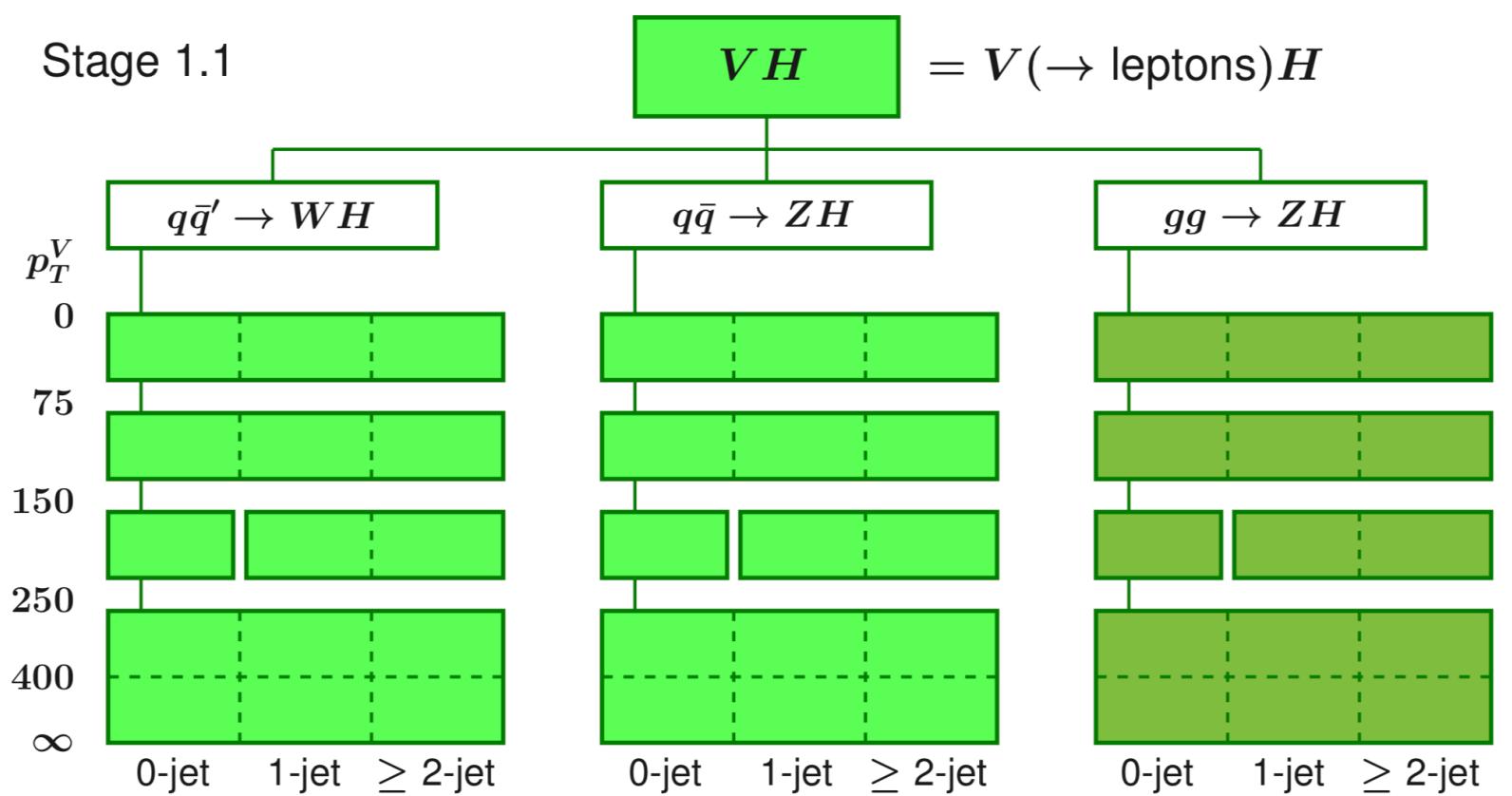


Benchmarking STXS in fully differential in WH

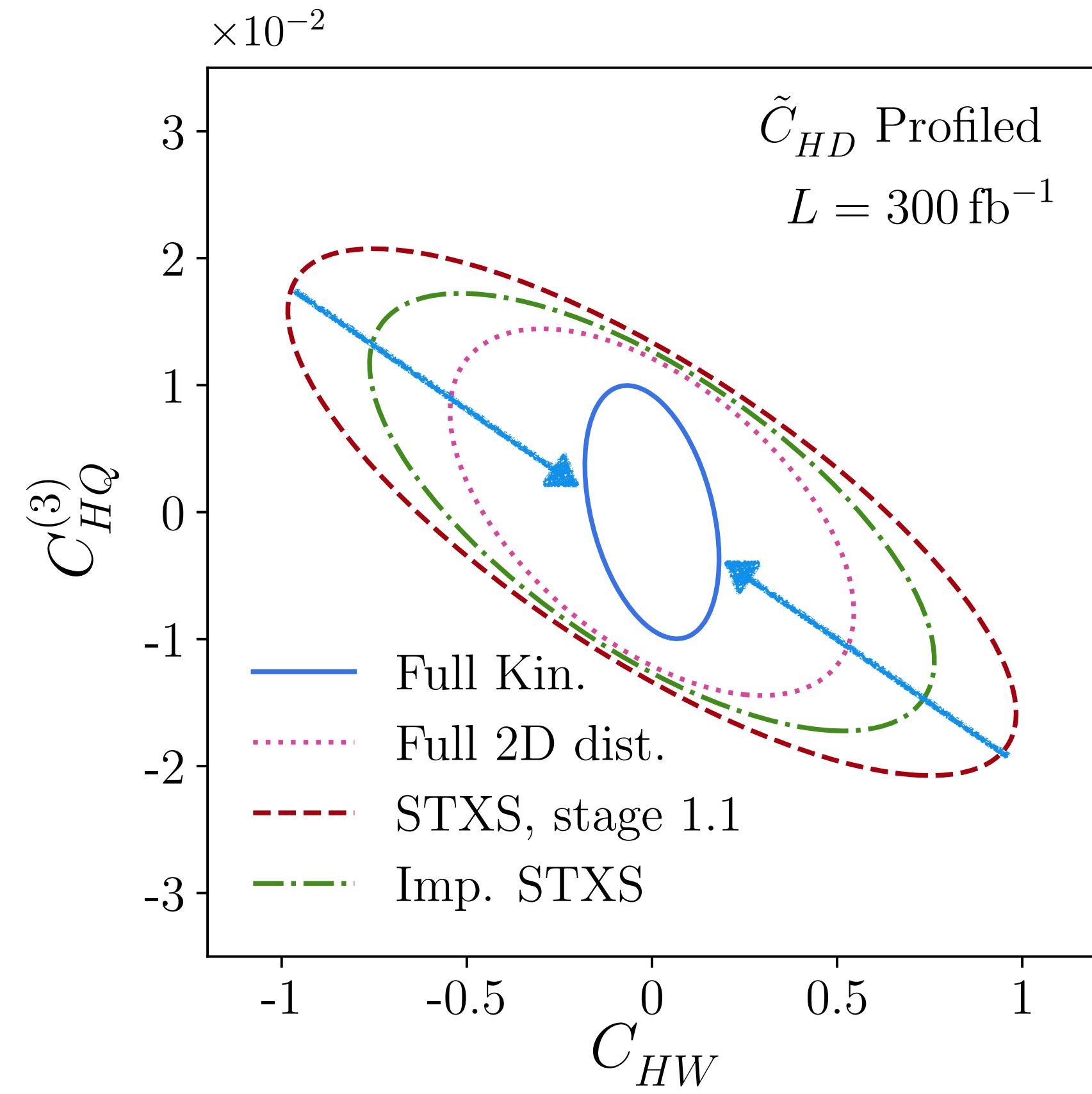
[JB, S. Dawson, S. Homiller, F. Kling, T. Plehn 1908.06980]

- Simplified Template Cross-Sections (STXS) define observable bins that are supposed to capture as much information on NP as possible

[N. Berger et al. 1906.02754; HXSWG YR4]



- Results: STXS are indeed sensitive to operators, adding a few more bins improve them, but a multivariate analysis is much stronger



- Let's check! How much information on

$$\tilde{\mathcal{O}}_{HD} = \mathcal{O}_{H\Box} - \frac{\mathcal{O}_{HD}}{4} = (\phi^\dagger \phi) \Box (\phi^\dagger \phi) - \frac{1}{4} (\phi^\dagger D^\mu \phi)^* (\phi^\dagger D_\mu \phi)$$

$$\mathcal{O}_{HW} = \phi^\dagger \phi W_{\mu\nu}^a W^{\mu\nu a}$$

$$\mathcal{O}_{Hq}^{(3)} = (\phi^\dagger i \overleftrightarrow{D}_\mu^a \phi) (\bar{Q}_L \sigma^a \gamma^\mu Q_L),$$

can we extract from $pp \rightarrow WH \rightarrow \ell\nu b\bar{b}$?

Hands-on Tutorial

- Step-by-step instructions
- Do Preliminaries
 - Need to install Docker
 - And then start Jupyter
- Step 1 is fast
- Step 2 takes ~25 min
- Step 3 takes ~20 min
- While they are running I will lecture
- Then we will finish with results

The screenshot shows the 'MadMiner Tutorial' website at <https://cranmer.github.io/madminer-tutorial/>. The page has a navigation bar with a back arrow and a download icon. The main content area is titled 'Introduction' and contains a brief introduction to the MadMiner tutorial, developed by Johann Brehmer, Felix Kling, Irina Espejo, and Kyle Cranmer, built using Jupyter Book. Below this is a diagram illustrating the workflow: 'Simulation' (parameter θ leads to latent z , which leads to observable x) → 'Machine Learning' (observable x leads to $r(x, z|\theta)$, $t(x, z|\theta)$, augmented data, and $\arg \min_g L[g]$) → 'Inference' (approximate likelihood ratio $\hat{r}(x|\theta)$ leads to θ_j and θ_i). The left side of the page lists the tutorial's contents: Introduction, MadMiner Tutorial, Preliminaries, Overview, Define process to study *, Morphing, Interactive Morphing Demo, Create training data, Set MadGraph Directory, Parton Level *, With Delphes, Train model, Likelihood Ratio *, Score *, Likelihood, Statistical Analysis, Limits on EFT parameters *, Fisher Information, Information Geometry, and Congratulations.

Introduction

MadMiner tutorial

This is a tutorial on [MadMiner](#) developed by Johann Brehmer, Felix Kling, Irina Espejo, and Kyle Cranmer. It is built using [Jupyter Book](#).

The diagram illustrates the workflow:

- Simulation:** A parameter θ is input into a simulation block. The simulation outputs a latent variable z , which is then used to generate an observable x .
- Machine Learning:** The observable x is used to train a machine learning model. This model outputs three types of data:
 - $r(x, z|\theta)$: raw data from the simulation.
 - $t(x, z|\theta)$: summary statistics or likelihood ratios.
 - augmented data: additional data used for training.The machine learning model also performs inference, calculating $\arg \min_g L[g]$ (the likelihood function) and determining the approximate likelihood ratio $\hat{r}(x|\theta)$.
- Inference:** The approximate likelihood ratio $\hat{r}(x|\theta)$ is used to find the best parameters θ_j and θ_i .

Introduction to MadMiner

Particle physics processes are usually modelled with complex Monte-Carlo simulations of the hard process, parton shower, and detector interactions. These simulators typically do not admit a tractable likelihood function: given a (potentially high-dimensional) set of observables, it is usually not possible to calculate the probability of these observables for some model parameters. Particle physicists usually tackle this problem of "likelihood-free inference" by hand-picking a few "good" observables or summary statistics and filling histograms of them. But this conventional approach discards the information in all other observables and often does not scale well to high-dimensional problems.

In the three publications "[Constraining Effective Field Theories With Machine Learning](#)", "[A Guide to Constraining Effective Field Theories With Machine Learning](#)", and "[Mining gold from implicit models to improve likelihood-free inference](#)", a new approach has been developed. In a nut shell, additional information is extracted from the simulations that is closely related to the matrix elements that determine the hard process. This "augmented data" can be used to train neural networks to efficiently approximate arbitrary likelihood ratios. We playfully call this process "mining gold" from the simulator, since this information may be hard to get, but turns out to be very valuable for inference.

The MadMiner for REANA

- Designed to scale to a cluster
 - Here six jobs for event generation and DELPHES simulation
 - Uses pre-built docker containers

reana

[Home](#) [Examples](#) [Get Started](#) [Documentation](#) [News](#) [Roadmap](#) [Contact](#) [Blog](#)

reana

Reproducible research data analysis platform

Flexible

Run many computational workflow engines.



Scalable

Support for remote compute clouds.

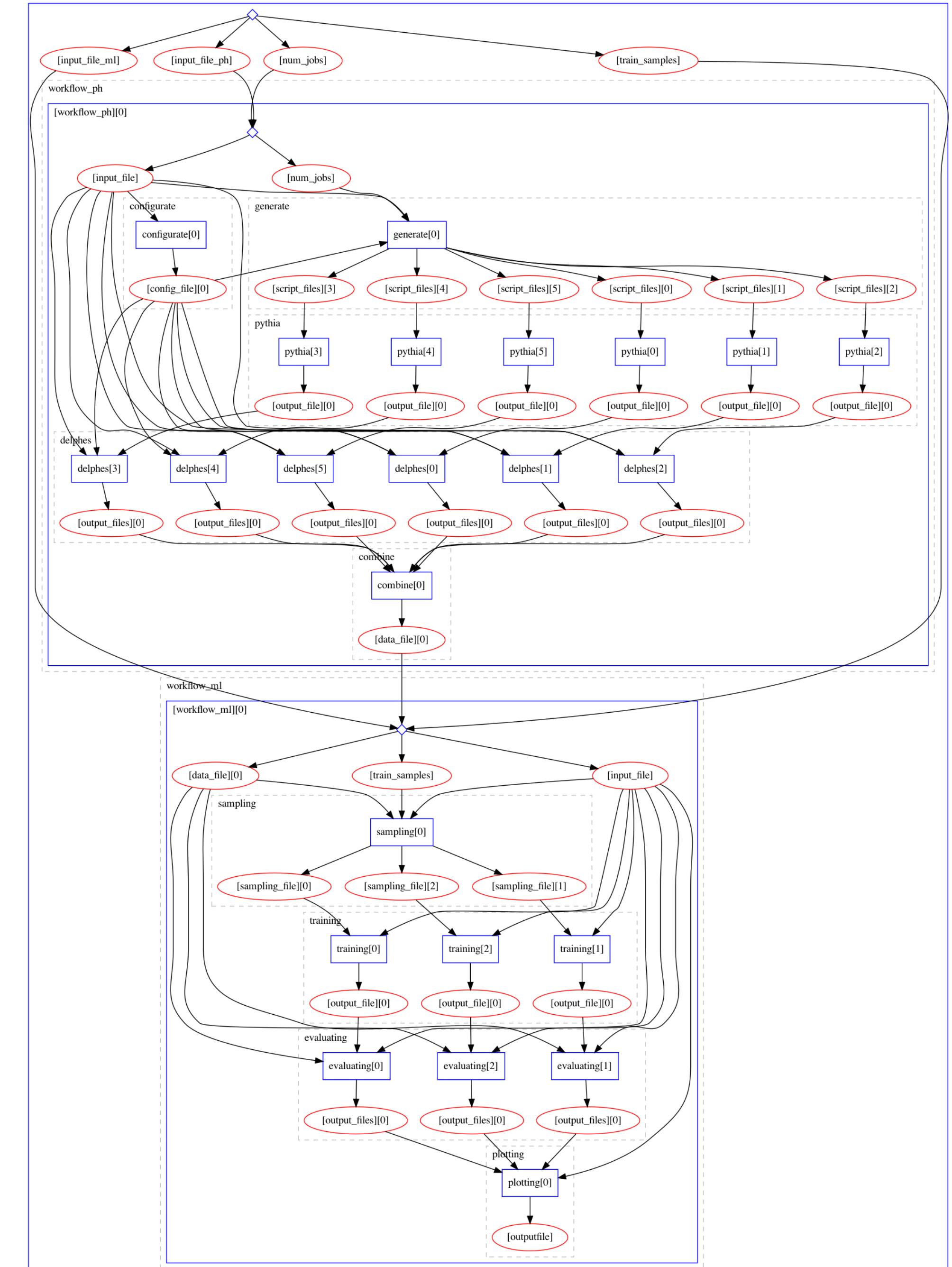


Reusable

Containerise once, reuse elsewhere. Cloud-native.

Free

Free Software. MIT licence.
Made with ❤ at CERN.



Now for some bad news....

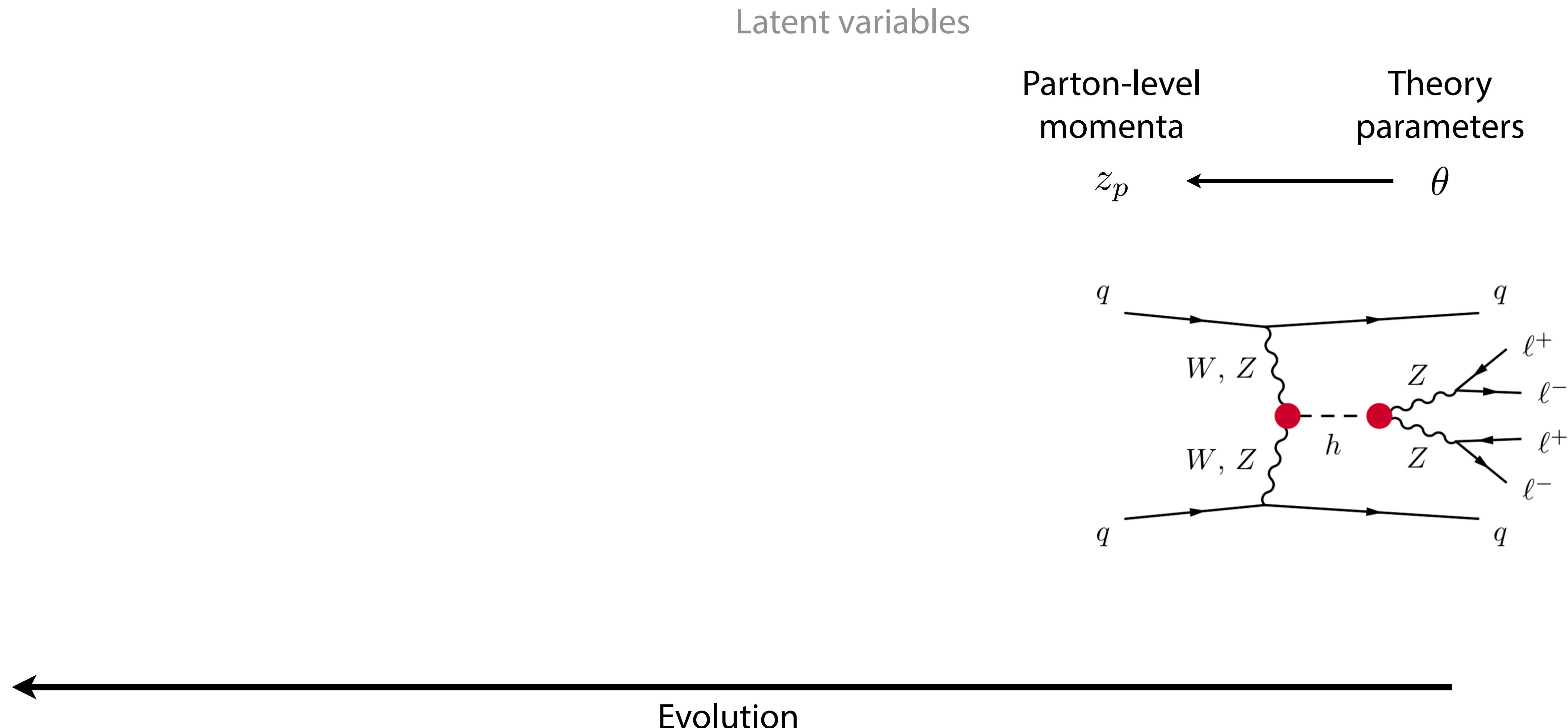
Particle physics processes do not have a tractable likelihood function.

Modeling particle physics processes

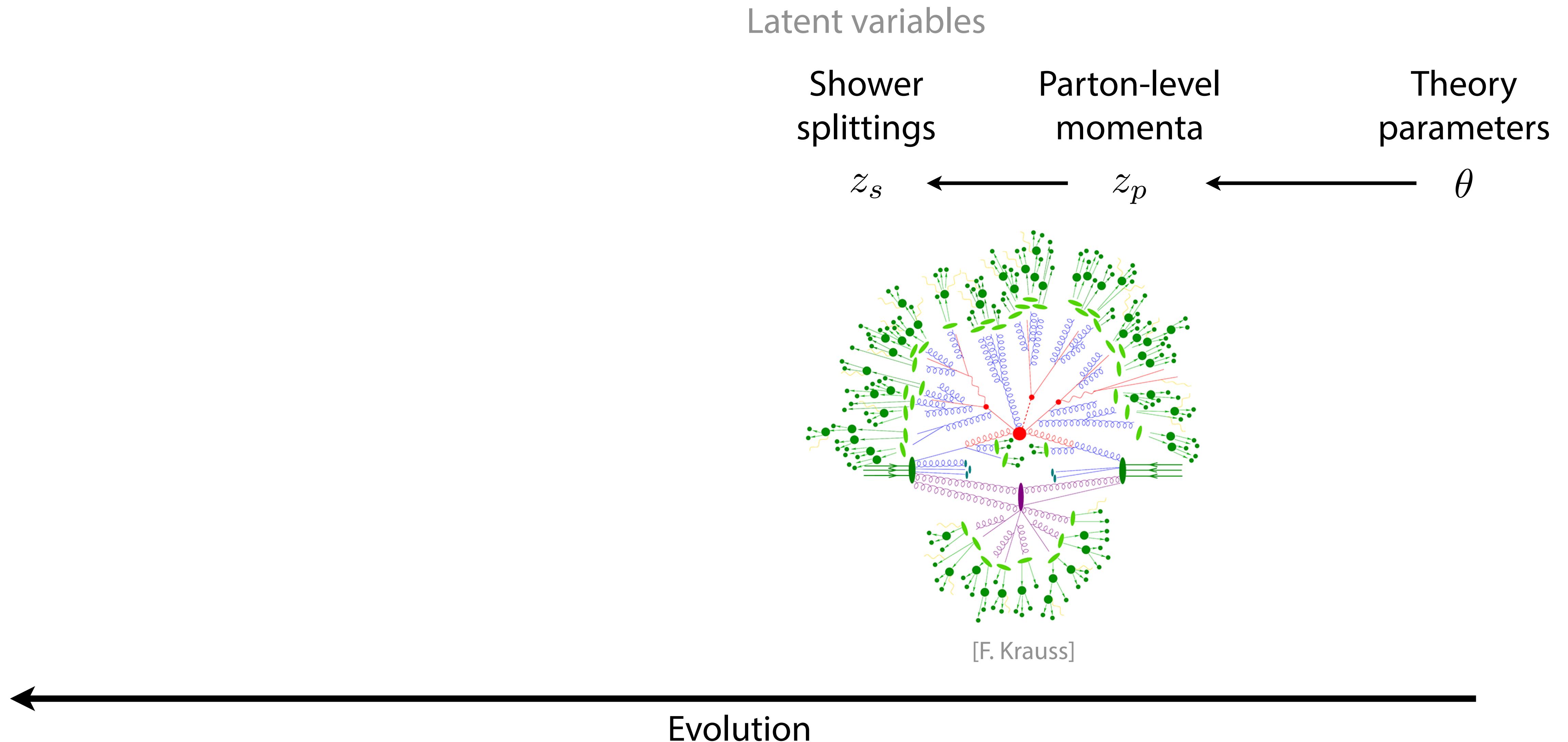
Theory
parameters
 θ



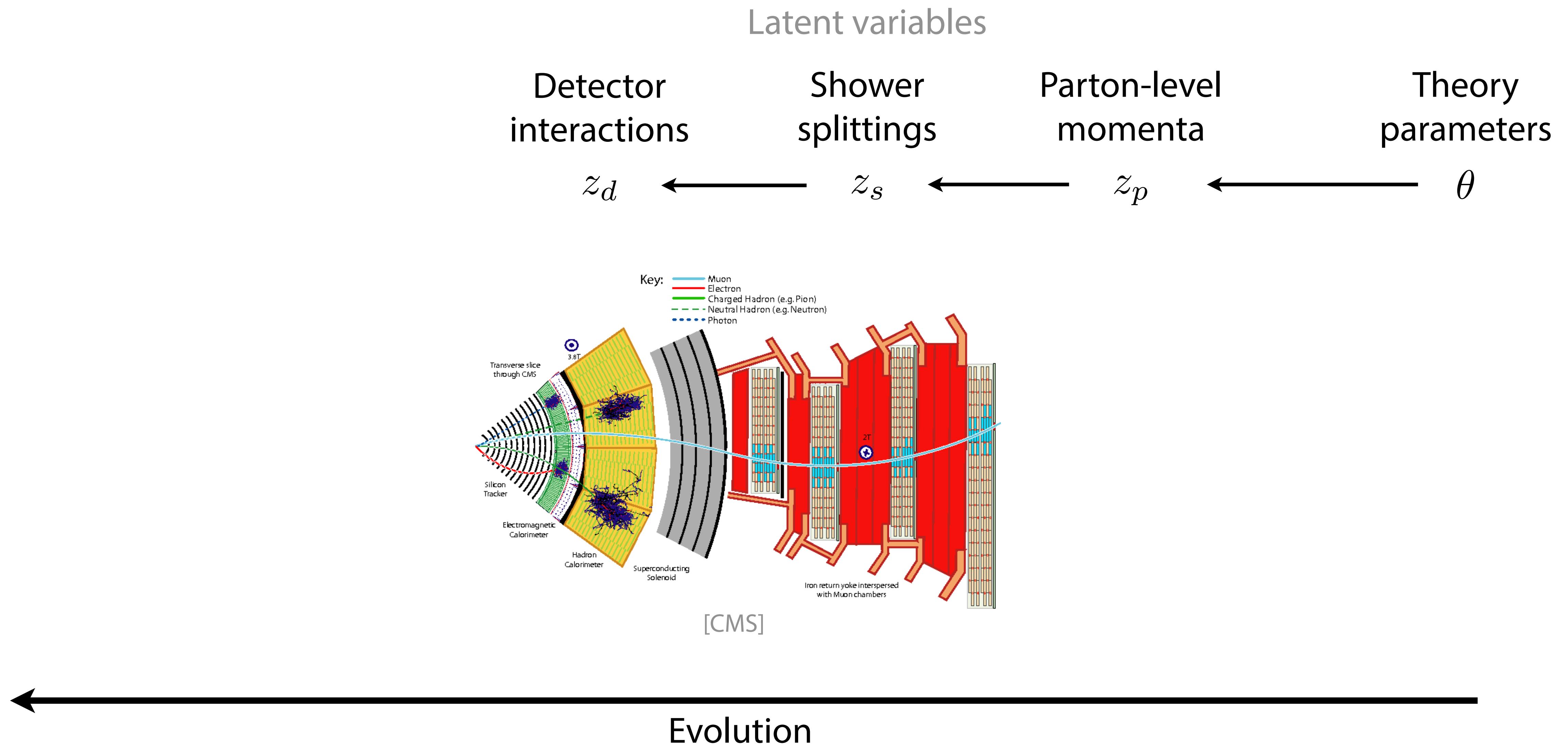
Modeling particle physics processes



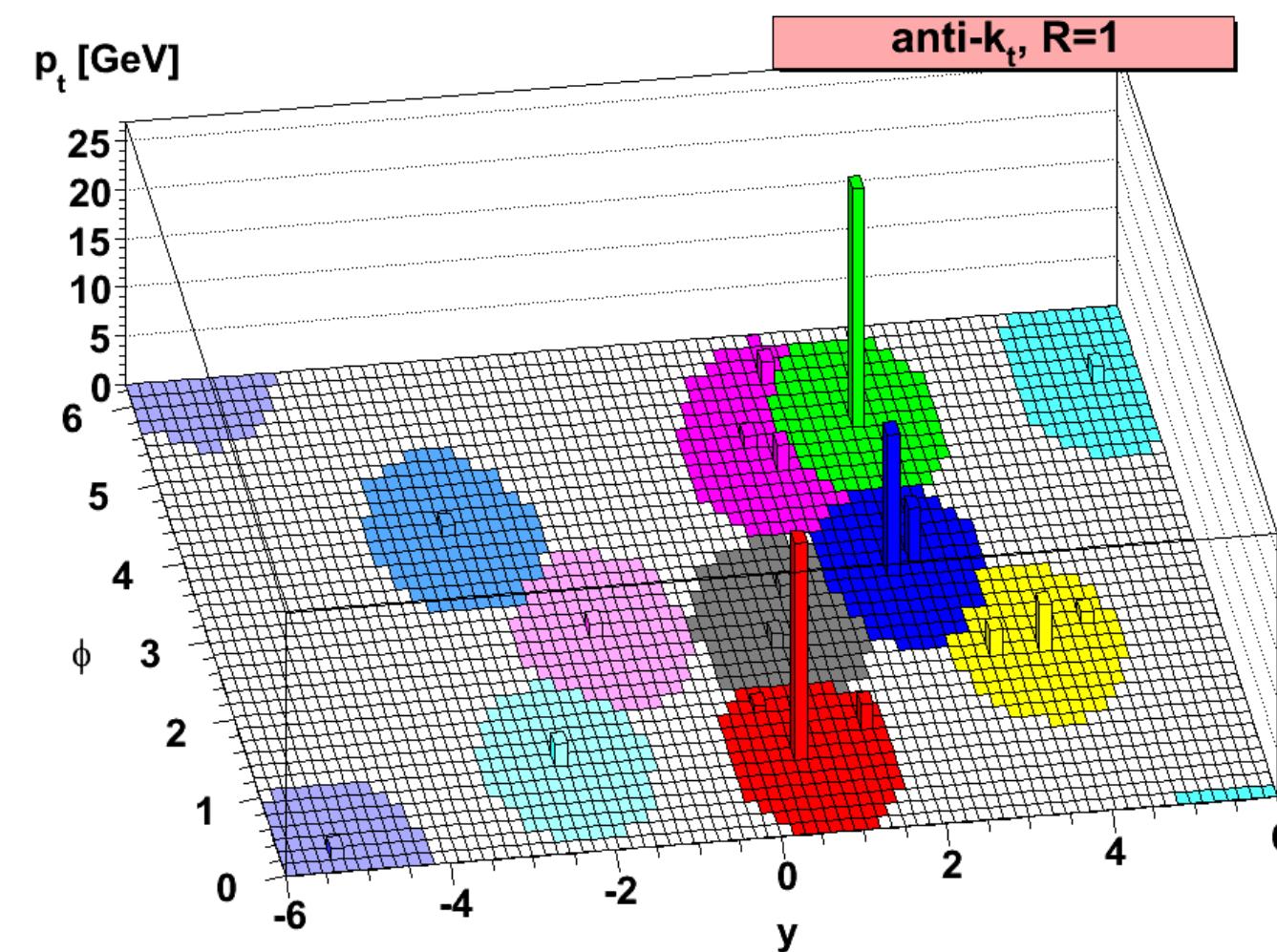
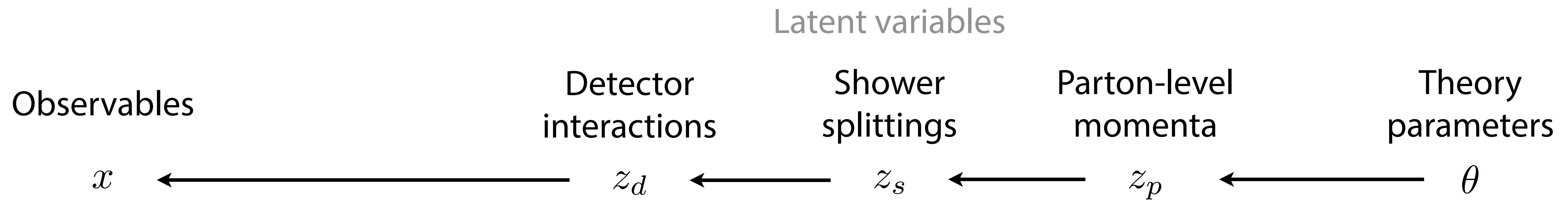
Modeling particle physics processes



Modeling particle physics processes



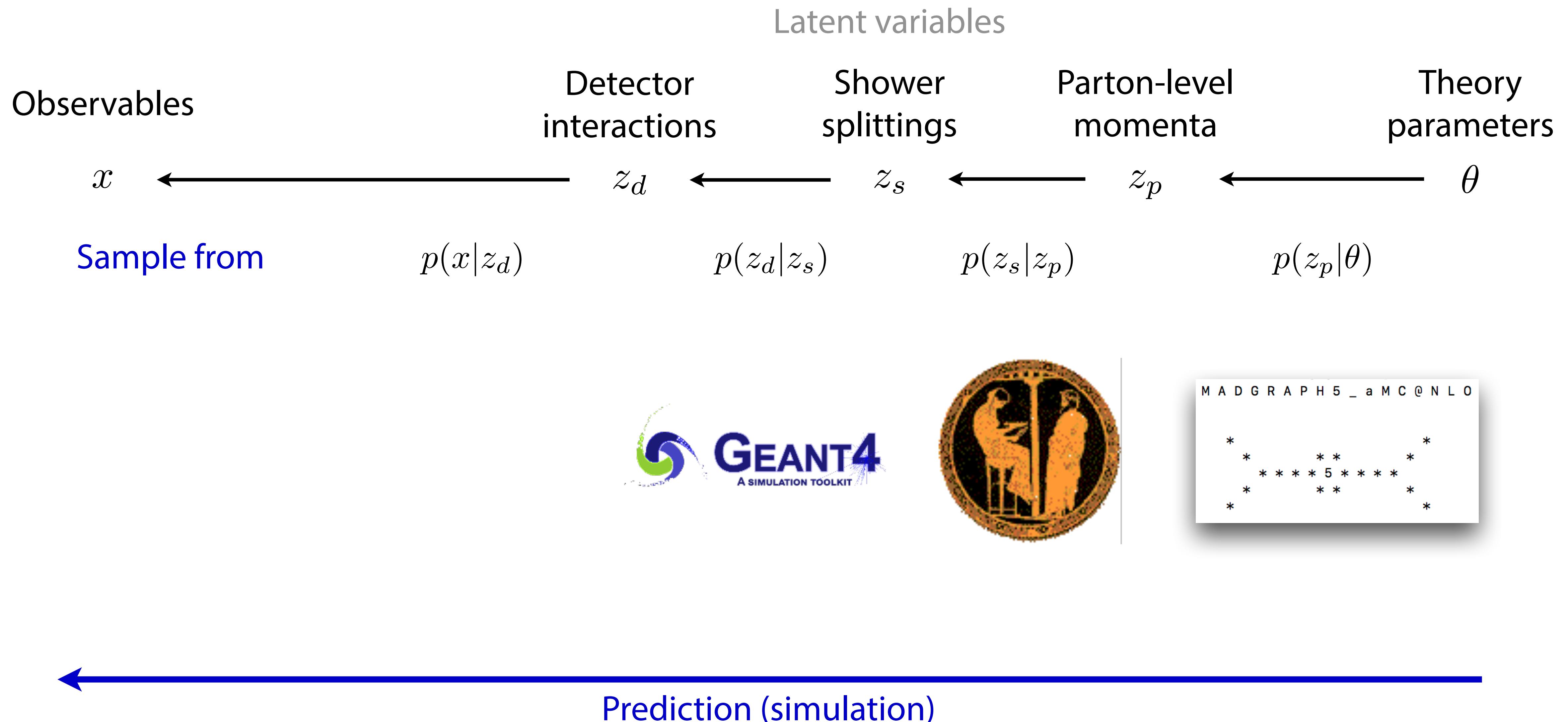
Modeling particle physics processes



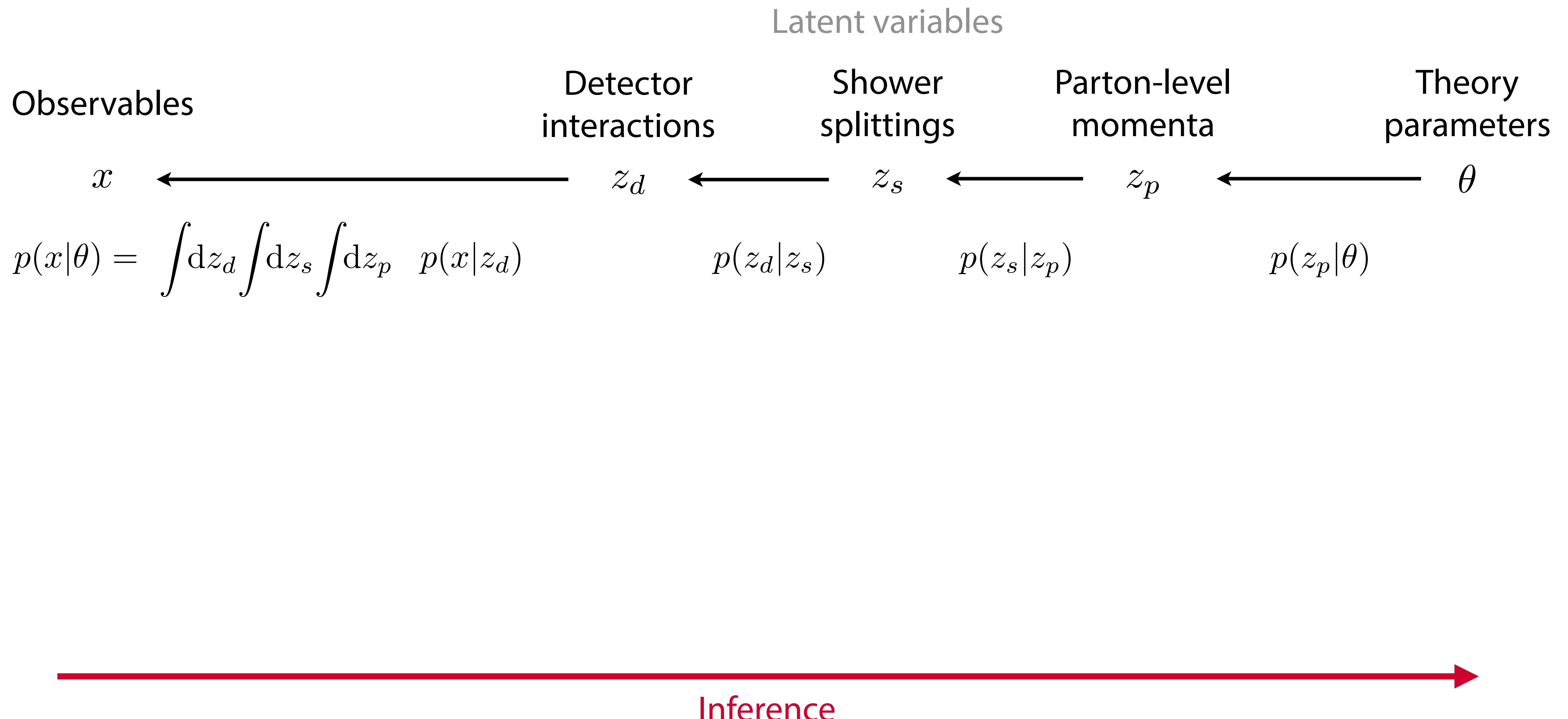
[M. Cacciari, G. Salam, G. Soyez 0802.1189]

Evolution

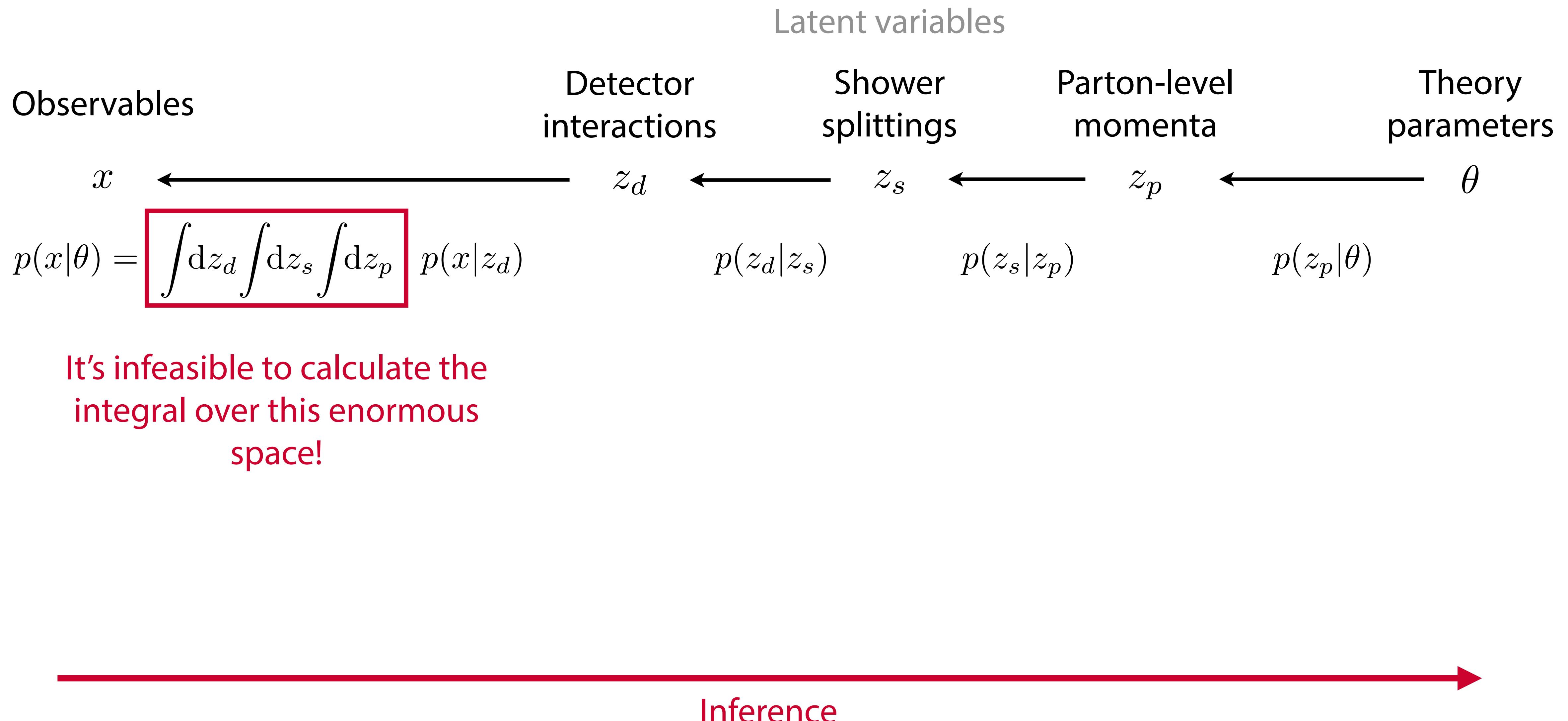
Modeling particle physics processes



Modeling particle physics processes

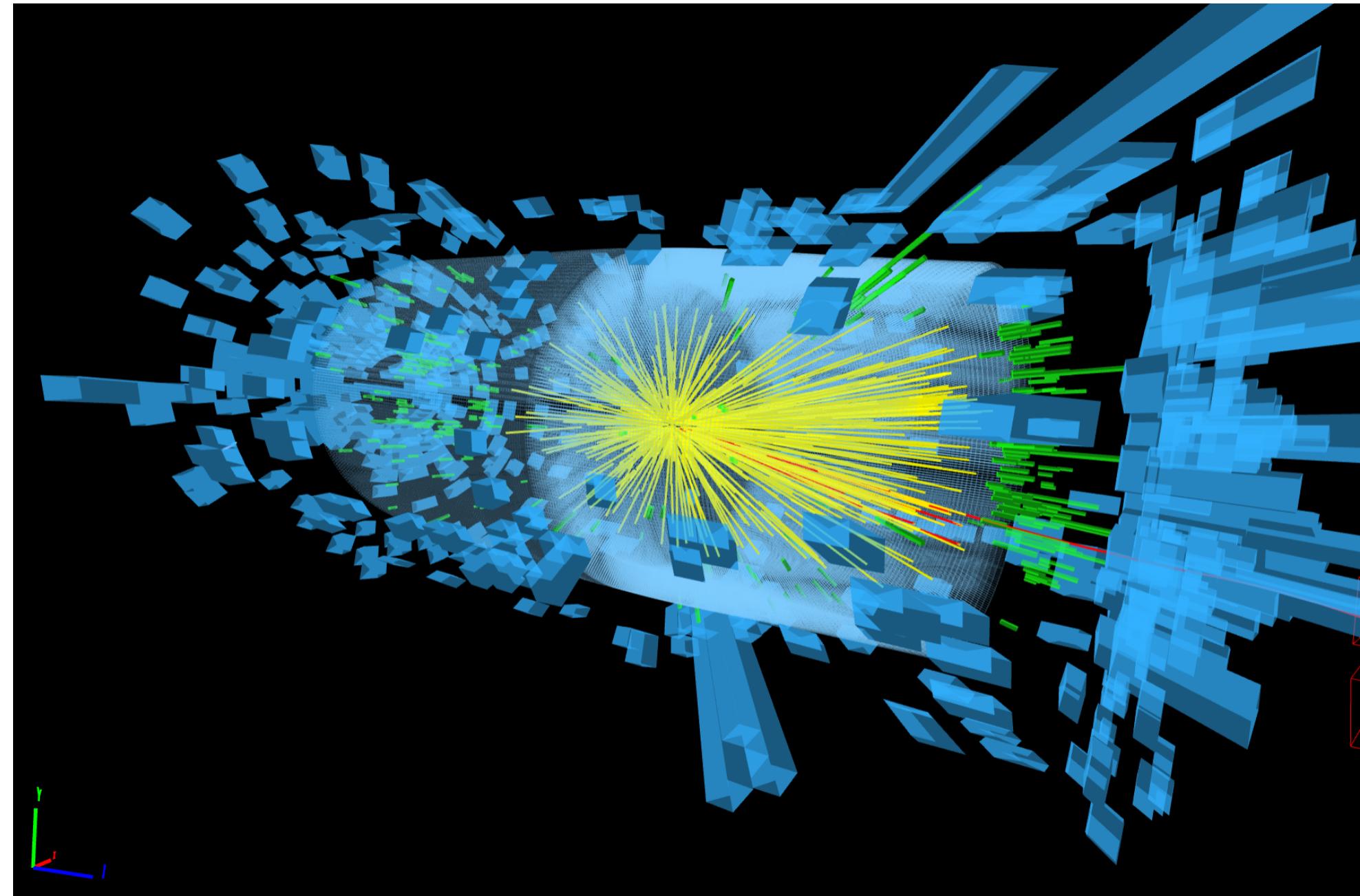


Modeling particle physics processes



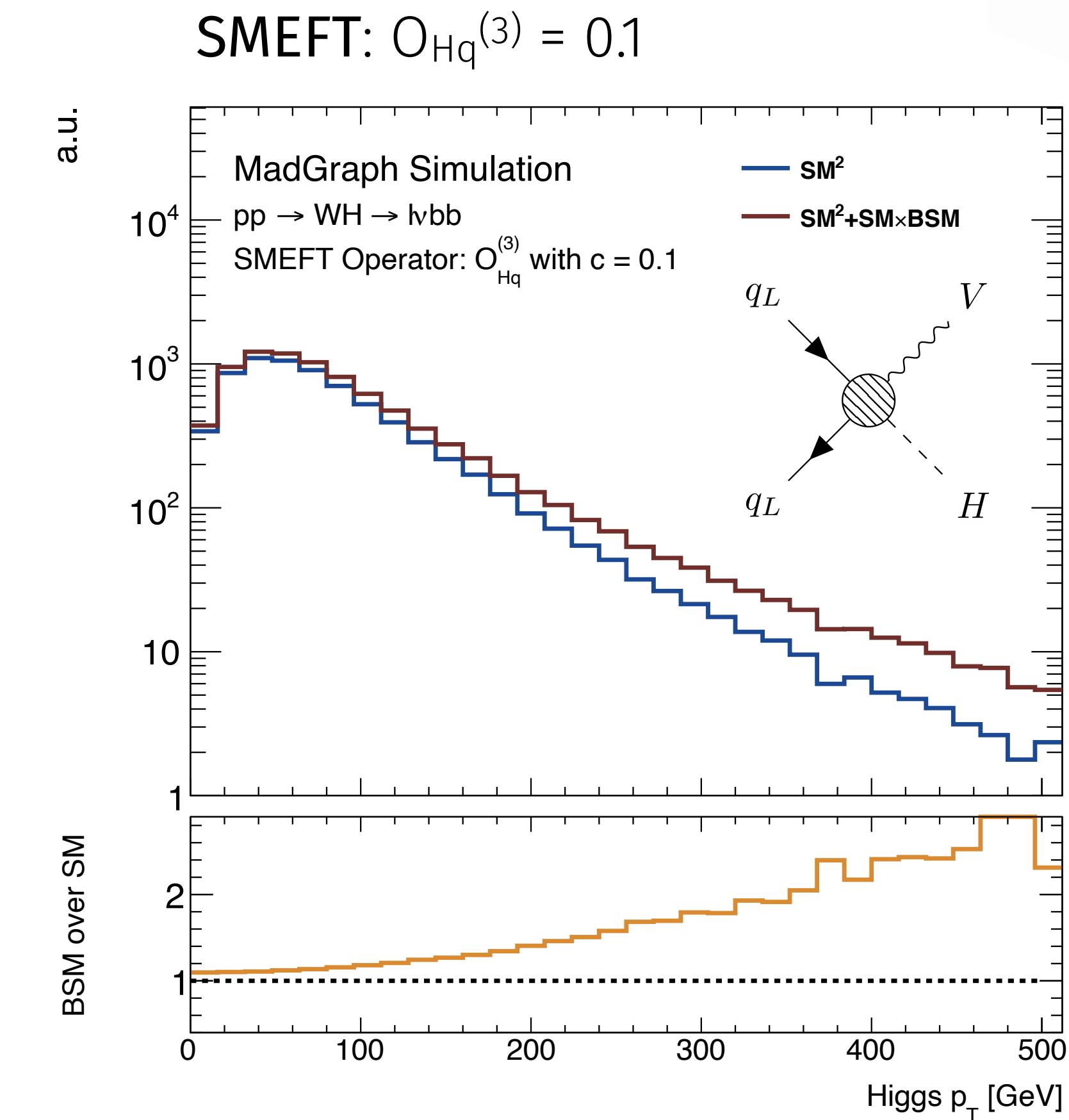
It's infeasible to calculate the integral over this enormous space!

What we usually do: number counting or singly differential

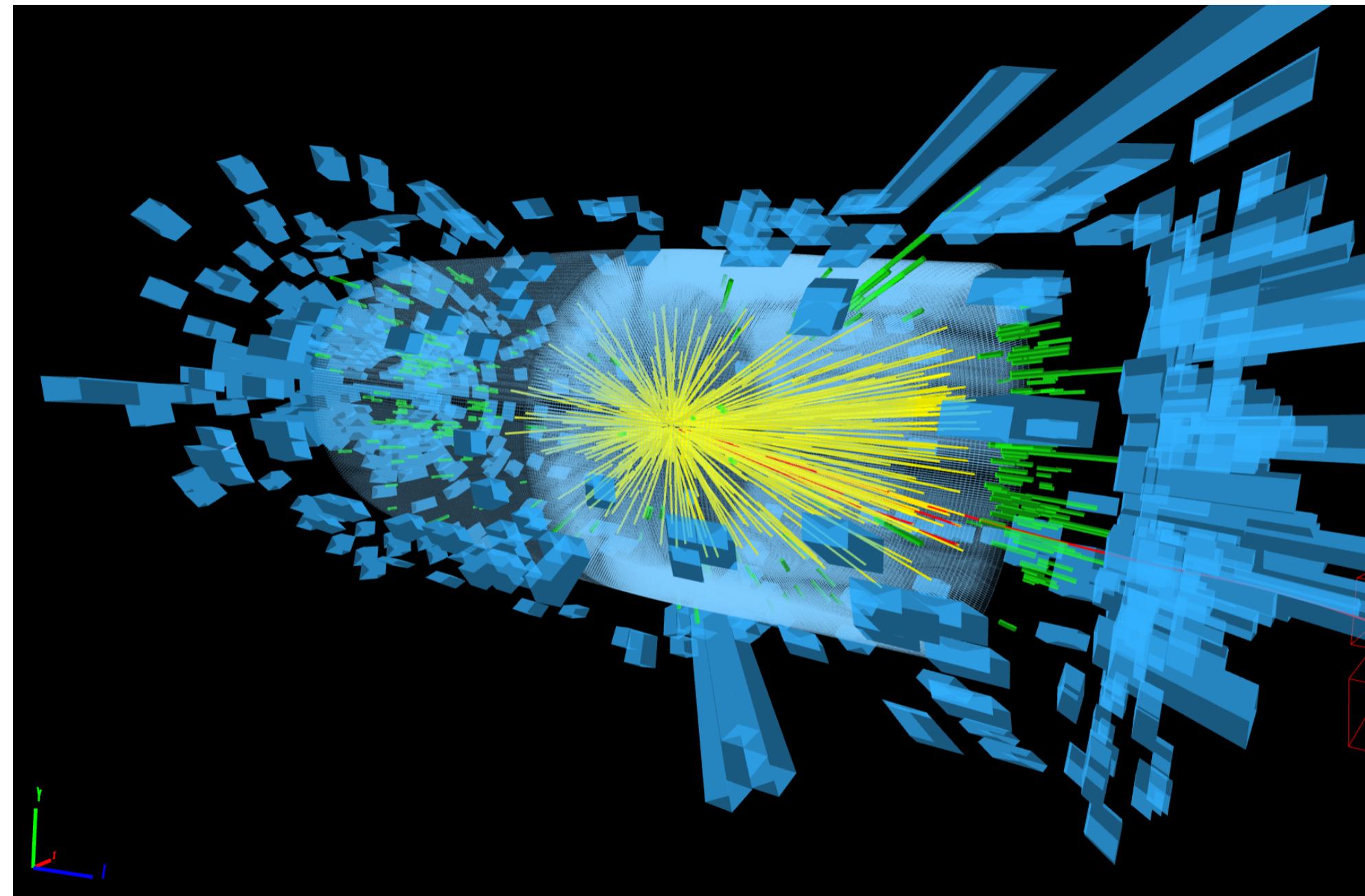


High-dimensional event data x

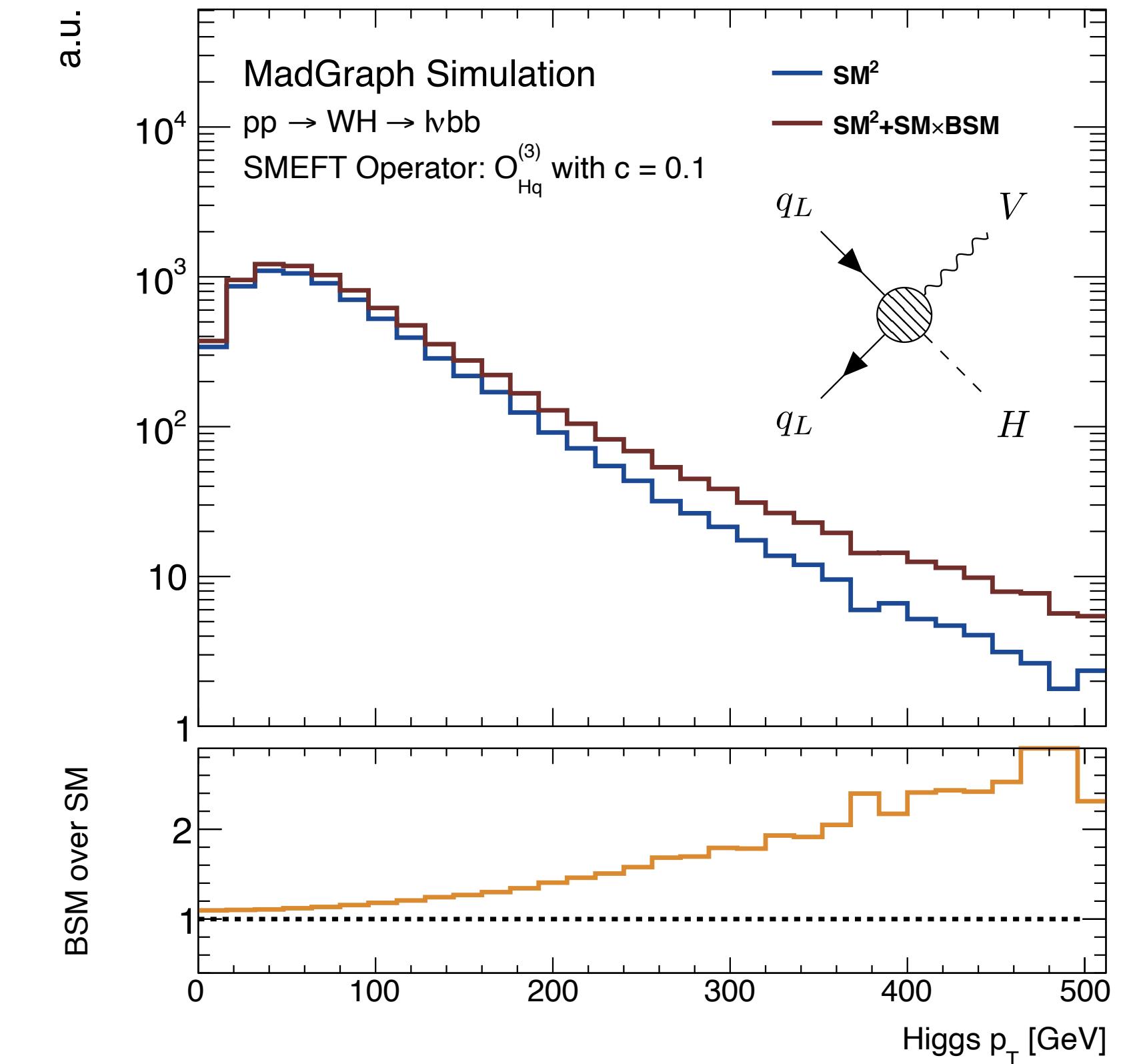
$p(x|\theta)$ cannot be calculated



What we usually do: number counting or singly differential



SMEFT: $O_{Hq}^{(3)} = 0.1$



High-dimensional event data x

$p(x|\theta)$ cannot be calculated

n.b. "summary statistic" = a sensitive observable

One or two summary statistics x'

$p(x'|\theta)$ can be estimated
with histograms

What if we could estimate the likelihood...

- for high-dimensional observables, including correlations?
like MEM: no need to pick summary statistics
- including state-of-the-art shower and detector models?
allowing for extra radiation, no need for transfer functions
- in microseconds?
amortized inference: train once, then always evaluate fast
- requiring less training examples than established machine learning methods?
using matrix element information: “ML version of MEM”

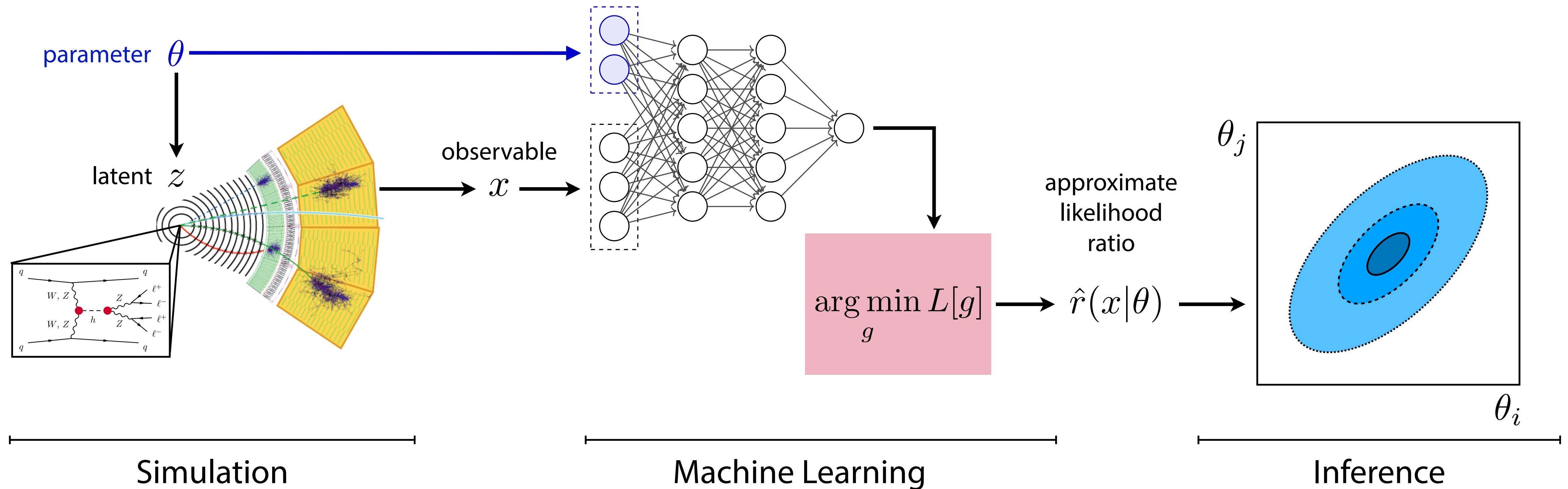
Learning with Simulated Data

arXiv:1805.12244

PRL, arXiv:1805.00013

PRD, arXiv:1805.00020

physics.aps.org/articles/v11/90



“Mining gold”: Extract additional information from simulator

Use this information to train estimator for likelihood ratio

Limit setting with standard hypothesis tests

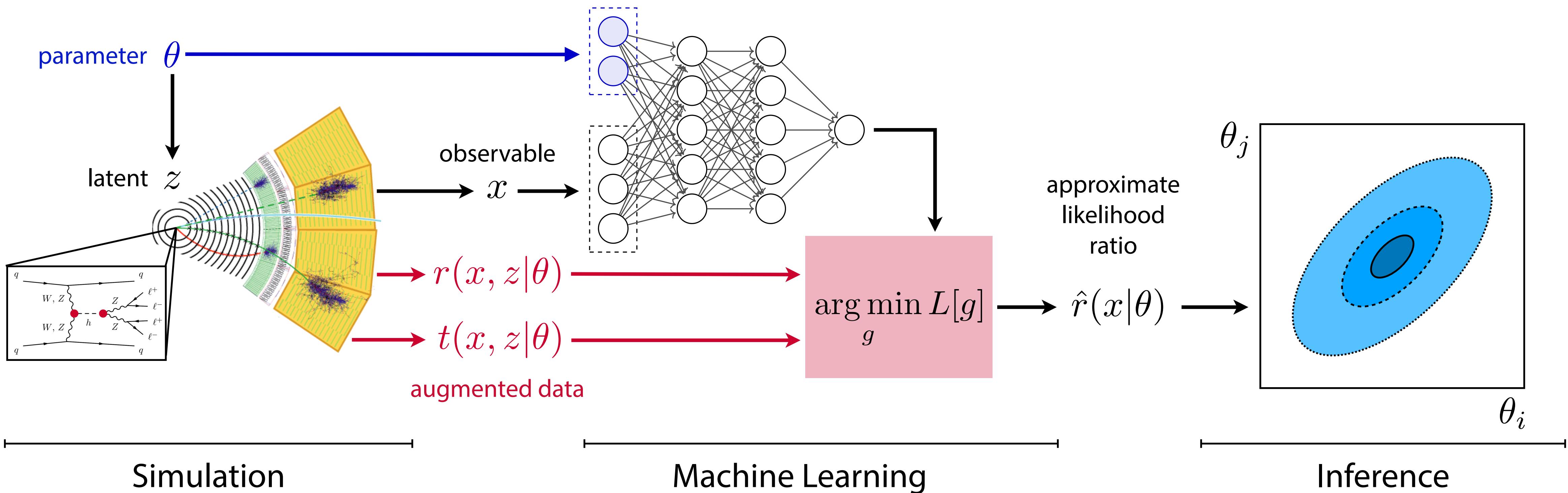
Learning with Augmented Data

arXiv:1805.12244

PRL, arXiv:1805.00013

PRD, arXiv:1805.00020

physics.aps.org/articles/v11/90



“Mining gold”: Extract additional information from simulator

Use this information to train estimator for likelihood ratio

Limit setting with standard hypothesis tests

EFT Decomposition

$$d\sigma \propto \left| \left(\mathcal{M}_{\text{SM}}^p + \sum_i \frac{f_i}{\Lambda^2} \mathcal{M}_i^p \right) \left(\mathcal{M}_{\text{SM}}^d + \sum_j \frac{f_j}{\Lambda^2} \mathcal{M}_j^d \right) \right|^2$$

Express EFT as a mixture:

$$p(x|\theta) = \sum_c w_c(\theta) p_c(x)$$

$w_c(\theta)$ are polynomials

$\nabla_\theta \log p(x|\theta)$ is now possible!

Process	Number of components for n operators					Σ
	$\mathcal{O}(\Lambda^0)$	$\mathcal{O}(\Lambda^{-2})$	$\mathcal{O}(\Lambda^{-4})$	$\mathcal{O}(\Lambda^{-6})$	$\mathcal{O}(\Lambda^{-8})$	
hV / WBF production	1	n	$\frac{n(n+1)}{2}$			$\frac{(n+1)(n+2)}{2}$
$h \rightarrow VV$ decay	1	n	$\frac{n(n+1)}{2}$			$\frac{(n+1)^2(n+2)}{2}$
Production + decay	1	n	$\frac{n(n+1)}{2}$	$\binom{n+2}{3}$	$\binom{n+3}{4}$	$\binom{n+4}{4}$

Table 1: Number of components c as given in Eq. (6) for different processes, sorted by their suppression by the EFT cutoff scale Λ .

For 2 BSM operators affecting VBF Higgs production and decay, we need a 15-D vector space

For 5 BSM operators we need 126-D vector space

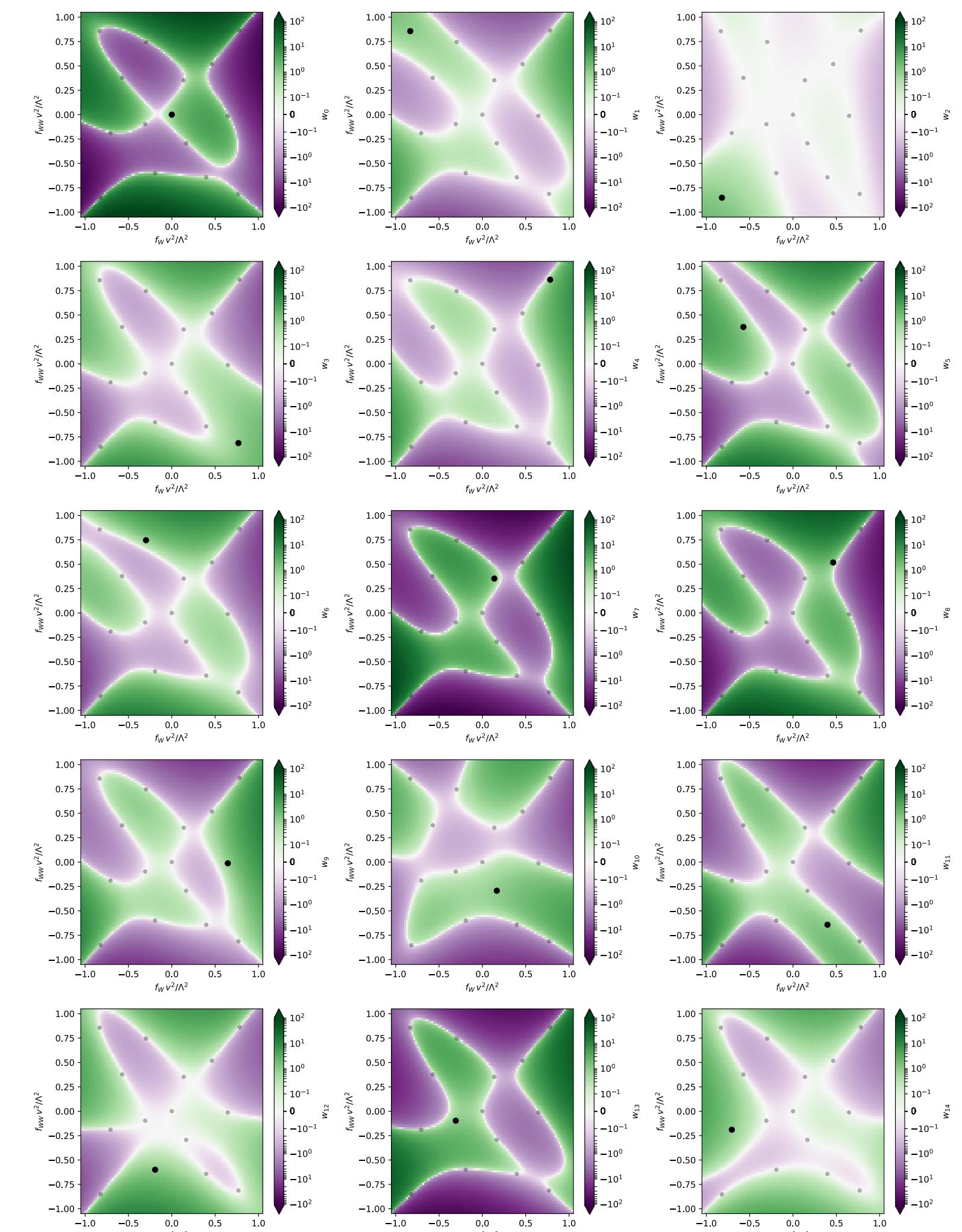
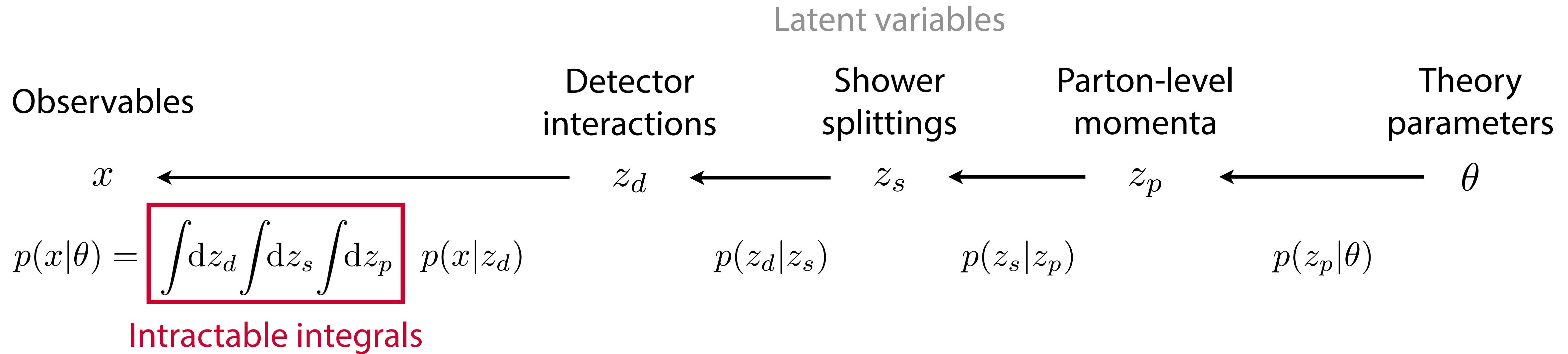
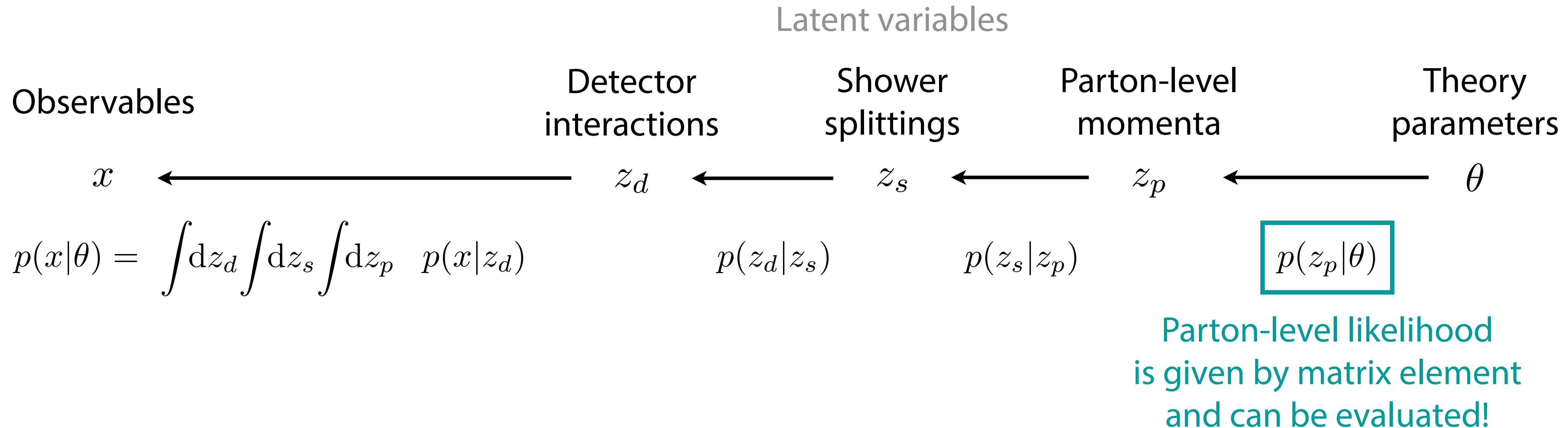


Figure 13: Morphing weights $w_i(\theta)$ for basis points distributed over the full relevant parameter space.

Mining gold from the simulator



Mining gold from the simulator



⇒ For each simulated event, we can calculate the **joint likelihood ratio** which depends on the specific evolution of the simulation:

$$r(x, z | \theta_0, \theta_1) \equiv \frac{p(x, z_d, z_s, z_p | \theta_0)}{p(x, z_d, z_s, z_p | \theta_1)} = \frac{p(x|z_d)}{p(x|z_d)} \frac{p(z_d|z_s)}{p(z_d|z_s)} \frac{p(z_s|z_p)}{p(z_s|z_p)}$$

$$\frac{p(z_p|\theta_0)}{p(z_p|\theta_1)} \sim \frac{|\mathcal{M}(z_p|\theta_0)|^2}{|\mathcal{M}(z_p|\theta_1)|^2}$$

The value of gold

We can calculate the **joint likelihood ratio**

$$r(x, z | \theta_0, \theta_1) \equiv \frac{p(x, z_d, z_s, z_p | \theta_0)}{p(x, z_d, z_s, z_p | \theta_1)}$$

("How much more likely is this simulated event, including all intermediate states, for θ_0 compared to θ_1 ?)



We want the **likelihood ratio function**

$$r(x | \theta_0, \theta_1) \equiv \frac{p(x | \theta_0)}{p(x | \theta_1)}$$

("How much more likely is the observation x for θ_0 compared to θ_1 ?)

The value of gold

We can calculate the **joint likelihood ratio**

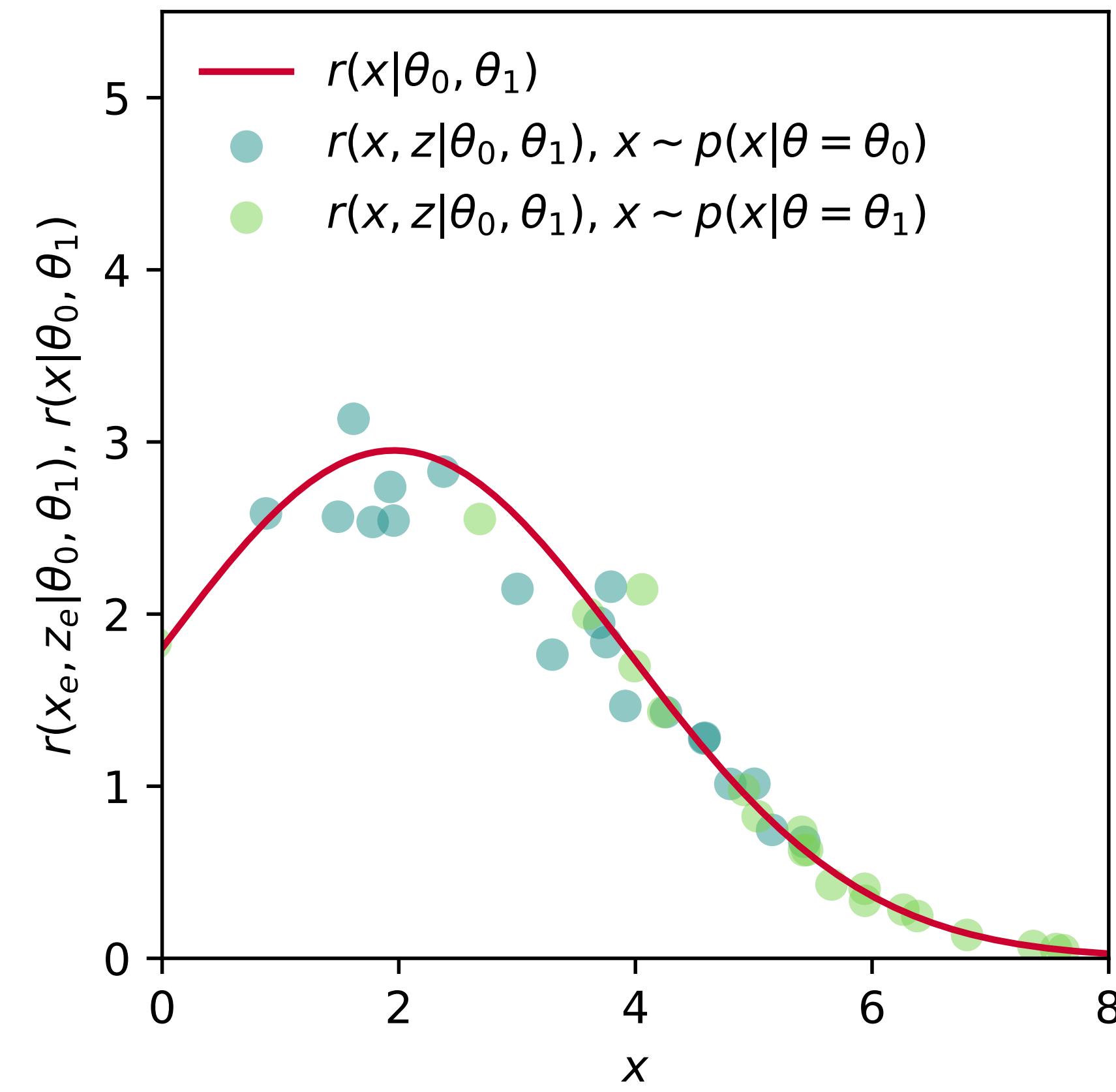
$$r(x, z | \theta_0, \theta_1) \equiv \frac{p(x, z_d, z_s, z_p | \theta_0)}{p(x, z_d, z_s, z_p | \theta_1)}$$



$r(x, z | \theta_0, \theta_1)$ are scattered around $r(x | \theta_0, \theta_1)$

We want the **likelihood ratio function**

$$r(x | \theta_0, \theta_1) \equiv \frac{p(x | \theta_0)}{p(x | \theta_1)}$$



The value of gold

We can calculate the joint likelihood ratio

$$r(x, z|\theta_0, \theta_1) \equiv \frac{p(x, z_d, z_s, z_p|\theta_0)}{p(x, z_d, z_s, z_p|\theta_1)}$$



We want the likelihood ratio function

$$r(x|\theta_0, \theta_1) \equiv \frac{p(x|\theta_0)}{p(x|\theta_1)}$$

With $r(x, z|\theta_0, \theta_1)$, we define a functional like

$$L_r[\hat{r}(x|\theta_0, \theta_1)] = \int dx \int dz p(x, z|\theta_1) \left[(\hat{r}(x|\theta_0, \theta_1) - r(x, z|\theta_0, \theta_1))^2 \right]$$

It is minimized by

$$\mathbb{E}_{z \sim p(z|x, \theta_1)} [r(x, z|\theta_0, \theta_1)] = \arg \min_{\hat{r}(x|\theta_0, \theta_1)} L_r[\hat{r}(x|\theta_0, \theta_1)]!$$

(And we can sample from $p(x, z|\theta)$ by running the simulator.)

The value of gold

With $r(x, z|\theta_0, \theta_1)$, we define a functional like

We can calculate the joint likelihood ratio

$$r(x, z|\theta_0, \theta_1) \equiv \frac{p(x, z_d, z_s, z_p|\theta_0)}{p(x, z_d, z_s, z_p|\theta_1)}$$

$$L_r[\hat{r}(x|\theta_0, \theta_1)] = \int dx \int dz p(x, z|\theta_1) [(\hat{r}(x|\theta_0, \theta_1) - r(x, z|\theta_0, \theta_1))^2]$$

It is minimized by

$$\mathbb{E}_{z \sim p(z|x, \theta_1)} [r(x, z|\theta_0, \theta_1)] = \arg \min_{\hat{r}(x|\theta_0, \theta_1)} L_r[\hat{r}(x|\theta_0, \theta_1)]!$$

(And we can sample from $p(x, z|\theta)$ by running the simulator.)

.... and then magic ...

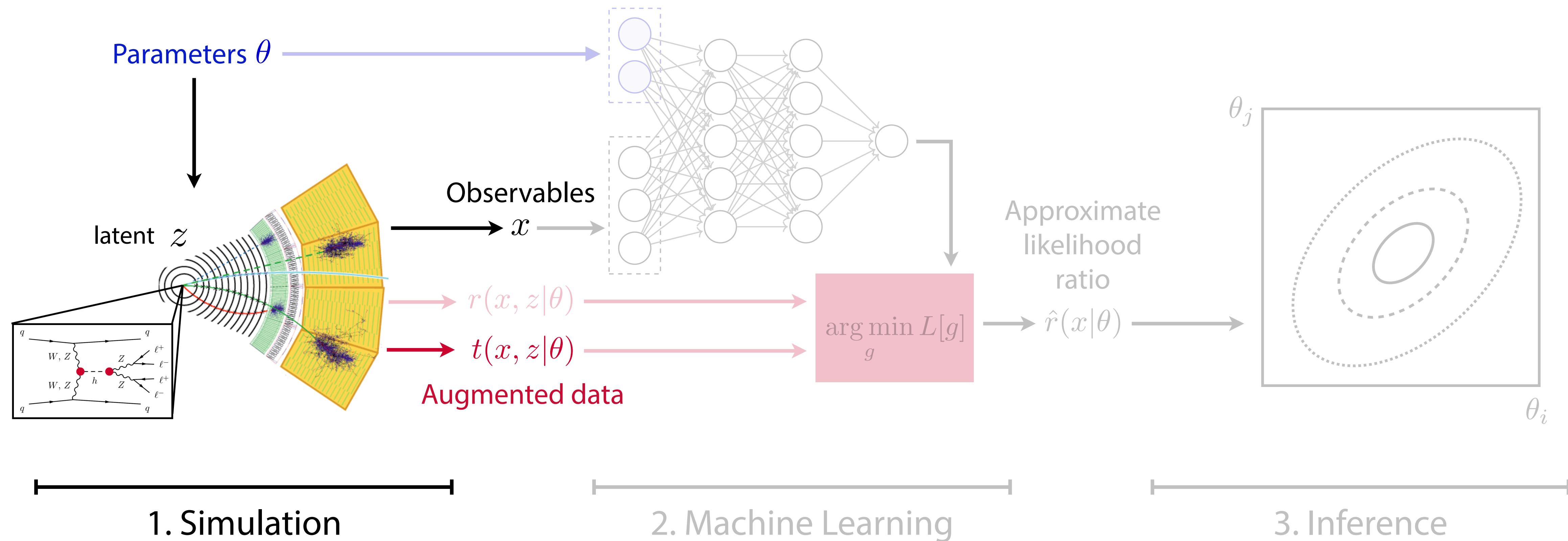
$$\begin{aligned} \mathbb{E}_{z \sim p(z|x, \theta_1)} [r(x, z|\theta_0, \theta_1)] &= \int dz p(z|x, \theta_1) \frac{p(x, z|\theta_0)}{p(x, z|\theta_1)} \\ &= \int dz \frac{p(x, z|\theta_1)}{p(x|\theta_1)} \frac{p(x, z|\theta_0)}{p(x, z|\theta_1)} \\ &= r(x|\theta_0, \theta_1) ! \end{aligned}$$

We want the likelihood ratio function

$$r(x|\theta_0, \theta_1) \equiv \frac{p(x|\theta_0)}{p(x|\theta_1)}$$



Learning with Augmented Data



Learning the score (related to optimal observables)

Similar to the joint likelihood ratio, from the simulator we can extract the **joint score**

$$t(x, z|\theta_0) \equiv \nabla_{\theta} \log p(x, z_d, z_s, z_p|\theta) \Big|_{\theta_0}$$



We want the **score**

$$t(x|\theta_0) \equiv \nabla_{\theta} \log p(x|\theta) \Big|_{\theta_0}$$

Learning the score (related to optimal observables)

Similar to the joint likelihood ratio, from the simulator we can extract the **joint score**

$$t(x, z|\theta_0) \equiv \nabla_{\theta} \log p(x, z_d, z_s, z_p|\theta) \Big|_{\theta_0}$$



We want the **score**

$$t(x|\theta_0) \equiv \nabla_{\theta} \log p(x|\theta) \Big|_{\theta_0}$$

Given $t(x, z|\theta_0)$,
we define the functional

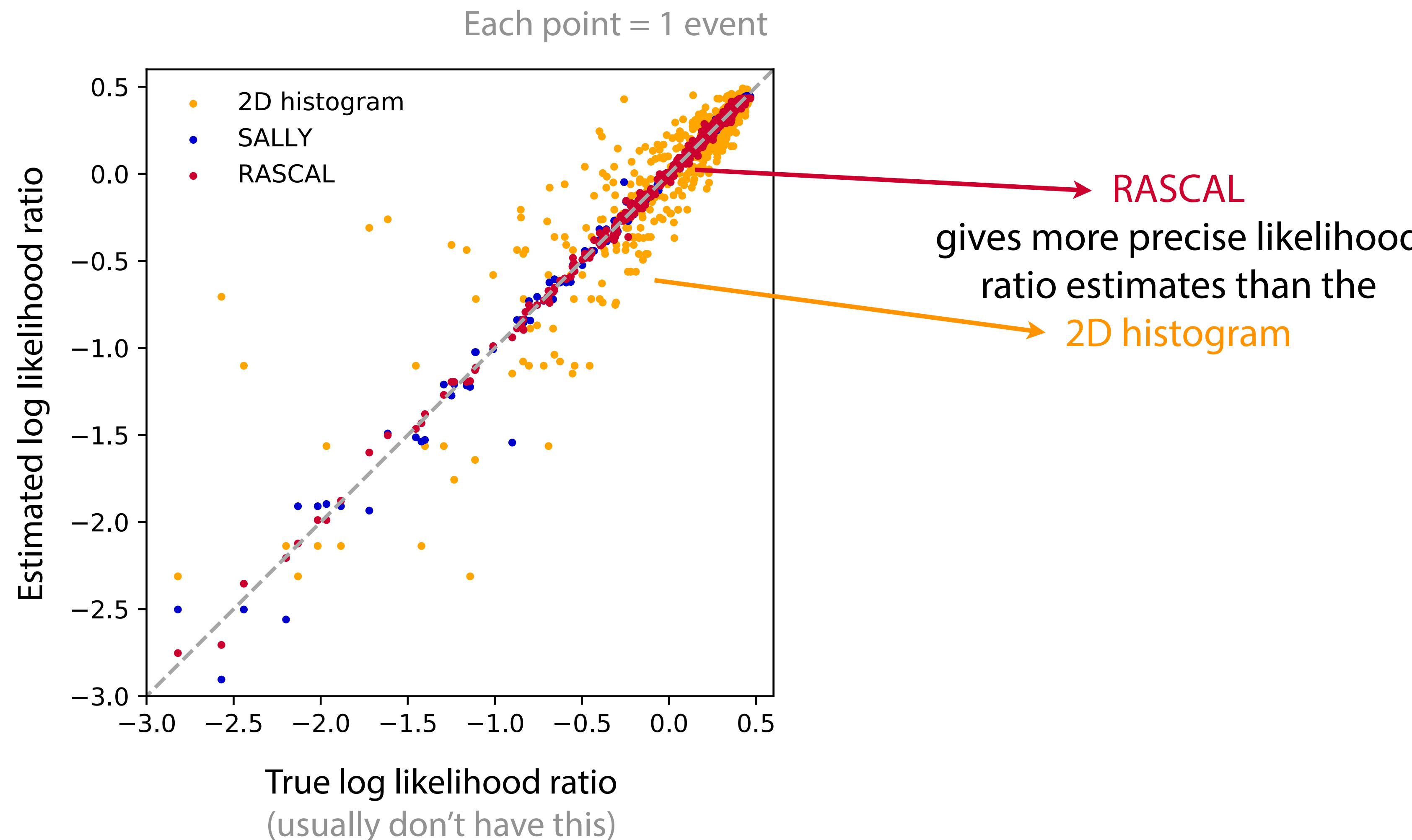
$$L_t[\hat{t}(x|\theta_0)] = \int dx \int dz \ p(x, z|\theta_0) \left[(\hat{t}(x|\theta_0) - t(x, z|\theta_0))^2 \right].$$

One can show it is minimized by

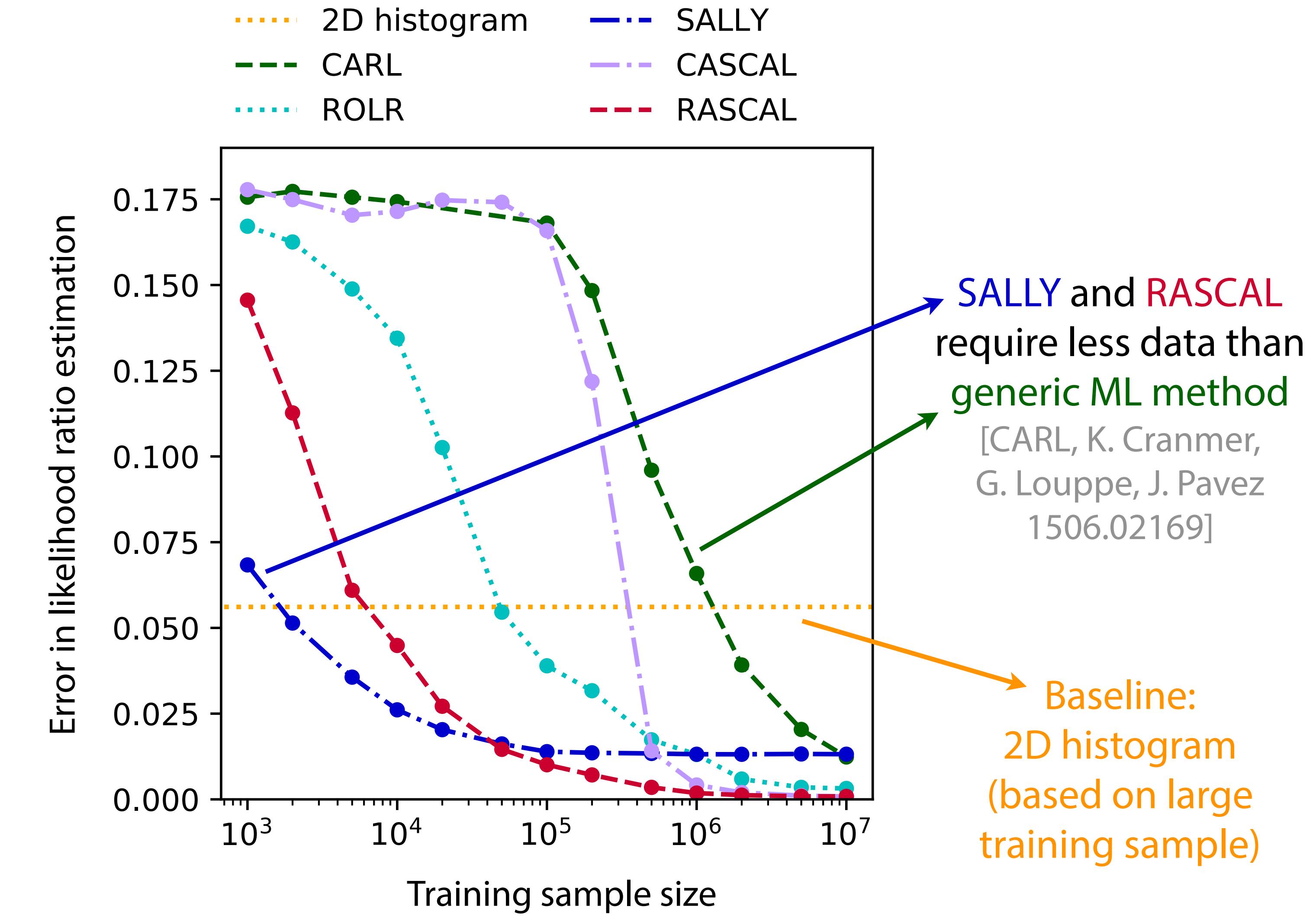
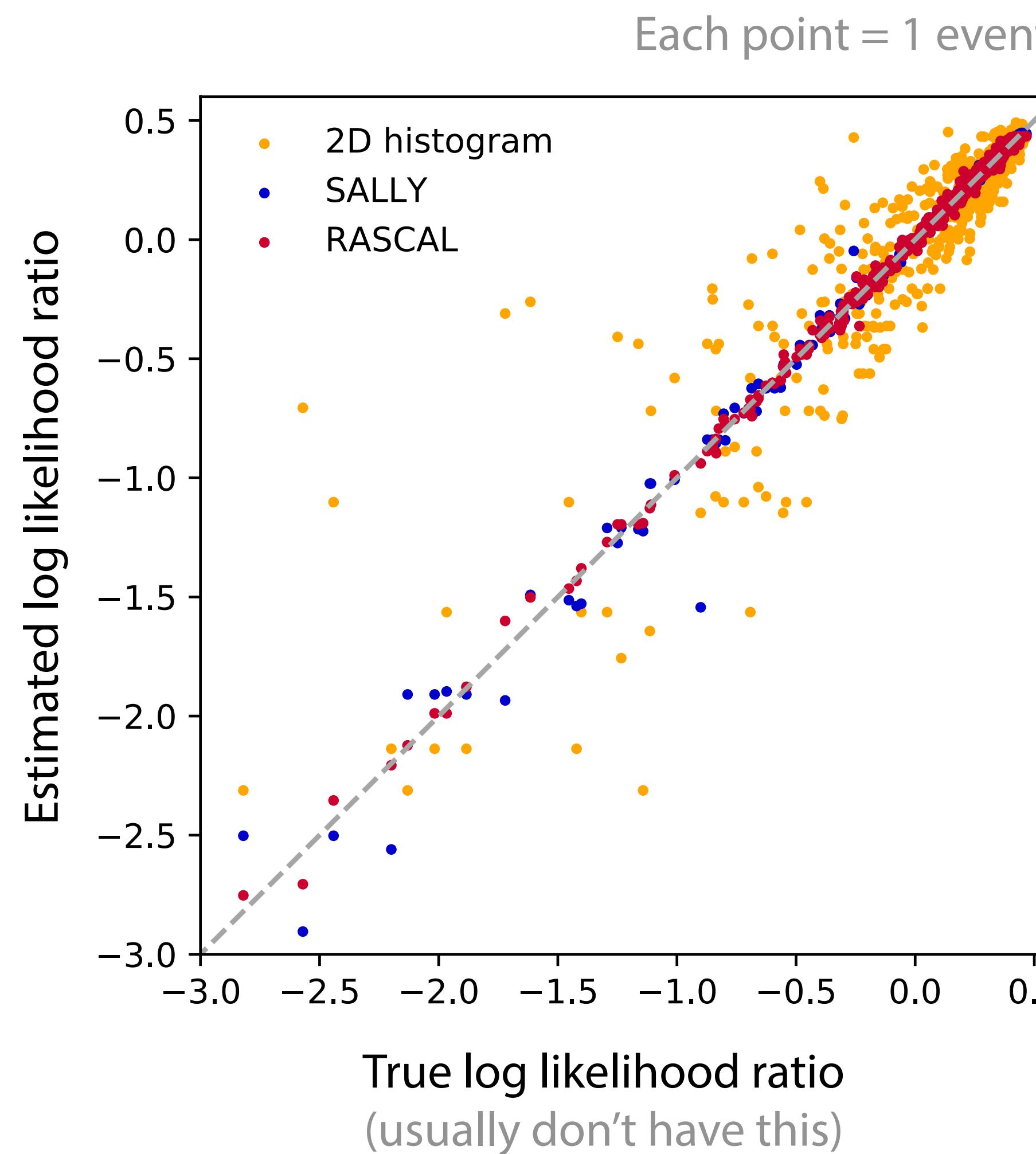
$$t(x|\theta_0) = \arg \min_{\hat{t}(x|\theta_0)} L_t[\hat{t}(x|\theta_0)].$$

Again, we implement this minimization through machine learning.

More precise likelihood ratio estimates with less training data



More precise likelihood ratio estimates with less training data



MadMiner automates all of these methods.

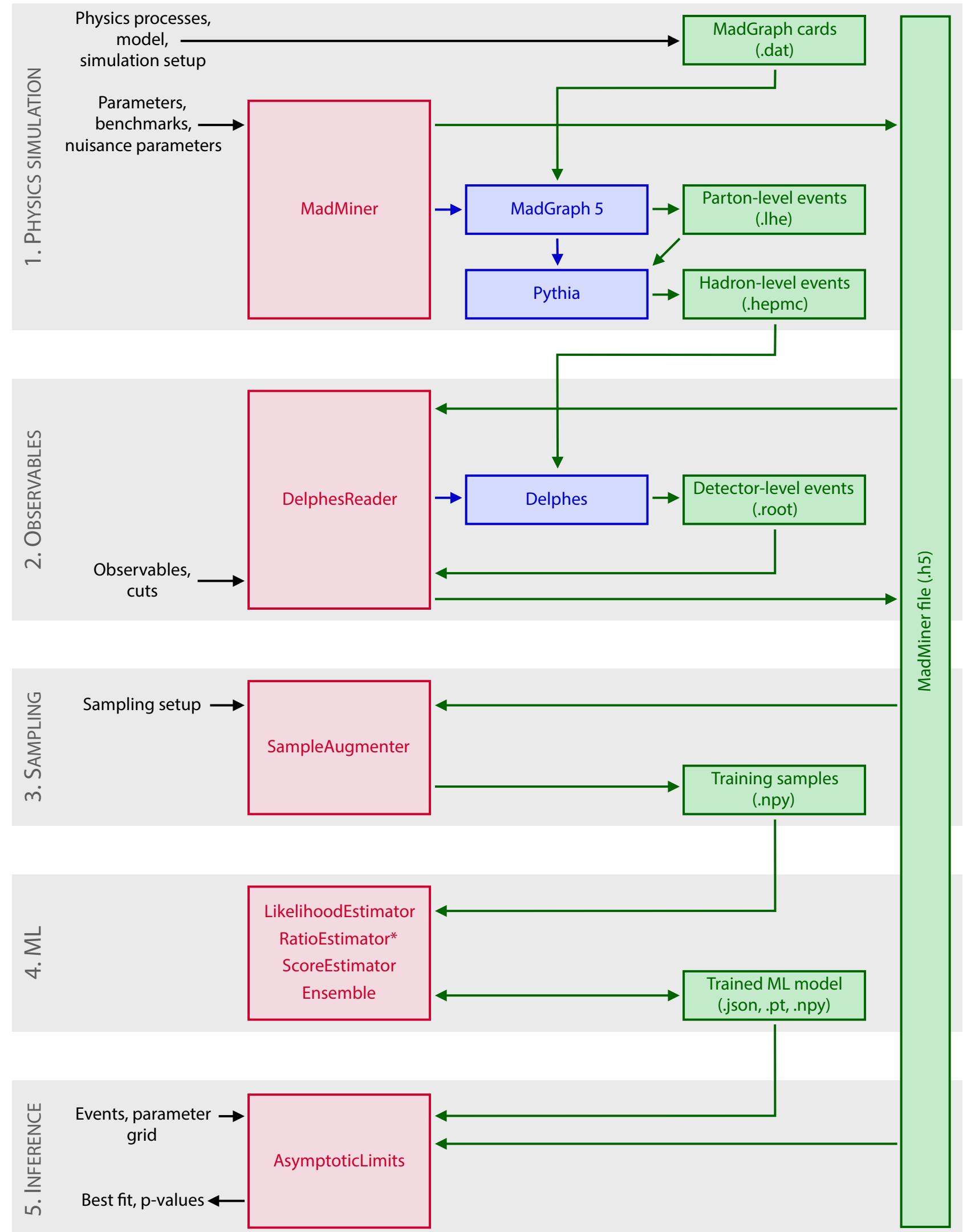
[JB, F. Kling, I. Espejo, K. Cranmer 1907.10621]

MadMiner

[JB, F. Kling, I. Espejo, K. Cranmer 1907.10621]

New Python package **MadMiner** makes it straightforward to apply the new techniques to LHC problems

- Out of the box: Pheno-level analyses
 - MadGraph, Pythia, Delphes, (could be GEANT4)
 - Systematic uncertainties from PDF / scale variation
- Scalable to state-of-the-art experimental tools
 - Mostly requires bookkeeping of fully differential cross sections
- Modular interface
 - Extensive documentation
 - Embedded into Python / ML ecosystem



MadMiner resources

MadMiner: Machine learning–based inference for particle physics

By Johann Brehmer, Felix Kling, Irina Espejo, and Kyle Cranmer

pypi package 0.6.3 build passing docs failing chat on gitter code style black License MIT DOI 10.5281/zenodo.1489147 arXiv 1907.10621

Introduction

Particle physics processes are usually modeled with complex Monte-Carlo simulations of the hard process, parton shower, and detector interactions. These simulators typically do not admit a tractable likelihood function: given a (potentially high-dimensional) set of observables, it is usually not possible to calculate the probability of these observables for some model parameters. Particle physicists usually tackle this problem of "likelihood-free inference" by hand-picking a few "good" observables or summary statistics and filling histograms of them. But this conventional

UCI-TR-2019-16, SLAC-PUB-17461

MadMiner: Machine learning–based inference for particle physics

Johann Brehmer,^{1,*} Felix Kling,^{2,3,†} Irina Espejo,^{1,‡} and Kyle Cranmer^{1,§}

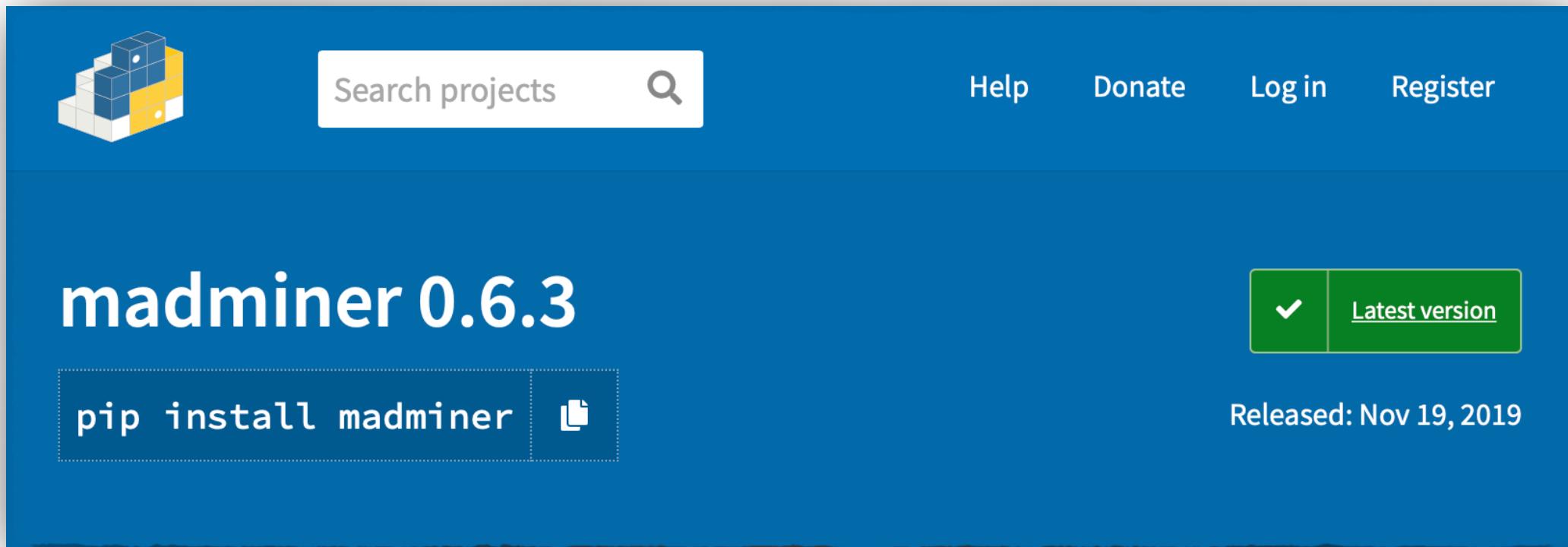
¹ Center for Data Science and Center for Cosmology and Particle Physics, New York University, New York, NY 10003, USA

² Department of Physics and Astronomy, University of California, Irvine, CA 92697, USA

³ SLAC National Accelerator Laboratory, 2575 Sand Hill Road, Menlo Park, CA 94025, USA

Precision measurements at the LHC often require analyzing high-dimensional event data for subtle kinematic signatures, which is challenging for established analysis methods. Recently, a powerful family of multivariate inference techniques that leverage both matrix element information and machine learning has been developed. This approach neither requires the reduction of high-dimensional data to summary statistics nor any simplifications to the underlying physics or detector response. In this paper we introduce **MadMiner**, a Python module

Repository and tutorials:
github.com/johannbrehmer/madminer



Installation:
`pip install madminer`

Docs » MadMiner Edit on GitHub

MadMiner

Johann Brehmer, Felix Kling, Irina Espejo, and Kyle Cranmer

An inference toolkit for LHC measurements

Note that this is a development version. Do not rely on anything being stable. If you have any questions, please contact us at johann.brehmer@nyu.edu.

Sites

- [Introduction to MadMiner](#)
- [Getting started](#)

REFERENCE

- madminer.analysis module
- madminer.core module
- madminer.delphes module

API documentation:
madminer.readthedocs.io

Hands-on Tutorial

- Step-by-step instructions
- Do Preliminaries
 - Need to install Docker
 - And then start Jupyter
- Step 1 is fast
- Step 2 takes ~25 min
- Step 3 takes ~20 min
- While they are running I will lecture
- Then we will finish with results

The screenshot shows the 'MadMiner Tutorial' website at <https://cranmer.github.io/madminer-tutorial/>. The page has a header with a back arrow and a download icon. The main content area is titled 'Introduction' and contains a brief introduction to the MadMiner tutorial, developed by Johann Brehmer, Felix Kling, Irina Espejo, and Kyle Cranmer, built using Jupyter Book. Below this is a diagram illustrating the workflow: 'Simulation' (parameter θ leads to latent z , which leads to observable x) → 'Machine Learning' (observable x leads to $r(x, z|\theta)$, $t(x, z|\theta)$, augmented data, and $\arg \min_g L[g]$) → 'Inference' ($\hat{r}(x|\theta)$ leads to approximate likelihood ratio, θ_j , and θ_i). The left sidebar lists the tutorial sections: Introduction, MadMiner Tutorial, Preliminaries, Overview, Define process to study *, Morphing, Interactive Morphing Demo, Create training data, Set MadGraph Directory, Parton Level *, With Delphes, Train model, Likelihood Ratio *, Score *, Likelihood, Statistical Analysis, Limits on EFT parameters *, Fisher Information, Information Geometry, and Congratulations.

Introduction

MadMiner tutorial

This is a tutorial on [MadMiner](#) developed by Johann Brehmer, Felix Kling, Irina Espejo, and Kyle Cranmer. It is built using [Jupyter Book](#).

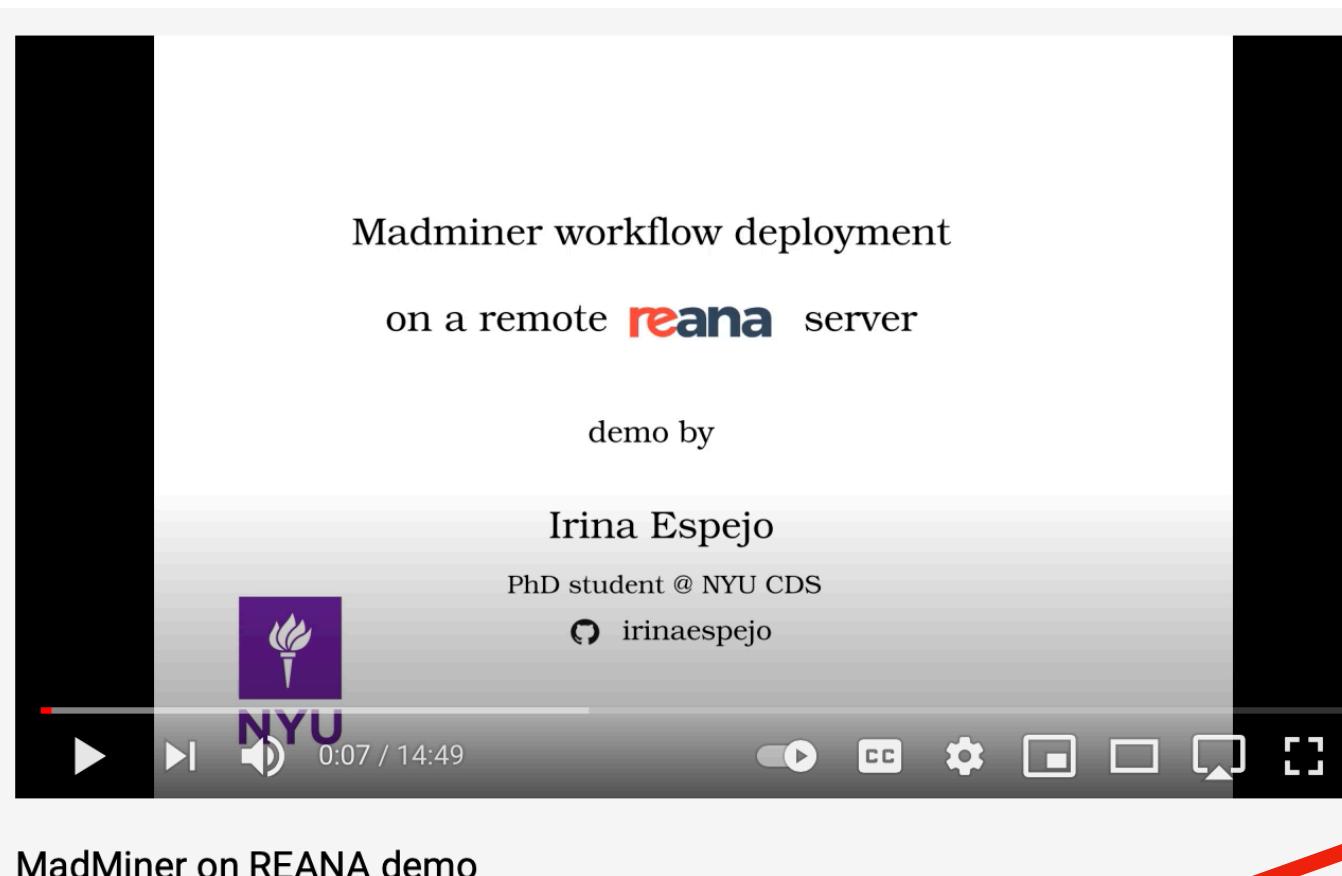
Introduction to MadMiner

Particle physics processes are usually modelled with complex Monte-Carlo simulations of the hard process, parton shower, and detector interactions. These simulators typically do not admit a tractable likelihood function: given a (potentially high-dimensional) set of observables, it is usually not possible to calculate the probability of these observables for some model parameters. Particle physicists usually tackle this problem of “likelihood-free inference” by hand-picking a few “good” observables or summary statistics and filling histograms of them. But this conventional approach discards the information in all other observables and often does not scale well to high-dimensional problems.

In the three publications “[Constraining Effective Field Theories With Machine Learning](#)”, “[A Guide to Constraining Effective Field Theories With Machine Learning](#)”, and “[Mining gold from implicit models to improve likelihood-free inference](#)”, a new approach has been developed. In a nut shell, additional information is extracted from the simulations that is closely related to the matrix elements that determine the hard process. This “augmented data” can be used to train neural networks to efficiently approximate arbitrary likelihood ratios. We playfully call this process “mining gold” from the simulator, since this information may be hard to get, but turns out to be very valuable for inference.

New: MadMiner on REANA

- Step-by-step instructions
- Do Preliminaries
 - Setup REANA Client
 - Get the workflow
 - Run on REANA
 - Run locally



- There's also a video

<https://www.youtube.com/watch?v=jzU5i-FKy6g>

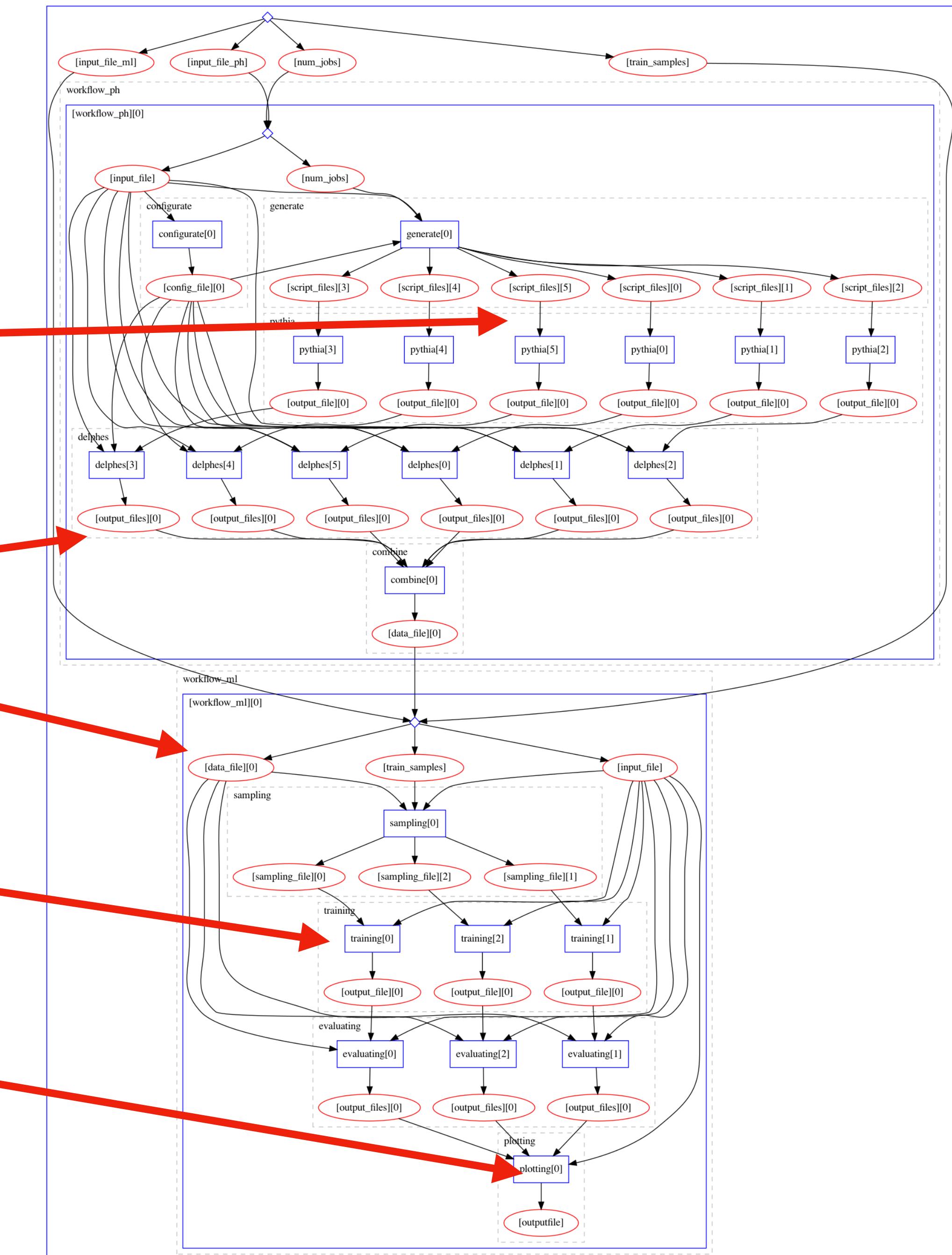
The screenshot shows a web browser window with the URL <https://theoryandpractice.org/madminer-tutorial/reana/preliminaries.html>. The page title is "MadMiner on REANA". On the left, there is a sidebar with a list of steps:

- Create training data
- Set MadGraph Directory
- Parton Level *
- With Delphes
- Train model
- Likelihood Ratio *
- Score *
- Likelihood
- Statistical Analysis
- Limits on EFT parameters *
- Fisher Information
- Information Geometry
- Congratulations
- Systematic uncertainties
- Systematic uncertainties
- Ensemble methods
- Reweighting existing samples

On the right, the main content area starts with a paragraph about the workflow integration. It mentions generating training data, running jobs on a cluster, comparing methods, tuning hyperparameters, and using the REANA execution system and Yadage workflow language. It also discusses scaling to large numbers of events and parameter scans, supported by the SCAILFIN project and IRIS-HEP. The text ends with information about the workflow containing all necessary elements for local and remote execution, integrating with MLFlow, and saving machine learning metrics. At the bottom, there is a section titled "Technologies" with a note about the technologies the workflow relies on.

The MadMiner for REANA

- Designed to scale to a cluster
 - Here six jobs for event generation and DELPHES simulation
 - Uses pre-built docker containers
- Combines data generation and machine learning sub-workflows
- ML / inference workflow is configurable
 - Here 3 methods (SALLY, ALICE, ALICES) are being trained and evaluated
 - Followed by plots
- Launch the entire workflow with a single command:
 - make run-reana (remote)
 - make run-yadage (locally)



Configuring the workflow

- You can get the workflow from GitHub
 - Has top-level combined workflow as well as
 - madminer-ph sub-workflow
 - madminer-ml sub-workflow
 - Uses pre-built docker containers
- See madminer-*/workflow/input.yml for configuration

These configuration options mirror the options you see in the python notebooks

- See madminer-*/workflow/input.yml for configuration

```
madminer-workflow$ git clone --recurse-submodules https://github.com/scailfin/madminer-workflow
madminer-workflow$ cd madminer-workflow/
madminer-workflow$ l
total 32
-rw-r--r-- 1 cranmer staff 1.1K Feb 11 16:11 LICENSE
-rw-r--r-- 1 cranmer staff 1.4K Feb 11 16:11 Makefile
-rw-r--r-- 1 cranmer staff 4.9K Feb 11 16:11 README.md
drwxr-xr-x 3 cranmer staff 96B Feb 11 16:11 docs
drwxr-xr-x 4 cranmer staff 128B Feb 11 16:11 modules
drwxr-xr-x 6 cranmer staff 192B Feb 11 16:14 reana
madminer-workflow$ l modules/
total 0
drwxr-xr-x 17 cranmer staff 544B Feb 11 16:11 madminer-workflow-ml
drwxr-xr-x 15 cranmer staff 480B Feb 11 16:11 madminer-workflow-ph
madminer-workflow$ l modules/madminer-workflow-ph/workflow/input.yml
-rw-r--r-- 1 cranmer staff 1.4K Feb 11 16:11 modules/madminer-workflow-ph/workflow/input.yml
madminer-workflow$ l modules/madminer-workflow-ml/workflow/input.yml
-rw-r--r-- 1 cranmer staff 2.4K Feb 11 16:11 modules/madminer-workflow-ml/workflow/input.yml
```

```
parameters:
  - parameter_name: 'CWL2'
    lha_block: 'dim6'
    lha_id: 2
    morphing_max_power: 2
    param_card_transform: '16.52*theta'
    parameter_range: (-10.0,10.0)

  - parameter_name: 'CPWL2'
    lha_block: 'dim6'
    lha_id: 5
    morphing_max_power: 2
    param_card_transform: '16.52*theta'
    parameter_range: (-10.0,10.0)

# The number of benchmark params must match the number of parameters above
benchmarks:
  - benchmark: 1
    name: 'sm'
    parameter_name_1: 'CWL2'
    value_1: 0.0
    parameter_name_2: 'CPWL2'
    value_2: 0.0

  - benchmark: 2
    name: 'w'
    parameter_name_1: 'CWL2'
    value_1: 10.0
    parameter_name_2: 'CPWL2'
    value_2: 0.0
```

```
#####
### SHARED AMONG STEPS #####
#####

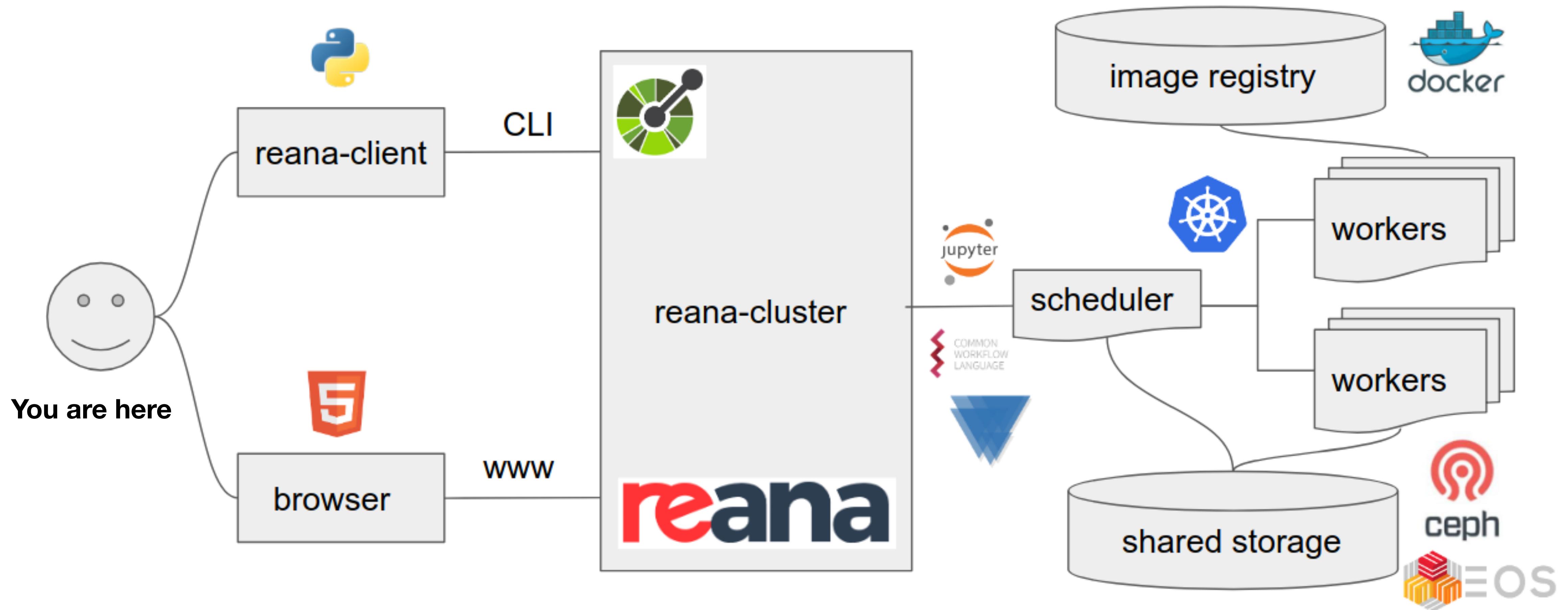
estimator: 'parameterized'
methods: ['alice', 'alices', 'sally']

# Region of theory we want to test
asymptotic_limits:
  region:
    theta_0:
      min: -20.0
      max: 20.0
    theta_1:
      min: -20.0
      max: 20.0
  resolutions: [25, 25]
  theta_true: [0.0 ,0.0]

#####
##### STEP: SAMPLING #####
#####

alice:
  # When selecting random_morphing points as sampling method, the number of prior parameters
  # must be equal to the parameters specified in the Physics workflow 'input.yml' file
  # Ref: https://github.com/scailfin/madminer-workflow-ph/blob/master/workflow/input.yml#L5
  theta_0:
    sampling_method: 'random_morphing_points'
    sampling_number: 20
  prior:
    - parameter_0:
        prior_shape: 'gaussian'
        prior_param_0: 0.0
        prior_param_1: 1.0
```

Schematic for REANA



REANA Monitor

https://localhost:30443

reana

Your workflows

Refreshed at 18:50:50 UTC

- madminer-workflow #6** finished in 10 min 32 sec
Finished 17 minutes ago step 23/1
- rootfit #1.1** finished in 33 seconds
Finished 19 days ago step 2/2

madminer-workflow #6
finished in 10 min 32 sec
step 23/1

workflow_ml/plotting/plots/all_methods.png

Download

https://localhost:30443/details/db99ad6c-0475-4a65 80%

reana

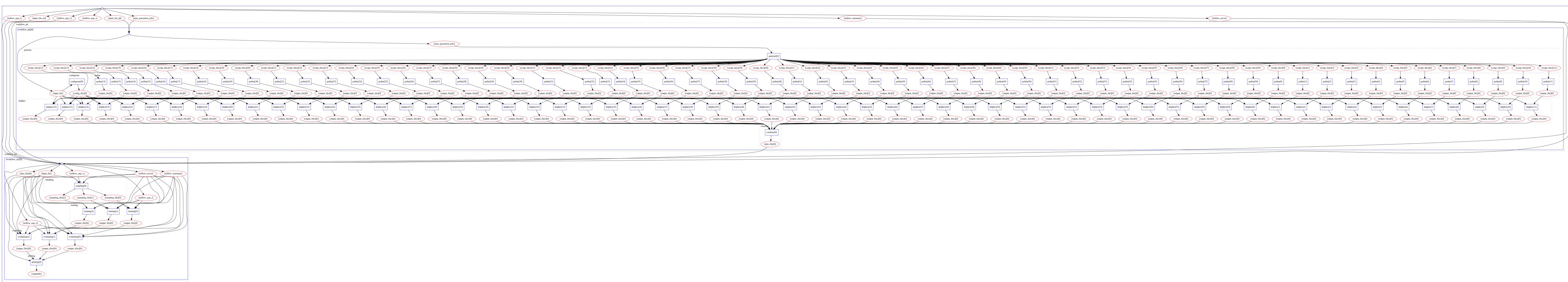
✓ **madminer-workflow #6**
Finished 18 minutes ago

finished in 10 min 32 sec
step 23/1

Logs Workspace Specification

Name	Modified	Size
workflow.yml	2021-02-10T18:22:52	884
ph/input.yml	2021-02-10T18:22:50	1421
_yadage/yadage_workflow_instance.dot	2021-02-10T18:33:25	21123
_yadage/yadage_snapshot_workflow.json	2021-02-10T18:33:23	159601
_yadage/yadage_workflow_instance.png	2021-02-10T18:33:24	564333
_yadage/yadage_workflow_instance.pdf	2021-02-10T18:33:25	31626
ml/input.yml	2021-02-10T18:22:51	2425
ph/yadage/workflow.yml	2021-02-10T18:22:50	1729
ph/yadage/steps.yml	2021-02-10T18:22:50	2029
_yadage/adage/adagesnap.txt	2021-02-10T18:33:23	58474
_yadage/adage/workflow.gif	2021-02-10T18:33:23	1814833
workflow_ph/pythia_4/nsqso_born.inc	2021-02-10T18:24:50	56
workflow_ph/pythia_4/py.py	2021-02-10T18:24:52	8778
workflow_ph/generate/folder_1.tar.gz	2021-02-10T18:23:54	5458911
workflow_ph/generate/py.py	2021-02-10T18:23:48	8778

REANA Monitor



Workflow #1: 6 generation jobs for the various benchmark points

Workflow #2: 60 generation jobs for 10x at each of the points

Note: both took ~17 min to run.

Your workflows

Refreshed at 12:25:15 UTC

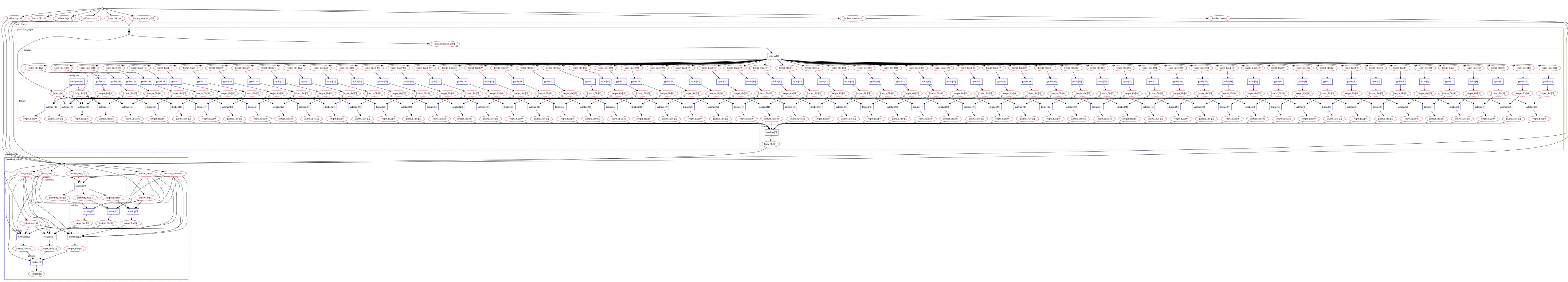
✓ madminer-workflow #2
Finished 9 hours ago

finished in 17 min 14 sec
step 131/1

✓ madminer-workflow #1
Finished 9 hours ago

finished in 16 min 37 sec
step 23/1

REANA Monitor



Workflow #1: 6 generation jobs for the various benchmark points

Workflow #2: 60 generation jobs for 10x at each of the points

Note: both took ~17 min to run.

Your workflows

Refreshed at 12:25:15 UTC

✓ madminer-workflow #2
Finished 9 hours ago

finished in 17 min 14 sec
step 131/1

✓ madminer-workflow #1
Finished 9 hours ago

finished in 16 min 37 sec
step 23/1