Recent progress in ML applications for LHC phenomenology

Benjamin Nachman

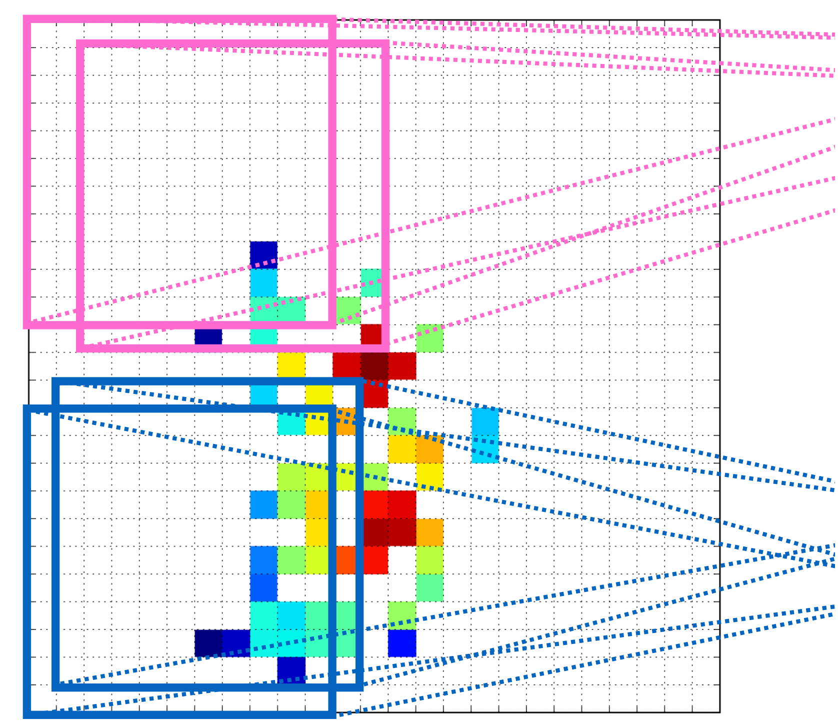
Lawrence Berkeley National Laboratory

bpnachman.com

bpnachman@lbl.gov

@bpnachman

bnachman



BERKELEY LAB

BIDS

BERKELEY INSTITUTE FOR DATA SCIENCE

Virtual Houches

June 17, 2021

Data analysis in particle physics

Theory of everything

↓

Physics simulators

↓

Detector-level observables

↓

Pattern recognition

↔

Nature

↓

Experiment

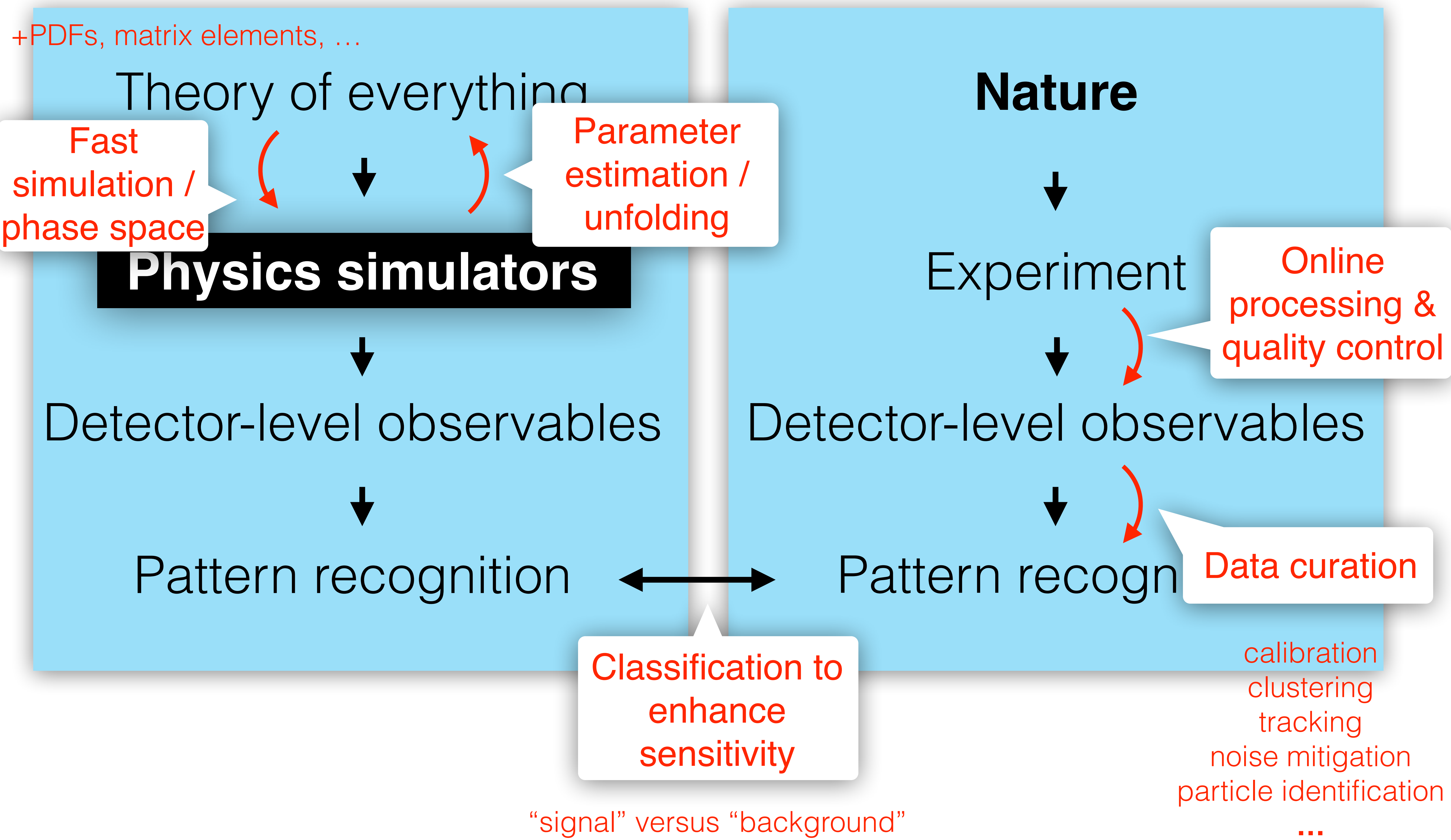
↓

Detector-level observables

↓

Pattern recognition

Data analysis in particle physics + ML



Data analysis in particle physics

Theory of everything

↓

Physics simulators

↓

Detector-level observables

↓

Pattern recognition

Nature

↓

Experiment

↓

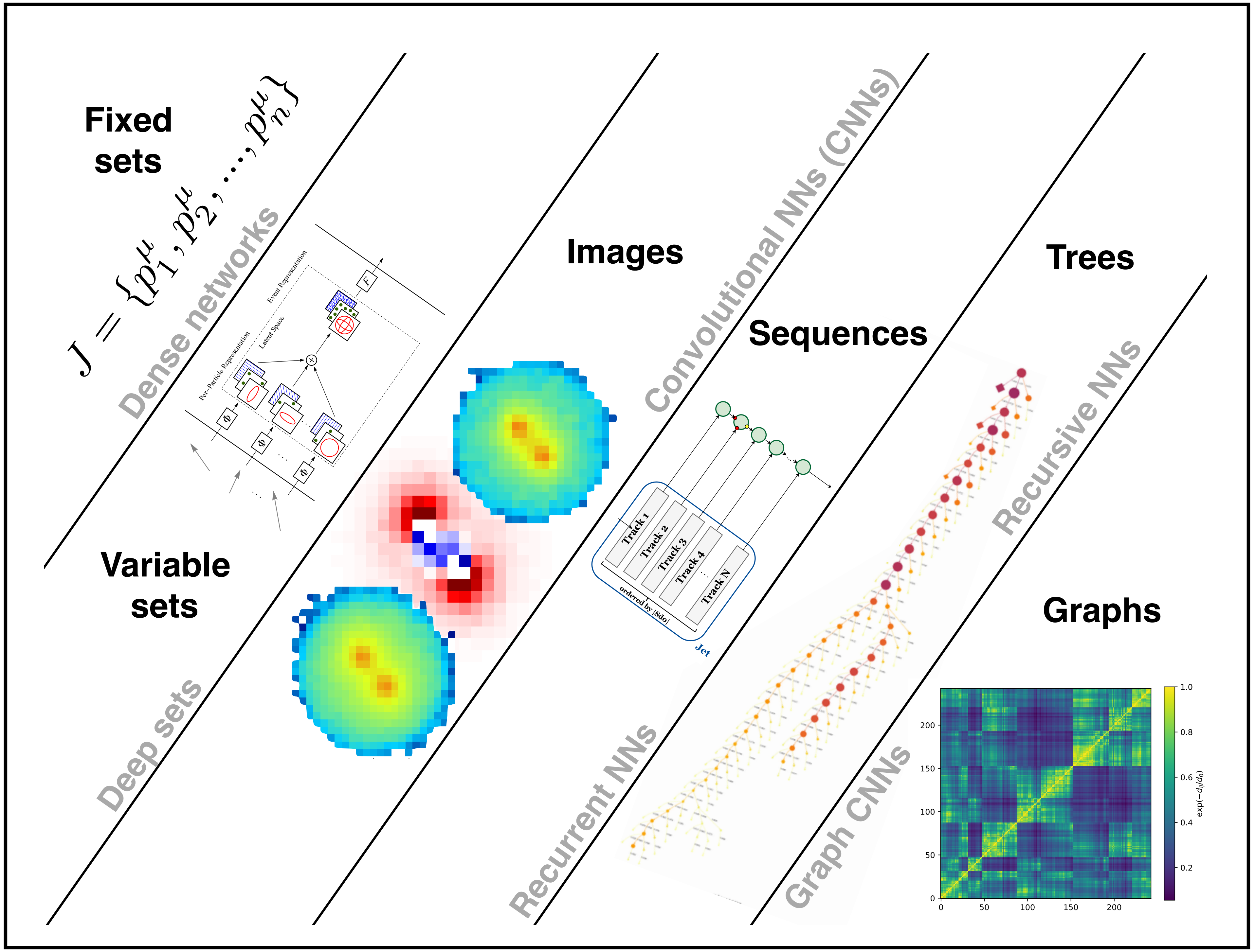
Detector-level observables

↓

Pattern recognition

This is where most machine learning is being applied.

Representing our data



Larkoski, Moult, and Nachman, 1709.04464, with images from Komiske, Metodiev, Thaler, 1810.05165; ATLAS, PUB-2017-003; T. Cheng, 1711.02633; Henrion et al. MLPS @ NeurIPS 2017

Data analysis in particle physics

Theory of everything

↓

Physics simulators

↓

Detector-level observables

↓

Pattern recognition ⟷ Pattern recognition

Nature

↓

Experiment

↓

Detector-level observables

↓

Pattern recognition

The growing toolkit of generative models are being developed to accelerate or augment simulations.

Deep Generative Models

A **generator** is nothing other than a function that maps random numbers to structure.

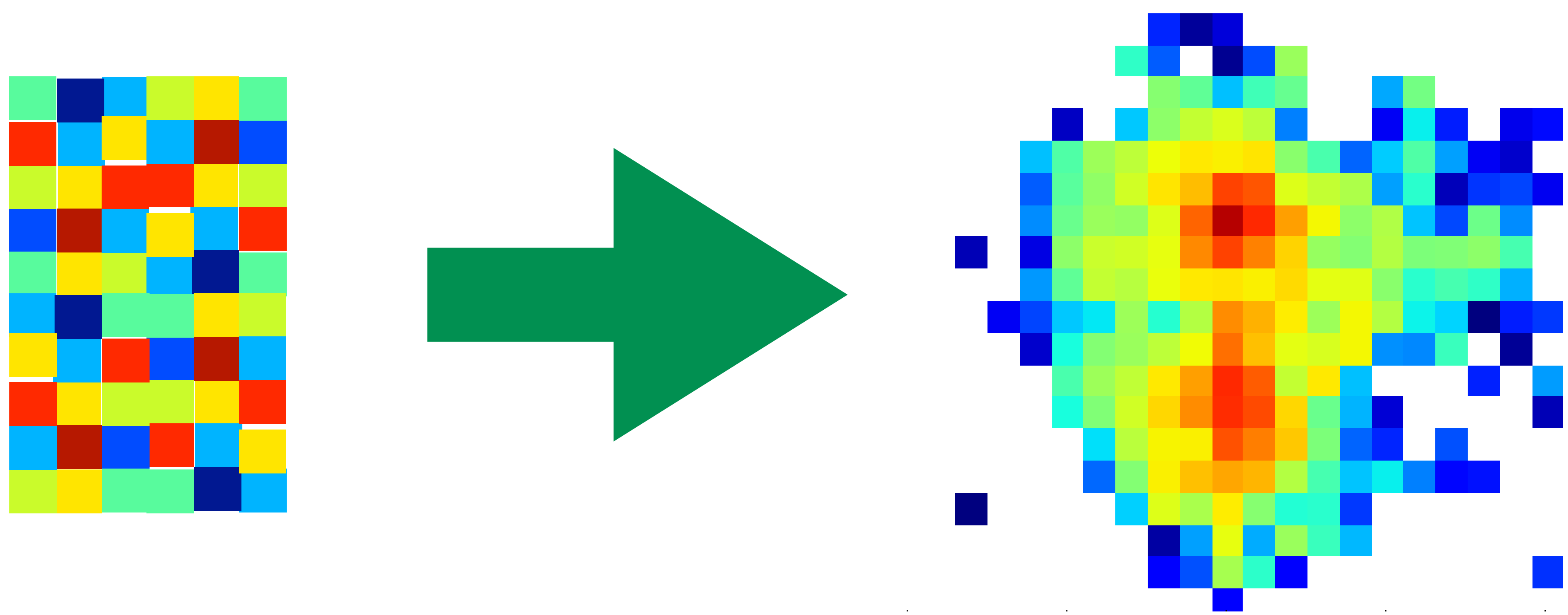


Image: Paganini, Oliveira, Nachman, 1705.02355

Deep Generative Models in HEP

Speeding up slow simulation

Generating Phase space

Estimating SM backgrounds

Measurements and Inference

BSM searches

N.B. being comprehensive with citations would fill up the slide - please see my link to the Living Review at the end for a comprehensive list

Deep Generative Models in HEP

Speeding up slow simulation

Generating Phase space

Estimating SM backgrounds

Measurements and Inference

BSM searches

N.B. being comprehensive with citations would fill up the slide - please see my link to the Living Review at the end for a comprehensive list

N.B. being comprehensive with citations would fill up the slide - please see my link to the Living Review at the end for a comprehensive list

Deep Generative Models in HEP

Speeding up slow simulation

Generating Phase space

Estimating SM backgrounds

Measurements and Inference

BSM searches

N.B. being comprehensive with citations would fill up the slide - please see my link to the Living Review at the end for a comprehensive list

Deep Generative Models in HEP

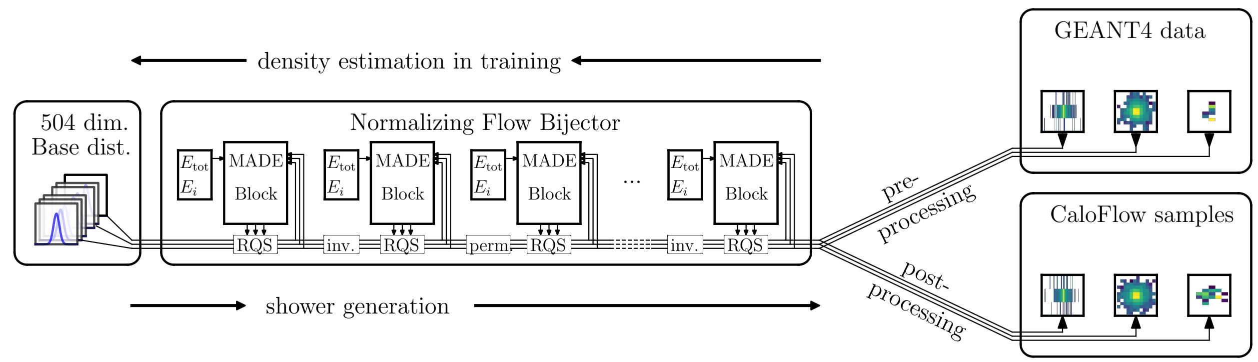
Speeding up slow simulation

Generating Phase space

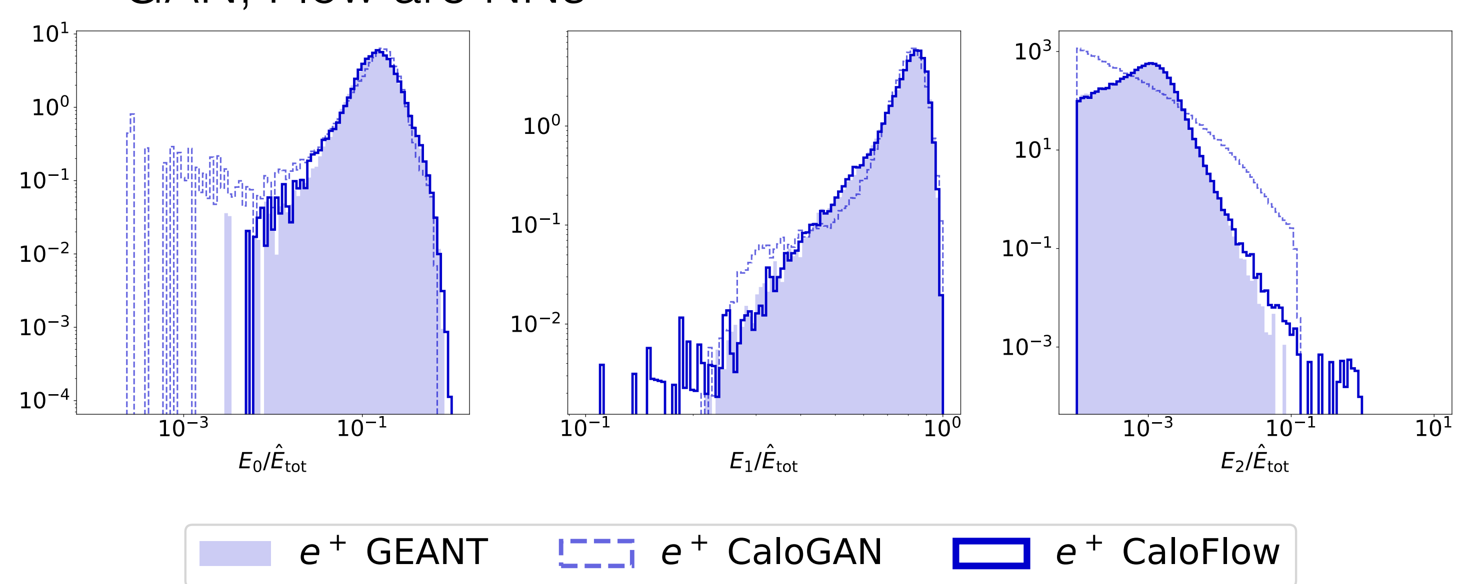
Estimating SM backgrounds

Measurements and Inference

BSM searches



GAN, Flow are NNs



CaloFlow: Krause and Shih, 2106.05285

CaloGAN: Paganini, Oliveira, Nachman, 1705.02355

not quite a fair comparison, but the state-of-the-art accuracy is highly non-trivial and very impressive!

N.B. being comprehensive with citations would fill up the slide - please see my link to the Living Review at the end for a comprehensive list

Deep Generative Models in HEP

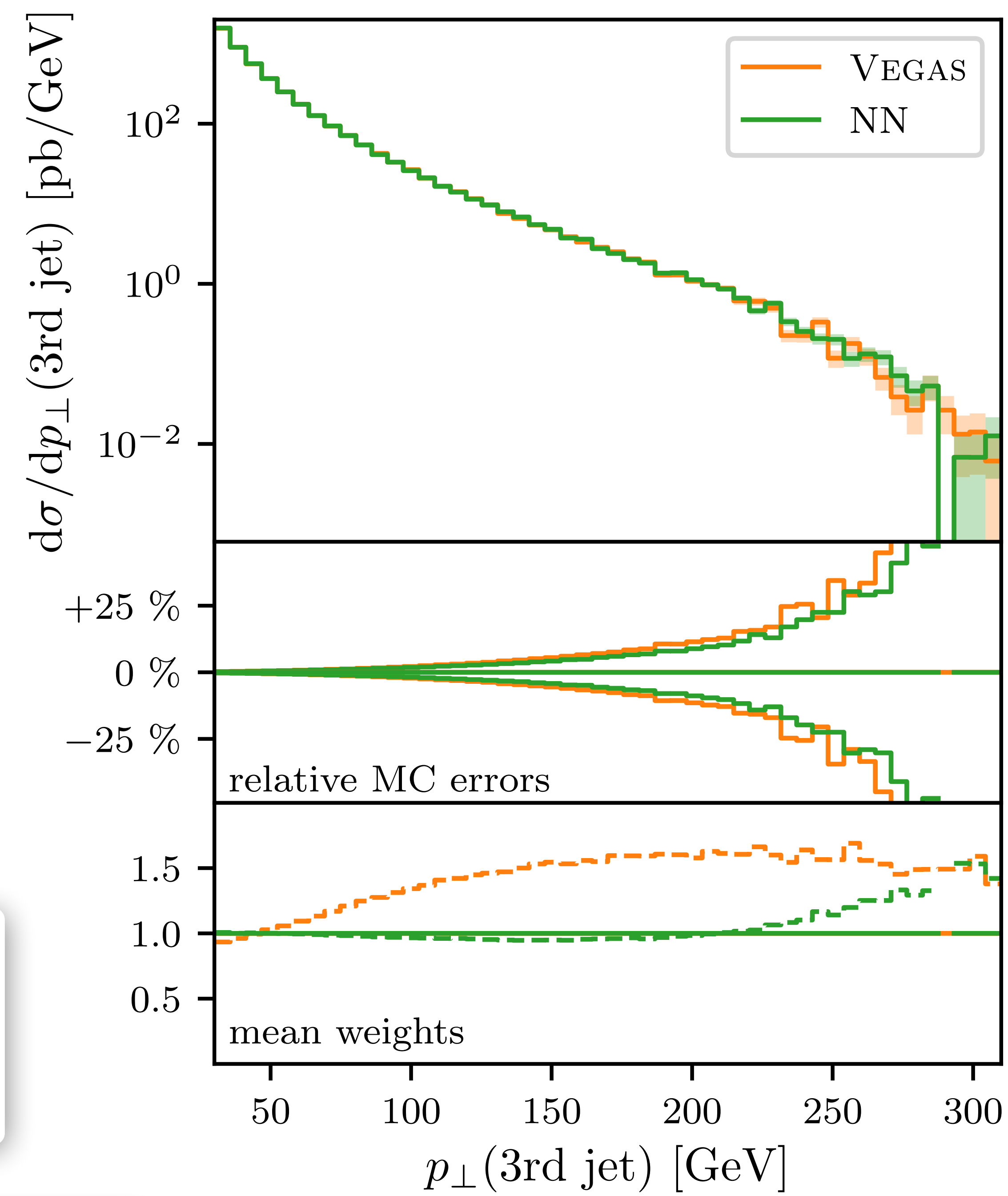
Speeding up slow simulation

Generating Phase space

Estimating SM backgrounds

Measurements and Inference

BSM searches



Bothmann et al., 2001.05478

N.B. being comprehensive with citations would fill up the slide - please see my link to the Living Review at the end for a comprehensive list

Deep Generative Models in HEP

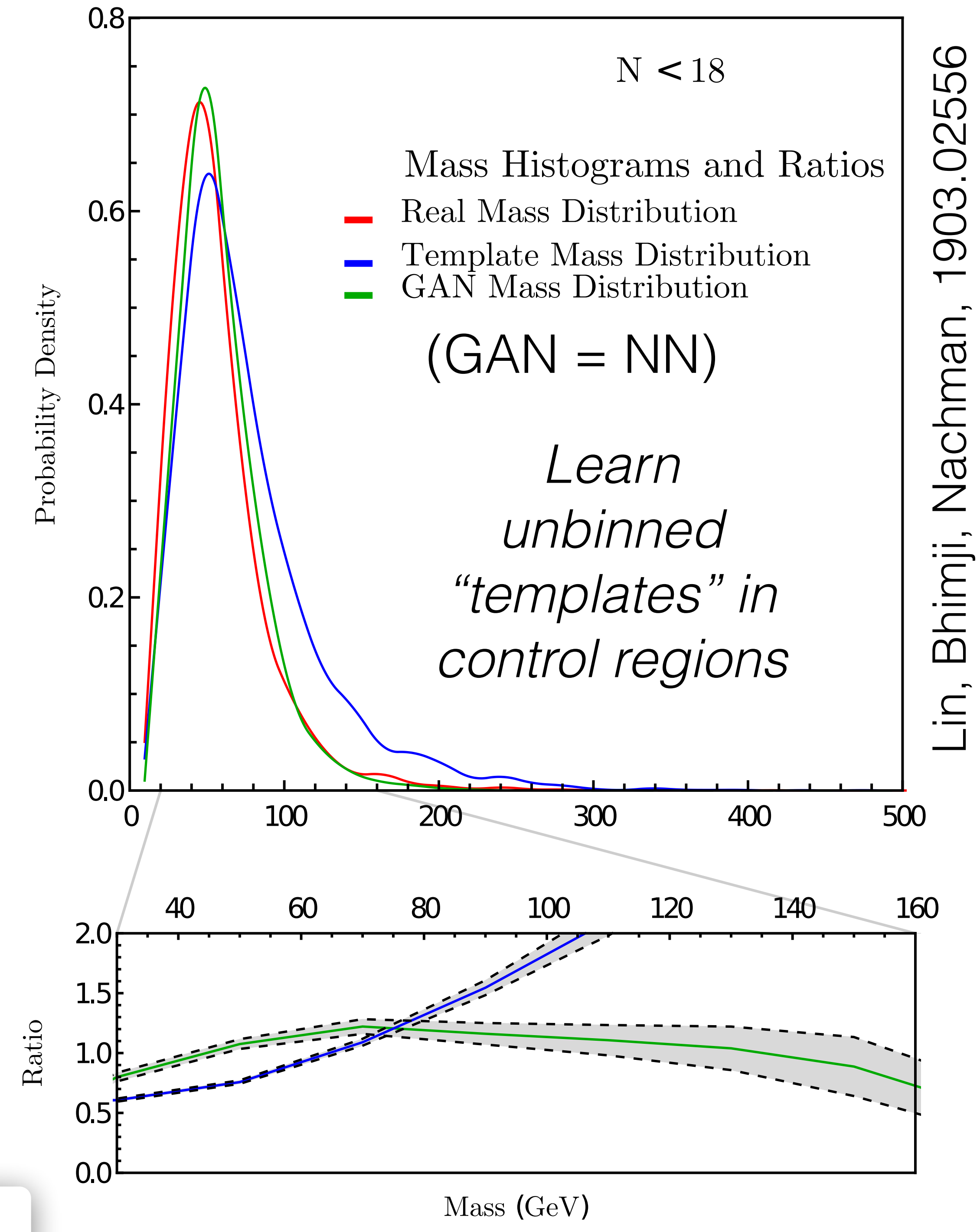
Speeding up slow simulation

Generating Phase space

Estimating SM backgrounds

Measurements and Inference

BSM searches



N.B. being comprehensive with citations would fill up the slide - please see my link to the Living Review at the end for a comprehensive list

Deep Generative Models in HEP

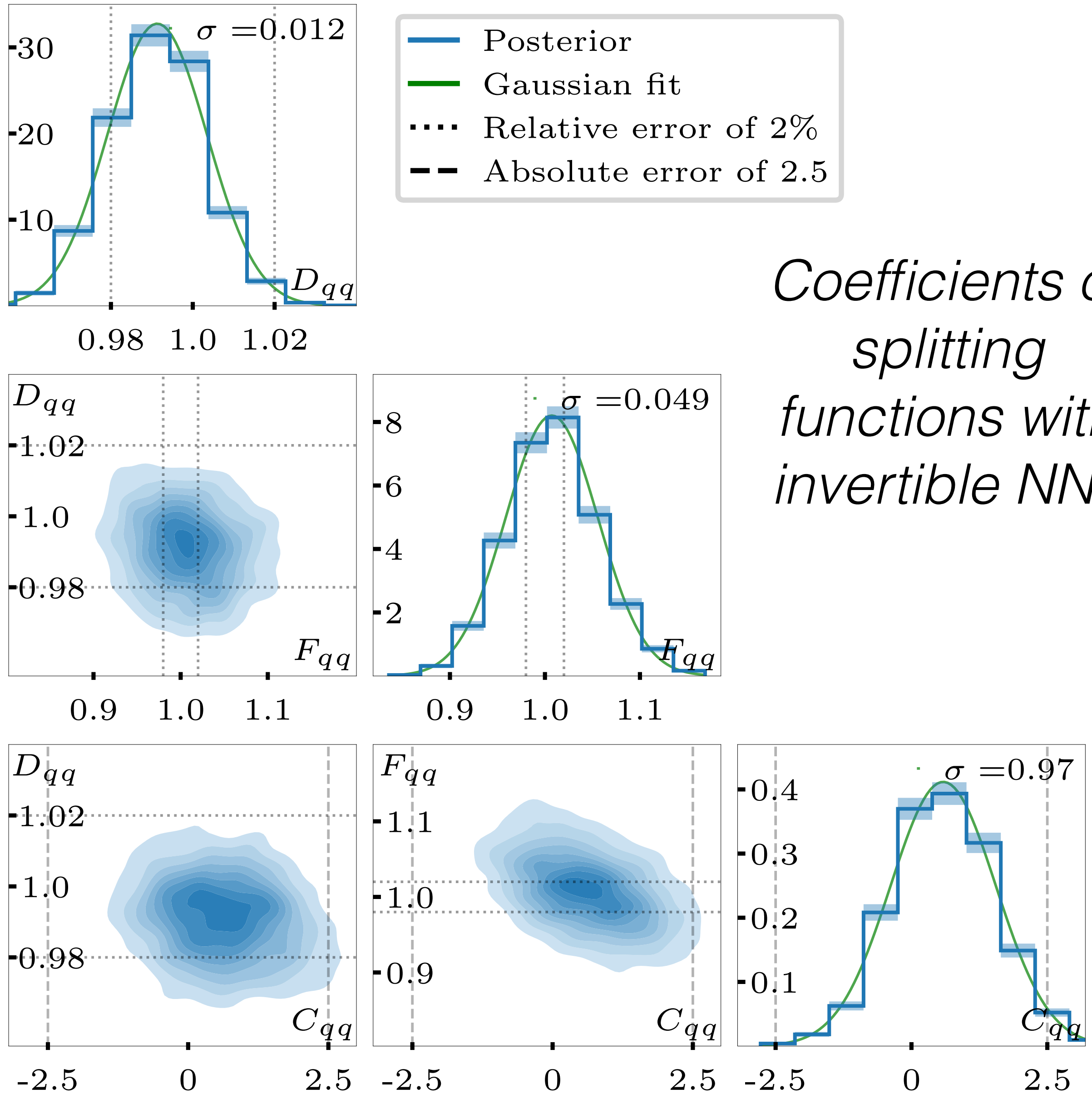
Speeding up slow simulation

Generating Phase space

Estimating SM backgrounds

Measurements and Inference

BSM searches



Coefficients of splitting functions with invertible NNs

Bieringer et al., 2012.09873

N.B. being comprehensive with citations would fill up the slide - please see my link to the Living Review at the end for a comprehensive list

Deep Generative Models in HEP

Speeding up slow simulation

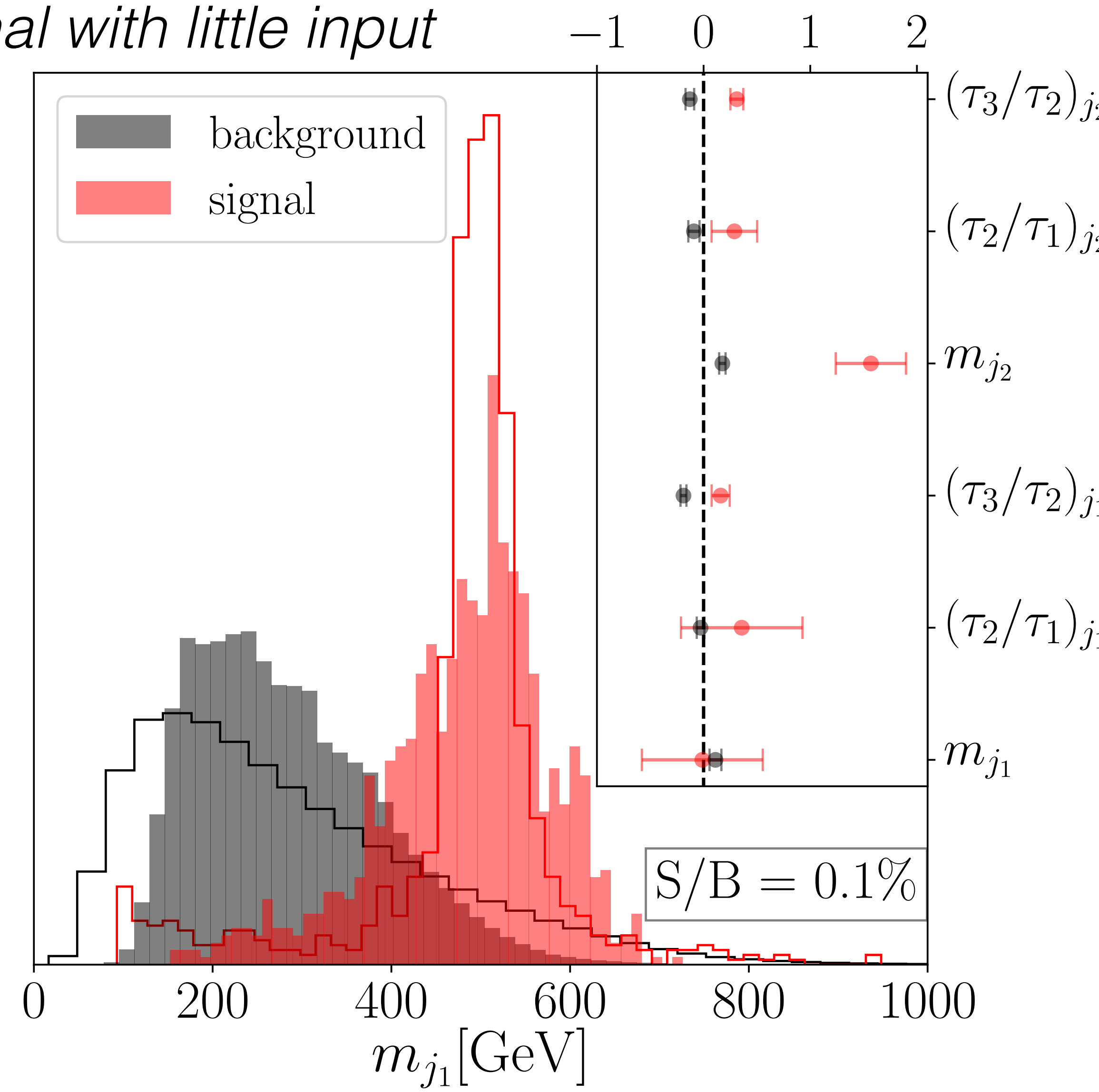
Generating Phase space

Estimating SM backgrounds

Measurements and Inference

BSM searches

Extract features of signal with little input

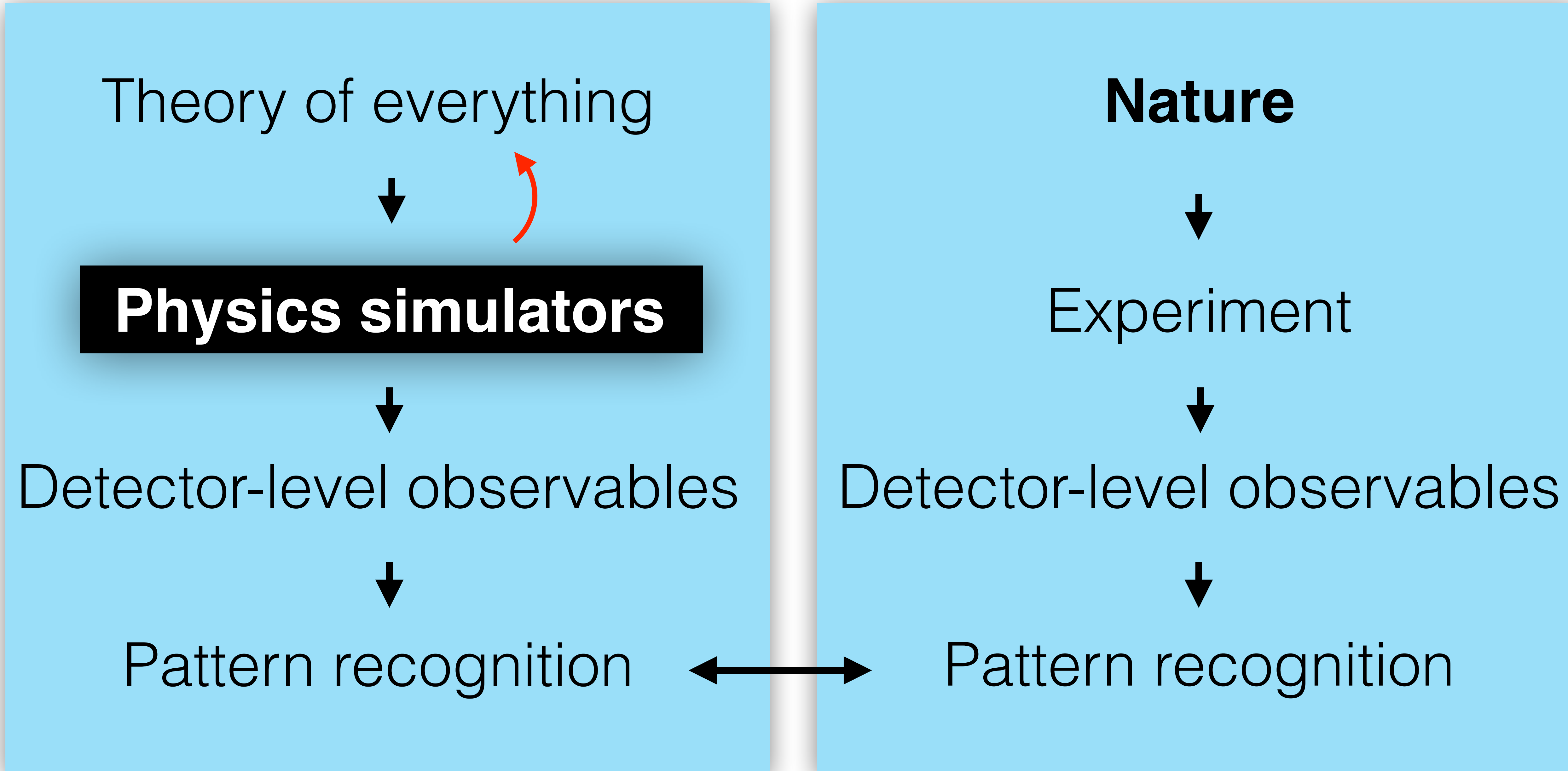


Bortolato et al., 2103.06595

See also the LHC Olympics 2020

N.B. being comprehensive with citations would fill up the slide - please see my link to the Living Review at the end for a comprehensive list

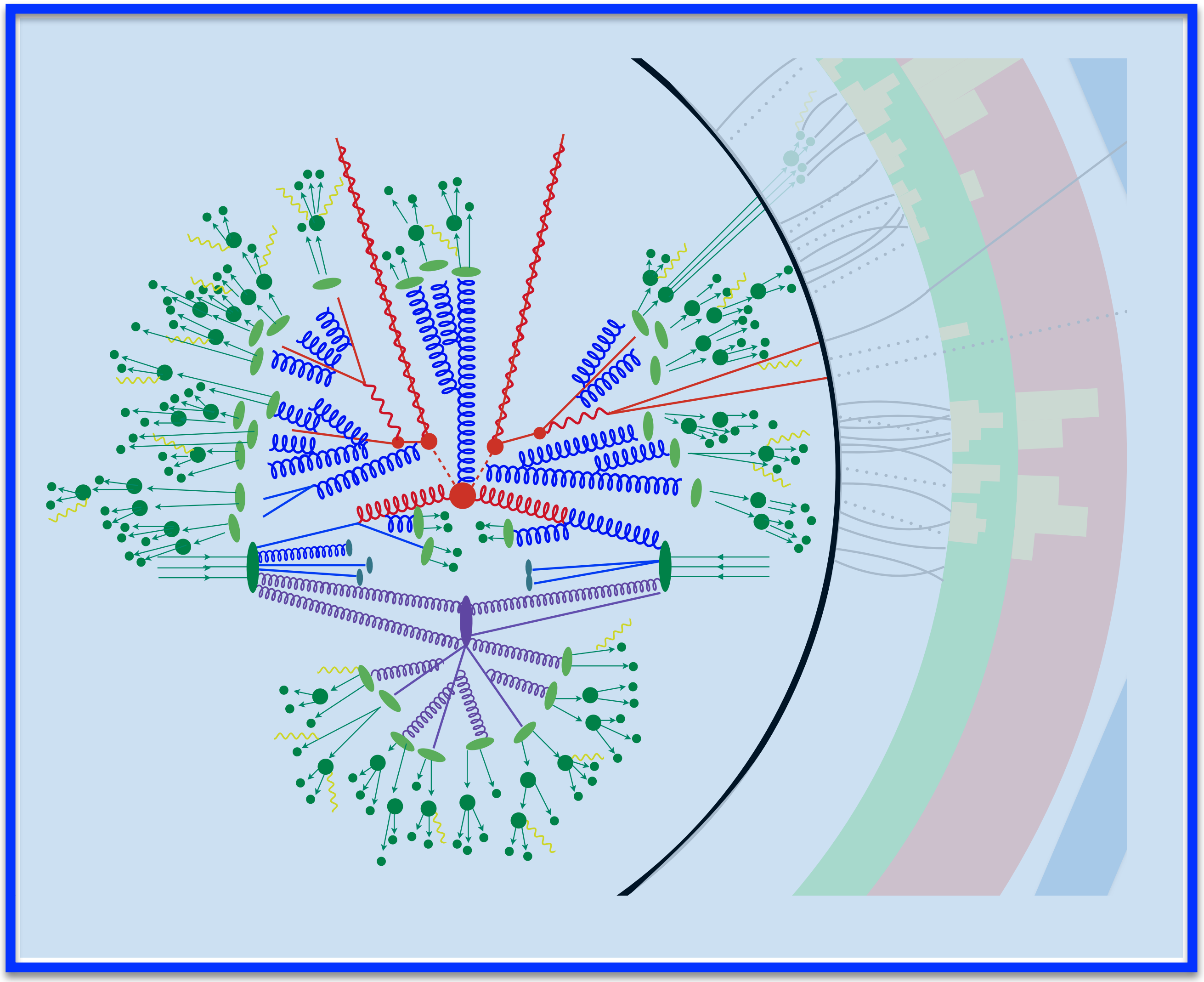
Data analysis in particle physics



Simulators are a unique and powerful aspect of particle physics, but, they do not allow us to go “backwards” !!

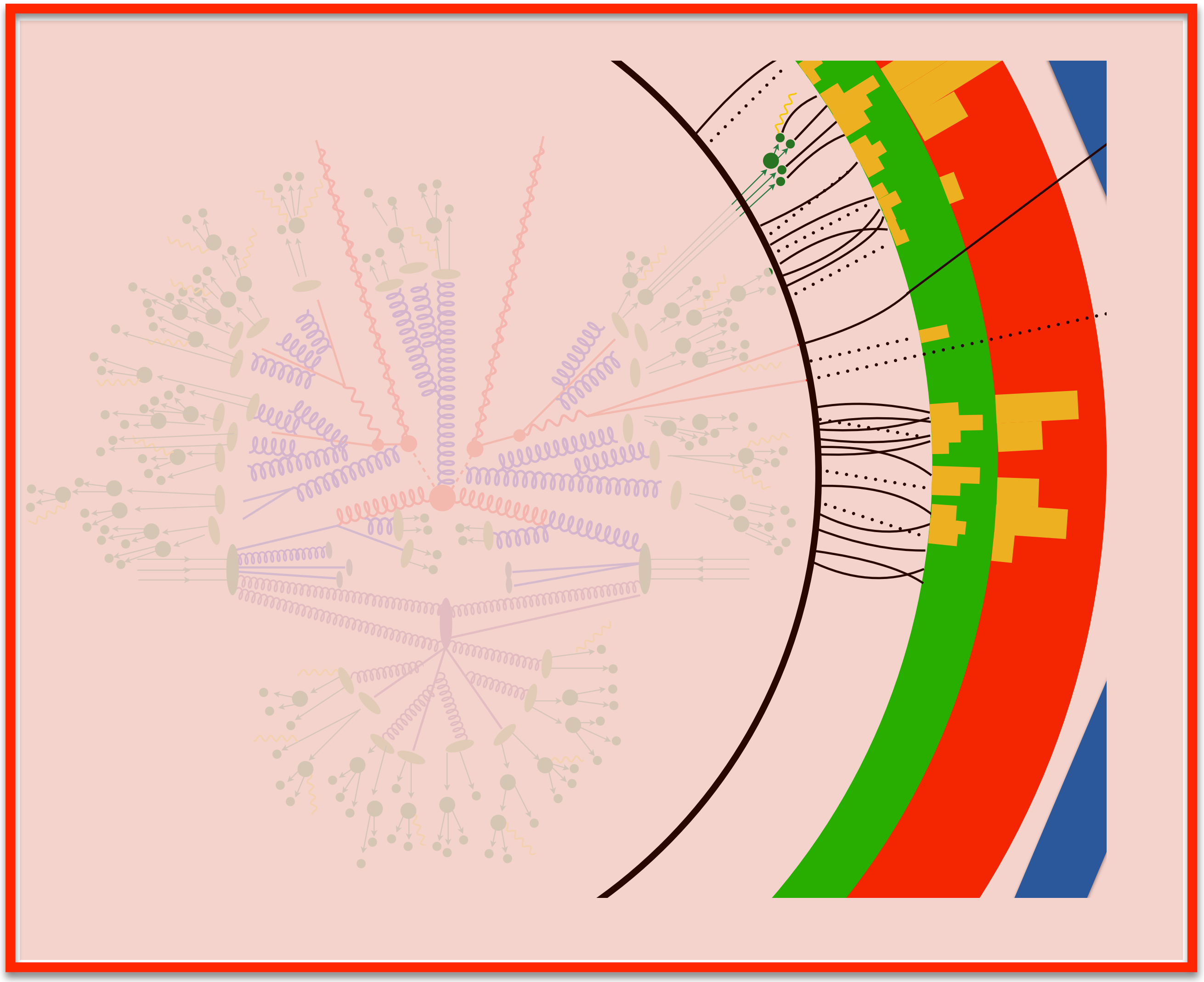
The Inference Challenge

Want this



(or sometimes the parameters of the model that generate this)

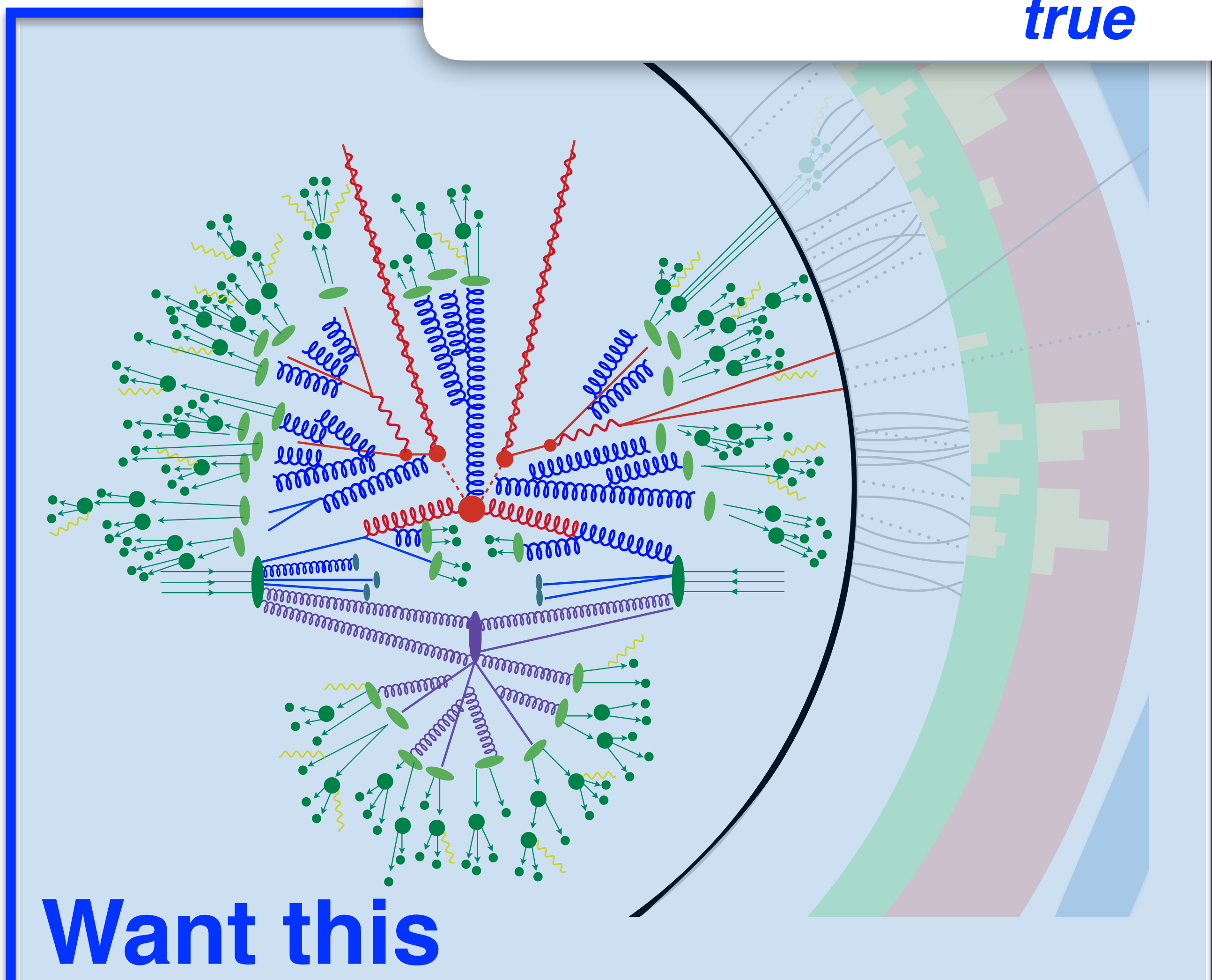
Measure this

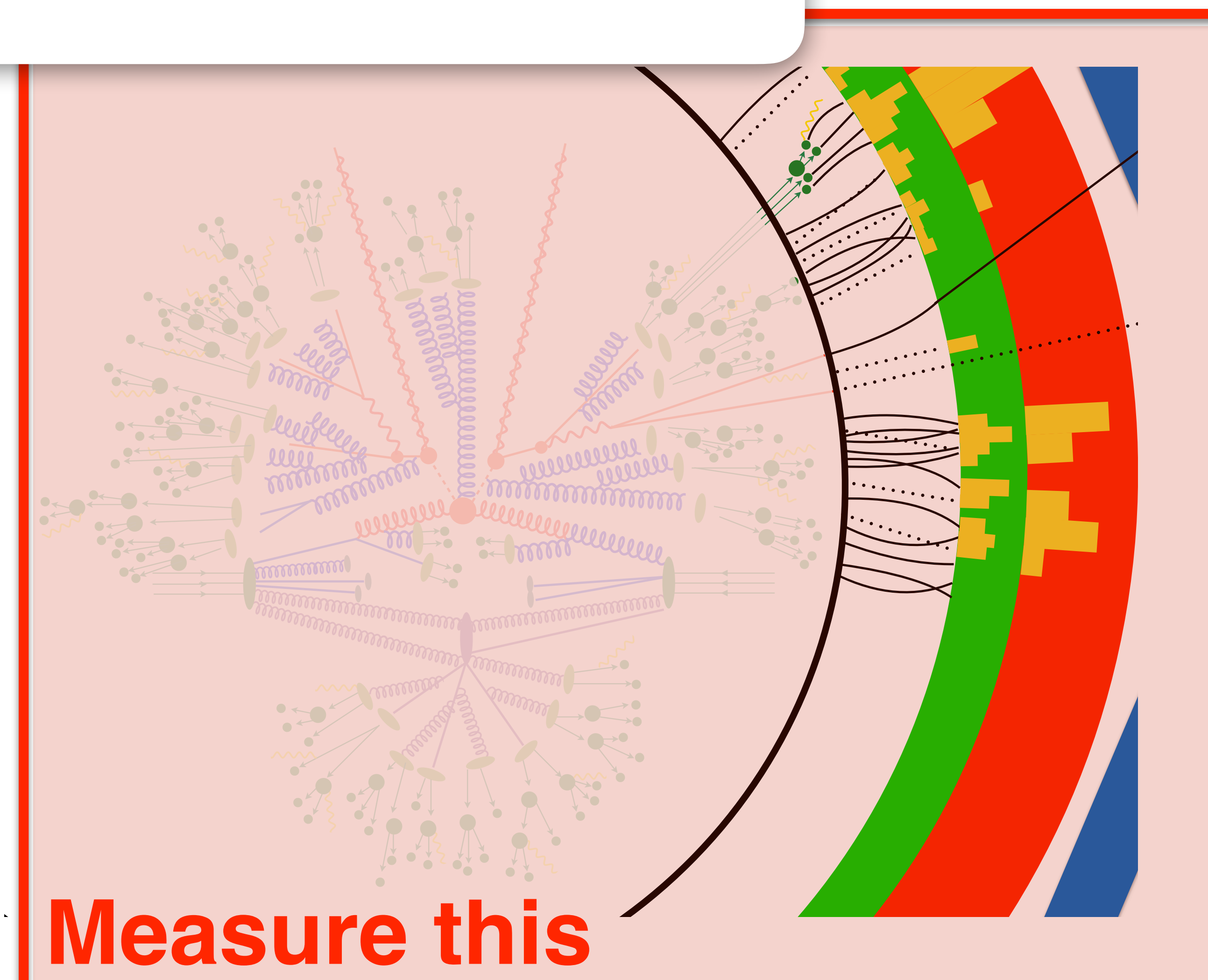


The Inference Challenge

If you know *p(meas. | true)*, could do maximum likelihood, i.e.

$$\text{unfolded} = \underset{\text{true}}{\operatorname{argmax}}\ p(\text{measured} \mid \text{true})$$





(or sometimes the parameters of the model that generate this)

The Inference Challenge

If you know *p*(***meas.*** / ***true***), could do maximum likelihood, i.e.

$$unfolded = \underset{true}{\operatorname{argmax}}\ p(\textbf{measured} \,/\, \textbf{true})$$

Challenge: **measured** is hyperspectral and **true** is hypervariate … *p*(***meas.*** | ***true***) *is* ***intractable*** *!*

The Inference Challenge

If you know *p(**meas.** | **true**)*, could do maximum likelihood, i.e.

$$\textit{unfolded} = \underset{\textit{true}}{\operatorname{argmax}}\; p(\textbf{measured} \mid \textbf{true})$$

Challenge: **measured** is hyperspectral and **true** is hypervariate … *p(**meas.** | **true**) is **intractable** !*

However: we have **simulators** that we can use to sample from *p(**meas.** | **true**)*

→ **Simulation-based (likelihood-free) inference !**

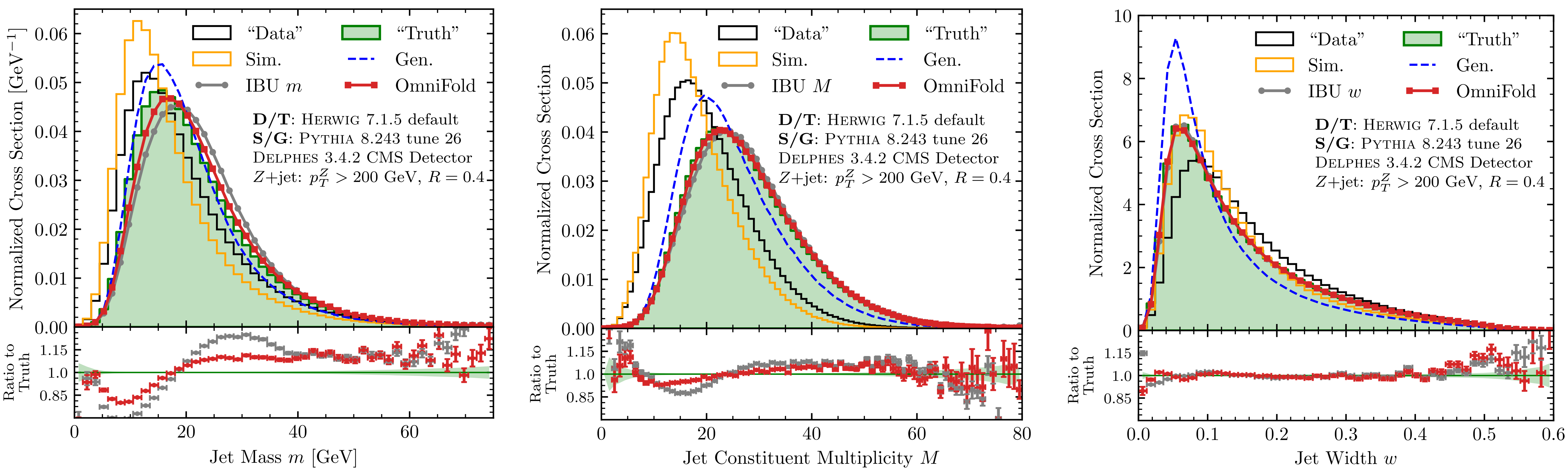
…an area of machine learning were particle physics is making key contributions!

See Cranmer, Brehmer, Louppe, 1911.01429 for a recent overview

Example: Unfolding

What if we could unfold all particles simultaneously?
We could then compute observables (and their bins)
AFTER doing the measurement (!)

…stick around for the second part of this
session for more discussions on this point



Andreassen et al., 1911.09107

Data analysis in fundamental physics

Theory of everything

↓

Physics simulators

↓

Detector-level observables

↓

Pattern recognition

Nature

↓

Experiment

↓

Detector-level observables

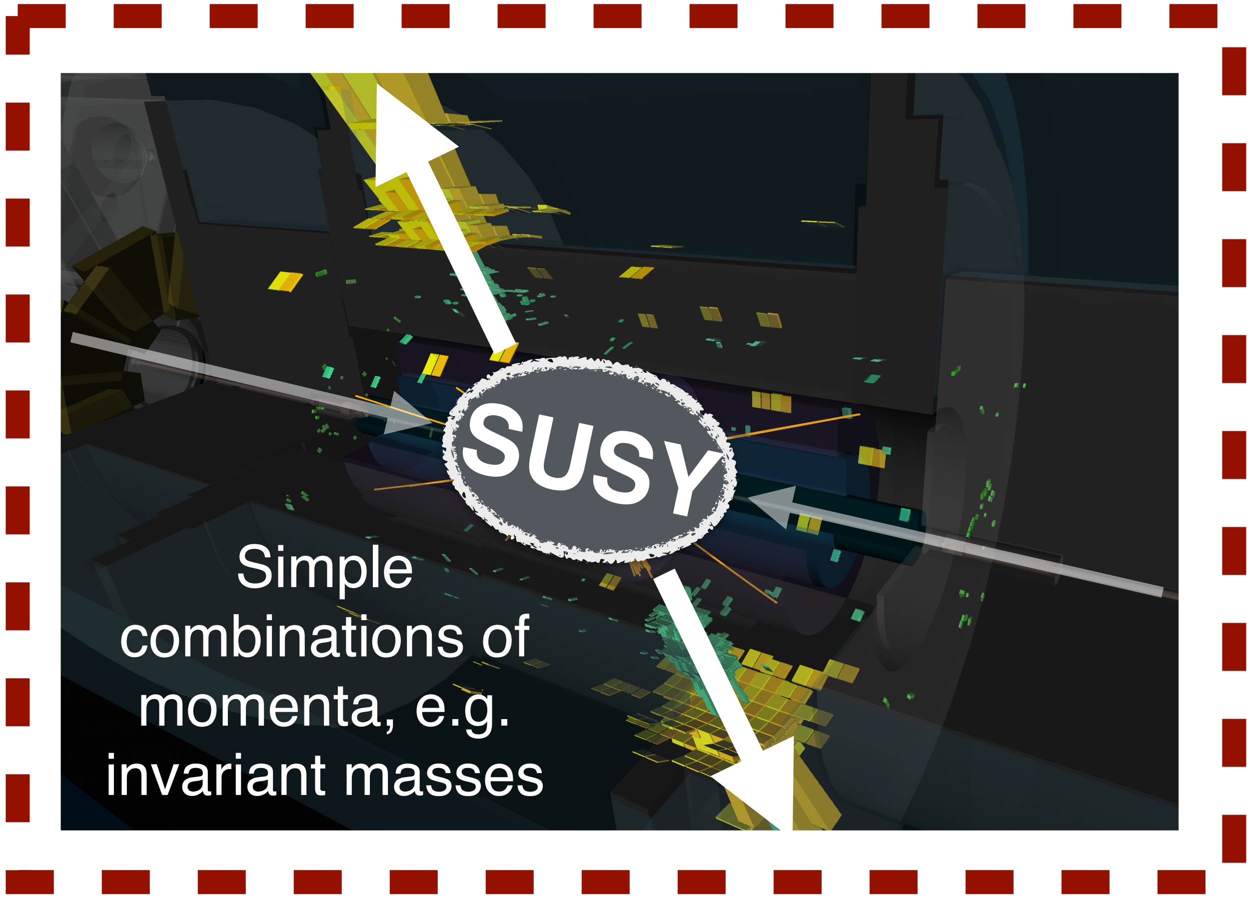
↓

Pattern recognition

Anomaly detection

Current Search Paradigm

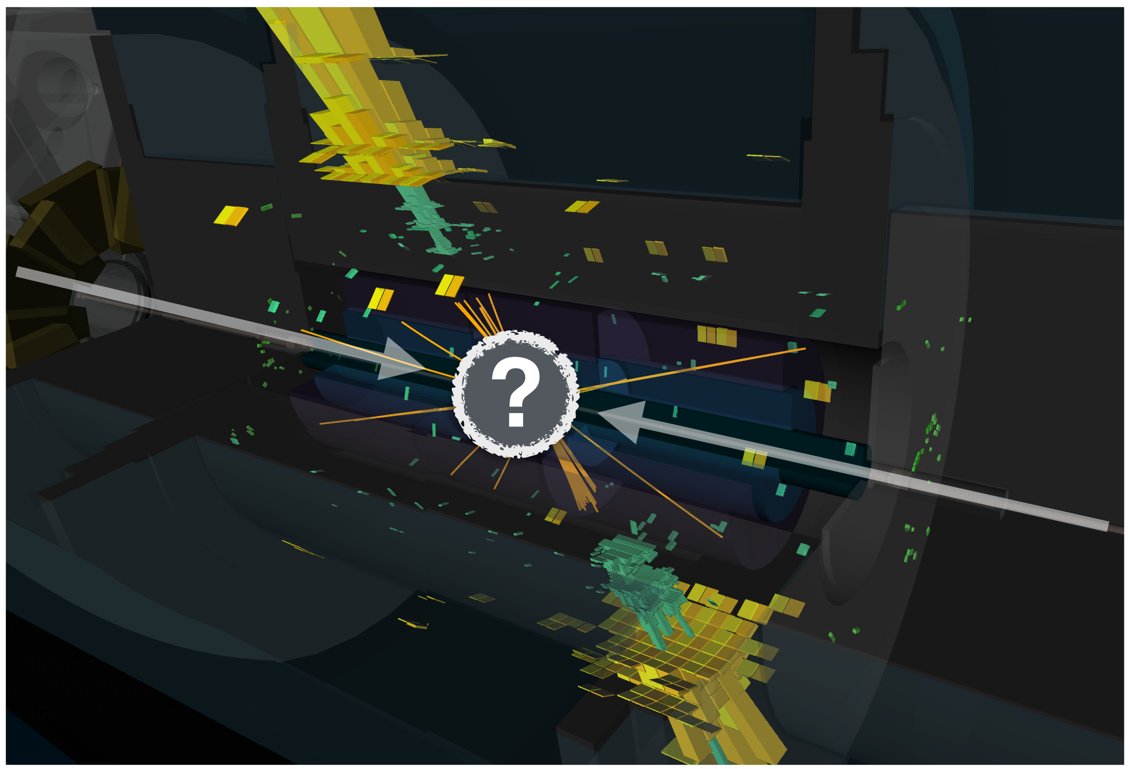
SUSY = Supersymmetry



(well-motivated) theory-biased
& low-dimensional observables

Current Search Paradigm

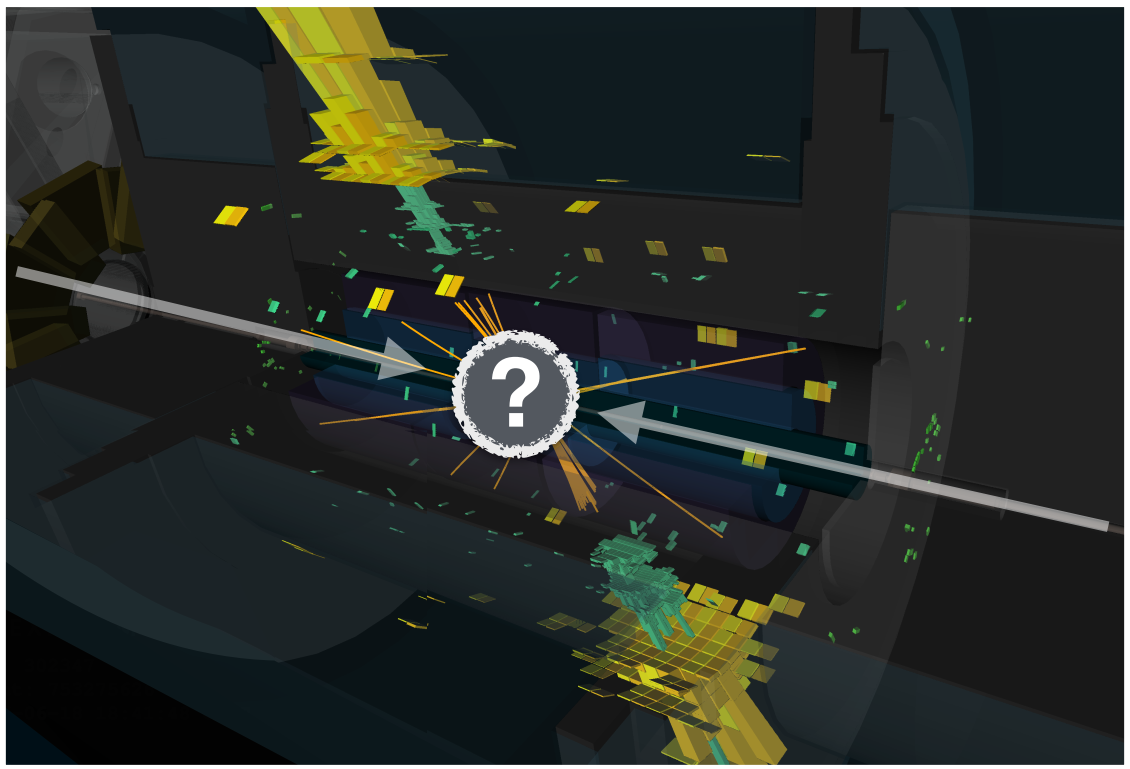
SUSY = Supersymmetry



Can we relax model assumptions and explore high-dimensional feature spaces?

(well-motivated) theory-biased & low-dimensional observables

Current Search Paradigm



What if we are not looking in the right place for the new phenomena?!

Can we relax model assumptions and explore high-dimensional feature spaces?

New search ideas

Supervision refers to the type of label information provided to the ML during training.

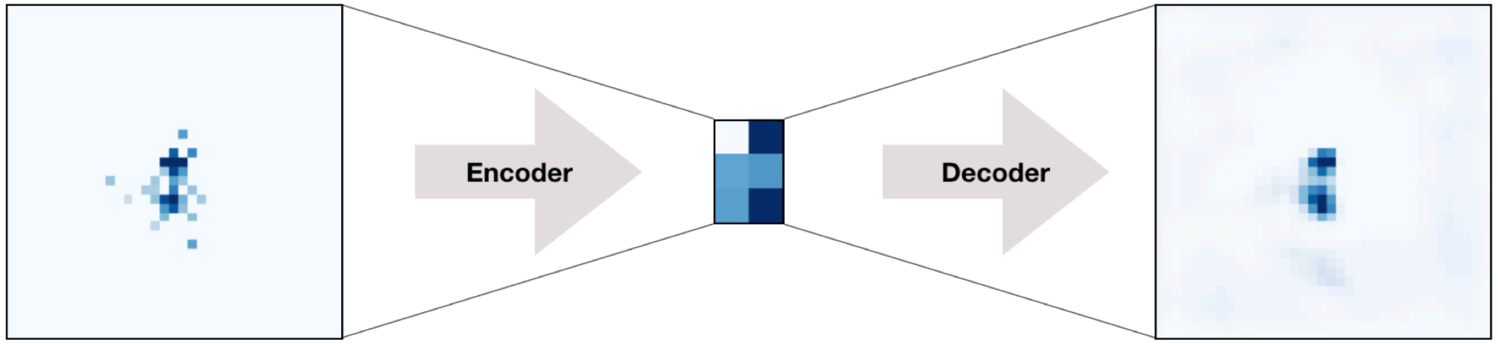
Unsupervised = no labels
Weakly-supervised = noisy labels
Semi-supervised = partial labels
Supervised = full label information

These categories are not exact and the boundaries are not rigid!

Solutions: Unsupervised

Unsupervised = no labels

Typically, the goal of these methods is to look for events with low *p(background)*



One strategy (autoencoders) is to try to compress events and then uncompress them. When x = uncompres(compress(x)), then x probably has low p(x).

Farina, Nakai, Shih, 1808.08992; Heimel, Kasieczka, Plehn, Thompson, 1808.08979; + many more including the recent LHC Olympics (Kasieczka et al., 2101.08320) and Dark Machines (Aarrestad et al., 2105.14027) reports

Solutions: Weakly-supervised

Weakly-supervised = noisy labels

Typically, the goal of these methods is to look for events with high *p(possibly signal-enriched)/p(possibly signal-depleted)*

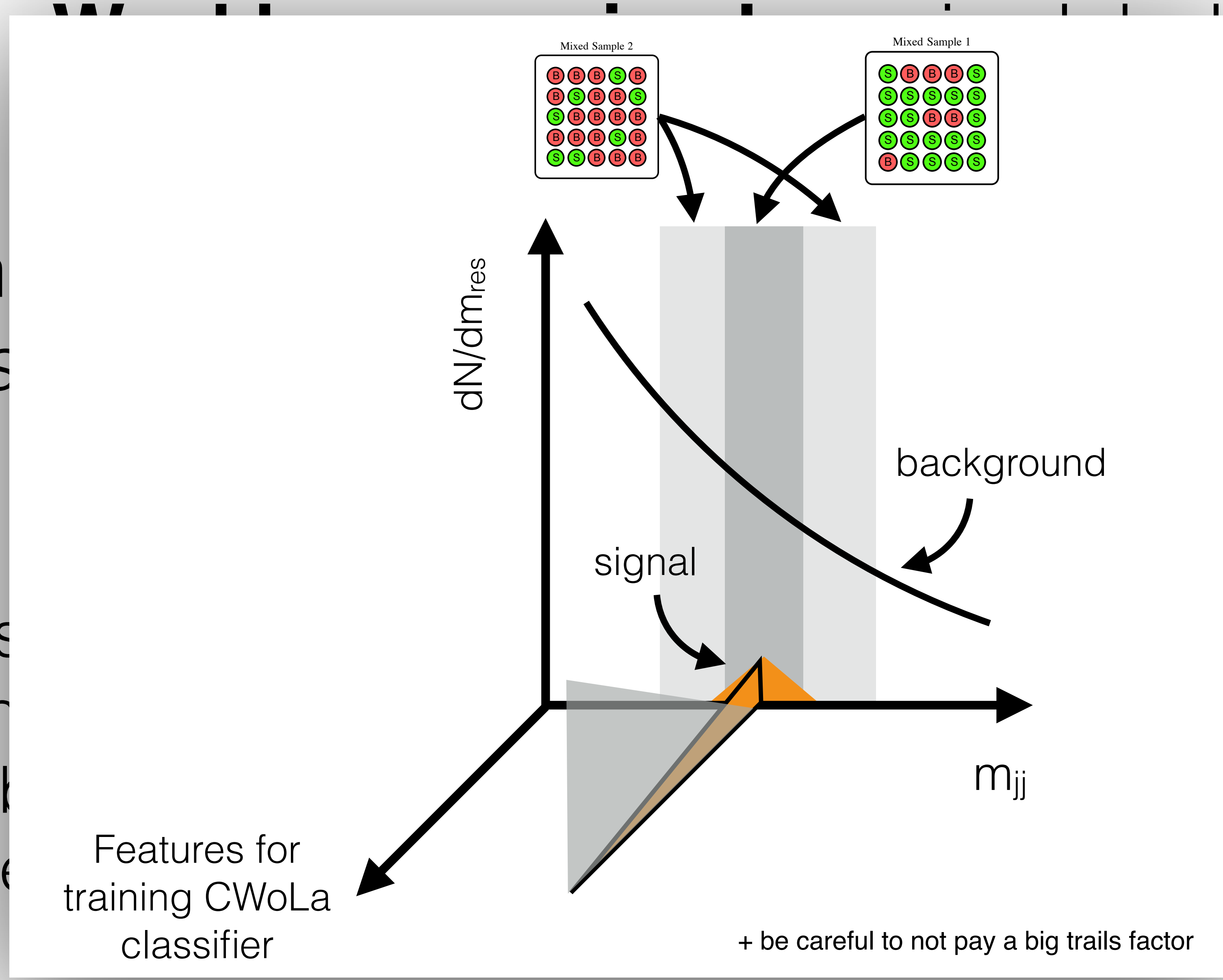
e.g. Classification Without Labels (CWoLa), events in a signal region are labeled “signal” and events in a sideband are labeled “background”. These labels are “noisy” but a classifier trained with them can detect the presence of a signal.

Metodiev, Nachman, Thaler, 1708.02949; Collins, Howe, Nachman, 1805.02664
+ many more including the recent LHC Olympics (Kasieczka et al., 2101.08320)

Solutions: Weakly-supervised

Typically, th … or events with high *p(poss … nal-depleted)*

e.g. Clas … in a signal region … band are labeled “b … a classifier traine … signal.

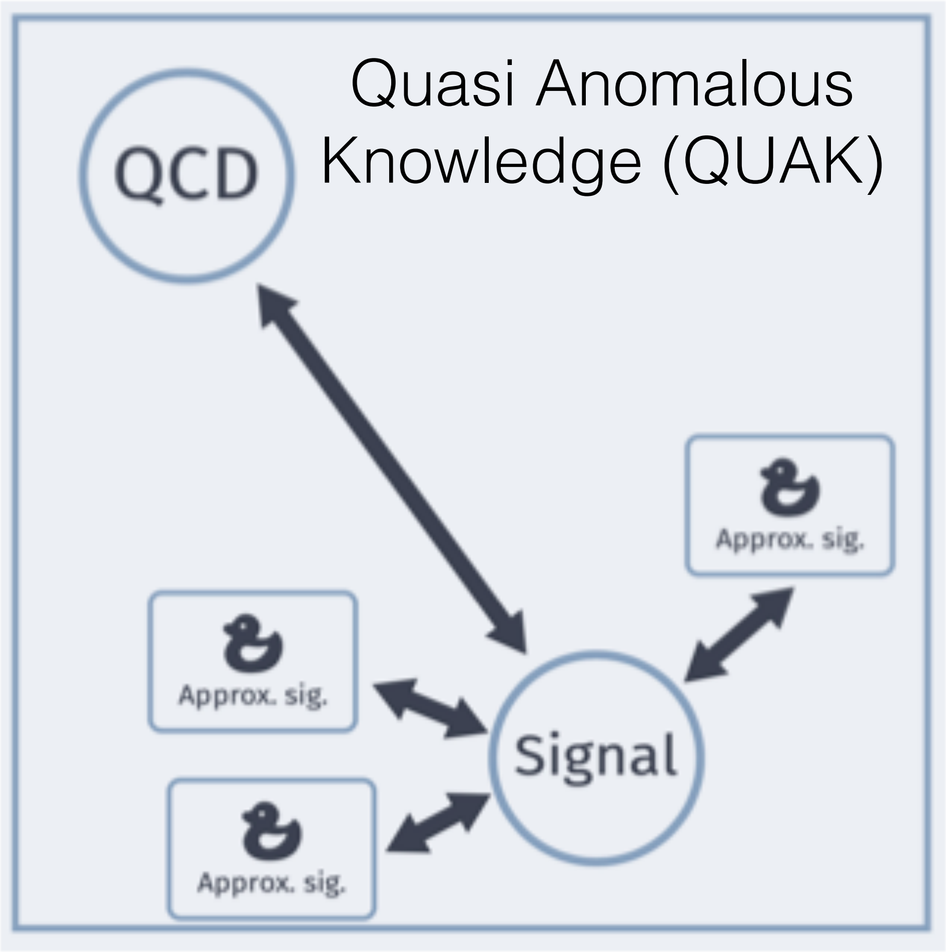


Metodiev, Nachman, Thaler, 1708.02949; Collins, Howe, Nachman, 1805.02664
+ many more including the recent LHC Olympics (Kasieczka et al., 2101.08320)

Solutions: Semi-supervised

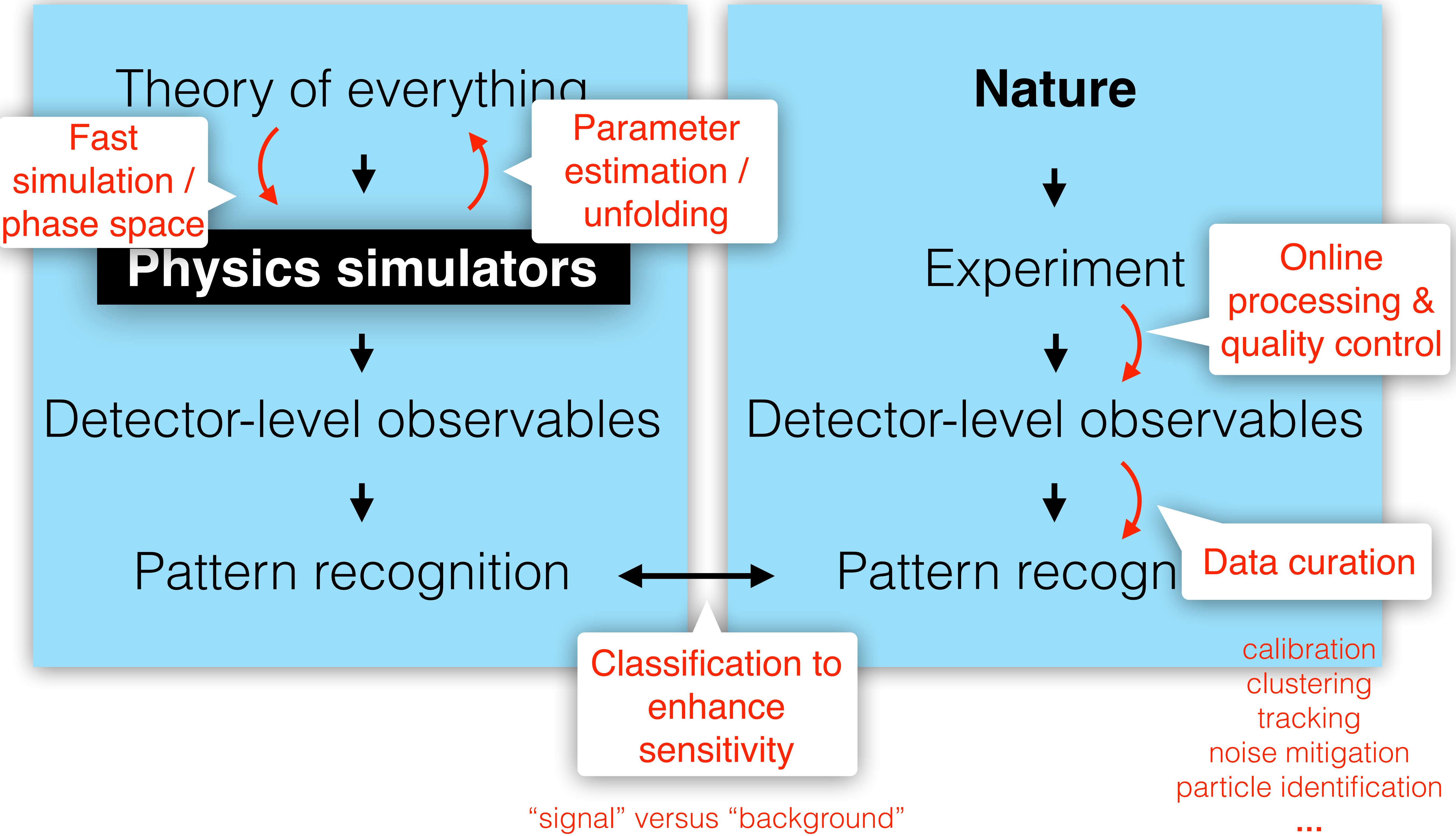
Semi-supervised = partial labels

Typically, these methods use some signal simulations to build signal sensitivity



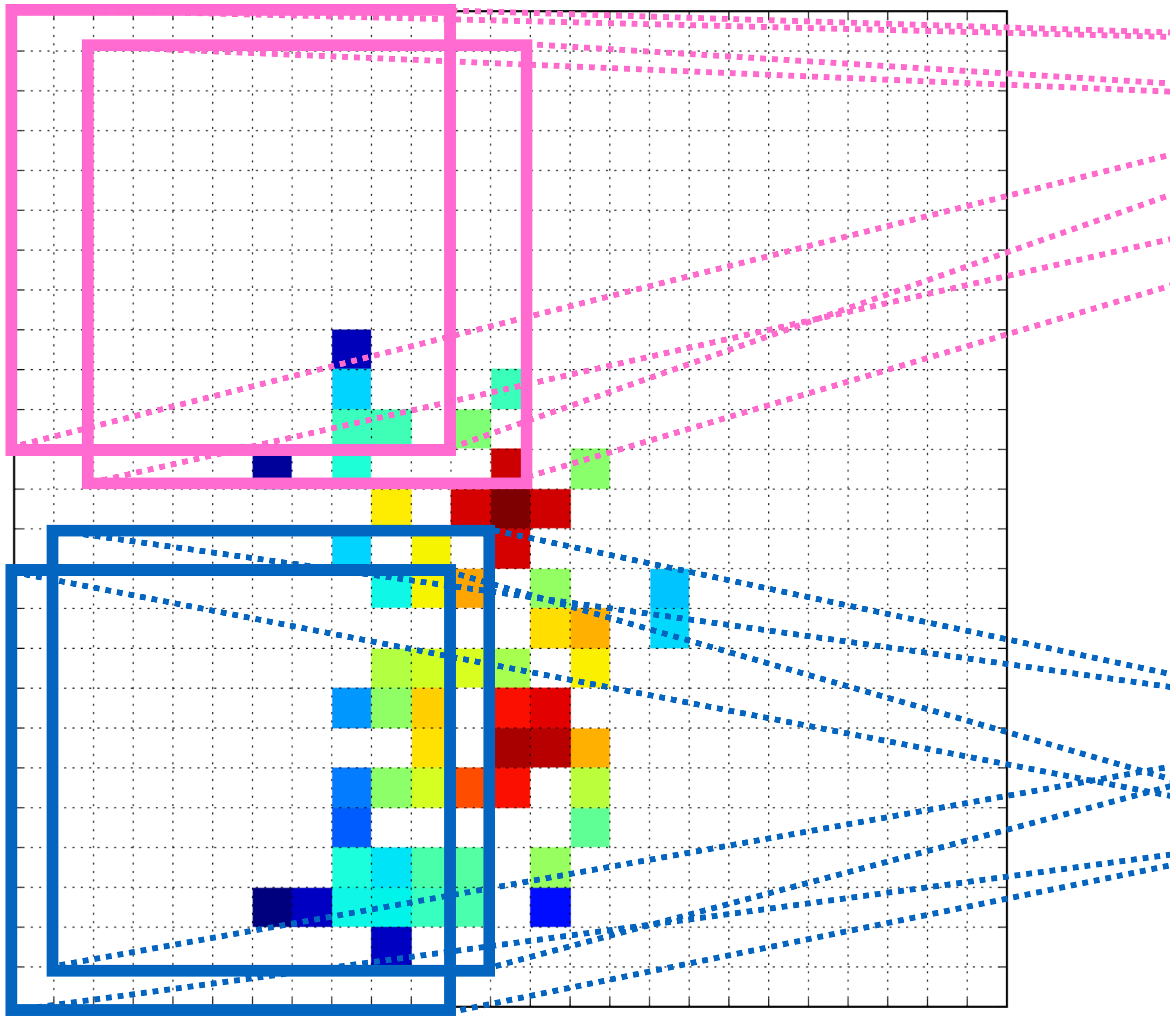
S. Park, D. Rankin, S.-M. Udrescu, M. Yunus, P. Harris, 2011.03550 + many more including the recent LHC Olympics (Kasieczka et al., 2101.08320)

Overview: Particle physics and ML



Conclusions and outlook

Deep learning has a great potential to **enhance**, **accelerate**, and **empower** discoveries in particle physics.



We have some unique challenges that require dedicated solutions.

With these new tools, we will be able to fully exploit the data in their natural high dimensionality enhancing the potential for **discovery**!

Conclusions and outlook

I only covered a small fraction of the exciting work from the hep-ph and hep-ex (friends in stat.ml++) communities!

My apologies to everyone who's awesome plot(s) I could not show…

Upcoming ML Workshops

Where can I learn more?

HEPML-LivingReview

A Living Review of Machine Learning for Particle Physics

Modern machine learning techniques, including deep learning, is rapidly being applied, adapted, and developed for high energy physics. The goal of this document is to provide a nearly comprehensive list of citations for those developing and applying these approaches to experimental, phenomenological, or theoretical analyses. As a living document, it will be updated as often as possible to incorporate the latest developments. A list of proper (unchanging) reviews can be found within. Papers are grouped into a small set of topics to be as useful as possible. Suggestions are most welcome.

download review

The purpose of this note is to collect references for modern machine learning as applied to particle physics. A minimal number of categories is chosen in order to be as useful as possible. Note that papers may be referenced in more than one category. The fact that a paper is listed in this document does not endorse or validate its content - that is for the community (and for peer-review) to decide. Furthermore, the classification here is a best attempt and may have flaws - please let us know if (a) we have missed a paper you think should be included, (b) a paper has been misclassified, or (c) a citation for a paper is not correct or if the journal information is now available. In order to be as useful as possible, this document will continue to evolve so please check back before you write your next paper. If you find this review helpful, please consider citing it using \cite{hepmllivingreview} in HEPML.bib.

https://iml-wg.github.io/HEPML-LivingReview/

https://github.com/iml-wg/HEPML-LivingReview

https://arxiv.org/abs/2102.02770

Questions?

Unbinned differential cross section measurements: *towards a common format*

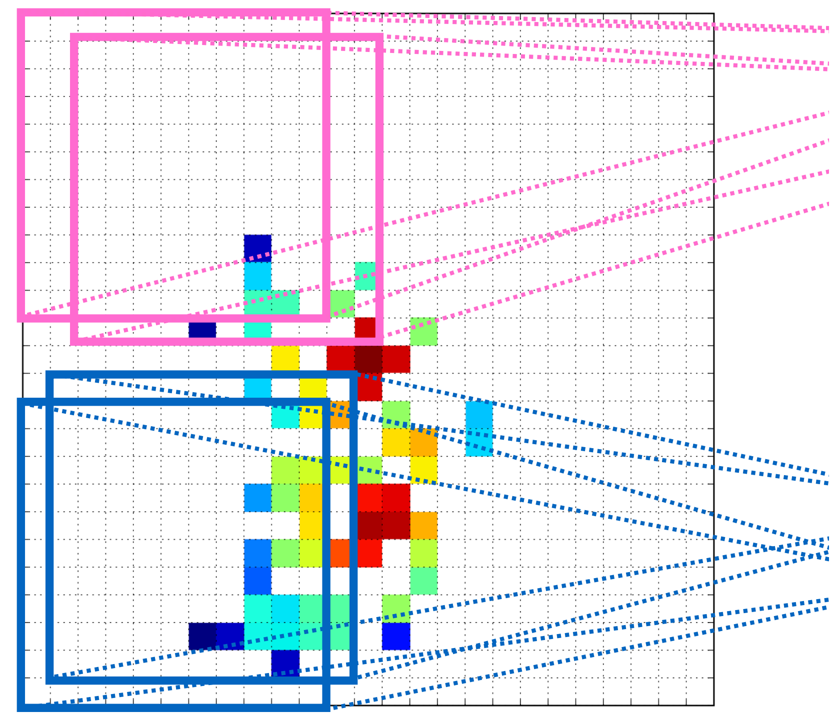
Benjamin Nachman

Lawrence Berkeley National Laboratory

bpnachman.com

bpnachman@lbl.gov

@bpnachman bnachman



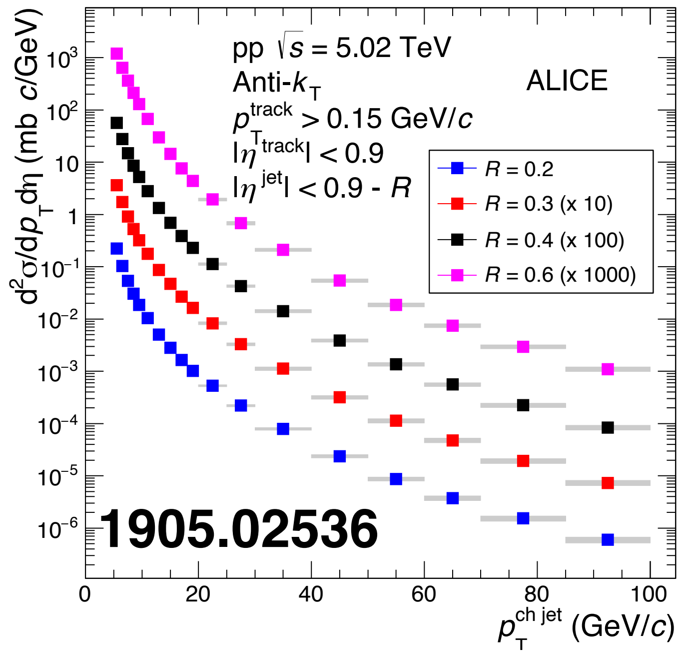
BERKELEY LAB

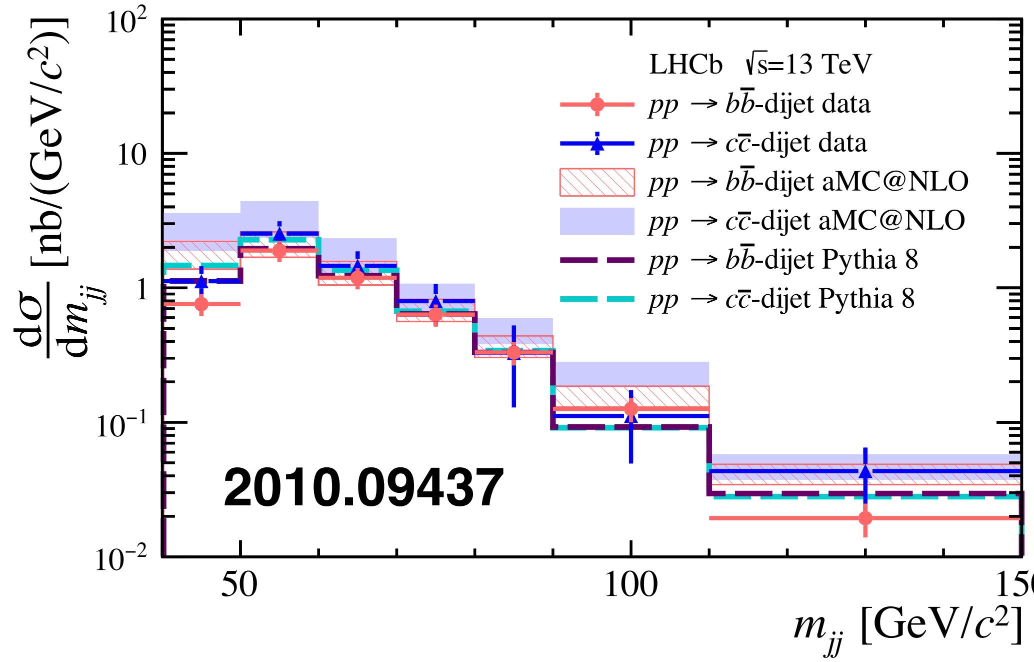
BIDS
BERKELEY INSTITUTE FOR DATA SCIENCE

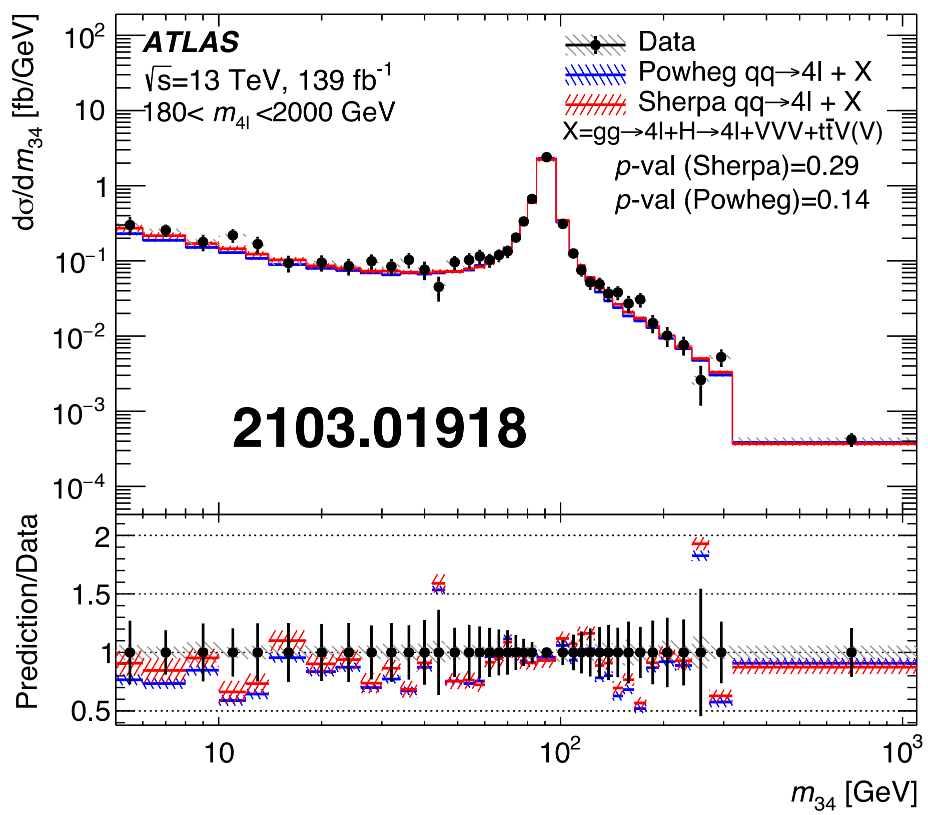
Virtual Houches

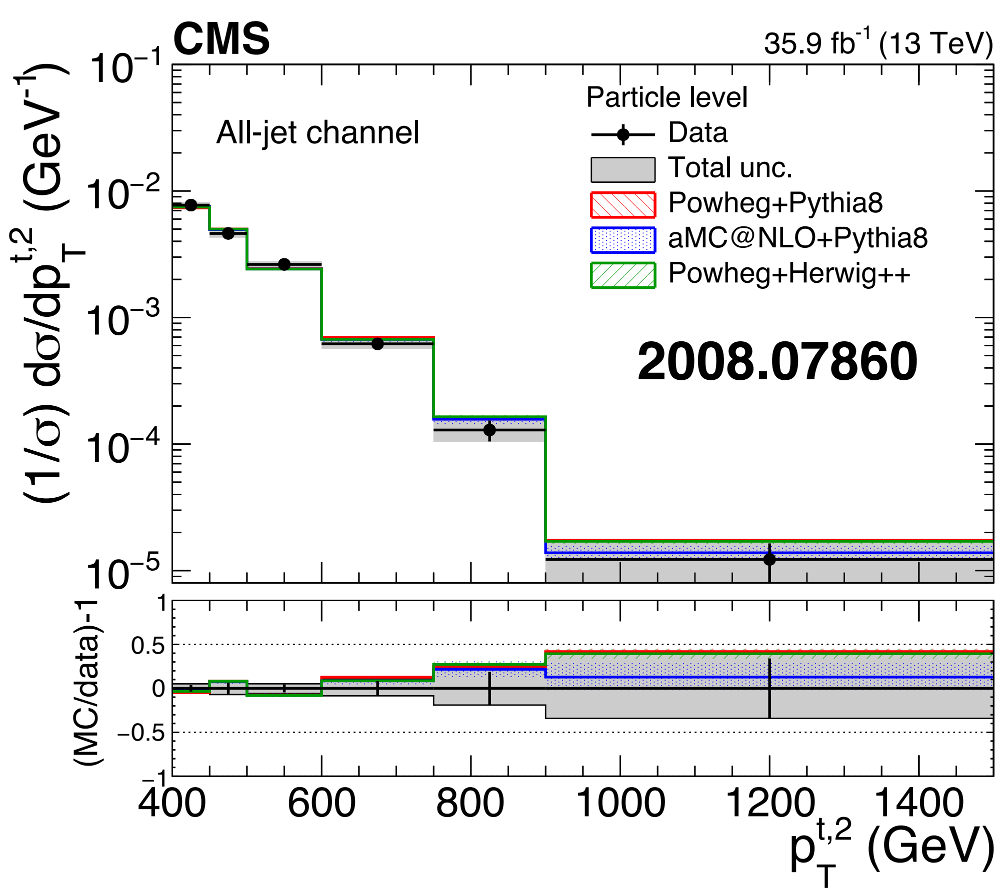
June 17, 2021

Differential Cross Section Measurements

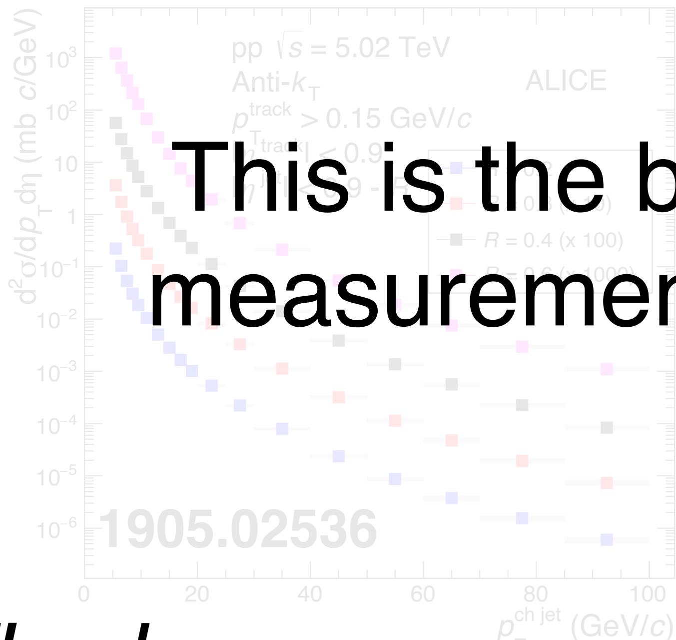


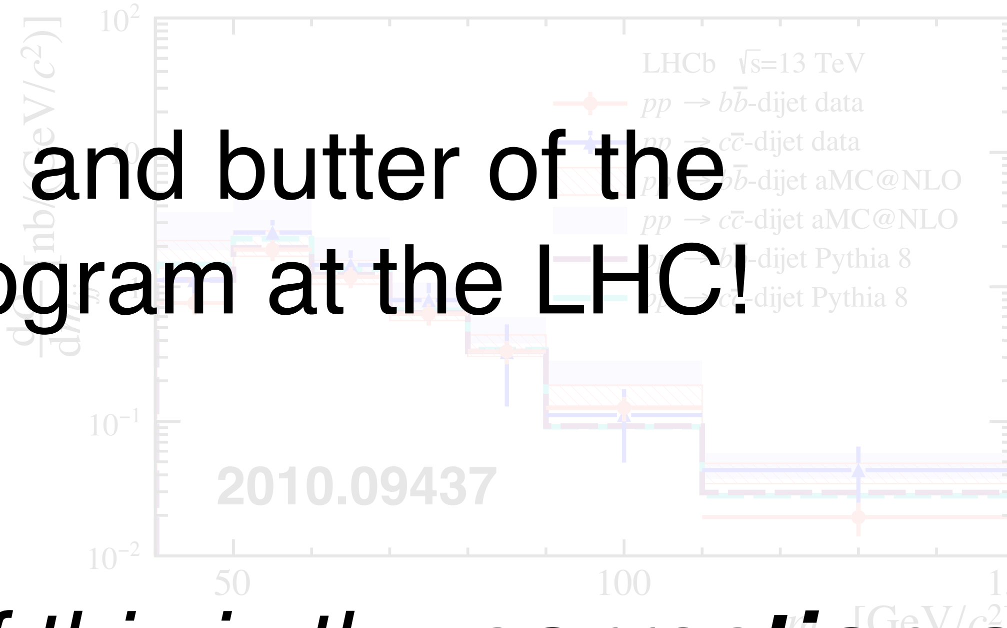






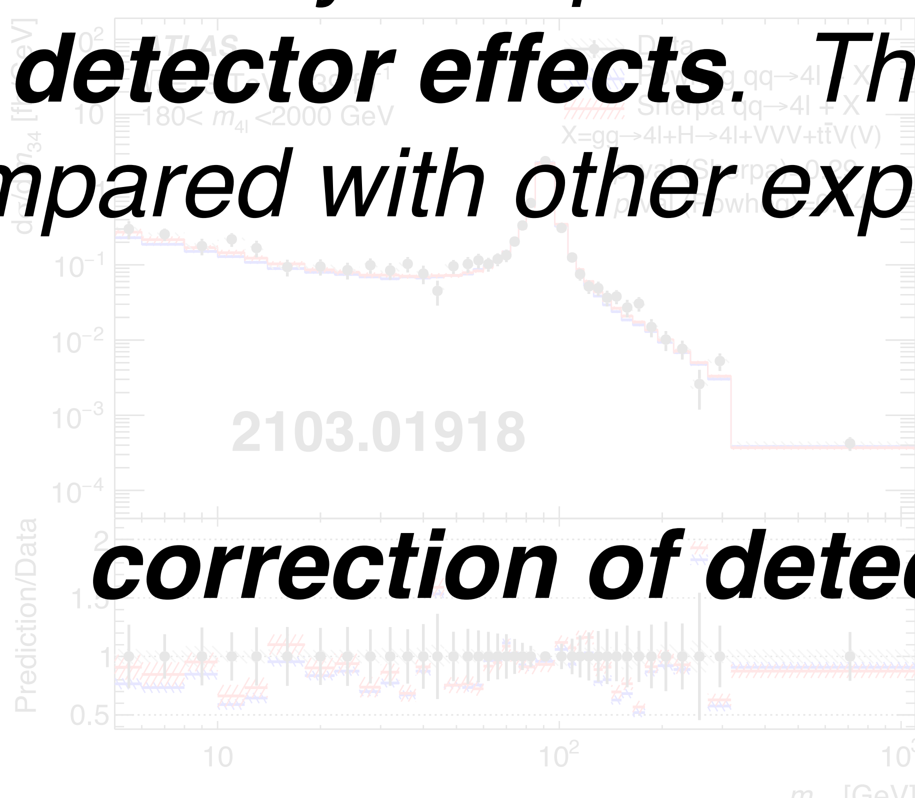
Differential Cross Section Measurements

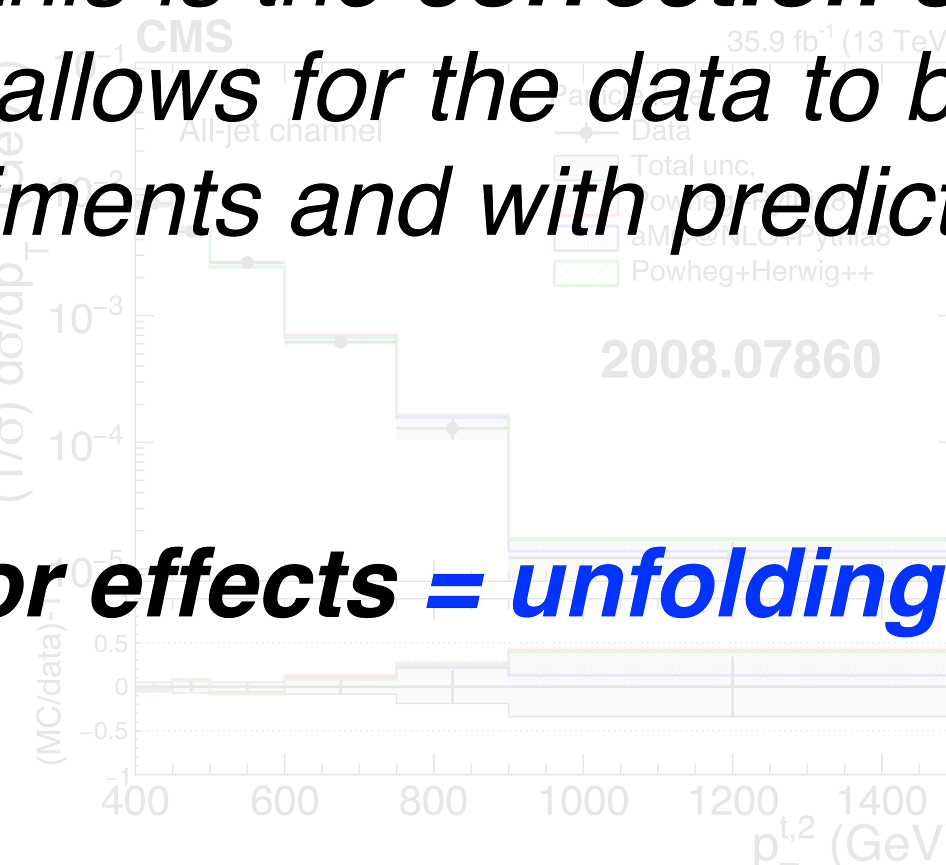




This is the bread and butter of the measurement program at the LHC!

The key component of this is the ***correction of detector effects****. This allows for the data to be compared with other experiments and with predictions.*

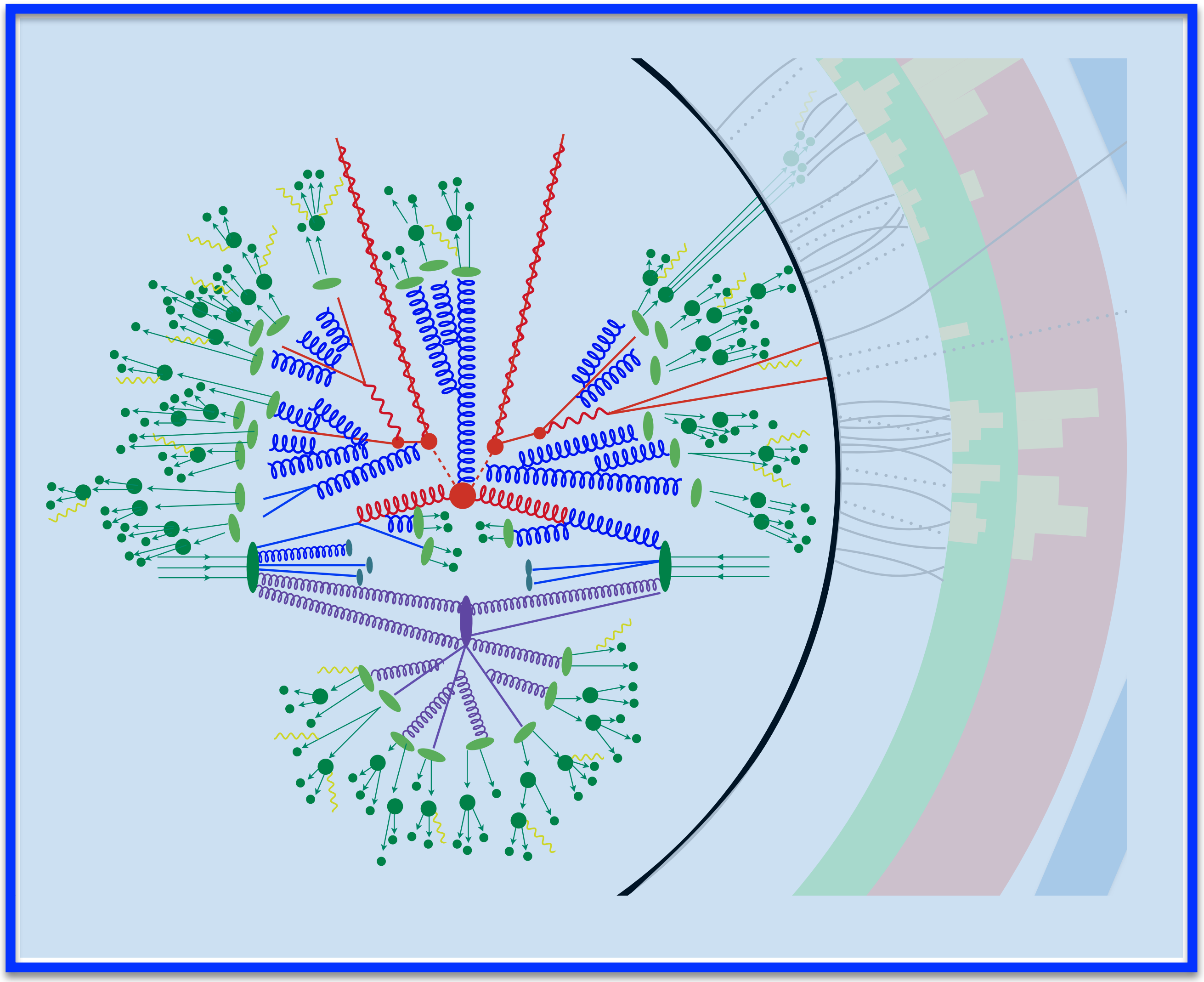




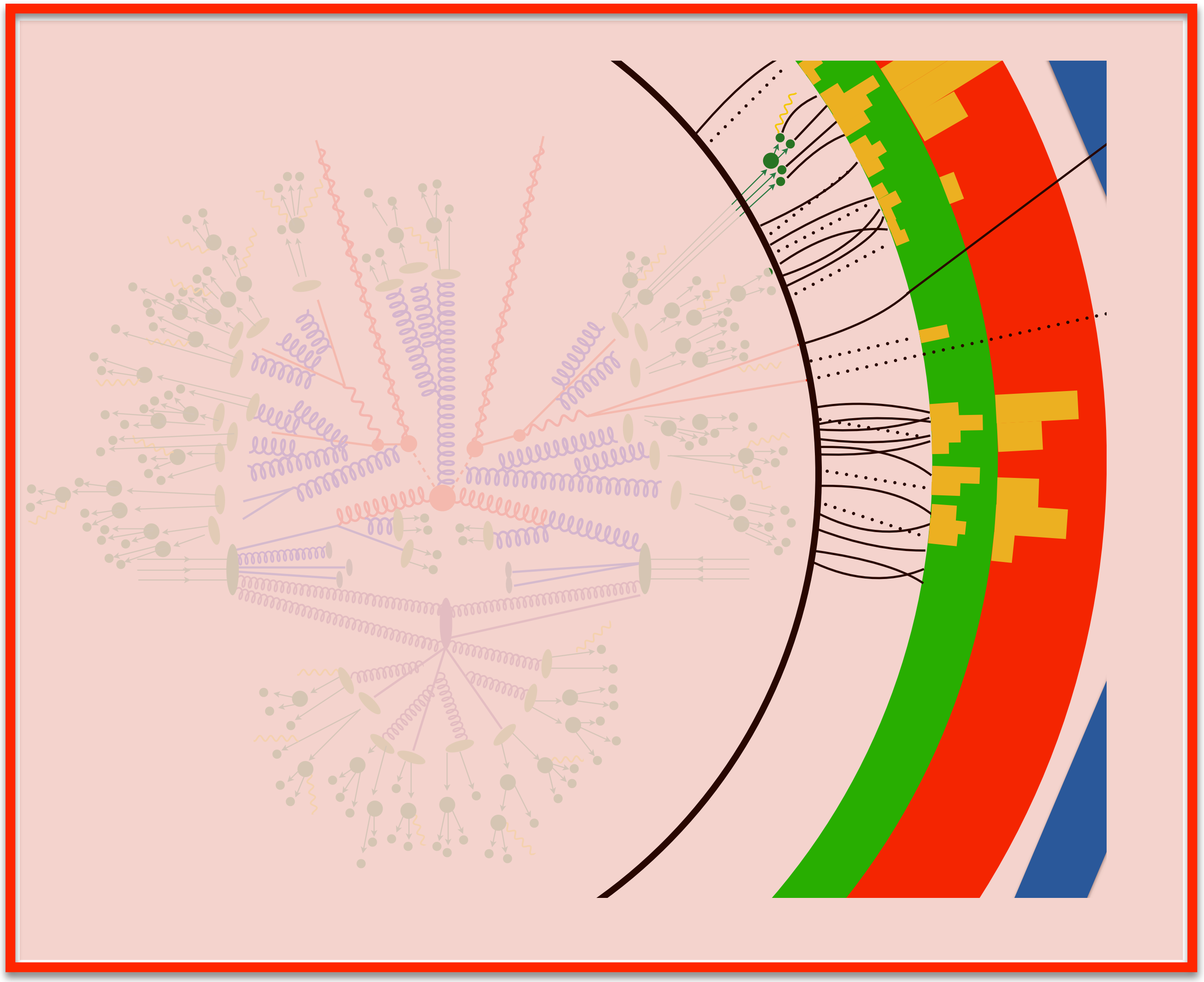
correction of detector effects = unfolding

The Unfolding Challenge

Want this

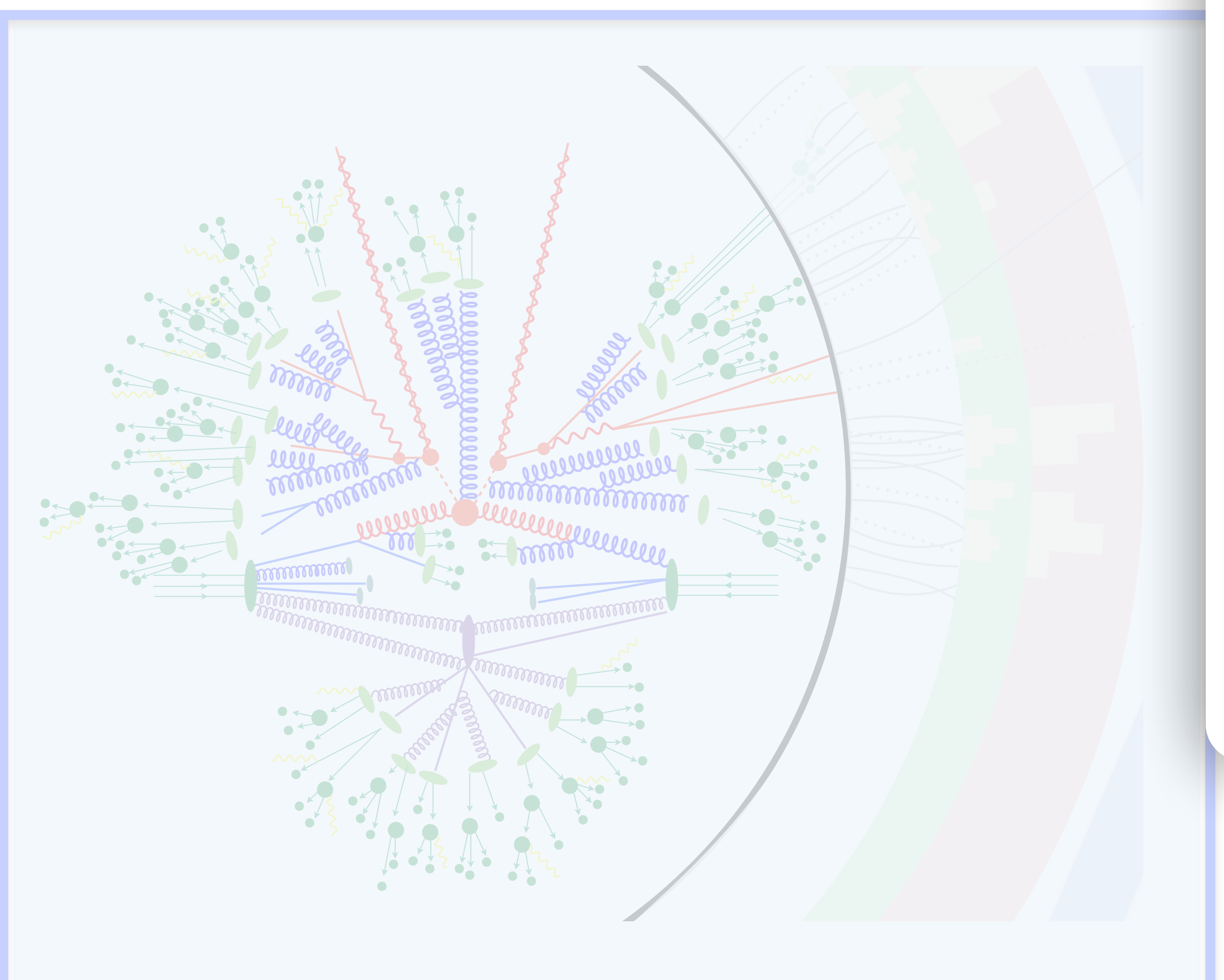


Measure this



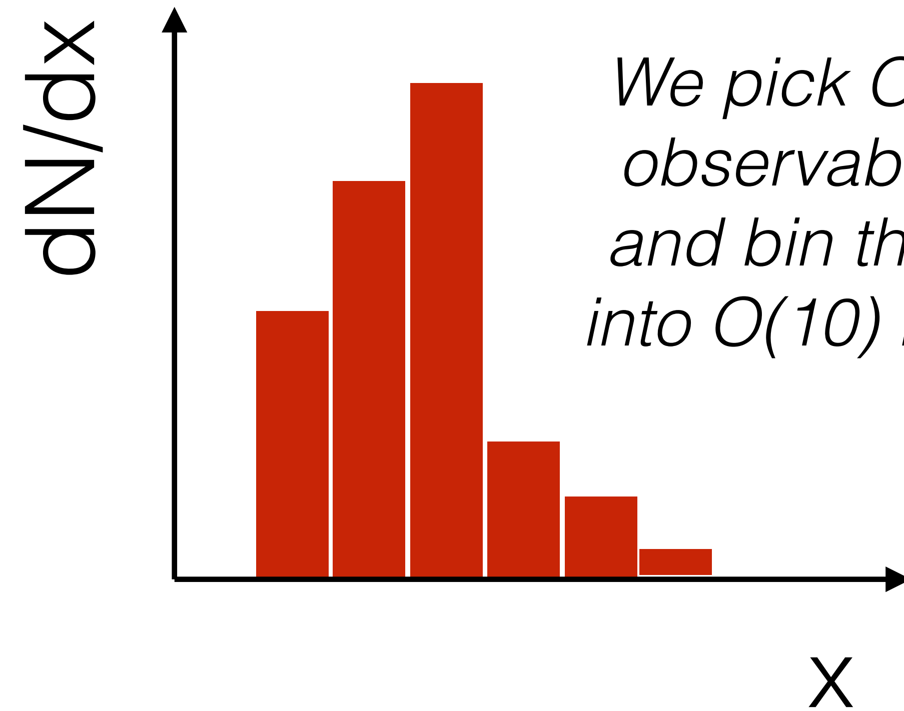
The Unfolding Challenge

Want this



Measure this

Usual solution:

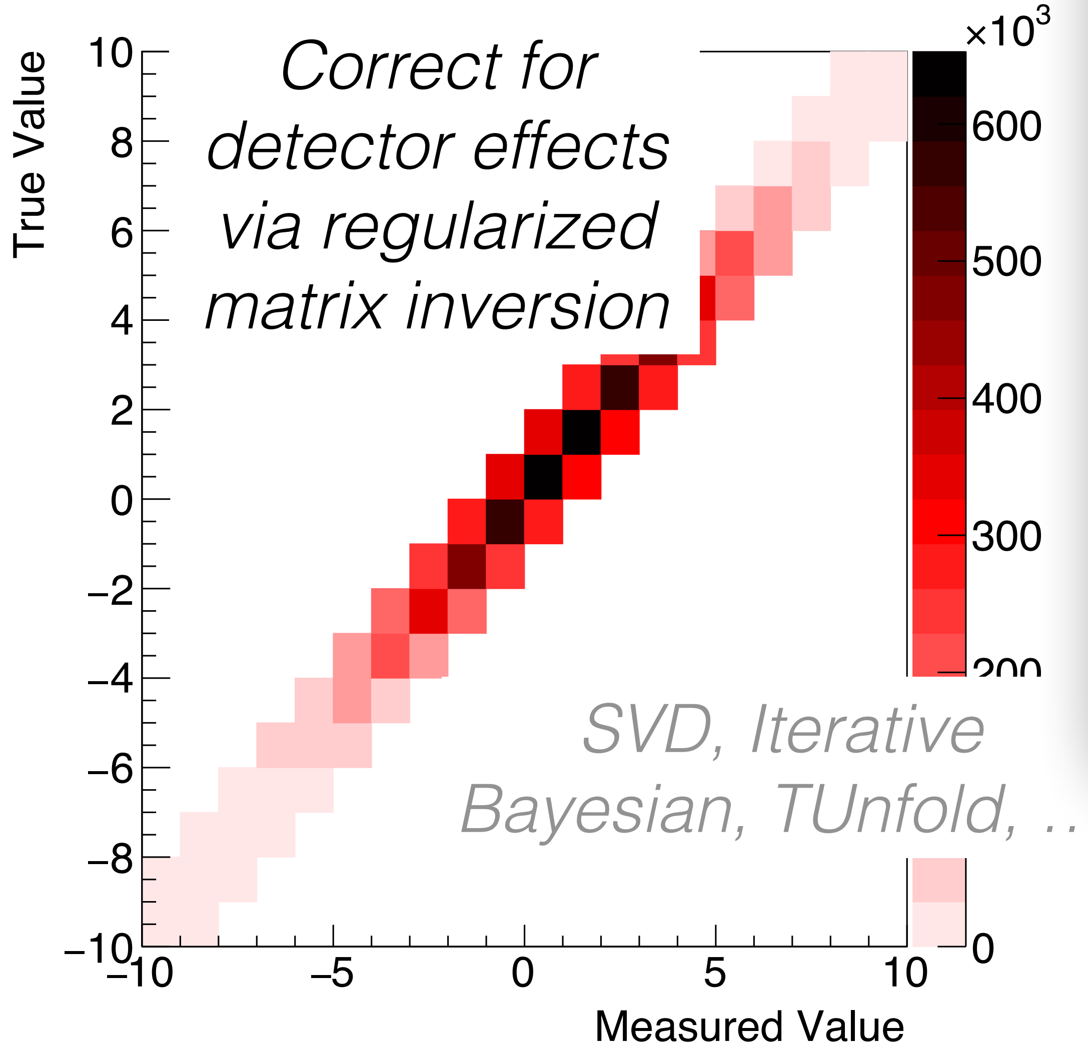


We pick O(1) observables and bin them into O(10) bins

The Unfolding Challenge

Want this

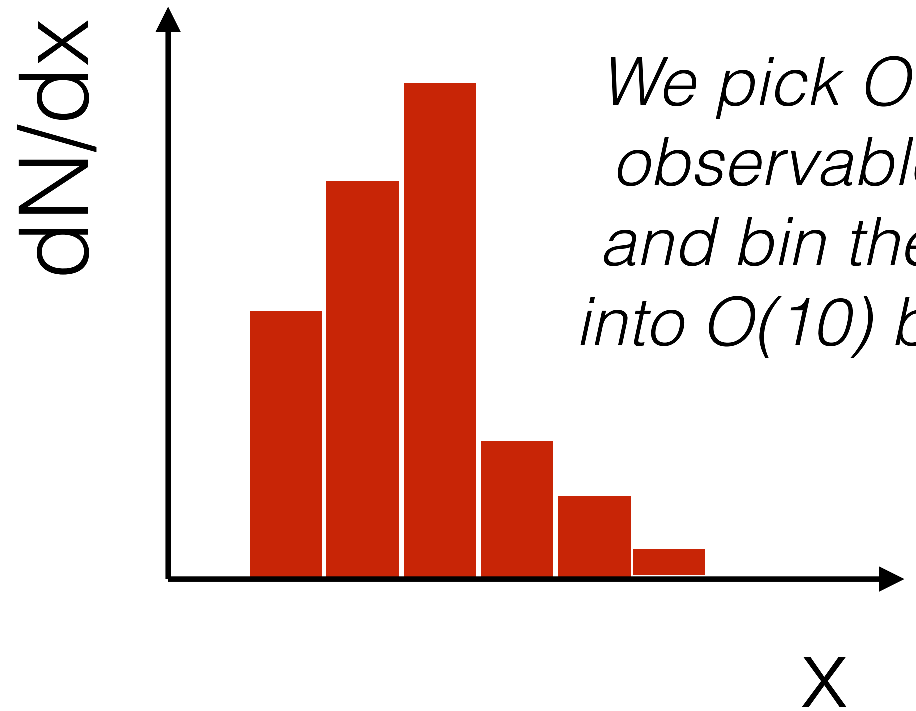
Measure this



Correct for detector effects via regularized matrix inversion

SVD, Iterative Bayesian, TUnfold, …

Usual solution:



We pick O(1) observables and bin them into O(10) bins

The Unfolding Challenge

Want this

Measure this

Usual solution:

pick O(1) observables bin them O(10) bins

Can we go unbinned?

Can we use many dimensions?

SVD, Iterative Bayesian, TUnfold, …

Unbinned Unfolding

There were some early proposals for unbinned unfolding* but as far as I am aware, they were not used for any measurements.

However, recent innovations in machine learning and resulted in new methods for unbinned unfolding, which are being used for data analysis+ (!)

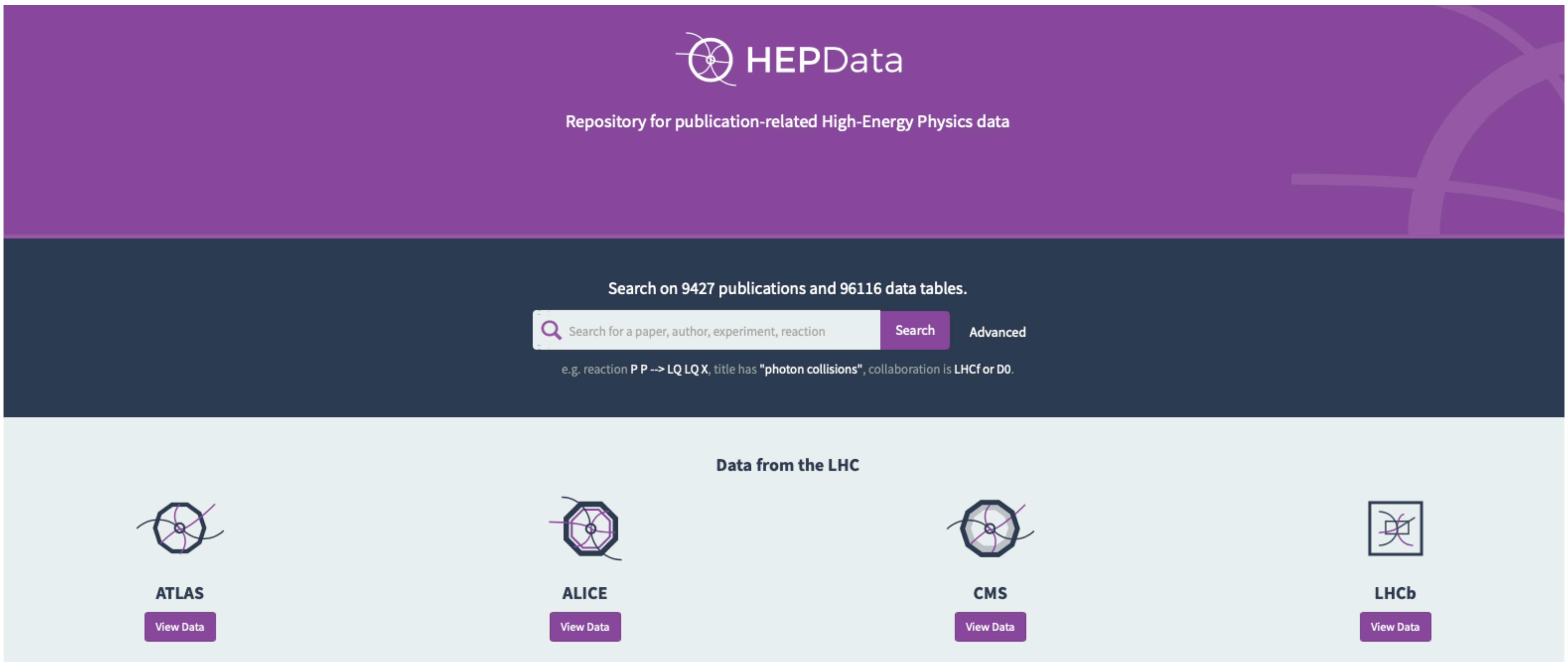
The goal of this discussion is to propose a common way for publishing unbinned results to maximize their science potential

We need input from both experimentalist and theorists (!)

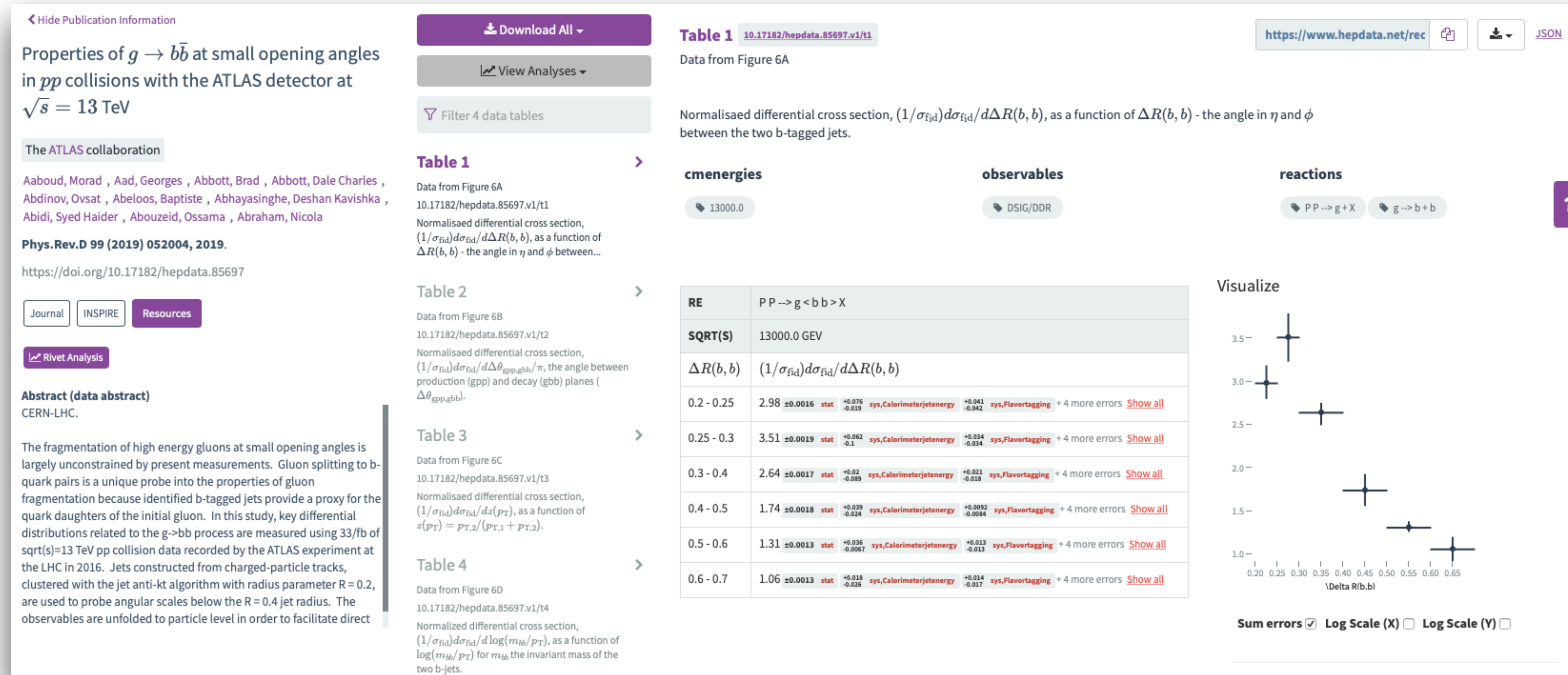
*see L. Lindemann and G. Zech, NIM A 354 (1995) 516 & related
+see https://www-h1.desy.de/h1/www/publications/htmlsplit/H1prelim-21-031.long.html

Publishing Binned Results

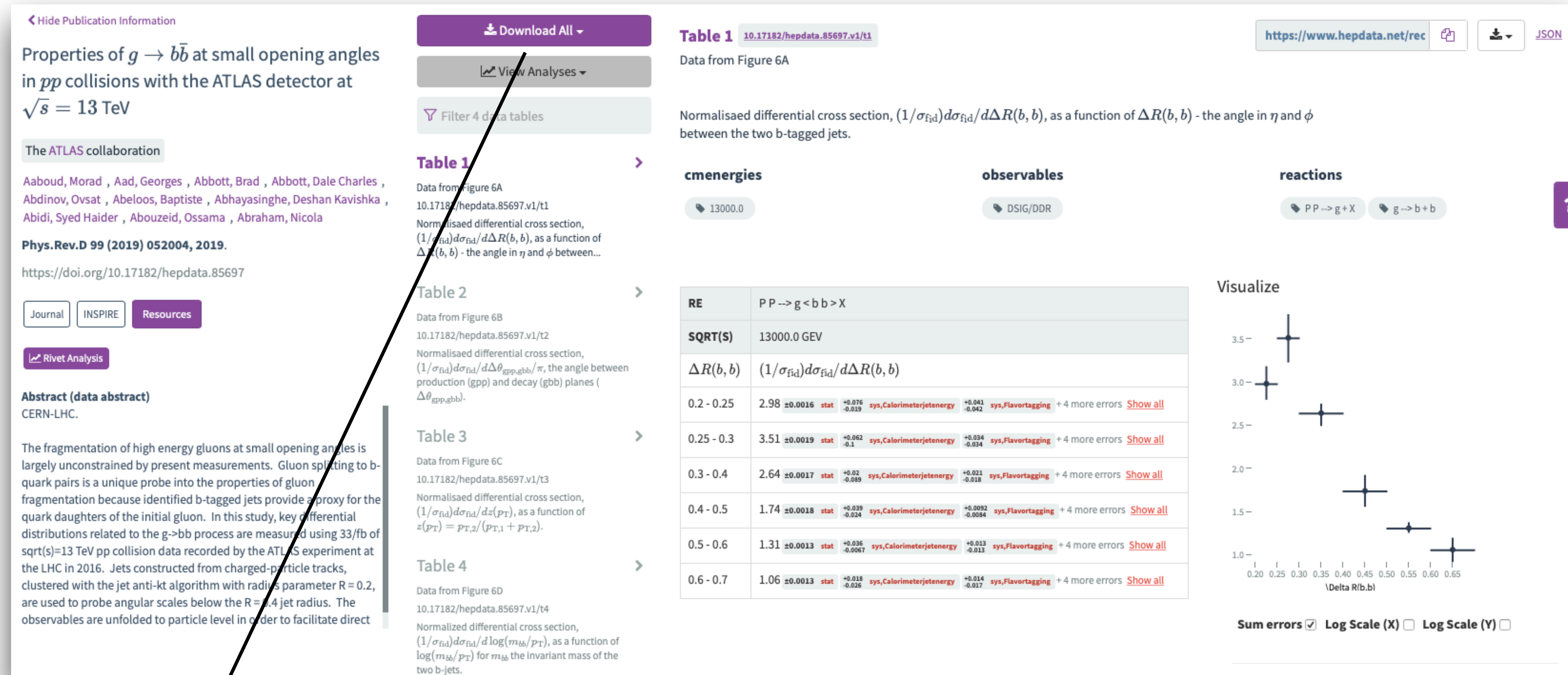
How do we publish binned results?



Publishing Binned Results

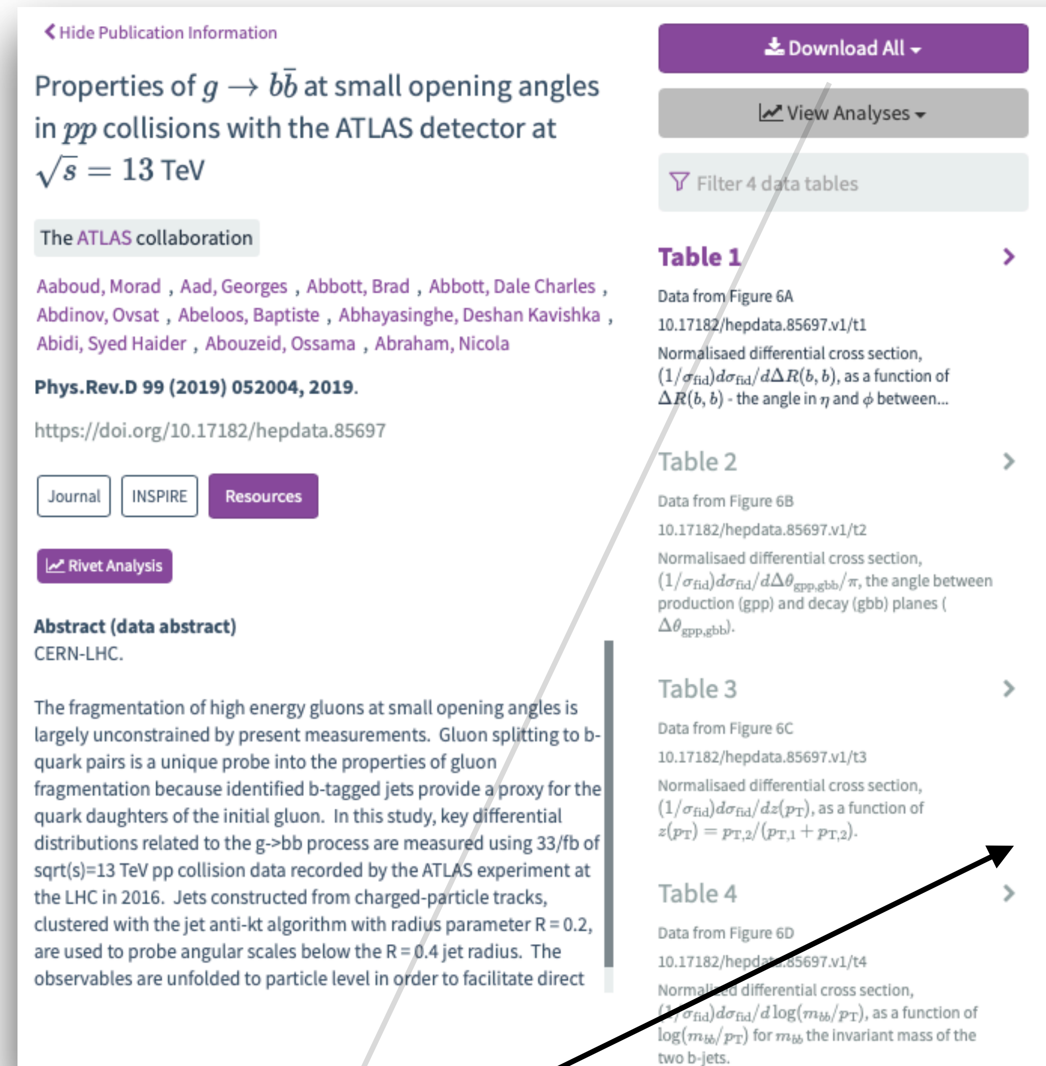


Publishing Binned Results



YAML with resource files
YAML
YODA
ROOT
CSV

Publishing Binned Results




```yaml
dependent_variables:
- header: {name: '$(1/\sigma_\text{fid})d\sigma_\text{fid}/d\Delta R(b,b)$'}
  qualifiers:
  - {name: RE, value: P P --> g < b b > X}
  - {name: SQRT(S), units: GEV, value: '13000.0'}
  values:
  - errors:
    - {label: stat, symerror: '0.0016'}
    - asymerror: {minus: '-0.019', plus: '0.076'}
      label: sys,Calorimeterjetenergy
    - asymerror: {minus: '-0.042', plus: '0.041'}
      label: sys,Flavortagging
    - asymerror: {minus: '-0.0069', plus: '0.03'}
      label: sys,Tracking
    - asymerror: {minus: '-0.1', plus: '0.1'}
      label: sys,Backgroundfit
    - asymerror: {minus: '-0.00099', plus: '0.00099'}
      label: sys,UnfoldingMethod
    - asymerror: {minus: '-0.15', plus: '0.15'}
      label: sys,Theoreticalmodeling
    value: '2.98'
  - errors:
    - {label: stat, symerror: '0.0019'}
    - asymerror: {minus: '-0.1', plus: '0.062'}
      label: sys,Calorimeterjetenergy
    - asymerror: {minus: '-0.034', plus: '0.034'}
      label: sys,Flavortagging
    - asymerror: {minus: '-0.0092', plus: '0.022'}
      label: sys,Tracking
    - asymerror: {minus: '-0.13', plus: '0.13'}
      label: sys,Backgroundfit
    - asymerror: {minus: '-0.0015', plus: '0.0015'}
      label: sys,UnfoldingMethod
    - asymerror: {minus: '-0.23', plus: '0.23'}
      label: sys,Theoreticalmodeling
    value: '3.51'
```


:


```yaml
independent_variables:
- header: {name: '$\Delta R(b,b)$'}
  values:
  - {high: 0.25, low: 0.2}
  - {high: 0.3, low: 0.25}
  - {high: 0.4, low: 0.3}
  - {high: 0.5, low: 0.4}
  - {high: 0.6, low: 0.5}
  - {high: 0.7, low: 0.6}
```


- YAML with resource files
- YAML
- YODA
- ROOT
- CSV

YAML files with metadata, bin contents, and uncertainties

How to represent unbinned data?

If the data can be fit with a function, you could publish the function (e.g. if it is a NN, you could publish the architecture and weights).

Another natural representation that doesn't require a function fit is to publish data sampled from the unfolded result.

My proposal is based on this idea.

Proposal

As in HEPData, I propose there is a “submission” YAML file with the same measurement metadata.

Each submeasurement* also has some metadata & points to a data file. In HEPData, the data file is itself a YAML file.

The files will have data with the “shape” [(M+1) x N(k+1)]

…where N is the number of sampled events

and M is the number of systematic uncertainties

and k is the number of dimensions per event

*this could be a single observable, or many observables

Proposal

The files will have data with the "shape" [(M+1) x N(k+1)]

Each event has k floats* and 1 event weight

There are N events

This is repeated for each of the M systematic uncertainties

For representations that don't have weights, the weights will be set to 1. For representations that only use weights, there will be M copies of the original array.

I have not thought deeply about file formats (npy, root, hdf5) and would be happy to hear opinions.

*For variable-length measurements, perhaps should use variable-length arrays like awkward for storage

Proposal - uncertainties

The submission YAML should give metadata about which uncertainties are included.

For statistical uncertainties, there should be Q replicas and the uncertainty in a given bin is computed by taking the standard deviation over replicas.

For systematic uncertainties, the difference between the nominal and varied bin content is the uncertainty.

There should be warnings in metadata and/or inflated uncertainties in regions of phase space that should not be studied with the data.

Proposal - where to store?

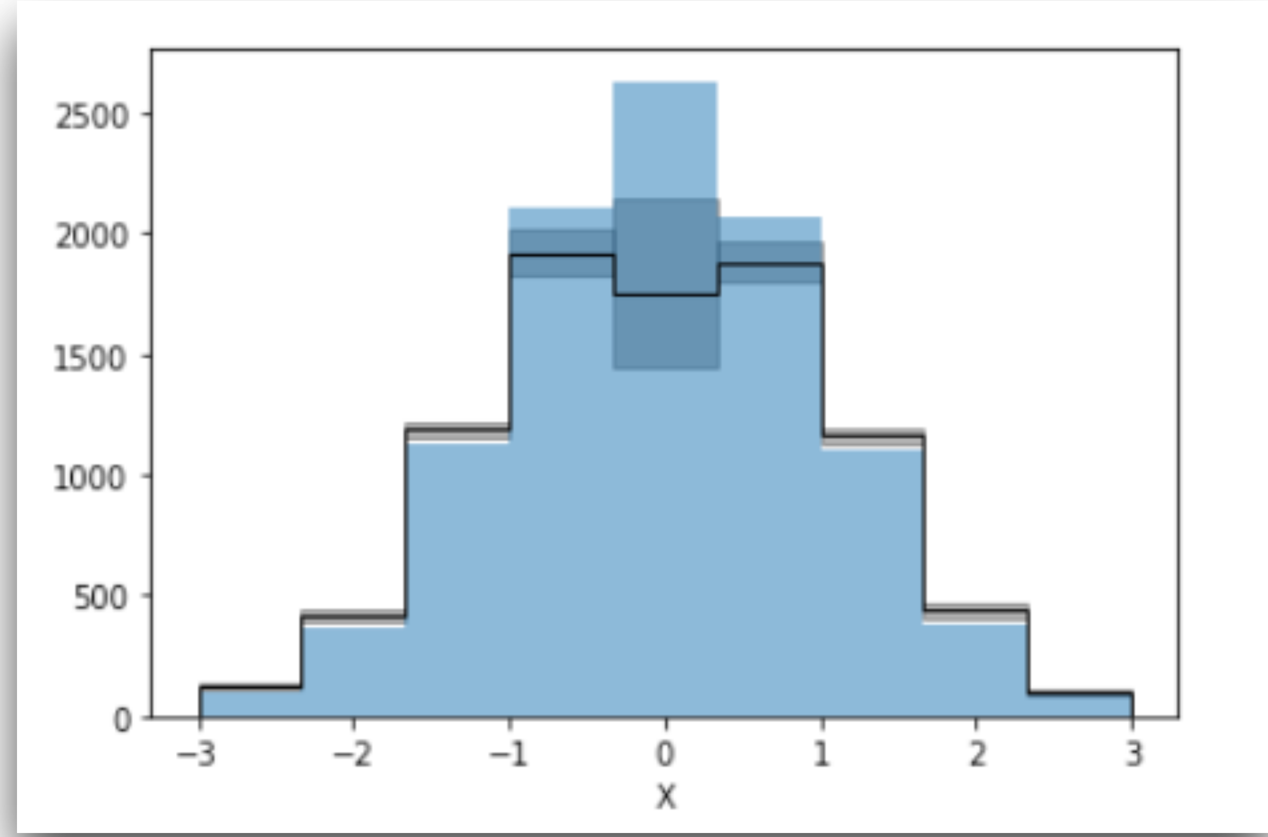
Zenodo is a very natural location. Maybe the submission YAML can also be hosted on HEPData and linked to Zenodo for each searching?

Proposal - example


```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt

In [2]: x = np.random.normal(0,1,10000)

In [3]: w = np.abs(x)**0.2
        w_syst_up = np.abs(x)**0.3
        w_syst_dn = np.abs(x)**0.1

In [4]: plt.hist(x,bins=np.linspace(-3,3,10),alpha=0.5)
        plt.hist(x,bins=np.linspace(-3,3,10),weights=w,histtype="step",color="black")
        n_syst_up,b=np.histogram(x,bins=np.linspace(-3,3,10),weights=w_syst_up)
        n_syst_dn,_=np.histogram(x,bins=np.linspace(-3,3,10),weights=w_syst_dn)
        for i in range(len(b)-1):
            plt.fill_between([b[i],b[i+1]],n_syst_dn[i],n_syst_up[i],color="black",alpha=0.3)
        plt.xlabel("X")
```

```
In [5]: d = {"nominal":x,
             "nominalw":w,
             "syst_up":x,
             "syst_upw":w_syst_up,
             "syst_dn":x,
             "syst_dnw":w_syst_dn}

In [6]: df = pd.DataFrame(data=d)

In [7]: df
Out[7]:
```

	nominal	nominalw	syst_up	syst_upw	syst_dn	syst_dnw
0	0.731914	0.939490	0.731914	0.910622	0.731914	0.969273
1	0.146232	0.680783	0.146232	0.561711	0.146232	0.825096
2	-0.629654	0.911634	-0.629654	0.870423	-0.629654	0.954795
3	0.581001	0.897089	0.581001	0.849675	0.581001	0.947148
4	-0.321038	0.796730	-0.321038	0.711159	-0.321038	0.892597
...	...	...	...	...	...	...



Discussion!