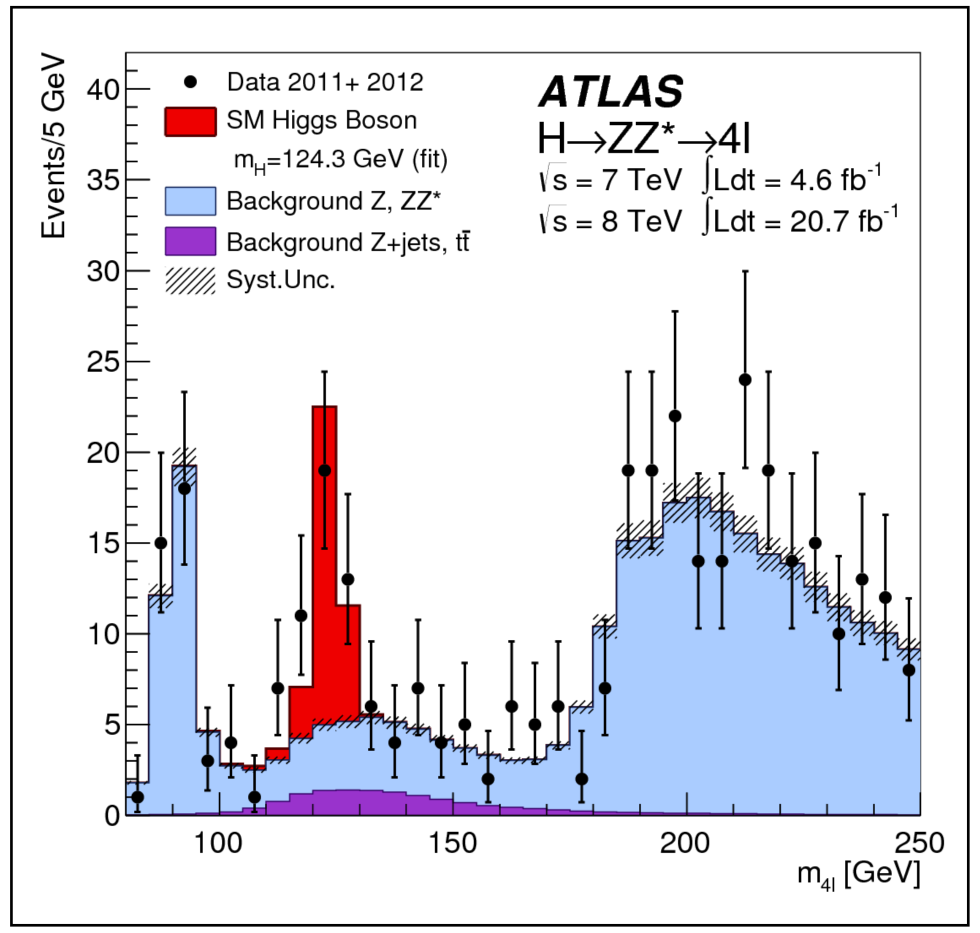
pyhf JSON and other thoughts

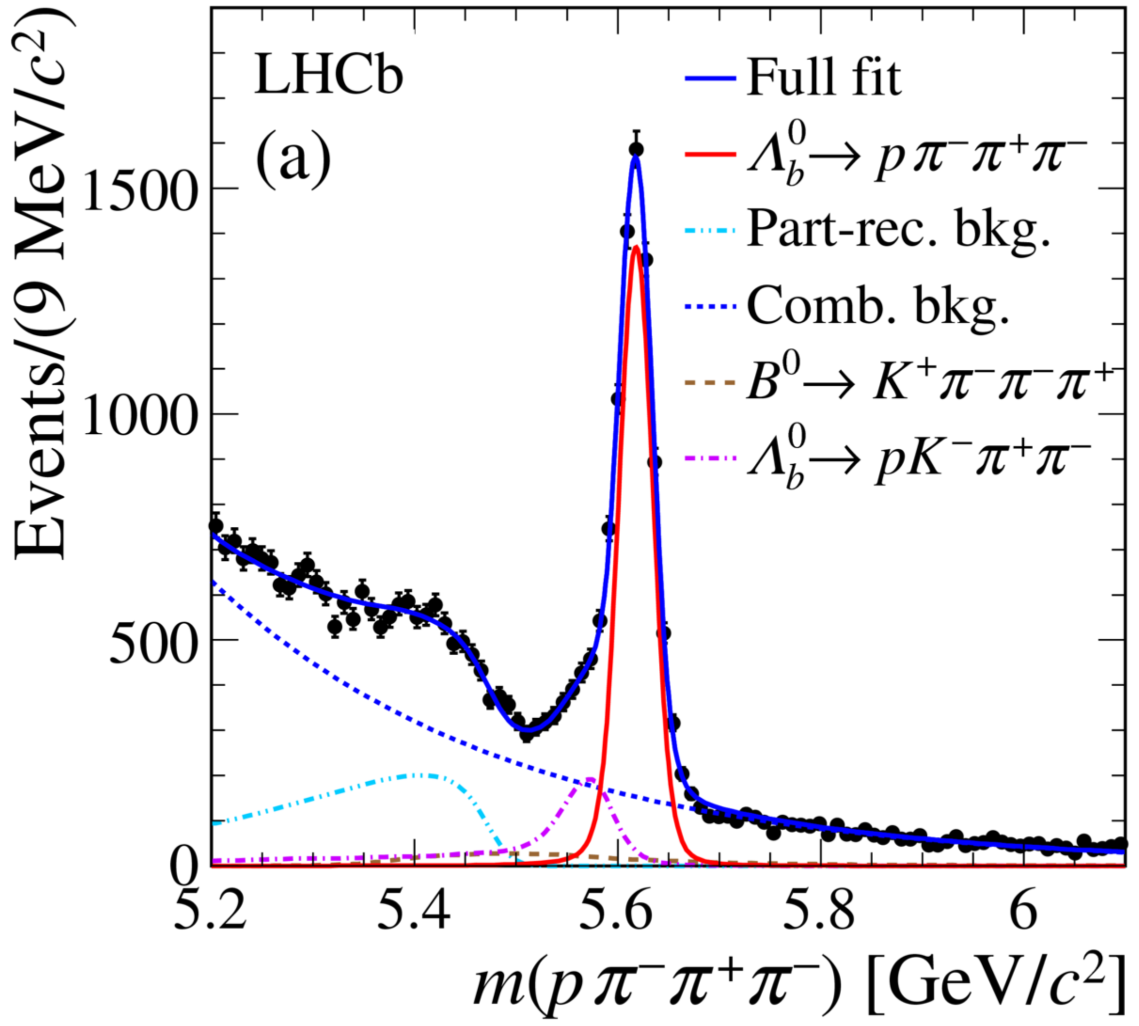
short overview

Lukas Heinrich, CERN

Binned vs Unbinned

HEP probability modelling splits along binned vs unbinned models





Unbinned: have access to a per-event model $p(x_i \mid \theta)$

$$p(\{x_i\} \mid \theta) = \mathrm{Pois}(N \mid \theta) \prod_{i=0}^{N} p(x_i \mid \theta)$$

Binned vs Unbinned

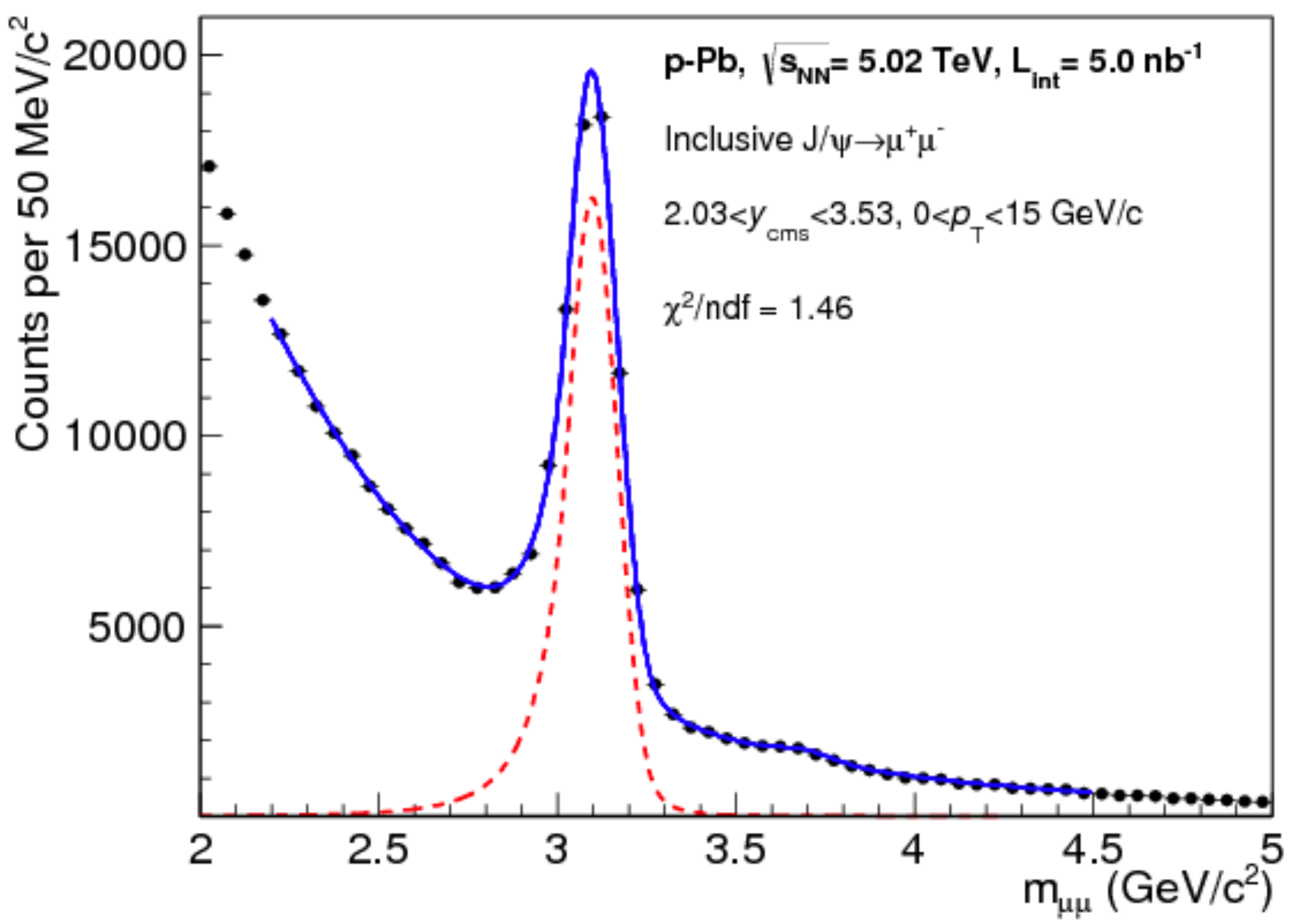
Unbinned: have access to a per-event model $p(x_i \,|\, \theta)$

$$p(\{x_i\} \,|\, \theta) = \mathrm{Pois}(N \,|\, \lambda(\theta)) \prod_{i=0}^{N} p(x_i \,|\, \theta)$$

Key Question: how to describe $p(x_i \,|\, \theta)$

1. from explicit **parametrized functions** (e.g. Gaussian, Crystal Ball, Exp ...)
2. density estimated from simulation (histograms, KDE, ...)

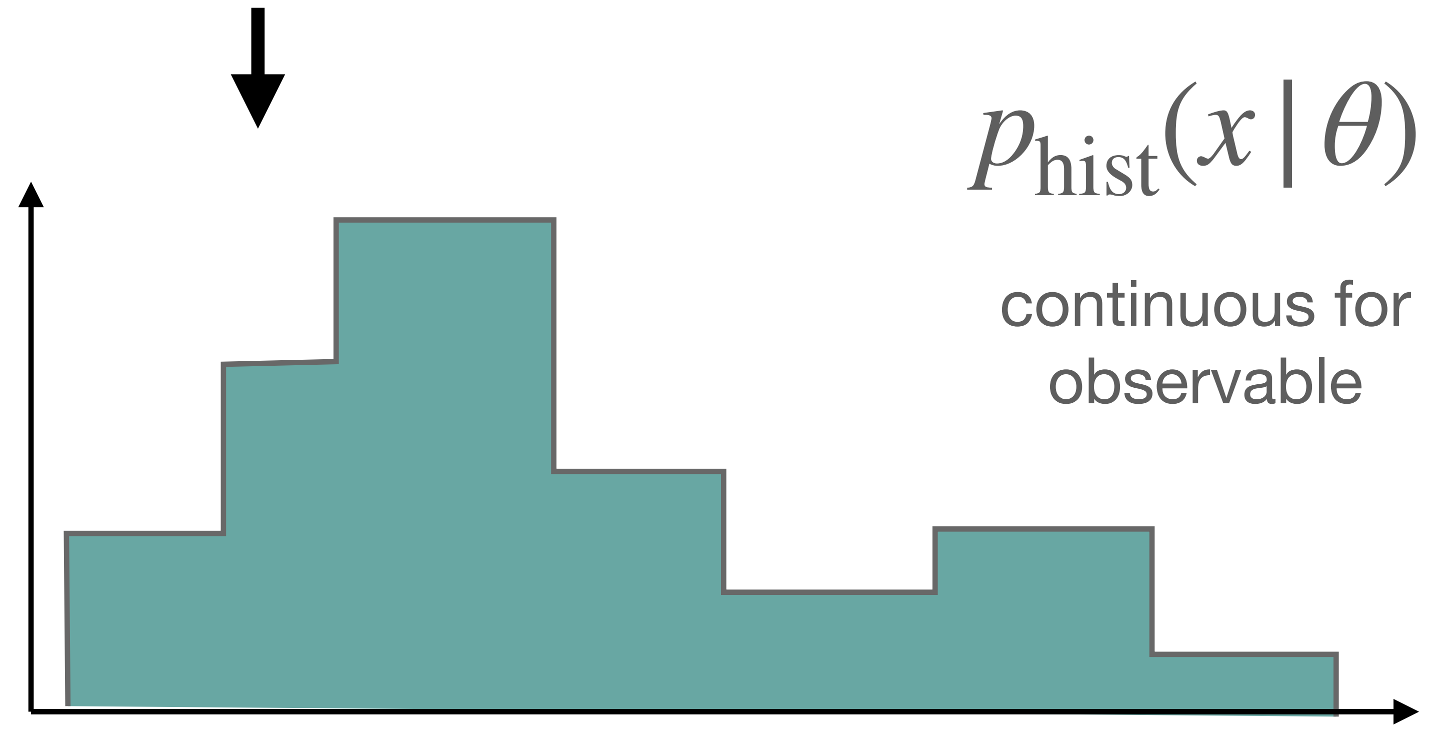
to-binned models



Binned vs Unbinned

Binned Models ~ Unbinned Models with **step-wise** per-event model

$$p(\{x_i\}\,|\,\theta) = \mathrm{Pois}(N\,|\,\lambda(\theta))\prod_{i=0}^{N} p(x_i\,|\,\theta)$$



$$p_{\mathrm{hist}}(x\,|\,\theta) = p(b(x)\,|\,\theta)$$

continuous for observable

discrete for each bin

equivalent to (up to const. factors)

$$p(\{n_b\}\,|\,\theta) = \prod_b \mathrm{Pois}(n_b\,|\,\lambda(\theta)p(b\,|\,\theta)) = \prod_b \mathrm{Pois}(n_b\,|\,\lambda_b(\theta))$$

For binned models: task to describe **parametrized histograms / yields**

(after that only need poisson)

$$h(\theta) = \lambda_b(\theta)$$

Open World vs Closed World

For **both** unbinned & binned we need to describe parametrized objects

$$p(x\,|\,\theta)$$

parametrized per-event model

$$h(\theta) = \lambda_b(\theta)$$

parametrized histograms

Open World: modeler has total freedom

$p(x\,|\,\theta)$ can be arbitrary function as long as it's a p.d.f.

$\lambda_b(\theta)$ arbitrary function producing yields

Difficult/Impossible to find a format: how to you describe arbitrary functions in finite amount of information?

Open World vs Closed World

For **both** unbinned & binned we need to describe parametrized objects

$$p(x\,|\,\theta)$$

parametrized per-event model

$$h(\theta) = \lambda_b(\theta)$$

parametrized histograms

Closed World: modeler has only finite choice to build up objects

$p(x\,|\,\theta)$ can only be mixture of gaussian, crystal ball + exponential

$\lambda_b(\theta)$ defined through a set of specific way to interpolated between histograms acquired from simulation

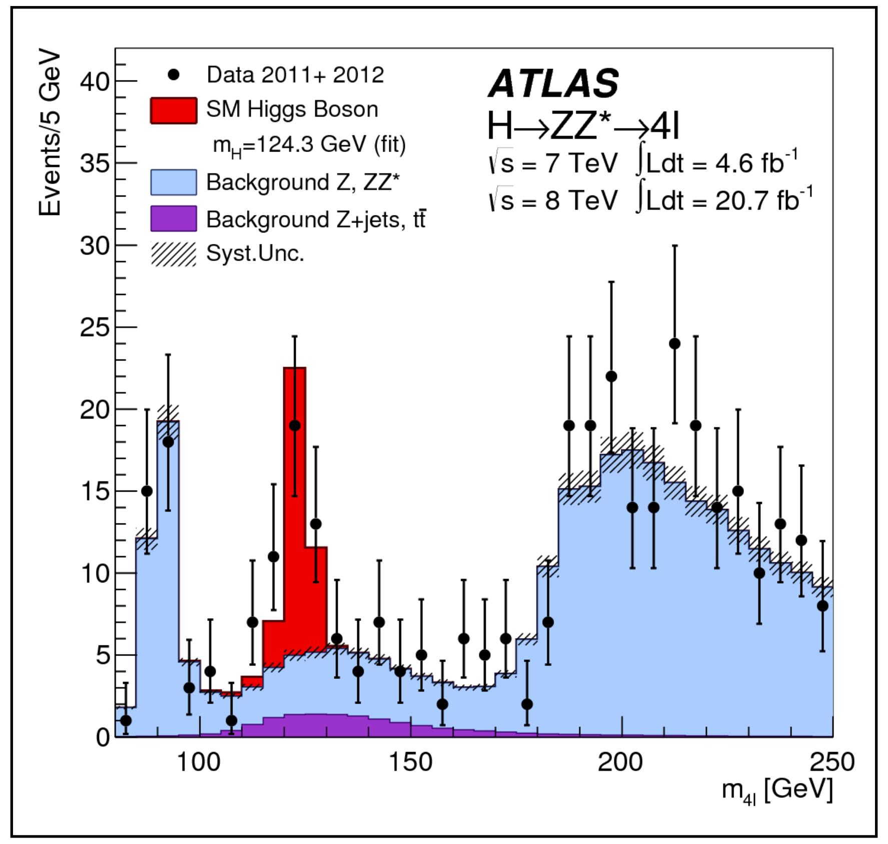
Tradeoff: small number of building blocks vs expressiveness of describable objects

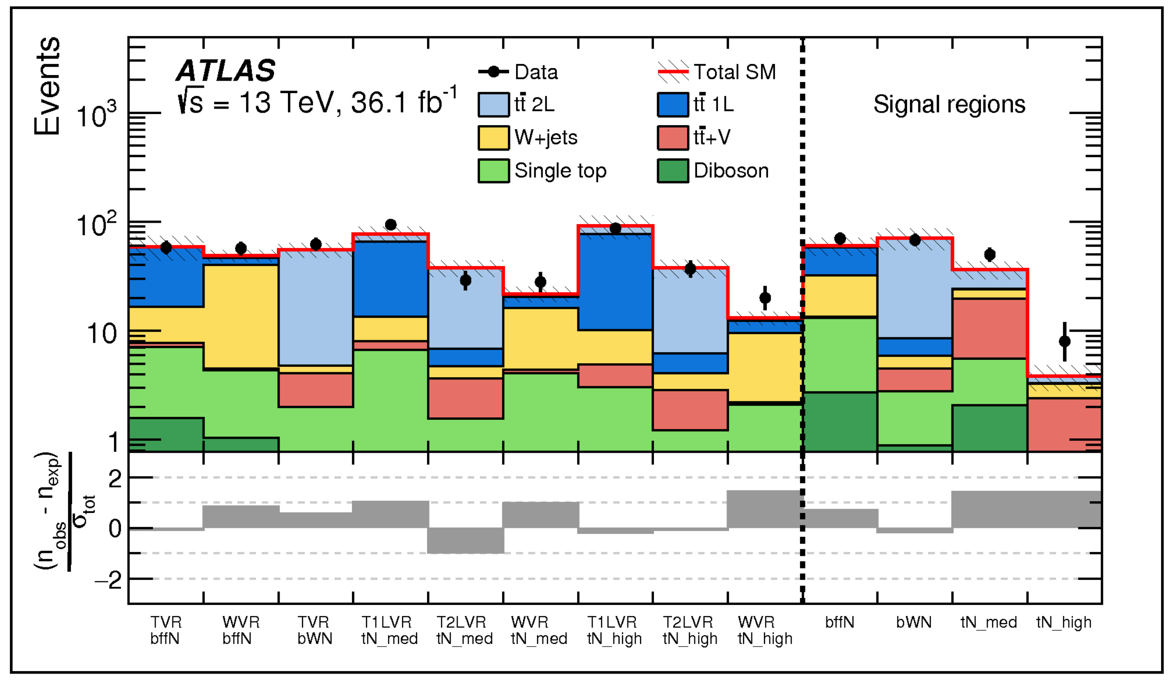
HistFactory

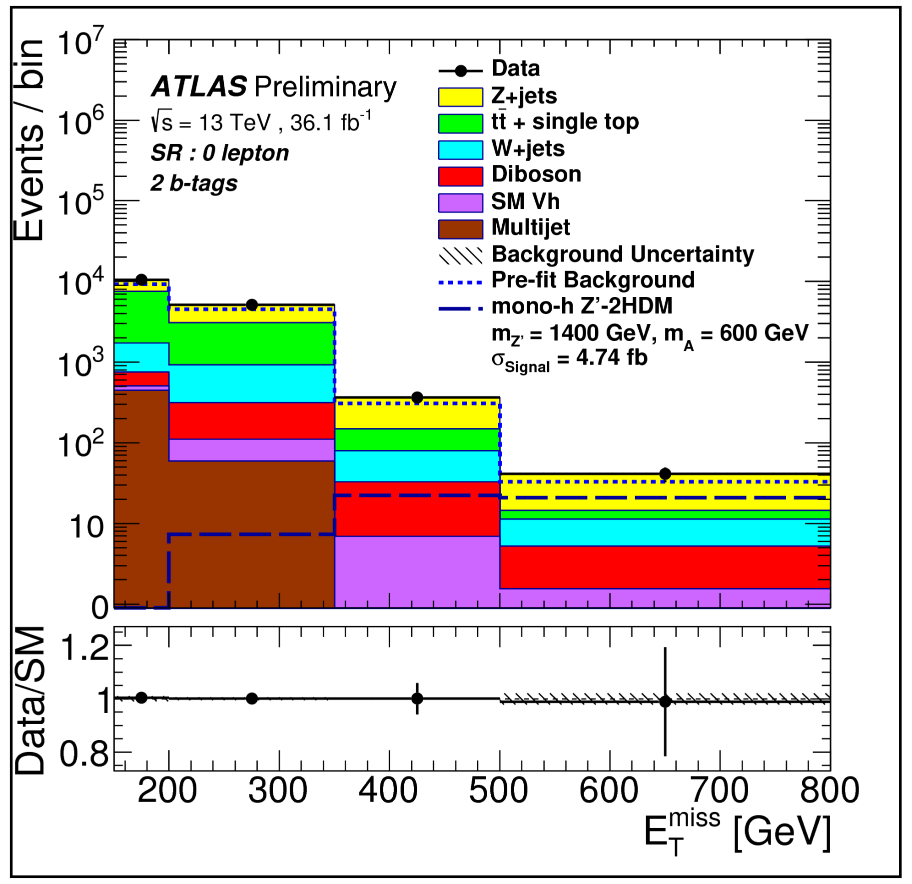
From experience: closed world approach for binned is very doable

- a huge amount of information comes from the simulation, not the parametrization of the histograms

HistFactory: surprisingly small set of building blocks suitable for wide range of physics





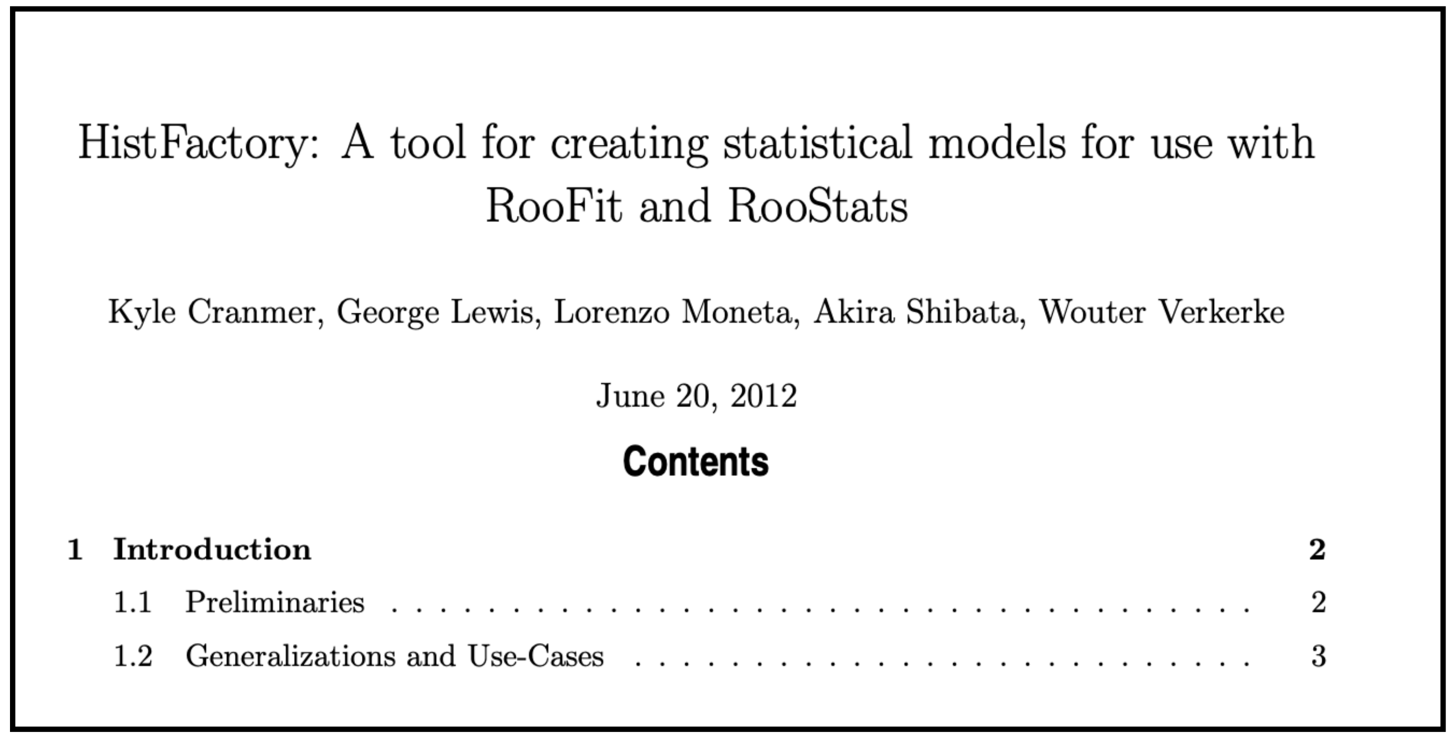


HistFactory

From experience: closed world approach for binned is very doable

- a huge amount of information comes from the simulation, not the parametrization of the histograms

HistFactory: surprisingly small set of building blocks suitable for wide range of physics



HistFactory: A tool for creating statistical models for use with RooFit and RooStats

Kyle Cranmer, George Lewis, Lorenzo Moneta, Akira Shibata, Wouter Verkerke

June 20, 2012

Contents

1 **Introduction** 2

1.1 Preliminaries 2

1.2 Generalizations and Use-Cases 3

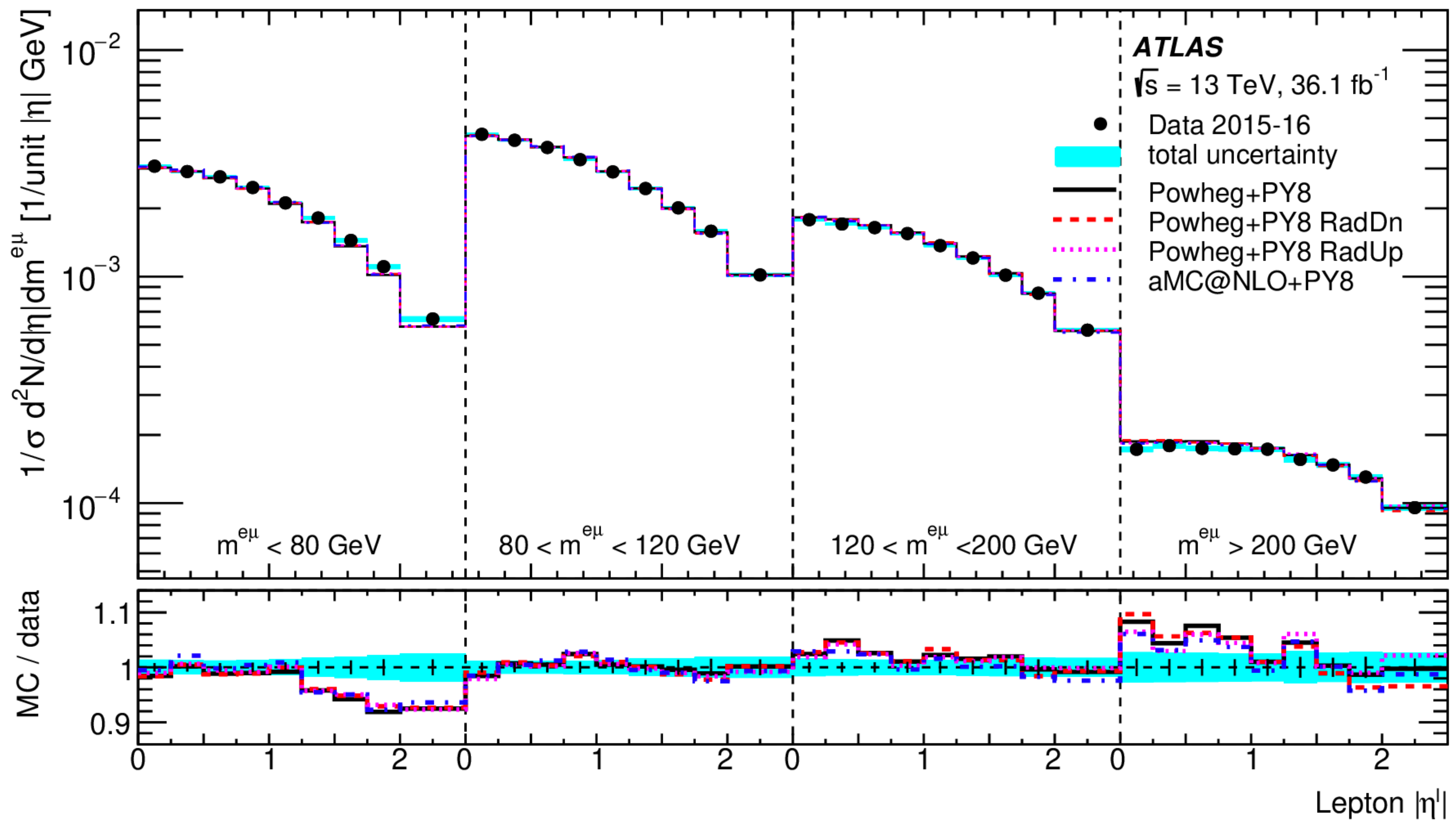
parametrized histograms

$$f(\boldsymbol{n},\boldsymbol{a}\mid\boldsymbol{\eta},\boldsymbol{\chi}) = \underbrace{\prod_{c\in\text{channels}}\prod_{b\in\text{bins}_c}\text{Pois}\left(n_{cb}\mid\nu_{cb}(\boldsymbol{\eta},\boldsymbol{\chi})\right)}_{\substack{\text{Simultaneous measurement}\\ \text{of multiple channels}}}\;\underbrace{\prod_{\chi\in\mathcal{X}} c_\chi(a_\chi\mid\chi)}_{\substack{\text{constraint terms}\\ \text{for ``auxiliary measurements''}}}$$

https://inspirehep.net/literature/1236448

HistFactory

Joint measurement of binned observable distributions accross multiple channels (phase space regions)



$$f(\boldsymbol{n}, \boldsymbol{a} \mid \boldsymbol{\eta}, \boldsymbol{\chi}) = \underbrace{\prod_{c \in \text{channels}} \prod_{b \in \text{bins}_c} \text{Pois}\left(n_{cb} \mid \nu_{cb}(\boldsymbol{\eta}, \boldsymbol{\chi})\right)}_{\text{Simultaneous measurement of multiple channels}} \underbrace{\prod_{\chi \in \boldsymbol{\chi}} c_\chi(a_\chi \mid \chi)}_{\text{constraint terms for "auxiliary measurements"}}$$

HistFactory

$$\prod_{c \in \text{channels}} \prod_{b \in \text{bins}_c} \text{Pois}\left(n_{cb} \,\middle|\, \boxed{\nu_{cb}\left(\boldsymbol{\eta}, \boldsymbol{\chi}\right))}\right.$$

$$\nu_{cb}(\boldsymbol{\phi}) = \sum_{s \in \text{samples}} \nu_{scb}(\boldsymbol{\eta}, \boldsymbol{\chi}) = \sum_{s \in \text{samples}} \underbrace{\left(\prod_{\kappa \in \boldsymbol{\kappa}} \kappa_{scb}(\boldsymbol{\eta}, \boldsymbol{\chi})\right)}_{\text{multiplicative modifiers}} \left(\underbrace{\nu^0_{scb}(\boldsymbol{\eta}, \boldsymbol{\chi})}_{\text{base histogram}} + \underbrace{\sum_{\Delta \in \boldsymbol{\Delta}} \Delta_{scb}(\boldsymbol{\eta}, \boldsymbol{\chi})}_{\text{additive modifiers}}\right).$$

Yields are build from

- base histogram (fixed, non-parametrized)
- parametrized modifier terms that act additively or multiplicatively
  → modifiers can be also derived on simulation input inpu

Question to CMS: does Combine fit into this scaffolding?

HistFactory

Seven Modifiers - six multiplicative, one additive

	Description	Modification	Constraint Term c_χ	Input
constrained	Uncorrelated Shape	$\kappa_{scb}(\gamma_b) = \gamma_b$	$\prod_b \mathrm{Pois}\left(r_b = \sigma_b^{-2} \middle\vert \rho_b = \sigma_b^{-2}\gamma_b\right)$	σ_b
	Correlated Shape	$\Delta_{scb}(\alpha) = f_p\left(\alpha \middle\vert \Delta_{scb,\alpha=-1}, \Delta_{scb,\alpha=1}\right)$	$\mathrm{Gaus}\,(a = 0\vert\, \alpha, \sigma = 1)$	$\Delta_{scb,\alpha=\pm 1}$
	Normalisation Unc.	$\kappa_{scb}(\alpha) = g_p\left(\alpha \middle\vert \kappa_{scb,\alpha=-1}, \kappa_{scb,\alpha=1}\right)$	$\mathrm{Gaus}\,(a = 0\vert\, \alpha, \sigma = 1)$	$\kappa_{scb,\alpha=\pm 1}$
	MC Stat. Uncertainty	$\kappa_{scb}(\gamma_b) = \gamma_b$	$\prod_b \mathrm{Gaus}\left(a_{\gamma_b} = 1 \middle\vert \gamma_b, \delta_b\right)$	$\delta_b^2 = \sum_s \delta_{sb}^2$
	Luminosity	$\kappa_{scb}(\lambda) = \lambda$	$\mathrm{Gaus}\,(l = \lambda_0\vert\, \lambda, \sigma_\lambda)$	$\lambda_0, \sigma_\lambda$
free	Normalisation	$\kappa_{scb}(\mu_b) = \mu_b$		
	Data-driven Shape	$\kappa_{scb}(\gamma_b) = \gamma_b$		

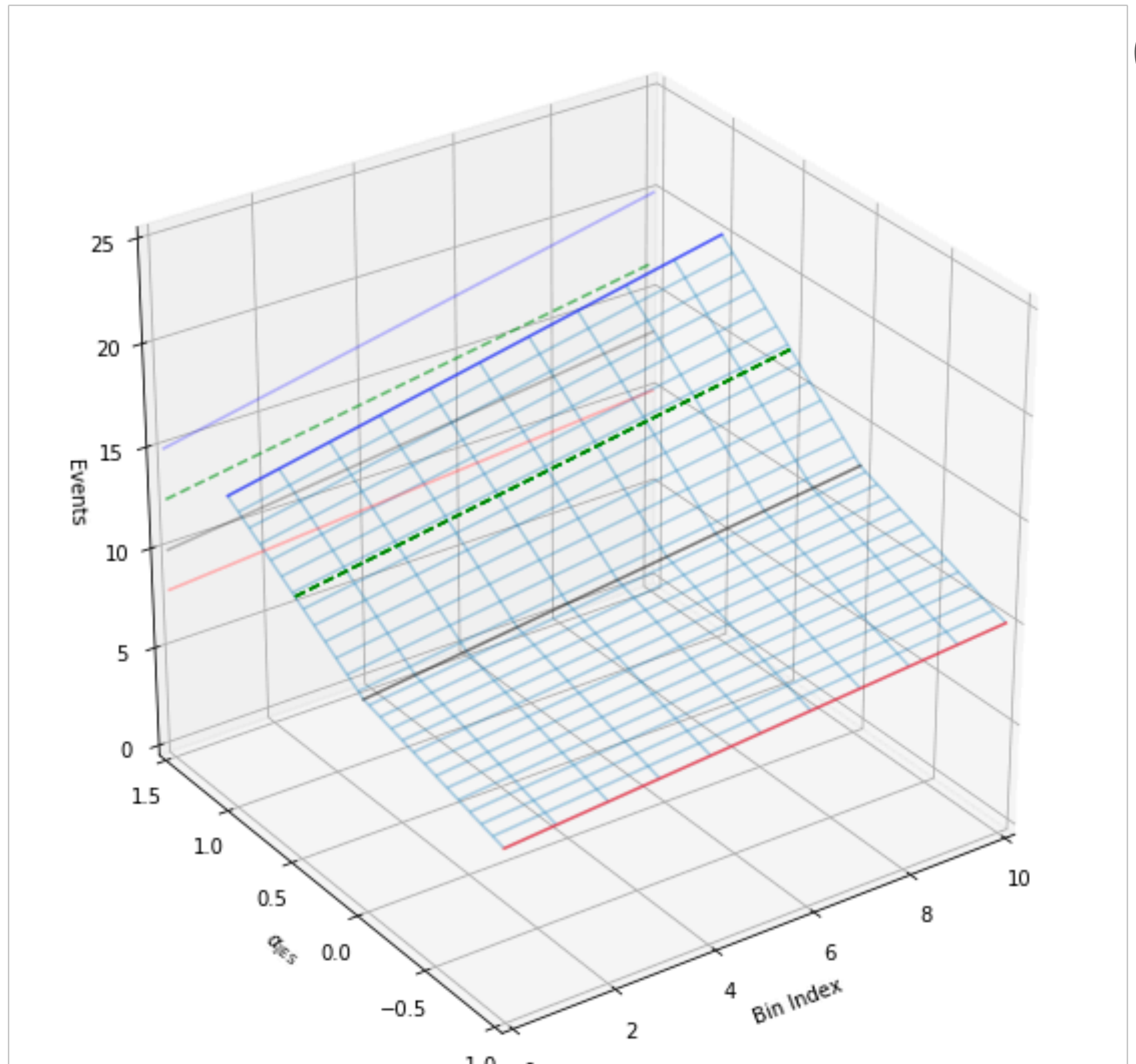
Example - Correlated Shape

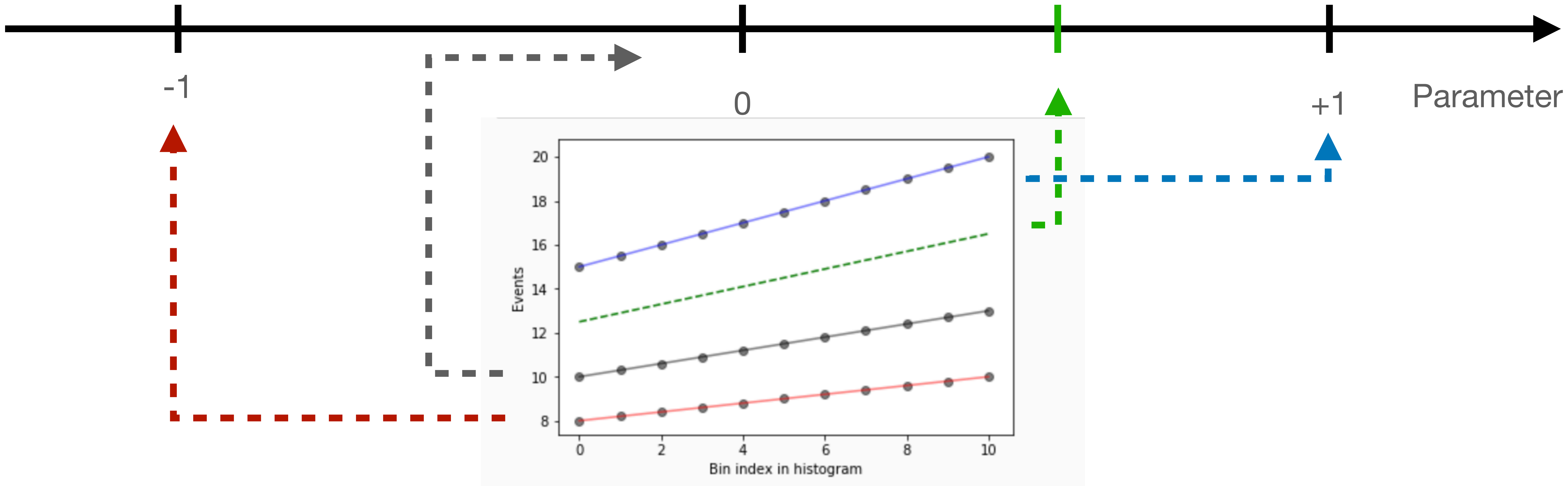
Single parameter $h(\alpha) = h(\alpha; h_0, h_{-1}, h_{+1})$

Input: three histograms (down, nominal, up)

- shape at par values -1,0,1
- choice of 4 interpolation funcs

gives you histogram at any parameter value



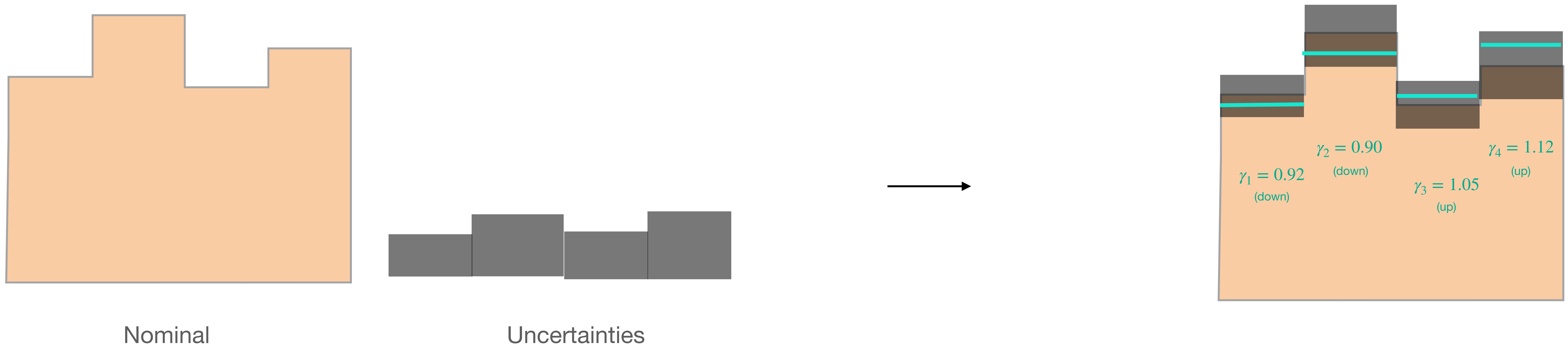


Example - Uncorrelated Shape

Multi-parametter $h(\gamma_1, \gamma_2, \ldots) = h(\gamma_1, \gamma_2, \ldots; h_0, \delta h)$

Input: two histograms/arrays:

- **nominal yields, per-bin uncertainties**
- **each parameter controls fluctuation for one bin**



Collecting the Model

To **describe the model** we need to track

```
for each channel
     for each sample
          ● nominal yields
          ● for each parameter that can affect this sample
            ● additional inputs to fully determine parametrized shape
```

Collecting the Model

To describe likelihoods (not just model) also need to track data

```
for each channel:
    ● observed counts
```

To describe inference (or measurements) we need to track configuration information

- **what are parameters of interest, what are NPs?**
- **what are ranges for parameters, ... (constraint terms, priors ...)**
    - **depends on inference method**

pyhf JSON

Original HistFactory Spec used XML + Data stored in ROOT files

pyhf JSON repackaged as JSON document with inlined data

Not a very fundamental difference. but a few advantages:

- single JSON file vs files in dir structure
- inlined data; no need to have ROOT-file reader to read histogram data / human readable
- JSON as simple/simpler to integrate in web services

Model

```
{
    "version": "1.0.0",
    "channels": [
        {
            "name": "single_channel",
            "samples": [
                {
                    "name": "signal",
                    "data": [5,10],
                    "modifiers": [
                        {"name": "mu", "type": "normfactor","data": null}
                    ]
                },
                {
                    "name": "background",
                    "data": [50,50],
                    "modifiers": [
                        {"name": "correlated_bkg_uncertainty", "type": "histosys", "data": {"hi_data": [45,40],"lo_data": [55,60]}}
                    ]
                }
            ]
        }
    ],
```

Data

```
    "observations": [
        {
            "name": "single_channel",
            "data": [50,50]
        }
    ],
```

Inference Config

```
   "measurements": [
        {
            "name": "measurement",
            "config": {
                "poi": "mu",
                "parameters": [
                    { "bounds": [[0,10]], "inits": [1.0], "fixed": false,"name": "mu" },
                    { "bounds": [[-5.0,5.0]], "inits": [0.0],"fixed": false, "name": "correlated_bkg_uncertainty"}
                ]
            }
        }
    ]
}
```

pyhf JSON

```
{
    "version": "1.0.0",
    "channels": [
        {
            "name": "single_channel",
            "samples": [
                {
                    "name": "signal",
                    "data": [5,10],
                    "modifiers": [
                        {"name": "mu", "type": "normfactor","data": null}
                    ]
                },
                {
                    "name": "background",
                    "data": [50,50],
                    "modifiers": [
                        {"name": "correlated_bkg_uncertainty", "type": "histosys", "data": {"hi_data": [45,40],"lo_data": [55,60]}}
                    ]
                }
            ]
        }
    ]
....
```

for each channel

for each sample

- nominal yields
- for each parameter that can affect this sample
  - additional inputs to fully determine parametrized shape

A word on Inference / Constraint Terms

somewhat geared towards frequentist inference: not fixed but default

$$f(\boldsymbol{n}, \boldsymbol{a} \mid \boldsymbol{\eta}, \boldsymbol{\chi}) = \underbrace{\prod_{c \in \text{channels}} \prod_{b \in \text{bins}_c} \text{Pois}\left(n_{cb} \mid \nu_{cb}(\boldsymbol{\eta}, \boldsymbol{\chi})\right)}_{\substack{\text{Simultaneous measurement} \\ \text{of multiple channels}}} \underbrace{\prod_{\chi \in \boldsymbol{\chi}} c_{\chi}(a_{\chi} \mid \chi)}_{\substack{\text{constraint terms} \\ \text{for ``auxiliary measurements''}}}$$

Constraint Terms: not priors but "subsidiary measurements"

- parameters that act in modifiers either constrained or not
- need to track "result" or subsidiary measurement (auxdata)

Effectively:

- experiments communicating their information about range of NPs
- analyzer can/should respect it (or not)

	Description	Modification	Constraint Term c_{χ}	Input
constrained	Uncorrelated Shape	$\kappa_{scb}(\gamma_b) = \gamma_b$	$\prod_b \text{Pois}\left(r_b = \sigma_b^{-2} \middle\| \rho_b = \sigma_b^{-2}\gamma_b\right)$	σ_b
	Correlated Shape	$\Delta_{scb}(\alpha) = f_p\left(\alpha \middle\| \Delta_{scb,\alpha=-1}, \Delta_{scb,\alpha=1}\right)$	$\text{Gaus}\,(a = 0 \| \alpha, \sigma = 1)$	$\Delta_{scb,\alpha=\pm1}$
	Normalisation Unc.	$\kappa_{scb}(\alpha) = g_p\left(\alpha \middle\| \kappa_{scb,\alpha=-1}, \kappa_{scb,\alpha=1}\right)$	$\text{Gaus}\,(a = 0 \| \alpha, \sigma = 1)$	$\kappa_{scb,\alpha=\pm1}$
	MC Stat. Uncertainty	$\kappa_{scb}(\gamma_b) = \gamma_b$	$\prod_b \text{Gaus}\left(a_{\gamma_b} = 1 \middle\| \gamma_b, \delta_b\right)$	$\delta_b^2 = \sum_s \delta_{sb}^2$
	Luminosity	$\kappa_{scb}(\lambda) = \lambda$	$\text{Gaus}\,(l = \lambda_0 \| \lambda, \sigma_\lambda)$	$\lambda_0, \sigma_\lambda$
free	Normalisation	$\kappa_{scb}(\mu_b) = \mu_b$		
	Data-driven Shape	$\kappa_{scb}(\gamma_b) = \gamma_b$		

pyhf JSON

Where do **constraint terms** fit in the spec?

Right now they are part of the inference configuration

a lot of implicit info not directlty ccoded in thte JSON, but published in the literature

Could introduce a separate new nuisance model spec

Bayes: just an additional Model

Freq: Subsidiary Model + "aux data"

Model

Data

Inference Config

```
{
    "version": "1.0.0",
    "channels": [
        {
            "name": "single_channel",
            "samples": [
                {
                    "name": "signal",
                    "data": [5,10],
                    "modifiers": [
                        {"name": "mu", "type": "normfactor","data": null}
                        {"name": "lumi", "type": "lumi","data": null}
                    ]
                },
                {
                    "name": "background",
                    "data": [50,50],
                    "modifiers": [
                        {"name": "correlated_bkg_uncertainty", "type": "histosys", "data": {"hi_data": [45,40],"lo_data": [55,60]}}
                        {"name": "lumi", "type": "lumi","data": null}
                    ]
                }
            ]
        }
    ],
    "observations": [
        {
            "name": "single_channel",
            "data": [50,50]
        }
    ],
   "measurements": [
        {
            "name": "measurement",
            "config": {
                "poi": "mu",
                "parameters": [
                    { "auxdata": [1.0], "bounds": [[0.5,1.5]], "fixed": false,"inits": [1.0], "name": "lumi", sigmas": [0.1]}
                    { "bounds": [[0,10]], "inits": [1.0], "fixed": false,"name": "mu" },
                    { "bounds": [[-5.0,5.0]], "inits": [0.0],"fixed": false, "name": "correlated_bkg_uncertainty"}
                ]
            }
        }
    ]
}
```

pyhf JSON

Where do constraint terms fit in the spec?

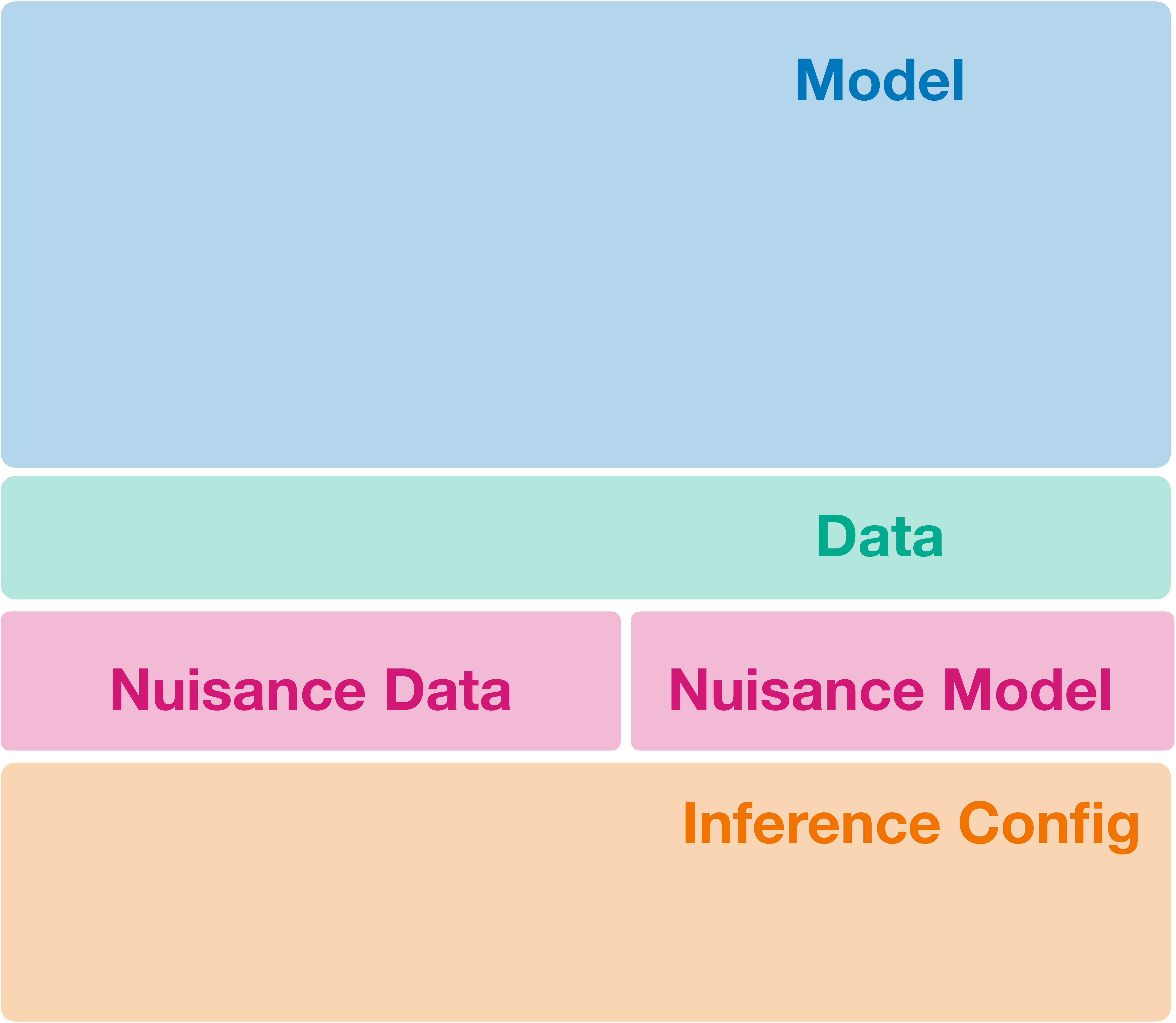
Right now they are part of the inference configuration

a lot of implicit info not directlty ccoded in thte JSON, but published in the literature

Could introduce a separate new nuisance model spec

Bayes: just an additional Model

Freq: Subsidiary Model + "aux data"



Beyond HistFactory

Four decision points to a likelihood spec:

- Binned or unbinned
- What's is the scaffolding/grammar to combine building blocks
- What building blocks exist
- How do you serialize them

HistFactory

binned

$$f(\boldsymbol{n}, \boldsymbol{a} \mid \boldsymbol{\eta}, \boldsymbol{\chi}) = \prod_{c \in \text{channels}} \prod_{b \in \text{bins}_c} \text{Pois}\left(n_{cb} \mid \nu_{cb}(\boldsymbol{\eta}, \boldsymbol{\chi})\right) \prod_{\chi \in \boldsymbol{\chi}} c_\chi(a_\chi \mid \chi)$$

$$\nu_{cb}(\boldsymbol{\phi}) = \sum_{s \in \text{samples}} \nu_{scb}(\boldsymbol{\eta}, \boldsymbol{\chi}) = \sum_{s \in \text{samples}} \underbrace{\left(\prod_{\kappa \in \boldsymbol{\kappa}} \kappa_{scb}(\boldsymbol{\eta}, \boldsymbol{\chi})\right)}_{\text{multiplicative modifiers}} \left(\nu_{scb}^0(\boldsymbol{\eta}, \boldsymbol{\chi}) + \underbrace{\sum_{\Delta \in \boldsymbol{\Delta}} \Delta_{scb}(\boldsymbol{\eta}, \boldsymbol{\chi})}_{\text{additive modifiers}}\right).$$

	Description	Modification	Constraint Term c_χ	Input
constrained	Uncorrelated Shape	$\kappa_{scb}(\gamma_b) = \gamma_b$	$\prod_b \text{Pois}\left(r_b = \sigma_b^{-2} \middle\| \rho_b = \sigma_b^{-2}\gamma_b\right)$	σ_b
	Correlated Shape	$\Delta_{scb}(\alpha) = f_p\left(\alpha \middle\| \Delta_{scb,\alpha=-1}, \Delta_{scb,\alpha=1}\right)$	$\text{Gaus}(a = 0 \| \alpha, \sigma = 1)$	$\Delta_{scb,\alpha=\pm 1}$
	Normalisation Unc.	$\kappa_{scb}(\alpha) = g_p\left(\alpha \middle\| \kappa_{scb,\alpha=-1}, \kappa_{scb,\alpha=1}\right)$	$\text{Gaus}(a = 0 \| \alpha, \sigma = 1)$	$\kappa_{scb,\alpha=\pm 1}$
	MC Stat. Uncertainty	$\kappa_{scb}(\gamma_b) = \gamma_b$	$\prod_b \text{Gaus}\left(a_{\gamma_b} = 1 \middle\| \gamma_b, \delta_b\right)$	$\delta_b^2 = \sum_s \delta_{sb}^2$
	Luminosity	$\kappa_{scb}(\lambda) = \lambda$	$\text{Gaus}(l = \lambda_0 \| \lambda, \sigma_\lambda)$	$\lambda_0, \sigma_\lambda$
free	Normalisation	$\kappa_{scb}(\mu_b) = \mu_b$		
	Data-driven Shape	$\kappa_{scb}(\gamma_b) = \gamma_b$		


```
{
    "version": "1.0.0",
    "channels": [
        {
            "name": "single_channel",
            "samples": [
                {
                    "name": "signal",
                    "data": [5,10],
                    "modifiers": [
                        {"name": "mu", "type": "normfactor","data": null}
                    ]
                },
                {
                    "name": "background",
                    "data": [50,50],
                    "modifiers": [
                        {"name": "correlated_bkg_uncertainty", "type": "histosys", "data": {"hi_data": [45,40],"lo_data": [55,60]}
                    ]
                }
            ]
        }
    ],
    "observations": [
        {
            "name": "single_channel",
            "data": [50,50]
        }
    ],
   "measurements": [
        {
            "name": "measurement",
            "config": {
                "poi": "mu",
                "parameters": [
                    { "bounds": [[0,10]], "inits": [1.0], "fixed": false,"name": "mu" },
                    { "bounds": [[-5.0,5.0]], "inits": [0.0],"fixed": false, "name": "correlated_bkg_uncertainty"}
                ]
            }
        }
    ]
}
```


Model

Data

Inference Config

Beyond HistFactory

Four decision points to a likelihood spec:

	Some imaginary unbinned spec
• **Binned or unbinned**	unbinned
• **What's is the scaffolding/grammar to combine building blocks**	mixtures, products, convolution
• **What building blocks exist**	gaussian, exponential, crystal ball
• **How do you serialize them**	domain specific expression lang

Common Building Blocks across Specs

Data Section can be probably shared formalized for any binned spec if they adopt "channel" semantics

Inference Config Language could be shared depending across specs with same inference type

- frequentist w/ subsidiary measurements
- bayesian w/ priors


```json
{
    "version": "1.0.0",
    "channels": [
        {
            "name": "single_channel",
            "samples": [
                {
                    "name": "signal",
                    "data": [5,10],
                    "modifiers": [
                        {"name": "mu", "type": "normfactor","data": null}
                    ]
                },
                {
                    "name": "background",
                    "data": [50,50],
                    "modifiers": [
                        {"name": "correlated_bkg_uncertainty", "type": "histosys", "data": {"hi_data": [45,40],"lo_data": [55,60]}}
                    ]
                }
            ]
        }
    ],
    "observations": [
        {
            "name": "single_channel",
            "data": [50,50]
        }
    ],
    "measurements": [
        {
            "name": "measurement",
            "config": {
                "poi": "mu",
                "parameters": [
                    { "bounds": [[0,10]], "inits": [1.0], "fixed": false,"name": "mu" },
                    { "bounds": [[-5.0,5.0]], "inits": [0.0],"fixed": false, "name": "correlated_bkg_uncertainty"}
                ]
            }
        }
    ]
}
```


Model

Data

Inference Config

A word on Unfolding

HistFactory is modelling a forward model before inference (together with data + inference config to run inference)

- **does not store inference result, idea is you can re-run it to get it**

$$p(x \,|\, \lambda(\theta))$$

Unfolding is more an inferenec result $p(\lambda \,|\, x), p(\theta \,|\, x) \text{ or } \hat{\lambda}, \hat{\theta}$

- **if inference result is a p.d.f. (i.e. Bayes) the modelling language could be reused, but could require new/other building blocks**
- **if inference result is max lhood, lhood scans, etc would need additional language (some earrly work in cabinetry (A. Held))**

NB:

X ± Y GeV
is a result
not a lhood