The CMS combine tool

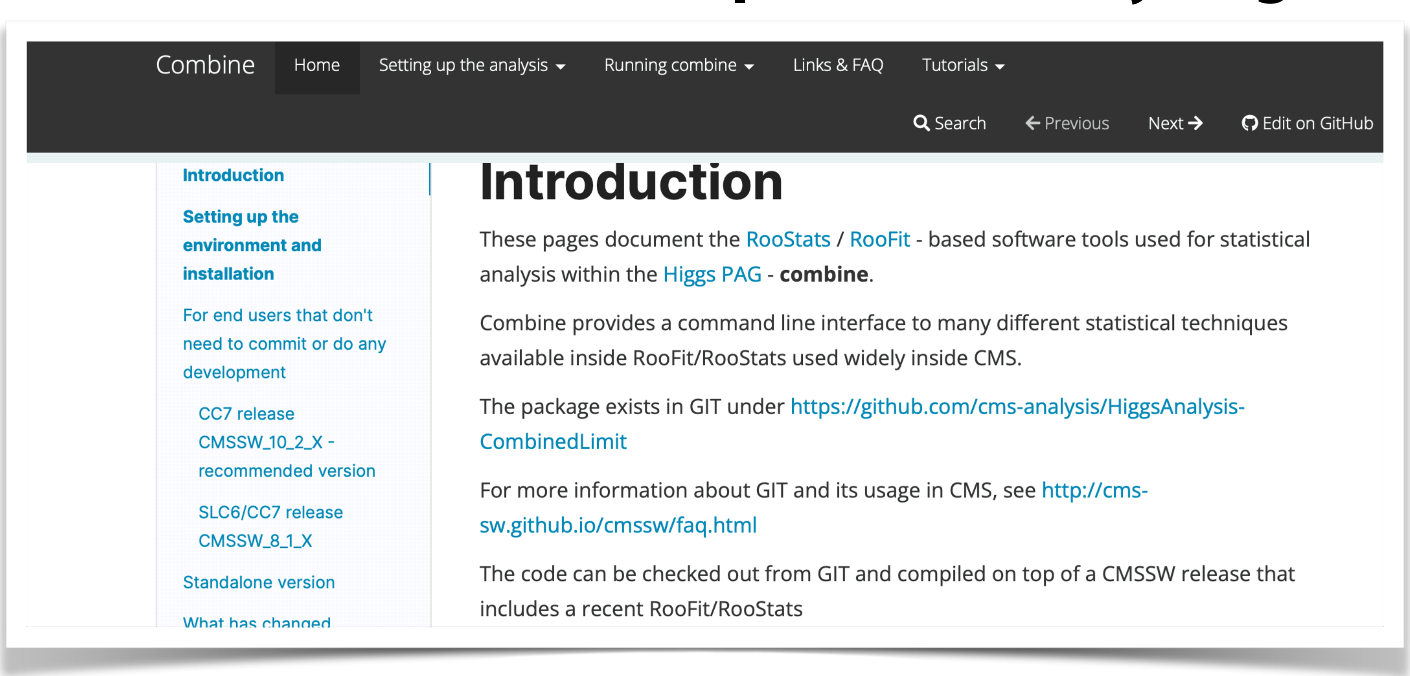
A. Gilbert

Publication of statistical models workshop
9 November 2021

Introduction

- The combine tool is the primary software framework used for statistical model building & inference in CMS physics analysis
  - Developed for Higgs analysis in Run 1, now used in all physics areas
- Built on top of ROOT and RooFit
  - Likelihood is persisted in a RooFit workspace
  - Input based on plain text "datacards"
- While combine is developed for CMS analysis, and with CMS users in mind, the code is public and can be compiled in "standalone" mode
  - An extensive manual is provided, along with links to tutorials and examples:

https://cms-analysis.github.io/HiggsAnalysis-CombinedLimit/



Combine · Home · Setting up the analysis · Running combine · Links & FAQ · Tutorials

Search · Previous · Next · Edit on GitHub

Introduction

Setting up the environment and installation

For end users that don't need to commit or do any development

CC7 release CMSSW_10_2_X - recommended version

SLC6/CC7 release CMSSW_8_1_X

Standalone version

What has changed

Introduction

These pages document the RooStats / RooFit - based software tools used for statistical analysis within the Higgs PAG - **combine**.

Combine provides a command line interface to many different statistical techniques available inside RooFit/RooStats used widely inside CMS.

The package exists in GIT under https://github.com/cms-analysis/HiggsAnalysis-CombinedLimit

For more information about GIT and its usage in CMS, see http://cms-sw.github.io/cmssw/faq.html

The code can be checked out from GIT and compiled on top of a CMSSW release that includes a recent RooFit/RooStats

Typical combine workflow

Typical combine workflow

- Text datacard for a single "channel"
  - In this case a one bin counting experiment
- Each channel and process has a unique label:

datacard.txt

Number of bins/channels

Number of processes

Number of nuisance parameters (*:determined automatically)

```
imax    1 number of bins
jmax    4 number of processes minus 1
kmax    * number of nuisance parameters
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
bin          signal_region
observation  10.0
--------------------------------------------------------------------------------
bin                          signal_region  signal_region  signal_region  signal_region  signal_region
process                      ttbar          diboson        Ztautau        jetFakes       bbHtautau
process                      1              2              3              4              0
rate                         4.43803        3.18309        3.7804         1.63396        0.711064
--------------------------------------------------------------------------------
CMS_eff_b            lnN     1.02           1.02           1.02           -              1.02
CMS_eff_t            lnN     1.12           1.12           1.12           -              1.12
CMS_eff_t_highpt     lnN     1.1            1.1            1.1            -              1.1
acceptance_Ztautau   lnN     -              -              1.08           -              -
acceptance_bbH       lnN     -              -              -              -              1.05
acceptance_ttbar     lnN     1.005          -              -              -              -
lumi_13TeV           lnN     1.025          1.025          1.025          -              1.025
norm_jetFakes        lnN     -              -              -              1.2            -
xsec_Ztautau         lnN     -              -              1.04           -              -
xsec_diboson         lnN     -              1.05           -              -              -
xsec_ttbar           lnN     1.06           -              -              -              -
```

Unique channel label

Number of observed events in channel

Process label

Process ID (<=0 for signal)

Expected number of events

Systematic uncertainties

Name

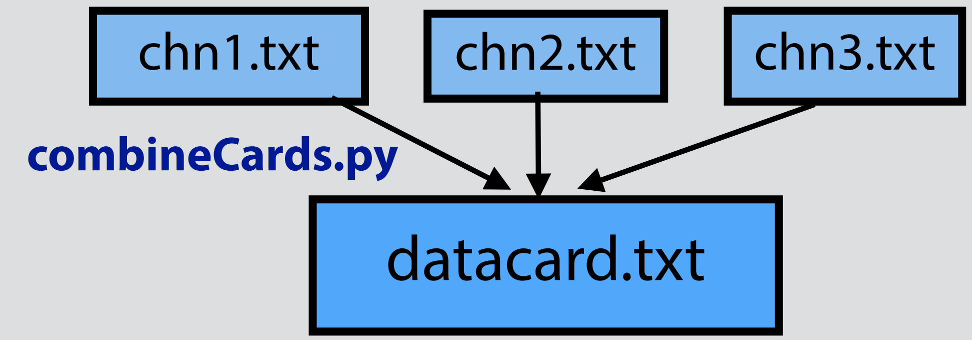
Type

Effect on process

Normalisation uncertainties, with a log-normal constraint pdf

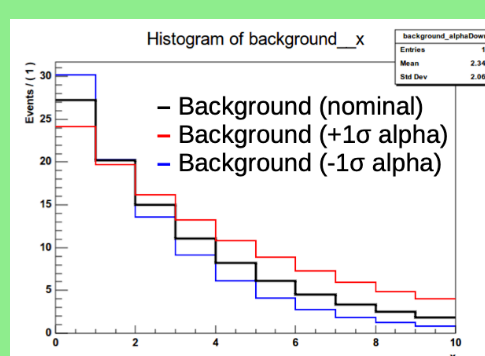
Typical combine workflow

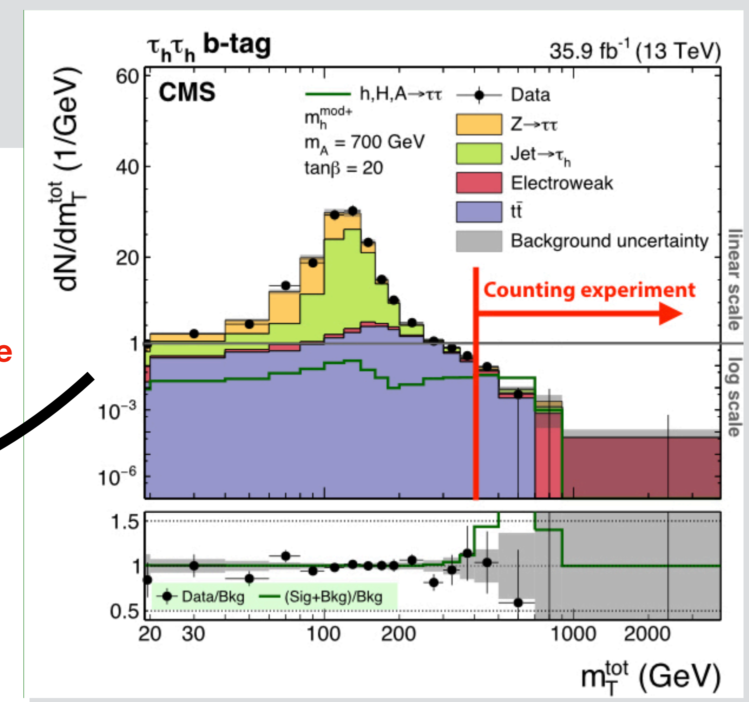
- Datacards can describe multiple channels
- Separate cards can be merged using the **combineCards.py** script
- Each channel can also represent a distribution
  - "shapes" directives link to input ROOT histograms for binned analyses:




```
imax 1
jmax 1
kmax *
---------------
shapes * * simple-shapes-TH1_input.root $PROCESS $PROCESS_$SYSTEMATIC
shapes signal * simple-shapes-TH1_input.root $PROCESS$MASS $PROCESS$MASS_$SYSTEMATIC
---------------
bin bin1
observation 85
------------------------------
bin             bin1        bin1
process         signal      background
process         0           1
rate            10          100
--------------------------------
lumi     lnN    1.10        1.0
bgnorm   lnN    1.00        1.3
alpha  shape    -           1
```

Links to the histograms saved in a root file





Shape uncertainty: 2 additional histograms per process supplied, with ±1σ shift

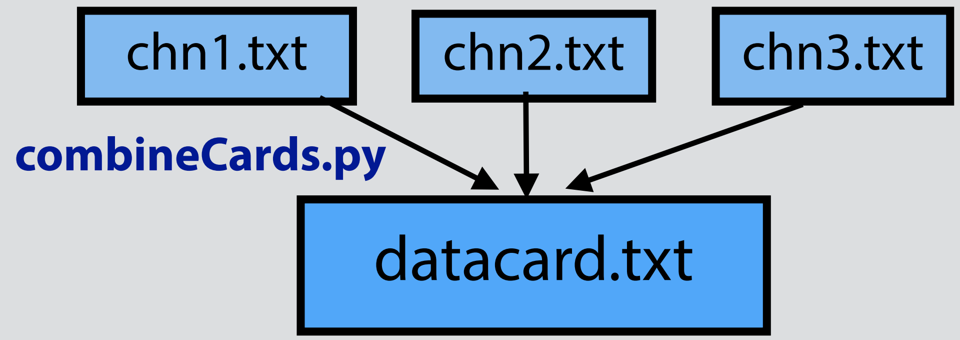
Shape uncertainties: per-bin interpolation of yield fractions between nominal, "up" and "down" templates:

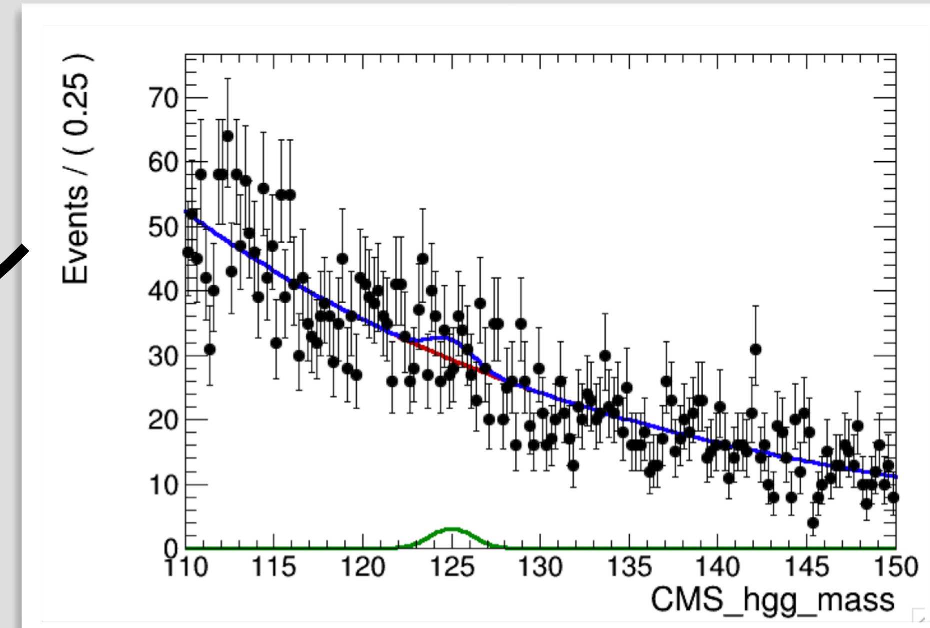
$$f(\theta) = \frac{1}{2}\left((\delta^+ - \delta^-)\theta + \frac{1}{8}(\delta^+ + \delta^-)(3\theta^6 - 10\theta^4 + 15\theta^2)\right)$$

Typical combine workflow

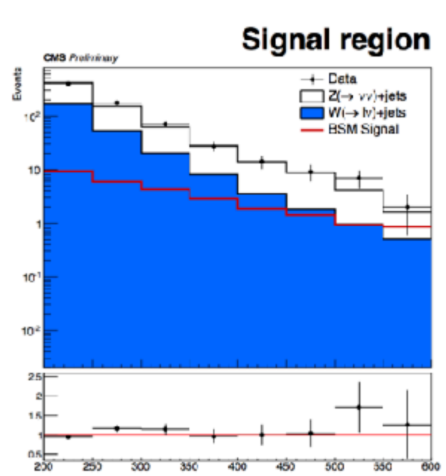
- Also possible to import any arbitrary binned/ unbinned RooFit pdfs
- Shape and normalisation systematics can be added in the same way

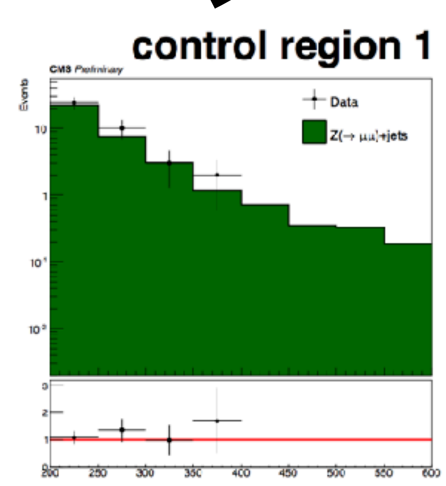
```
imax 1
jmax 1
kmax *
---------------
shapes * * simple-shapes-parametric_input.root w:$PROCESS

bin bin1
observation -1
------------------------------
bin            bin1        bin1
process        sig    bkg
process        0           1
rate           1           1
--------------------------------
lumi    lnN  1.1          1.0
vogian_sigma    param 1.0       0.1
```





Parametric signal and background functions

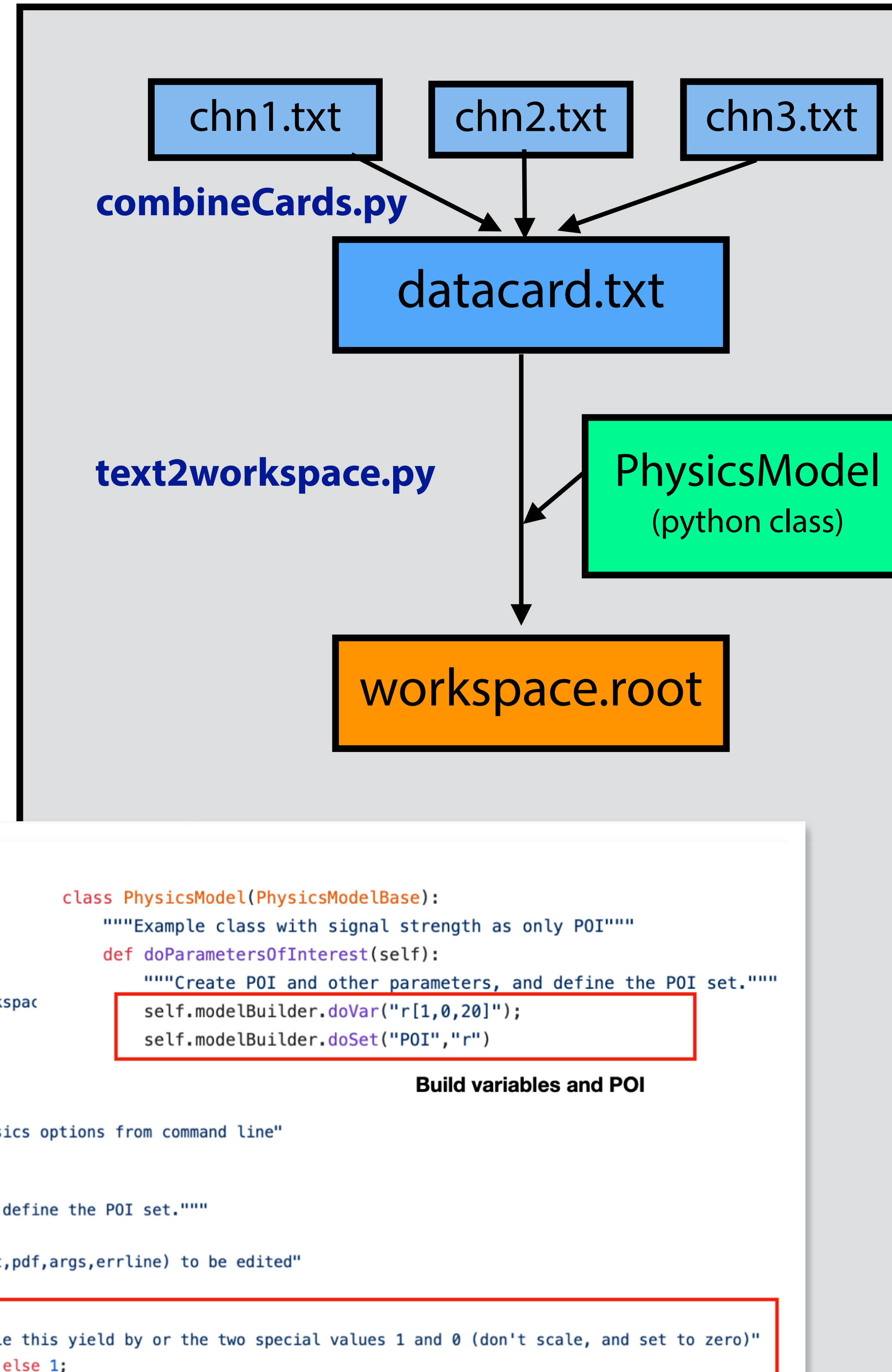




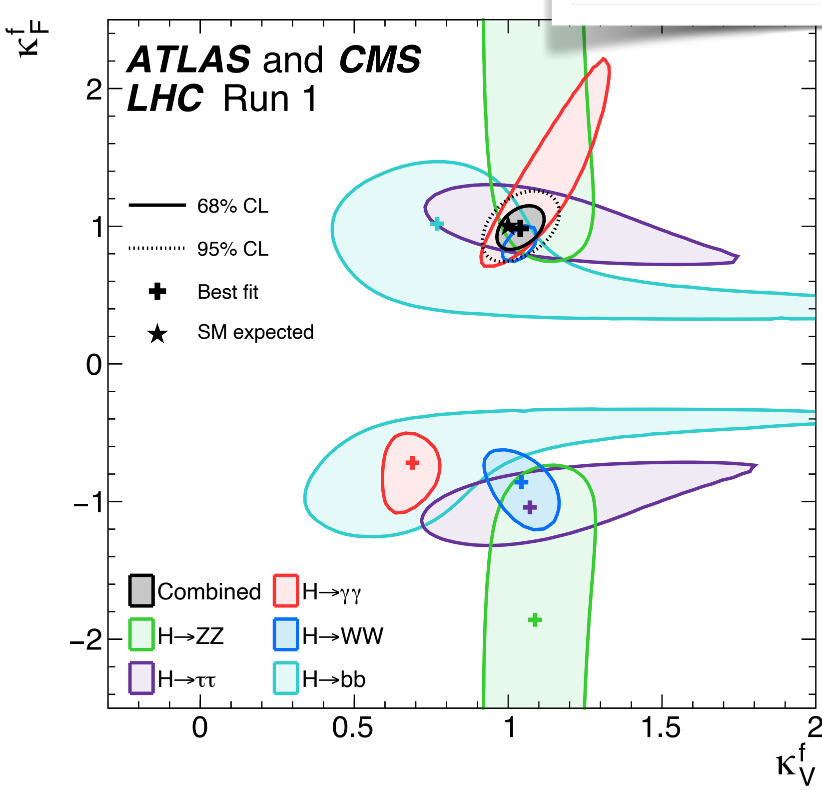
Histograms with parametric bin contents (e.g. signal region bins modelled as functions of control region bins)

Typical combine workflow

- The **text2workspace.py** script converts the datacard into a self-contained RooFit workspace
- Also introduces a "physics model"
  - By default, adds a floating parameter "r" that multiples the normalisation of all processes marked as signal in the datacard
  - Customised models can be applied by providing a simple extension of the **PhysicsModel** class
  - E.g. coupling modifier parameterisation of Higgs processes:



$$\kappa_j^2 = \sigma_j/\sigma_j^{\mathrm{SM}} \quad \text{or} \quad \kappa_j^2 = \Gamma^j/\Gamma^j_{\mathrm{SM}},$$




```
class PhysicsModelBase(object):
    __metaclass__ = ABCMeta
    def __init__(self):
        pass
    def setModelBuilder(self, modelBuilder):
        "Connect to the ModelBuilder to get workspac
        self.modelBuilder = modelBuilder
        self.DC = modelBuilder.DC
        self.options = modelBuilder.options
    def setPhysicsOptions(self,physOptions):
        "Receive a list of strings with the physics options from command line"
    @abstractmethod
    def doParametersOfInterest(self):
        """Create POI and other parameters, and define the POI set."""
    def preProcessNuisances(self,nuisances):
        "receive the usual list of (name,nofloat,pdf,args,errline) to be edited"
        pass # do nothing by default
    def getYieldScale(self,bin,process):
        "Return the name of a RooAbsReal to scale this yield by or the two special values 1 and 0 (don't scale, and set to zero)"
        return "r" if self.DC.isSignal[process] else 1;
```

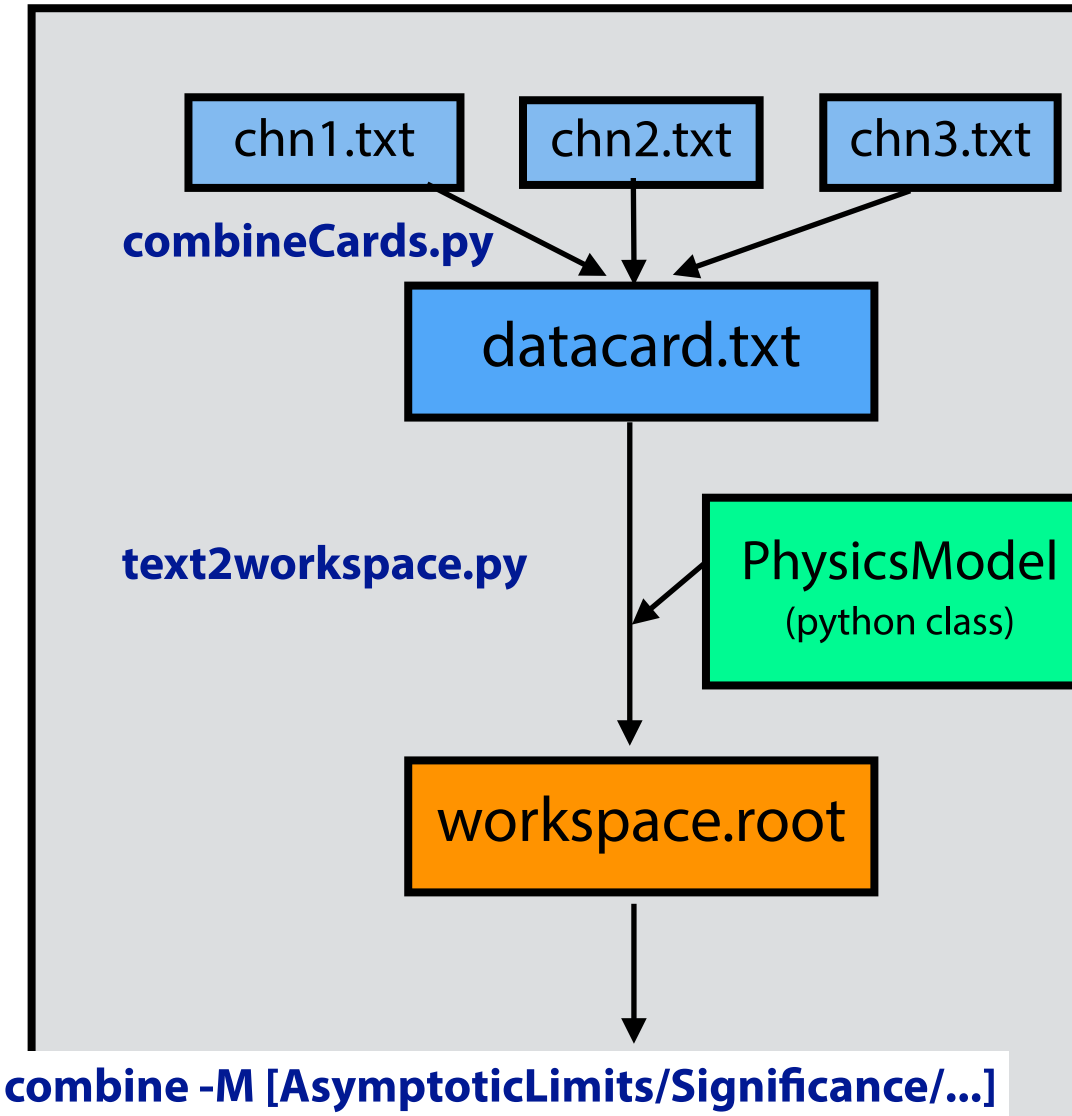
```
class PhysicsModel(PhysicsModelBase):
    """Example class with signal strength as only POI"""
    def doParametersOfInterest(self):
        """Create POI and other parameters, and define the POI set."""
        self.modelBuilder.doVar("r[1,0,20]");
        self.modelBuilder.doSet("POI","r")
```

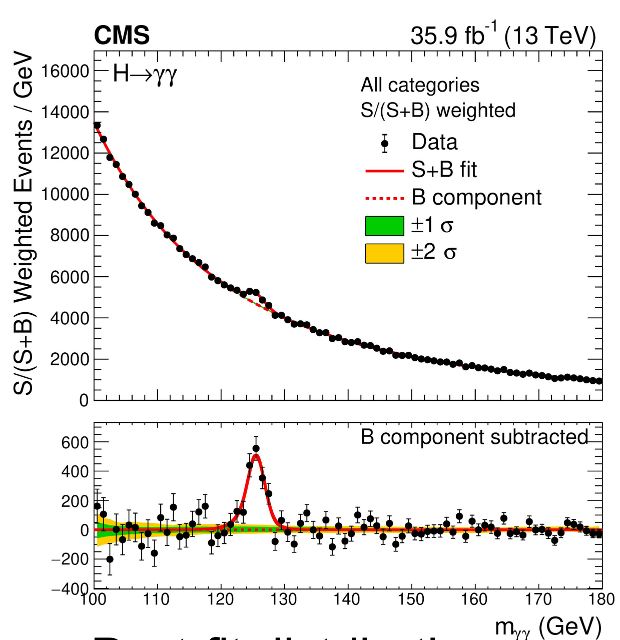
Build variables and POI

How yield gets scaled

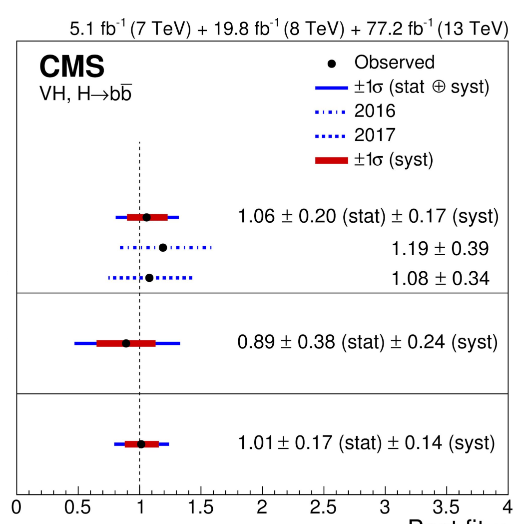
Typical combine workflow

- The combine program implements the commonly used statistical methods
  - Limit setting (asymptotic and toy based)
  - Significance / p-value calculation
  - Confidence intervals
- All methods can run on real data or internally generated toys/Asimov datasets
- Also diagnostics and model information
  - Pre-/post-fit yields, shapes and uncertainties
  - Covariance matrices
- Output in ROOT file format

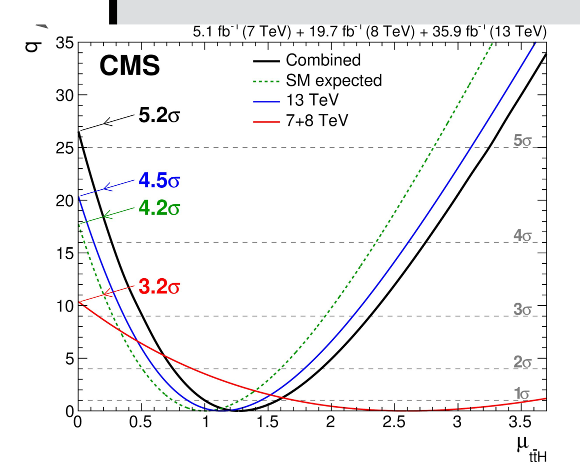




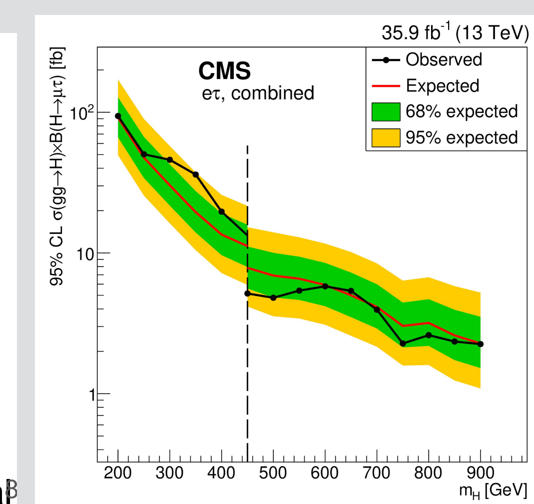
Post-fit distribution



Confidence interval



Significance / confidence interval

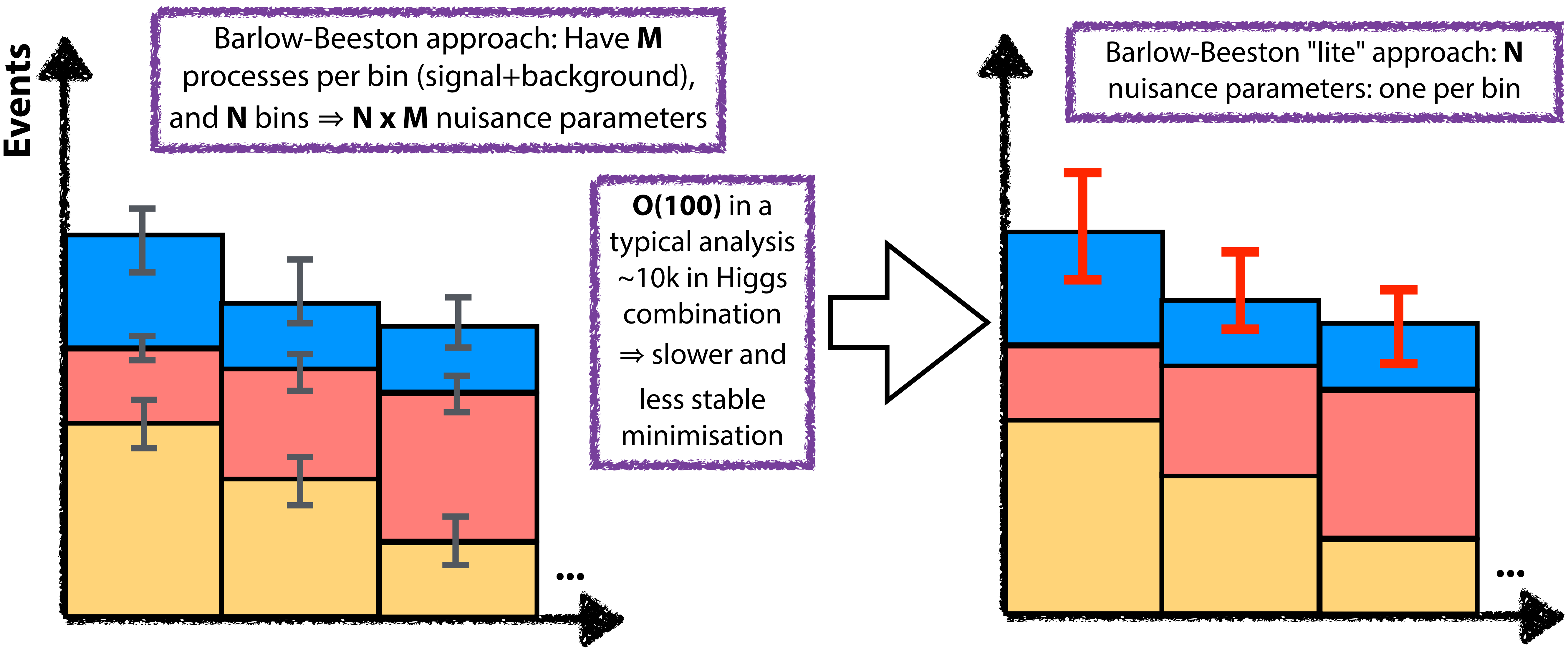


Upper limit

(Non exhaustive) summary of other features

MC statistical uncertainties

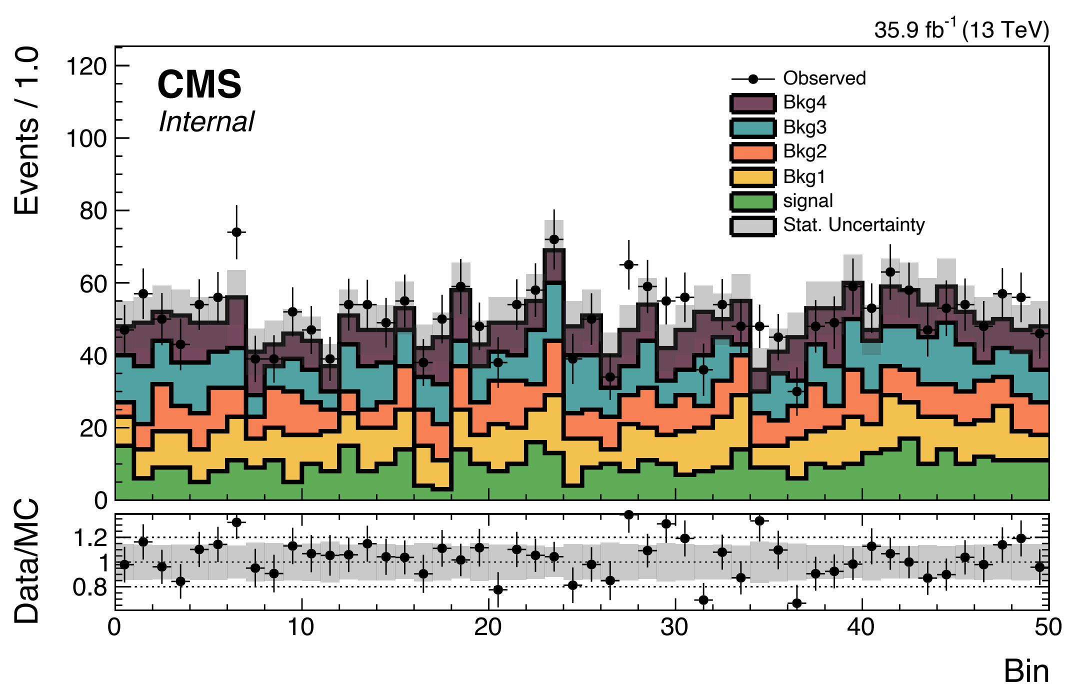
- **autoMCStats:** A feature in combine for incorporating uncertainties due to finite event counts in templates
- Full documentation here, more background in [Barlow, Beeston '93] [Conway '11]
- Automatically models total uncertainty in each bin with a single Gaussian ("lite" approach)
  - Analysts only have to add a single line in the datacard to enable
  - Falls back to per-process Poisson if MC stats too low in any particular bin

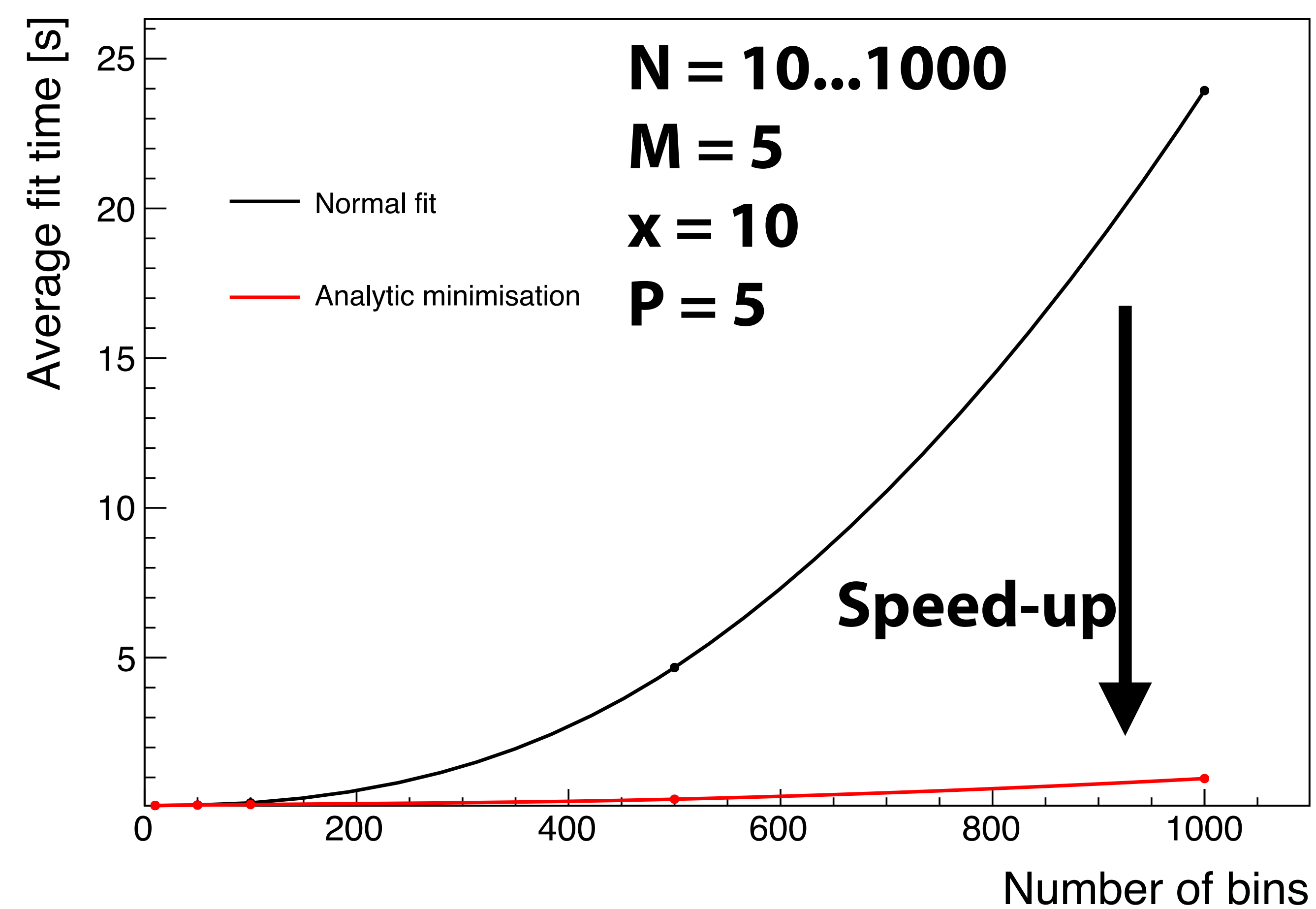


MC statistical uncertainties

- There is no pruning of uncertainties in this implementation (too error prone) - there will be one nuisance parameter for every populated bin
  - Fitting time can still be long if many bins
- But with the lite approach the maximum likelihood for each parameter is independent of the others and has a simple form that we can solve
- The custom minimizer in combine handles the analytic minimisation of these parameters

- Large speed-up possible compared to using normal numeric minimisation:

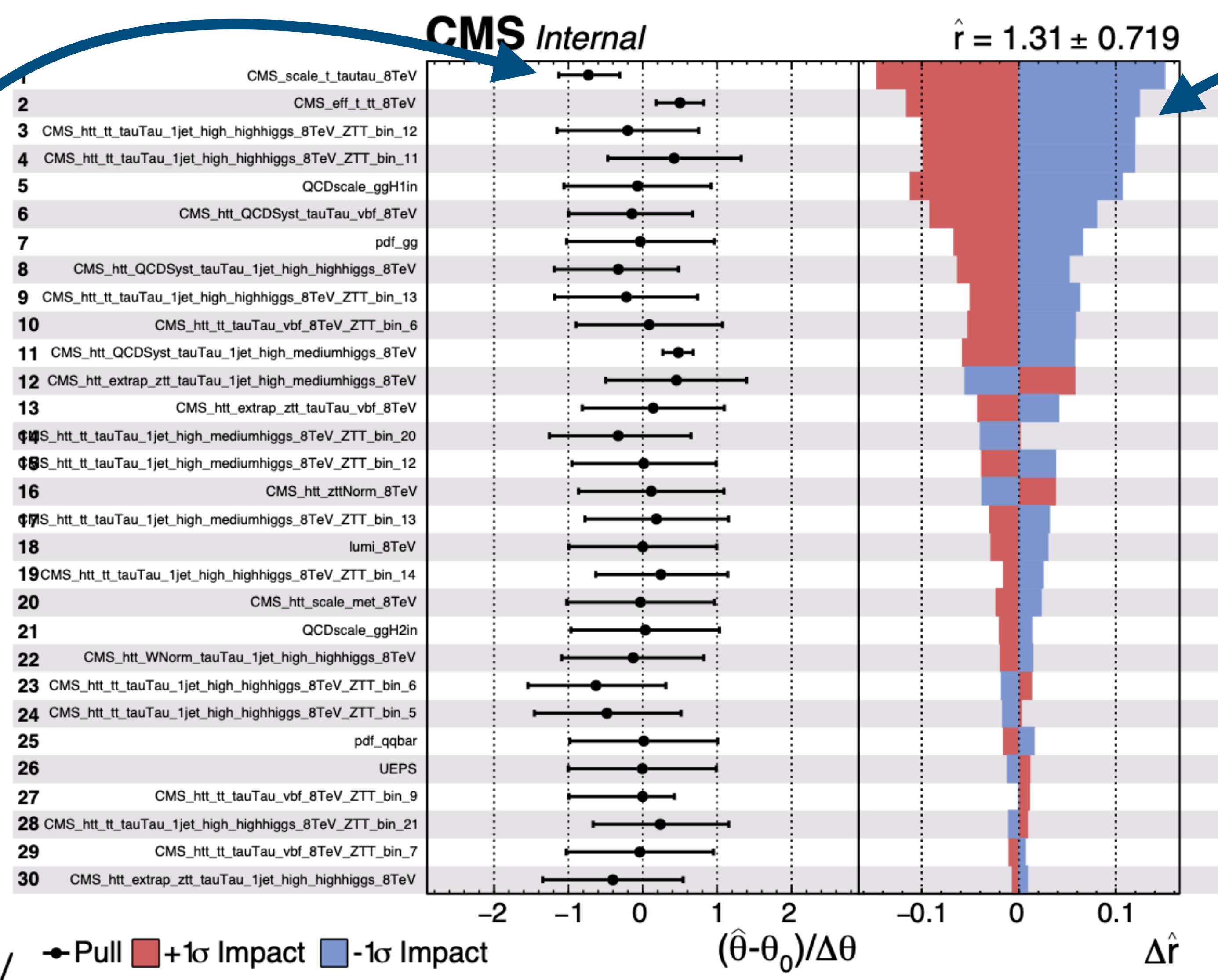




Nuisance parameter impacts

- Combine automates the calculation of **impacts** for the nuisance parameters
  - Define the **impact** of a nuisance parameter on the POI as the shift in the POI that is induced as the NP is fixed and brought to its +1σ or -1σ post-fit values

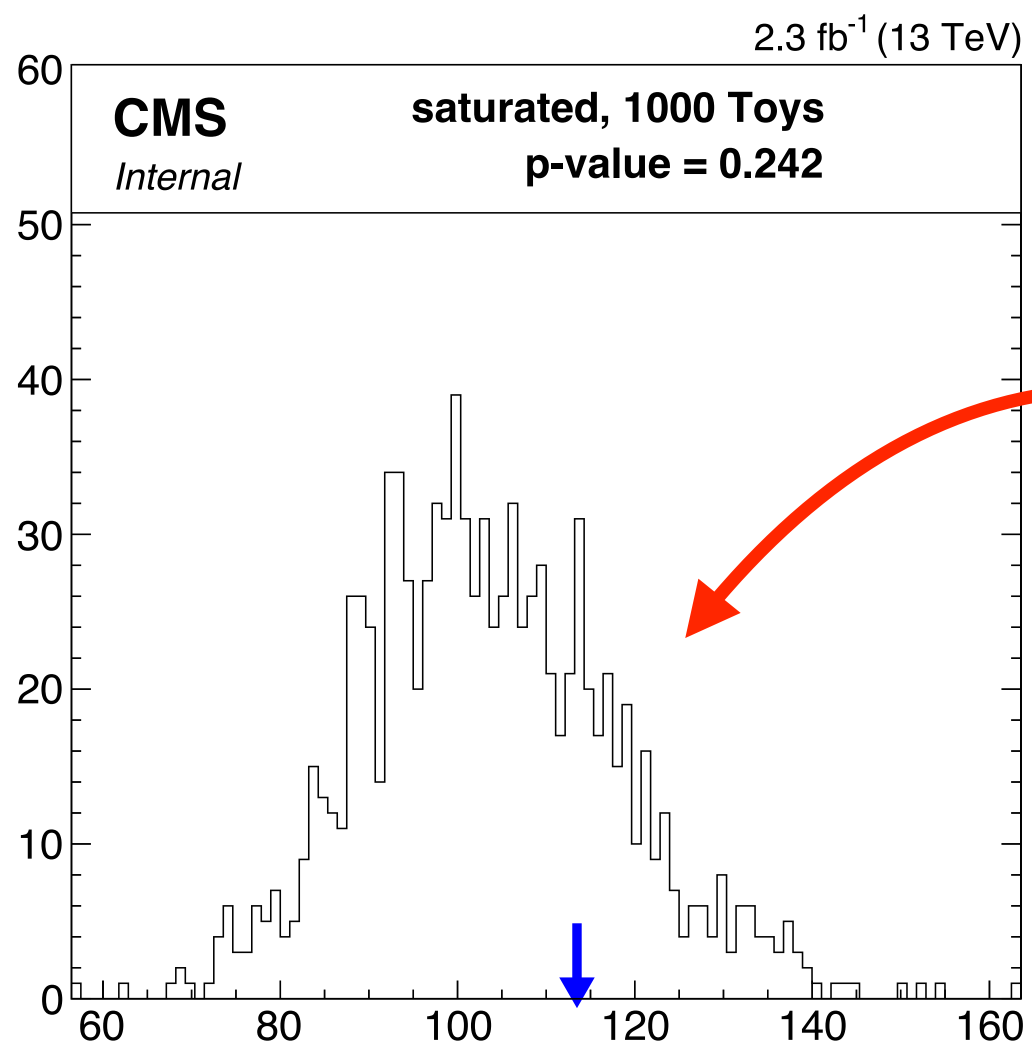
Also see the parameter constraint relative to the input uncertainty



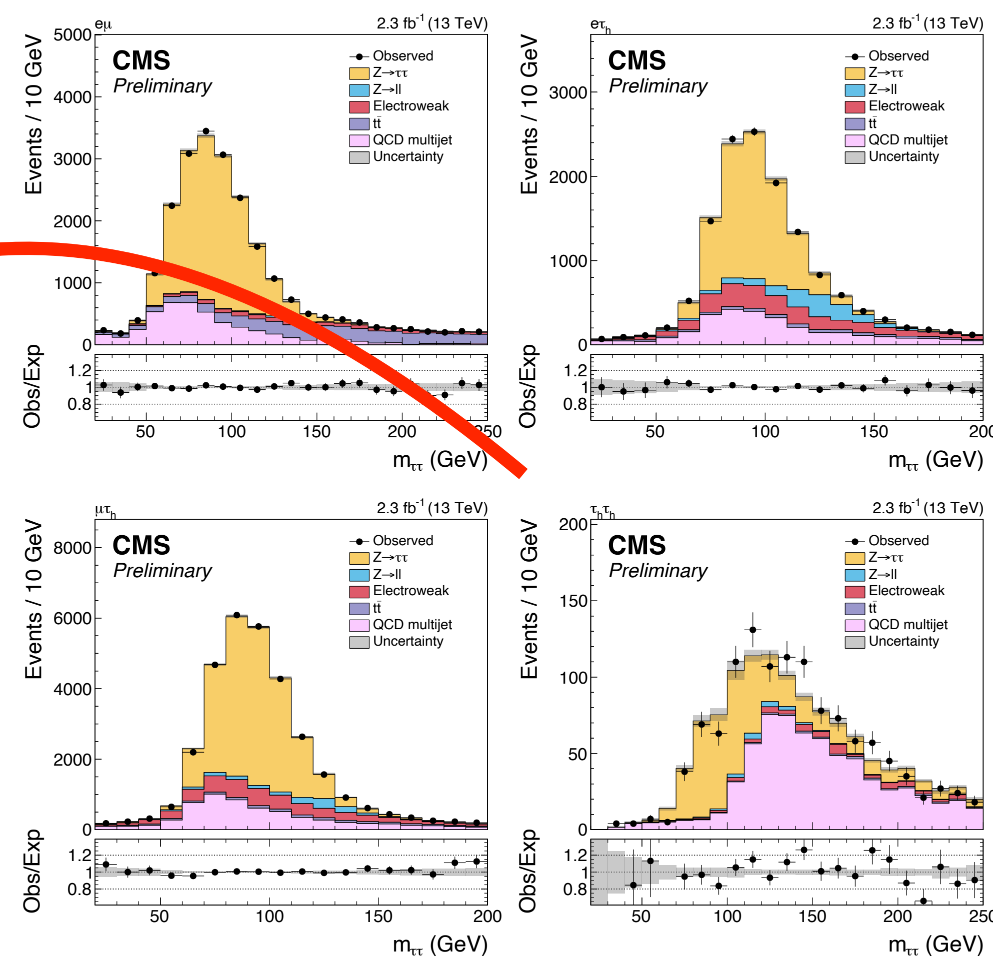
Size of the bar ∝ Impact on “r”

Goodness-of-fit

- Support for calculating saturated model, Kolmogorov-Smirnov and Anderson-Darling test statistics
  - Combine's toy generation routines used for building up expected distributions

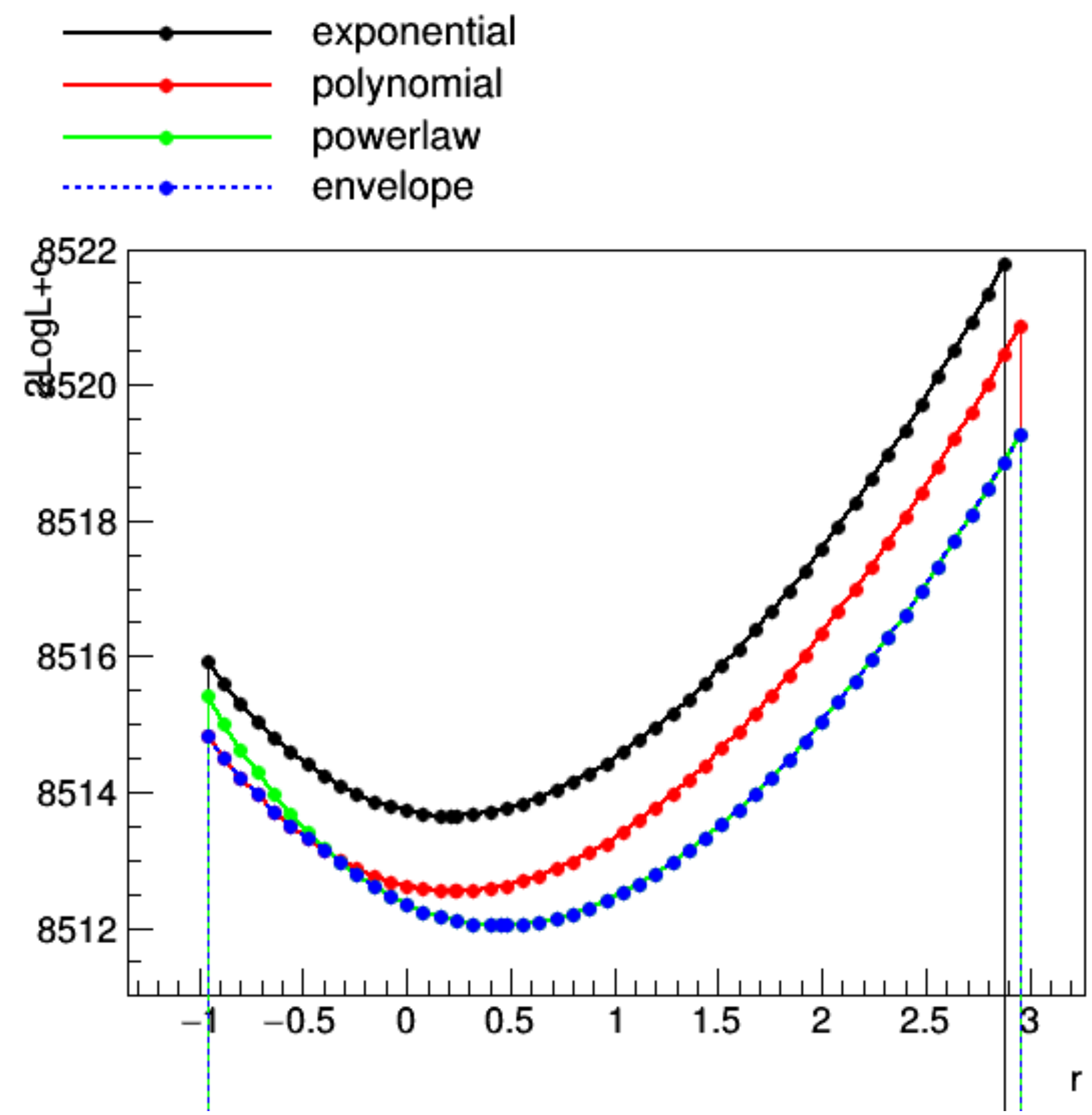


Saturated model: $-2\ln\lambda = 2\sum_i f_i - d_i + d_i \ln(d_i/f_i).$



Discrete profiling

- Method first proposed in https://arxiv.org/abs/1408.6865
- Introduces discrete nuisance parameters (implemented via RooCategory) that correspond to the choice of pdf for a given process (RooMultiPdf)
- Allow the discrete parameter to vary in the maximum likelihood fit
  - Gives an uncertainty due to uncertainty on the choice of PDF functional form
  - Can be considered an alternative to traditional "spurious signal" approach
- NB: Minuit does not support fitting for discrete parameters
  - Handled directly by combine



Unfolding

- The physics model flexibility makes it straightforward to perform unfolding of distributions
  - Datacard processes should be defined in terms of fiducial bins
  - Max. likelihood fit for normalisations in unfolded space
    - Takes the place of traditional matrix inversion

$$\chi^2 = \left(\vec{x}_{\mathrm{reco}} - \vec{b} - \mathbf{R}\vec{\mu}\right)^{\mathrm{T}} \mathbf{\Sigma}^{-1} \left(\vec{x}_{\mathrm{reco}} - \vec{b} - \mathbf{R}\vec{\mu}\right)$$

$$\Downarrow$$

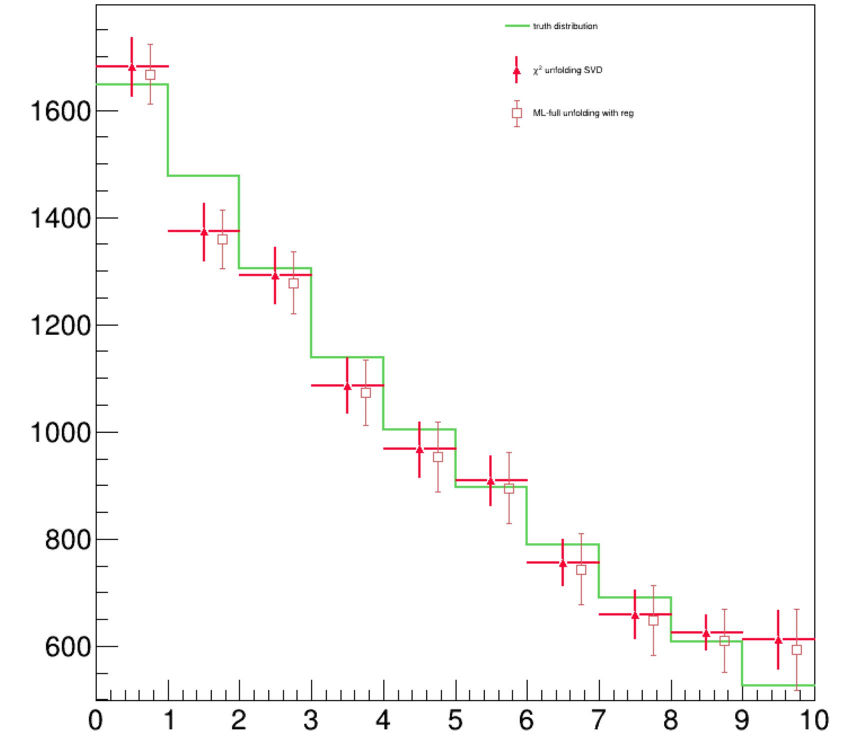
$$\mathcal{L} = \prod_{i\in\mathrm{reco}} \mathcal{P}(x_{\mathrm{reco},i} | \sum_{j\in\mathrm{gen}} \mu_j \mathbf{R}_{ij} + \vec{b}_i)$$

- Possible to add penalty term to the likelihood to perform regularisation
  - Flexible datacard syntax to introduce constraints

```
name constr @0-2*@2+@1 r_Bin0,r_Bin1,r_Bin2 0.03
```

$$-2\log\mathcal{L} = -2\log\mathcal{L}_{\mathrm{stat}} + \tau||\mathbf{L}\cdot\vec{\mu}||^2$$

$$\mathcal{L} = \mathcal{L}_{\mathrm{stat}} \cdot \mathcal{N}(\mathbf{L}\vec{\mu}|1,\delta) \cdot \ldots$$



Discussion points

Preserving the full likelihood

- General likelihood:

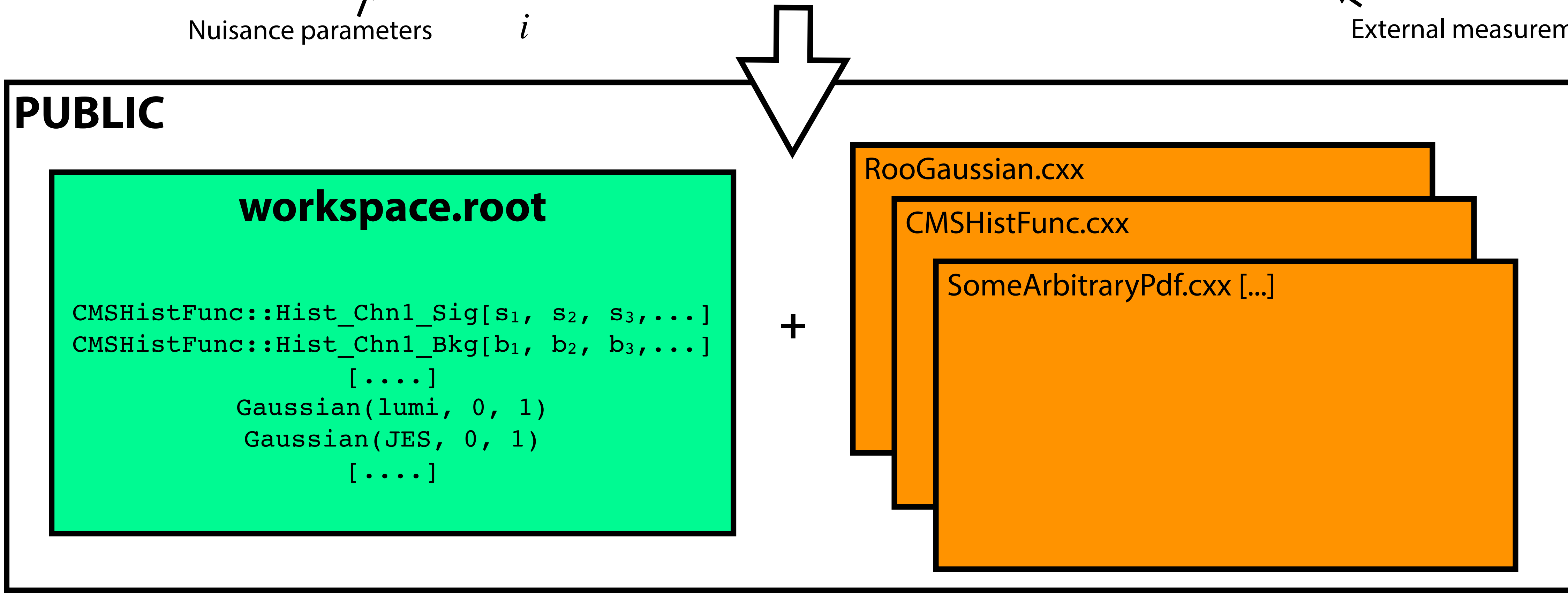
Parameters of interest

Signal and background pdfs

$$\mathscr{L}(\text{data} \mid \vec{\alpha}, \vec{\theta}) = \prod_i \text{Poisson}(n_i \mid s_i(\vec{\alpha}, \vec{\theta}) + b_i(\vec{\theta}))\, p(\vec{\tilde{\theta}} \mid \vec{\theta})$$

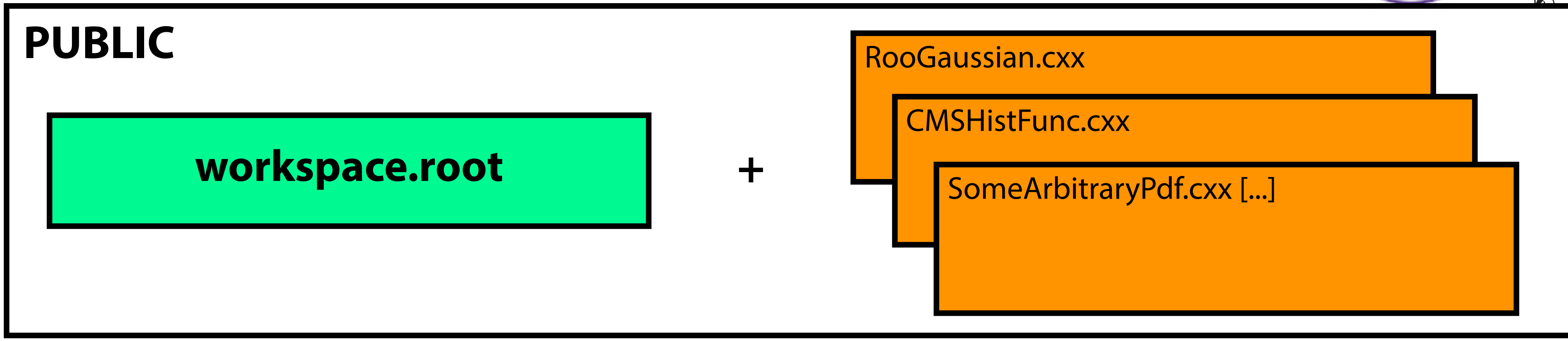
Nuisance parameters

External measurement pdfs



- Natural division between:
  - Input values specific to the analysis (observed data, list of pdfs, pdf input data...) ⇒ **Workspace**
  - General specifications of pdfs that define s_i, b_i, and $p(\theta)$ ⇒ **C++ class definitions**
- **Both must be made public to claim we have "published the full likelihood"**

Preserving the full likelihood



- Some thoughts on use cases. I want to...

[A] **Inspect** the full form of the **likelihood**
  - Requires reading code and/or comments, but possible to extract full definition and reimplement

[B] **Evaluate** the **likelihood** as a function of all **POIs** and **NPs**
  - Can treat the above as a black box, with external handles for setting the parameter values

[C] **Evaluate** the **profiled likelihood** as a function of the **POIs**
  - Can treat the above as a black box, with handles for the POIs, and some minimizer algo provided

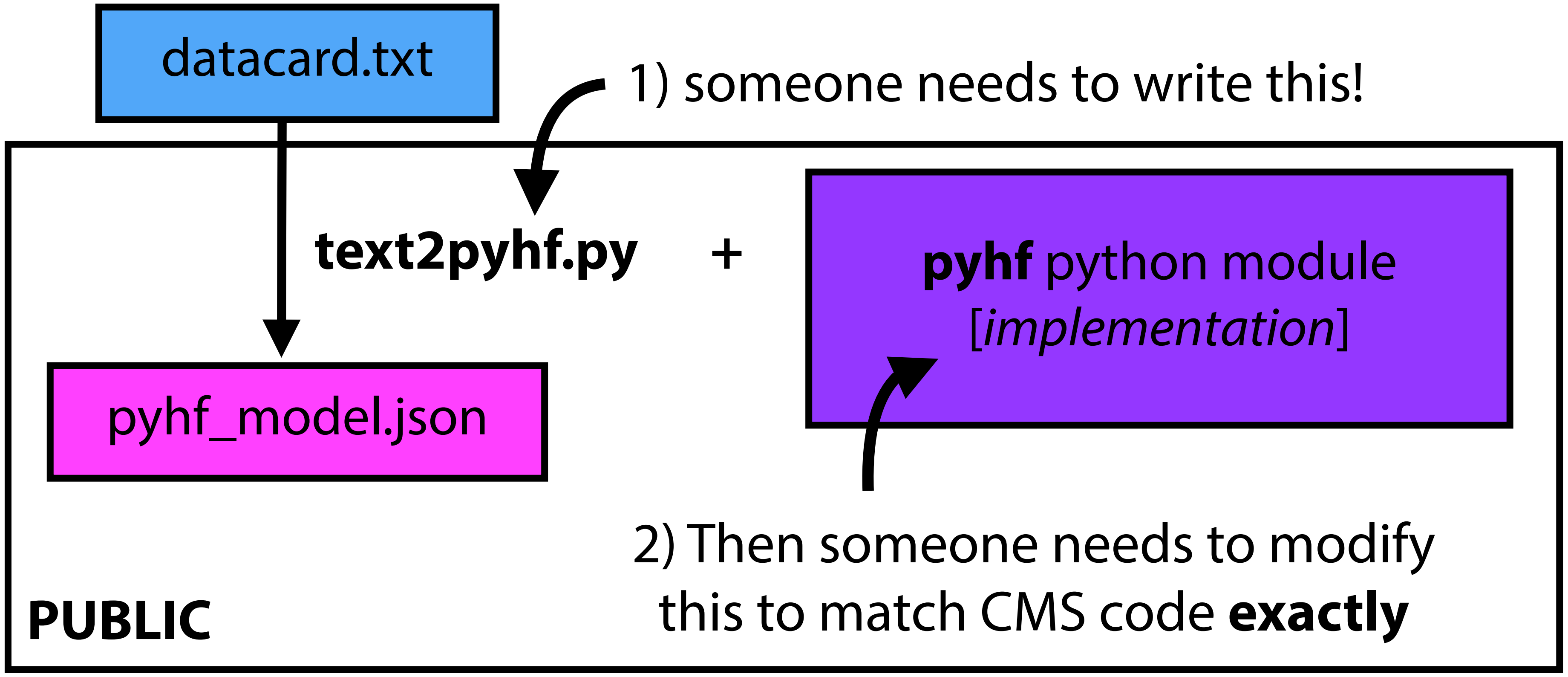
[D] **Evaluate** the **profiled likelihood** as a function of **reparametrised POIs**
  - As above, but take diff. or Higgs STXS measurement cross sections σ_i, reparametrize in coupling modifiers or EFT coefficients

[E] **Combine likelihoods** from multiple analyses
  - Possible (done by experiments in some cases), but requires care - may be incompatibilities

[F] **Modify** the **(s+b) PDF(s)**
  - E.g. to add a different signal prediction. Possible, but RooFit manipulation can be non-trivial (esp. without expts. providing more useful wrapper tools)

Serialising combine models

- Could pyhf be used?
  - The combine and HistFactory/pyhf feature sets are roughly similar
    - Close enough that a basic converter from datacards to pyhf JSON format should not to too difficult
    - Harder to make the pyhf likelihood **exactly** equivalent to the combine one (and if not identical, the likelihood is not preserved)
  - Some things (MC stat uncertainties) are definitely handled differently... other things (e.g. shape morphing) may appear to be the same, but subtle details may differ
  - Unclear if other commonly used features available (e.g. writing bin contents for some processes as generic formulae (RooFormulaVars))



Differences to HistFactory

- Disclaimer: I am not a HistFactory expert - observations are based on public documentation, not detailed comparison of the codes

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

- **Uncorrelated shape:** for single-bin counting channels (gmN), for shapes, RooParametricHist with CR
- **Correlated shape:** unclear if default CMS interpolation available (6th order poly interp. + linear extrap)
- **Normalisation:** CMS lnN with single value [u] : $\kappa = u^{\alpha}$, with asymmetric [d]/[u], $f(\alpha,d,u)^{\alpha}$, where f interpolates between log(u) and log(d)
- **MC Stat. uncertainty:** HF approach similar for combine Barlow-Beeston lite (δ_b^2 updated dynamically)
- **Luminosity**: not commonly used (treated with lnN)
- **Normalisation:** OK
- **Data-driven shape:** RooParametric hist