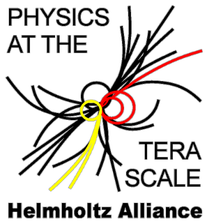
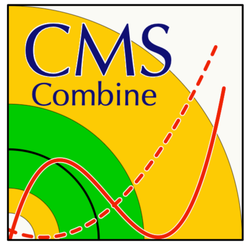
Combine overview

Kyle Cormier (UZH), Aliya Nigamova (UHH) and Nicholas Wardle (IC)

Terascale Statistics School | 02/04/2024

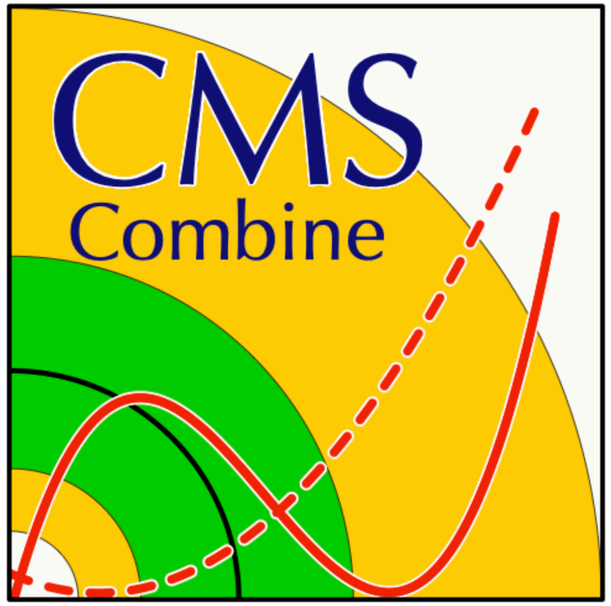
Imperial College London





Introduction

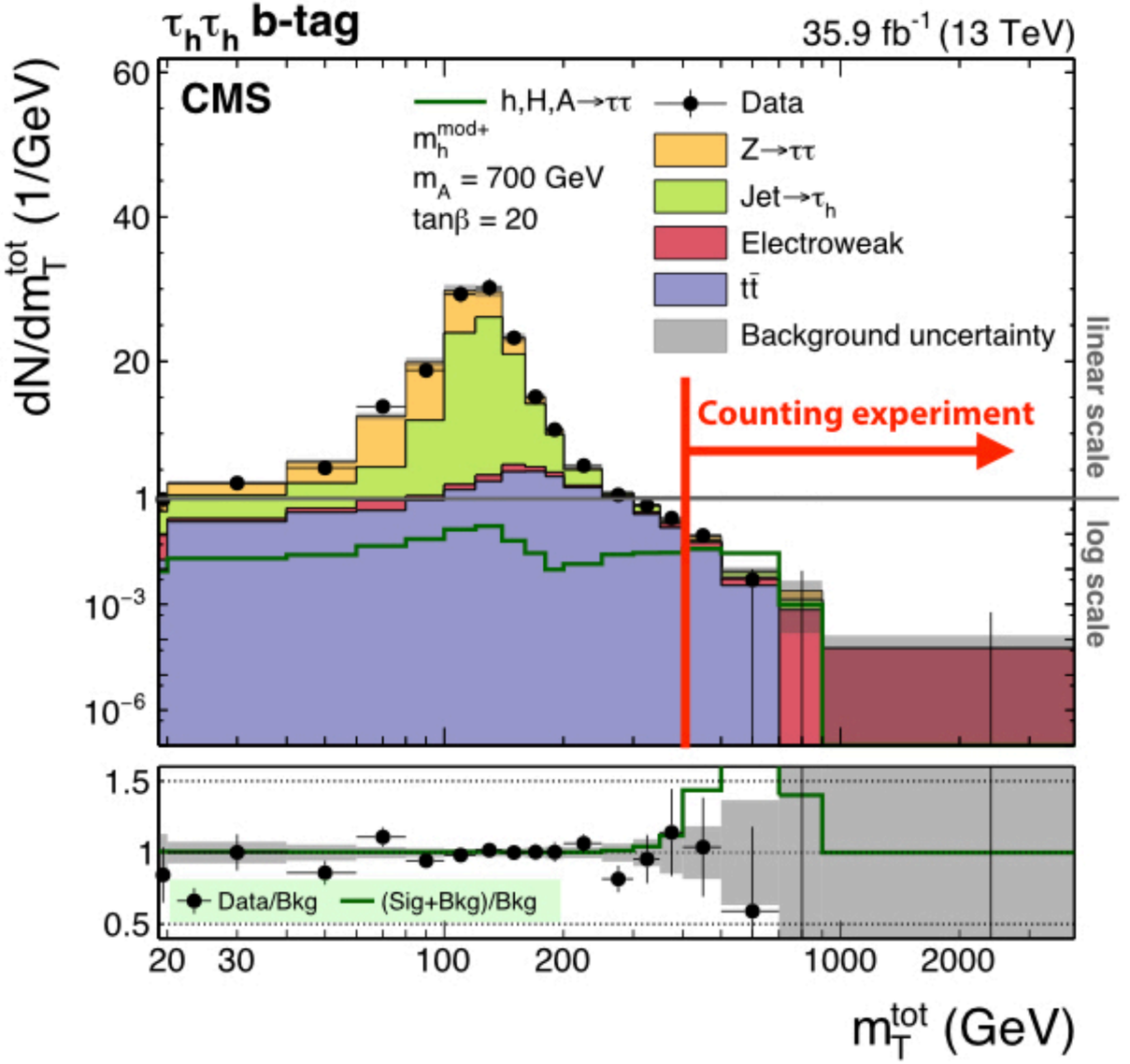
- Combine - the most popular tool for statistical inference in CMS
- Command-line interface to RooStats and RooFit methods, and even more:
  - Builds statistical model
  - Runs statistical test
  - Provides tool set for validation
- GitHub page: https://github.com/cms-analysis/HiggsAnalysis-CombinedLimit



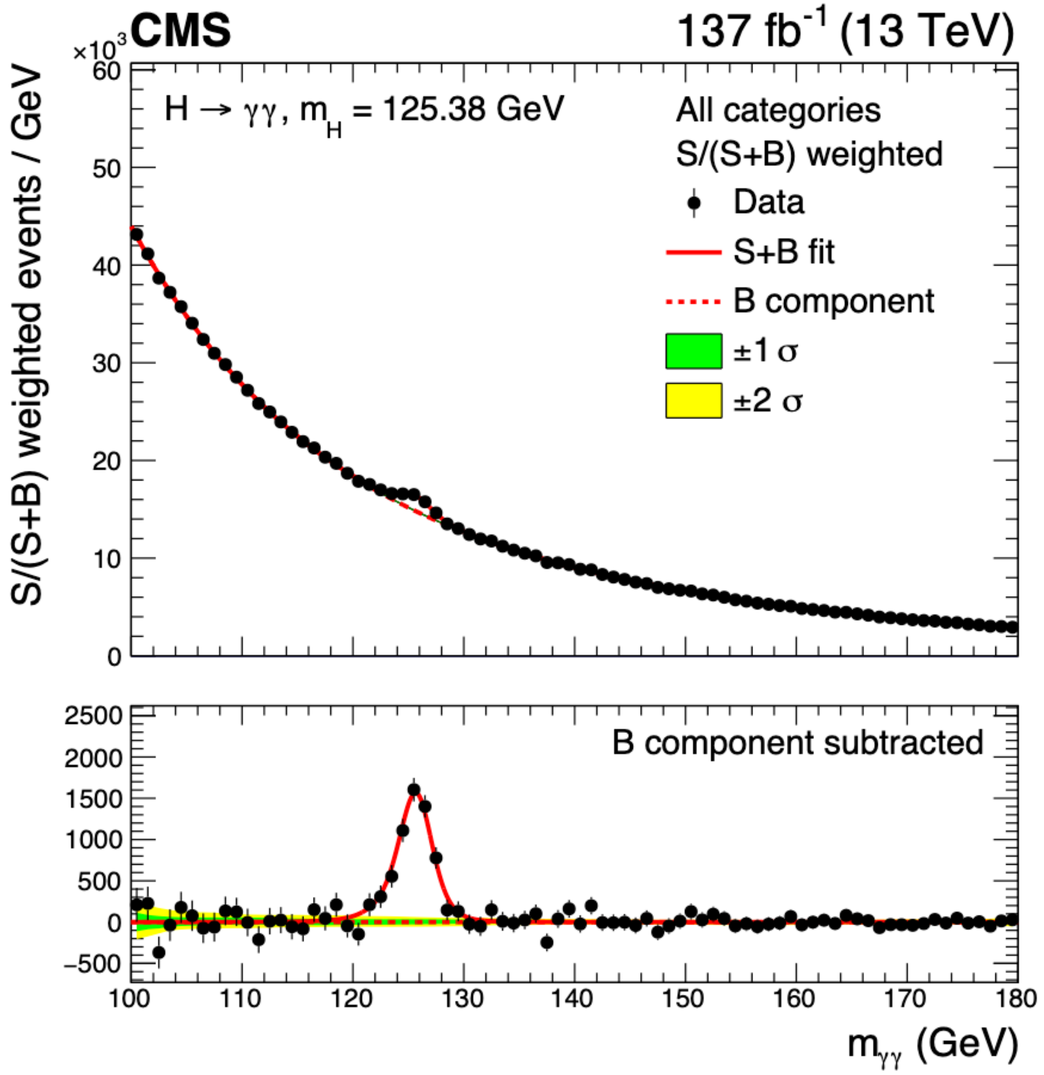
Analysis types: input

Template based model:

- When no simple analytic form can be assigned for the signal(background) description
- Templates (histograms) are provided for all signal and background processes and shape systematic uncertainties
- MC statistical uncertainties are modelled with Barlow-Beeston-lite approach (autoMCStats)



Analysis types: input



Parametric model:

- Suitable for the analysis with analytically described bkg and signal: e.g. Gaussian signal on smoothly falling polynomial background
- In most cases used in the analyses with data-driven bkg description
- The systematic uncertainties assigned on the parameters of the model

Analysis types: input

Template based model:

- When no simple analytic form can be assigned for the signal(background) description
- Templates (histograms) are provided for all signal and background processes and shape systematic uncertainties
- MC statistical uncertainties are modelled with Barlow-Beeston-lite approach (autoMCStats)

Parametric model:

- Suitable for the analysis with analytically described bkg and signal: e.g. Gaussian signal on smoothly falling polynomial background
- Data-driven bkg description
- The systematic uncertainties assigned on the parameters of the model

Counting experiment:

- “One-bin analysis”, i.e. the input parameters in the model are the signal, bkg and data yields

Likelihood function

Likelihood is a probability to observe data given parameters Φ, which we want to measure

$$\mathcal{L}_{\mathcal{M}}(\vec{\Phi}) = p_{\mathcal{M}}(\text{data}; \vec{\Phi})$$

Often we factorize the likelihood into primary and auxiliary components

$$\mathcal{L} = \mathcal{L}_{\text{primary}} \cdot \mathcal{L}_{\text{auxiliary}}$$

$\vec{\nu}$ auxiliary observables, from external measurement:
E.g. systematic uncertainties from exp. corrections (b-tagging, reconstruction efficiency); theory uncertainties

If we have several independent measurements $\overrightarrow{x_d}$ (histogram bins) the total likelihood constructed with product of individual likelihoods

$$\mathscr{L} = \prod_k \mathscr{L}_k(\mathrm{x}, \mu, \vec{\nu})$$

Same applies for individual independent categories (each with multiple bins)

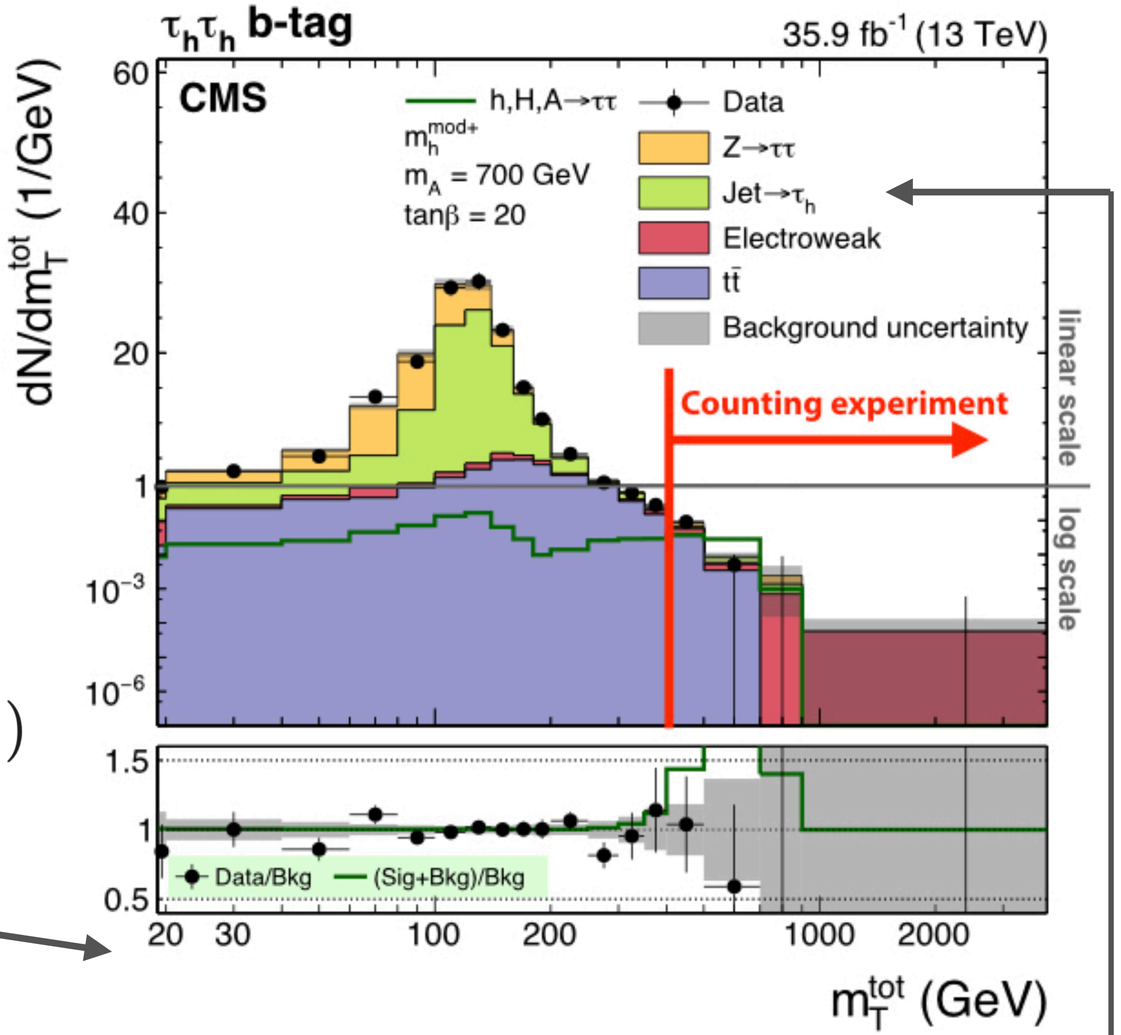
Binned likelihood example

$$\mathcal{L} = \mathcal{L}_{\text{primary}} \cdot \mathcal{L}_{\text{auxiliary}} =$$

C runs over categories

$$= \prod_{c=1}^{N_c} \prod_{b=1}^{N_b^c} \text{Poiss}(n_{cb}; n_{cb}^{\text{exp}}(\vec{\mu}, \vec{\nu})) \cdot \prod_{e=1}^{N_E} p_e(y_e; \nu_e)$$

Product over bins in a histogram



$$n_{cb}^{\text{exp}} = \max(0, \sum_p M_{cp}(\vec{\mu}) N_{cp}(\nu_G, \vec{\nu}_L, \vec{\nu}_S, \vec{\nu}_\rho) \omega_{cbp}(\vec{\nu}_S) + E_{cb}(\vec{\nu}_B))$$

Sum over the processes

Constraint terms

When we want to model systematic uncertainties given by external measurements we introduce constraint terms:

$$\mathscr{L} = \prod_k \mathscr{L}_k(\vec{x}, \mu, \vec{\nu}) \boxed{\prod_j p_j(\nu)}$$

With j running over NP

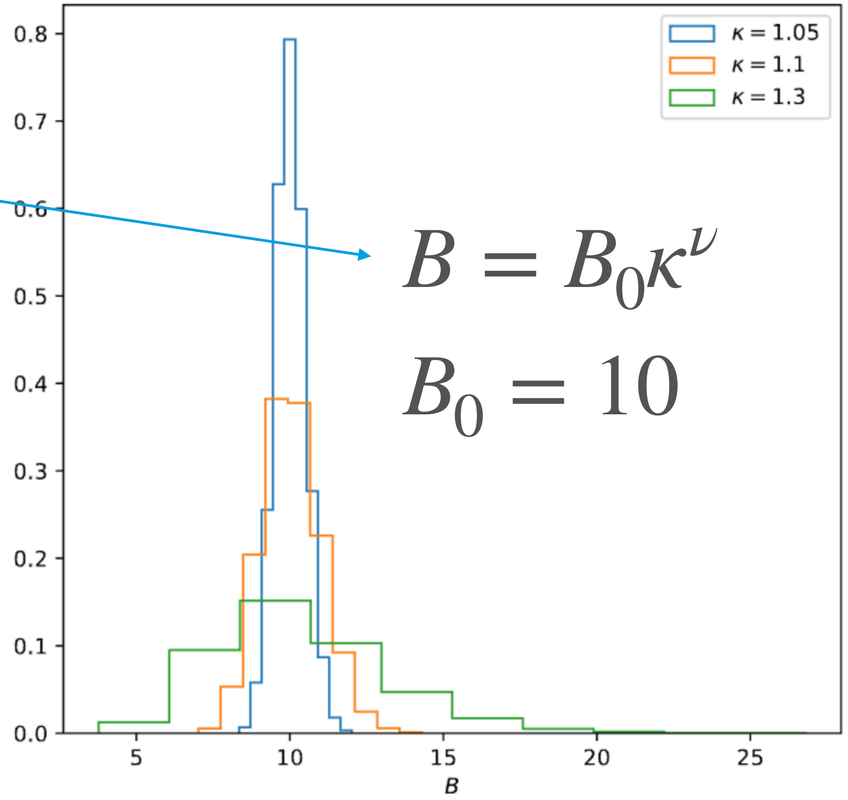
Uncertainty type	Directive	Inputs	Multiplicative factor, $f(\nu)$	$p(y;\nu)$	Default values
Log-normal	`lnN`	`kappa`	κ^{ν}	$\mathcal{N}(y;\nu,1)$	$\nu = y = 0$
Asymmetric log-normal	`lnN`	`kappaDown`, `kappaUp`	$\left(\kappa^{\text{Down}}\right)^{-\nu}$ if $\nu < -0.5$, $\left(\kappa^{\text{Up}}\right)^{\nu}$ if $\nu > 0.5$, $e^{\nu K\left(\kappa^{\text{Down}},\kappa^{\text{Up}},\nu\right)}$ otherwise.*	$\mathcal{N}(y;\nu,1)$	$\nu = y = 0$
Log-uniform	`lnU`	`kappa`	κ^{ν}	$\mathcal{U}(y, 1/\kappa, \kappa)$	$\nu = y = \frac{1}{2}(\kappa + 1/\kappa)$
Gamma	`gmN`	`N`, `alpha`†	ν/N	$\mathcal{P}(y;\nu)$	$\nu = N+1, y = N^{\ddagger}$

$$\kappa^{\text{Up/Down}} = \frac{\sum_b y_b^{+/-}}{\sum_b y_b^0}$$

lnN uncertainties

Uncertainty type	Directive	Inputs	Multiplicative factor, $f(\nu)$	$p(y;\nu)$	Default values
Log-normal	`lnN`	`kappa`	κ^{ν}	$\mathcal{N}(y;\nu,1)$	$\nu = y = 0$
Asymmetric log-normal	`lnN`	`kappaDown`, `kappaUp`	$(\kappa^{\text{Down}})^{-\nu}$ if $\nu < -0.5$, $(\kappa^{\text{Up}})^{\nu}$ if $\nu > 0.5$, $e^{\nu K(\kappa^{\text{Down}},\kappa^{\text{Up}},\nu)}$ otherwise.*	$\mathcal{N}(y;\nu,1)$	$\nu = y = 0$
Log-uniform	`lnU`	`kappa`	κ^{ν}	$\mathcal{U}(y,1/\kappa,\kappa)$	$\nu = y = \frac{1}{2}(\kappa + 1/\kappa)$
Gamma	`gmN`	`N`, `alpha`†	ν/N	$\mathcal{P}(y;\nu)$	$\nu = N+1, y = N^{\ddagger}$

$$p_e \propto \exp\left(-0.5\left(\frac{(\nu_e - y_e)}{\sigma}\right)^2\right);\ p_e = \text{Poiss}(\nu_e; y_e);\ \text{or } p_e \propto \text{constant (on some interval [a, b])}.$$



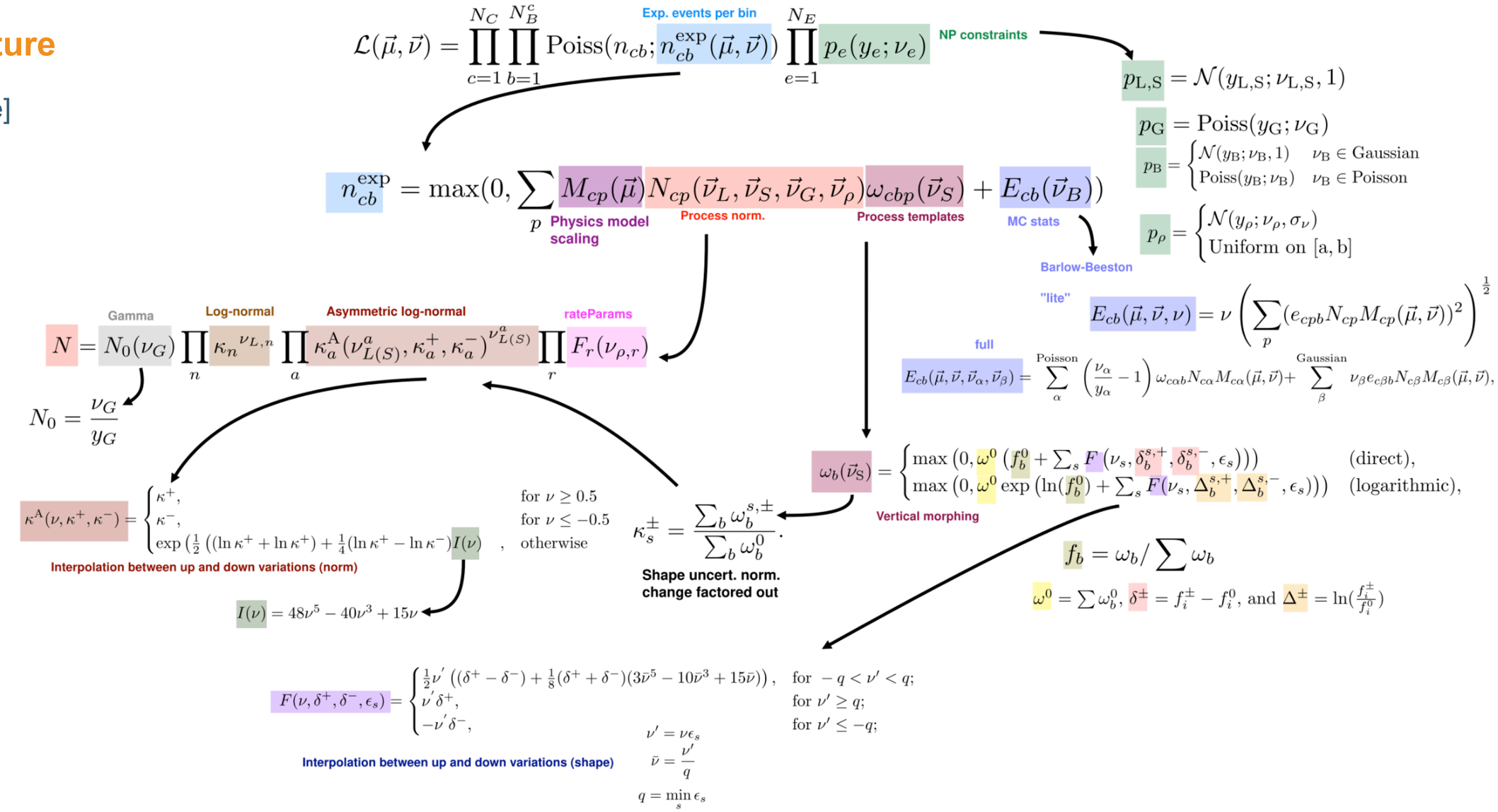
$$B = B_0\kappa^{\nu}$$

$$B_0 = 10$$

Likelihood definition

Complete picture

[More details here]



Datacard structure

Complete statistical model = datacard + inputs + physics model **= RooFit workspace**

(full likelihood)

Configuration file: maps the histograms with processes and assigns NP(lnN, shape, shapeN, gmN)

```
imax 1
jmax 1
kmax *
---------------
shapes * * simple-shapes-TH1_input.root $PROCESS $PROCESS_$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   uncertainty on background shape and normalization
sigma  shape    0.5        -   uncertainty on signal resolution. Assume the histogram is a 2 sigma shift,
#                              so divide the unit gaussian by 2 before doing the interpolation
```

analysis categories

process yield

background_alphaUp(Down) and signal_sigmaUp(Down) histograms taken from simple-shapes-TH1_input.root file

process id: >0 for bkg, ≤0 for signal

10% on signal from limi

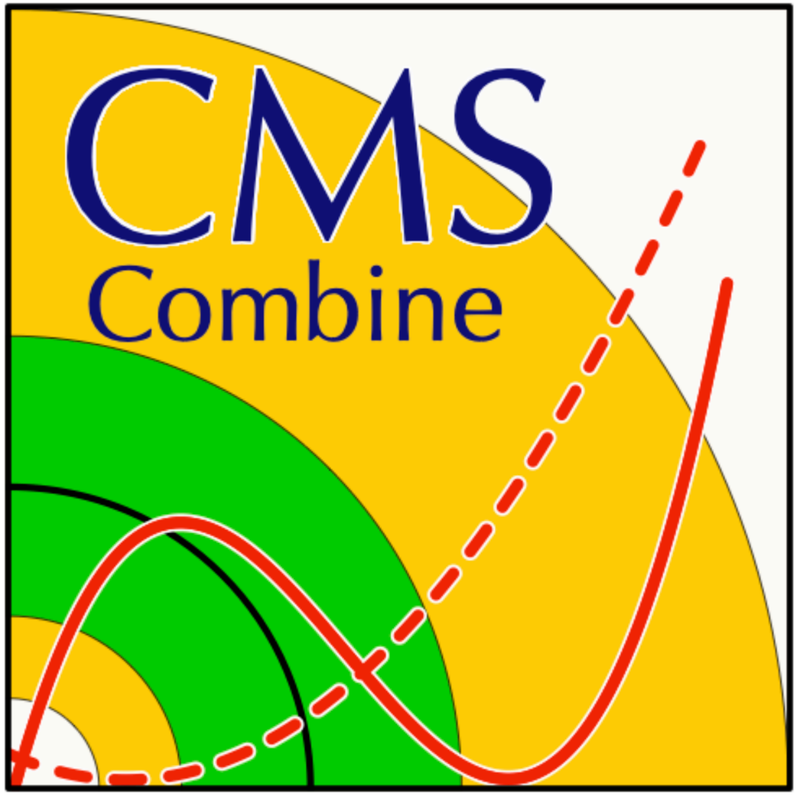
30% on bkg from bgnorm

Documentation

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

Combine v9.2.0 | Search | GitHub v9.2.0 ☆68 369

Introduction



These pages document the RooStats / RooFit - based software tool used for statistical analysis within the CMS experiment - COMBINE. Note that while this tool was originally developed in the Higgs Physics Analysis Group (PAG), its usage is now widespread within CMS.

COMBINE provides a command-line interface to many different statistical techniques, available inside RooFit/RooStats, that are used widely inside CMS.

The package exists on GitHub under https://github.com/cms-analysis/HiggsAnalysis-CombinedLimit

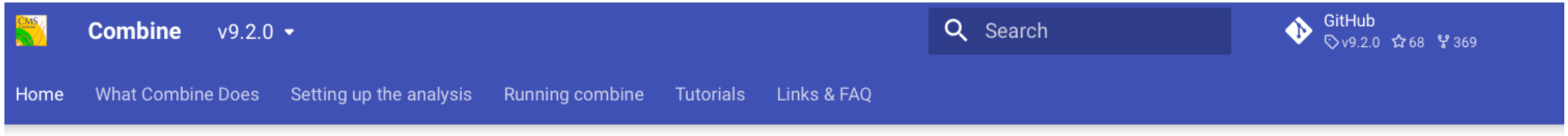
For more information about Git, GitHub and its usage in CMS, see http://cms-sw.github.io/cmssw/faq.html

The code can be checked out from GitHub and compiled on top of a CMSSW release that includes a recent RooFit/RooStats, or via standalone compilation without CMSSW dependencies. See the instructions for installation

Table of contents

Documentation

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

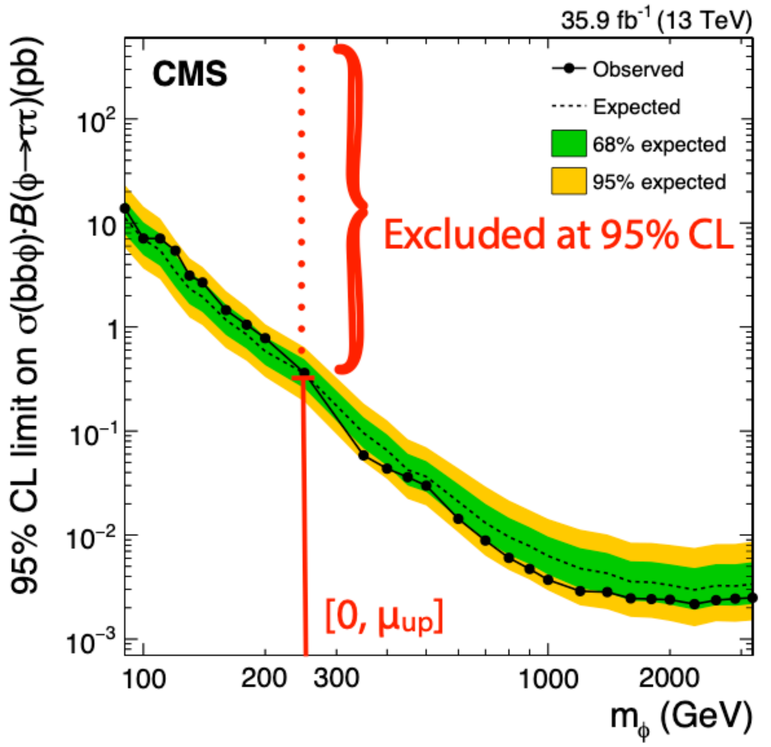


- Installation (w/ CMSSW, standalone, using LCG, conda, latest release) [link]
- Setting up the analysis (counting, template based, parametric) [link]
- Running Combine [link]
- Underlying statistics
  - Likelihood definition: [link]
  - How the fits are performed (profiling, marginalization, confidence intervals): [link]
  - Statistical tests (test statistic, GOF): [link]
- Tutorials: main features, parametric, unfolding, RooFit

Main methods

[Documentation]

- **Asymptotic** likelihood methods:
  - AsymptoticLimits: limits calculated according to the asymptotic formulas in arxiv:1007.1727, valid for large event counts
  - Significance: simple profile likelihood approximation for calculating significances.
- **Frequentist** or hybrid bayesian-frequentist methods:
  - HybridNew: compute modified frequentist limits with toys, significance/p-values and confidence intervals with several options, `--LHCmode LHC-limits` is the recommended one
- **Bayesian** methods:
  - BayesianSimple: performing a classical numerical integration (for simple models only)
  - MarkovChainMC: performing Markov Chain integration (for arbitrarily complex models)



$$\tilde{q}_\mu = \begin{cases} -2\log\left(\frac{\mathcal{L}(\mu)}{\mathcal{L}(\mu=0)}\right) & \hat{\mu} < 0 \\ -2\log\left(\frac{\mathcal{L}(\mu)}{\mathcal{L}(\hat{\mu})}\right) & 0 < \hat{\mu} < \mu \\ 0 & \mu < \hat{\mu} \end{cases}$$

Main methods

- **Fitting**
  - [MultiDimFit]: perform maximum likelihood fits with multiple POIs and likelihood scans

```
combine -M MultiDimFit --algo grid --points 50 -d ws.root
```

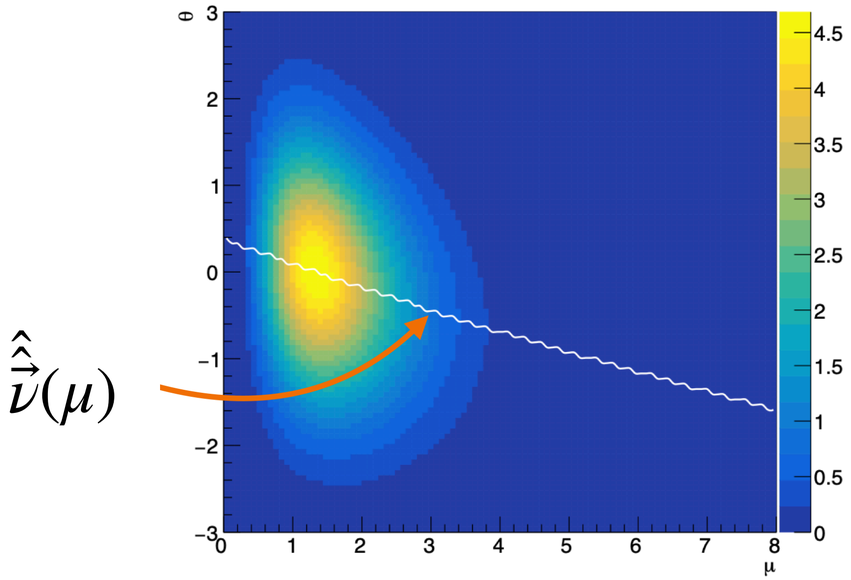
  - [FitDiagnostics]: performs maximum likelihood fits to extract the signal yield and provides diagnostic tools

```
combine -M FitDiagnostics -d ws.root
```

- **Other modules**:
  - [GoodnessOfFit]: perform a goodness of fit test for models including shape information using several GOF estimators (AD, KS, **saturated**)
  - [ChannelCompatibiltyCheck]: check how consistent the individual channels of your analysis are
  - [GenerateOnly]: generate random or asimov toy datasets for use as input to other methods
  - [Impacts]: evaluate the shift in POI from $\pm\sigma_{\text{postfit}}$ variation for each NP

[Documentation]

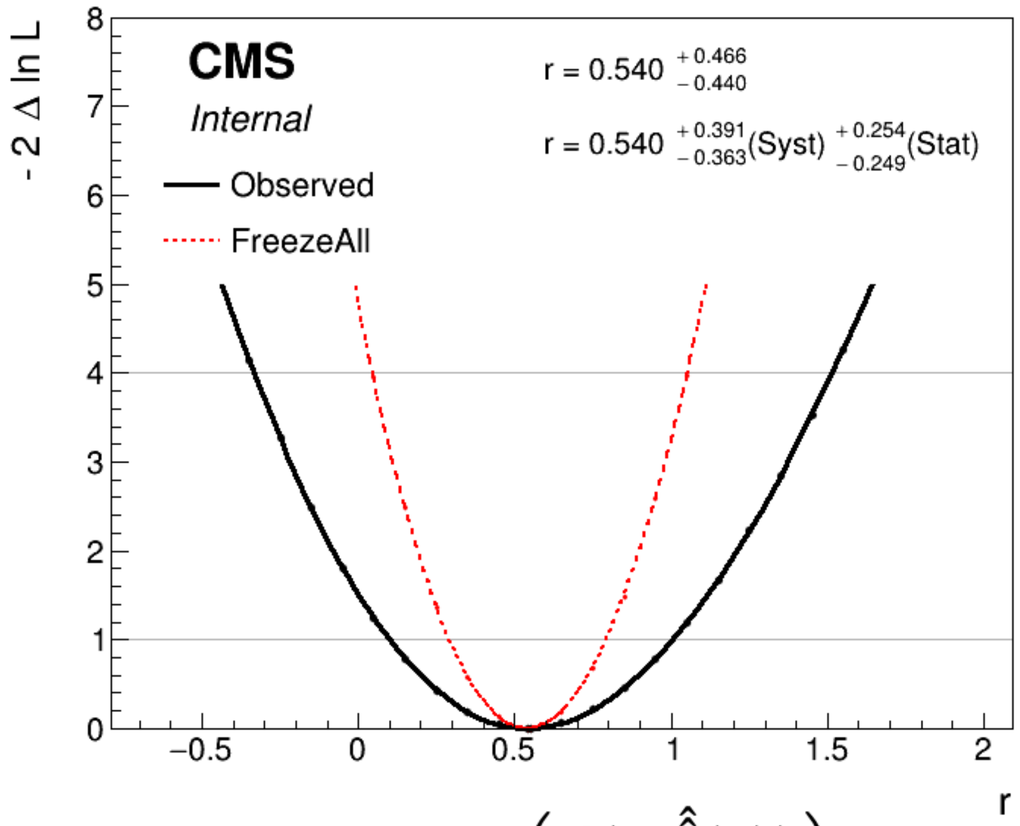
Profiled likelihood



$\hat{\hat{\vec{\nu}}}(\mu)$

In a measurement where no need to measure the intervals for NP $\vec{\nu}$

➡ Profiling: find a $\hat{\hat{\vec{\nu}}}(\mu)$ which maximise the likelihood for each value POI



$$q(\vec{\mu}) = -\ln\left(\frac{\mathcal{L}(\vec{\mu}, \hat{\hat{\vec{\nu}}}(\vec{\mu}))}{\mathcal{L}(\hat{\vec{\mu}}, \hat{\vec{\nu}})}\right)$$

FitDiagnostics

[Documentaion]

- Provides more information than `-M MultiDimFit`
- Runs background only fit first (`r=0` fixed), followed by s+b (`r` is floating).

  `combine -M FitDiagnostics -d ws.root`; output: `fitDiagnostics.root`

  - Provides full list of NP constraints and pulls for both fits
  - Covariance matrix is saved, access to all correlations
  - Using fit results from `fitDiagnostic.root` one can check the NP shifts and uncertainties wrt their input values:

    `python diffNuisances.py fitDiagnostics.root --all` [instructions]
  - Post(pre)-fit shapes can be saved with option `--saveShapes`, additional directories inside output file will be created for each category (can only be used when covariance matrix is properly estimated [see the debugging session in the afternoon])

Uncertainty on the measurement (r) should be estimated from full likelihood scans `combine -M MultiDimFit --algo grid --points 50 -d ws.root …`

Generating toys in combine

[Documentation]

- To compute expected significance, intervals, perform bias tests the toy experiments have to be generated.
- Adding `-t -1,` for Asimov dataset, `-t N` for N random datasets, options to any method
- The method `-M GenerateOnly --saveToys` allows to generate toys that can be later used with other methods (`-t -1` or `-t N`)
- 2 options for systematic uncertainties:

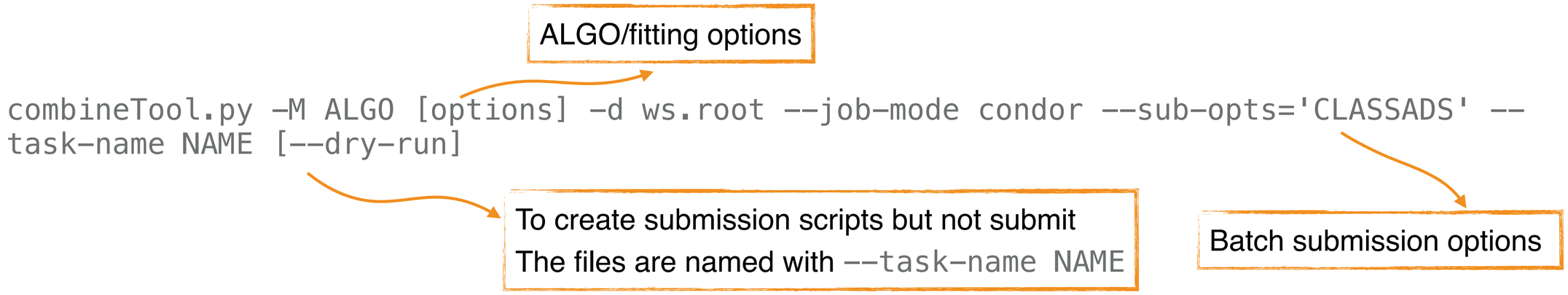
  `--toysNoSystematics` nominal (prefit) NP values are used to generated toys .

  `--toysFrequentist` first the fit to data is performed, then the nuisance parameters in each toy are set to their post-fit values, with POIs fixed (eg. `--setParameters r=1.`), before generating the data. The constraint terms are randomized within their Gaussian constraint pdfs around the post-fit NP values.
- To read the toy dataset: `-toysFile=higgsCombineTest.GenerateOnly.root`

Batch submission

[Detailed instructions]

- Useful when generating toys, running likelihood scans, other tasks that can be parallelised



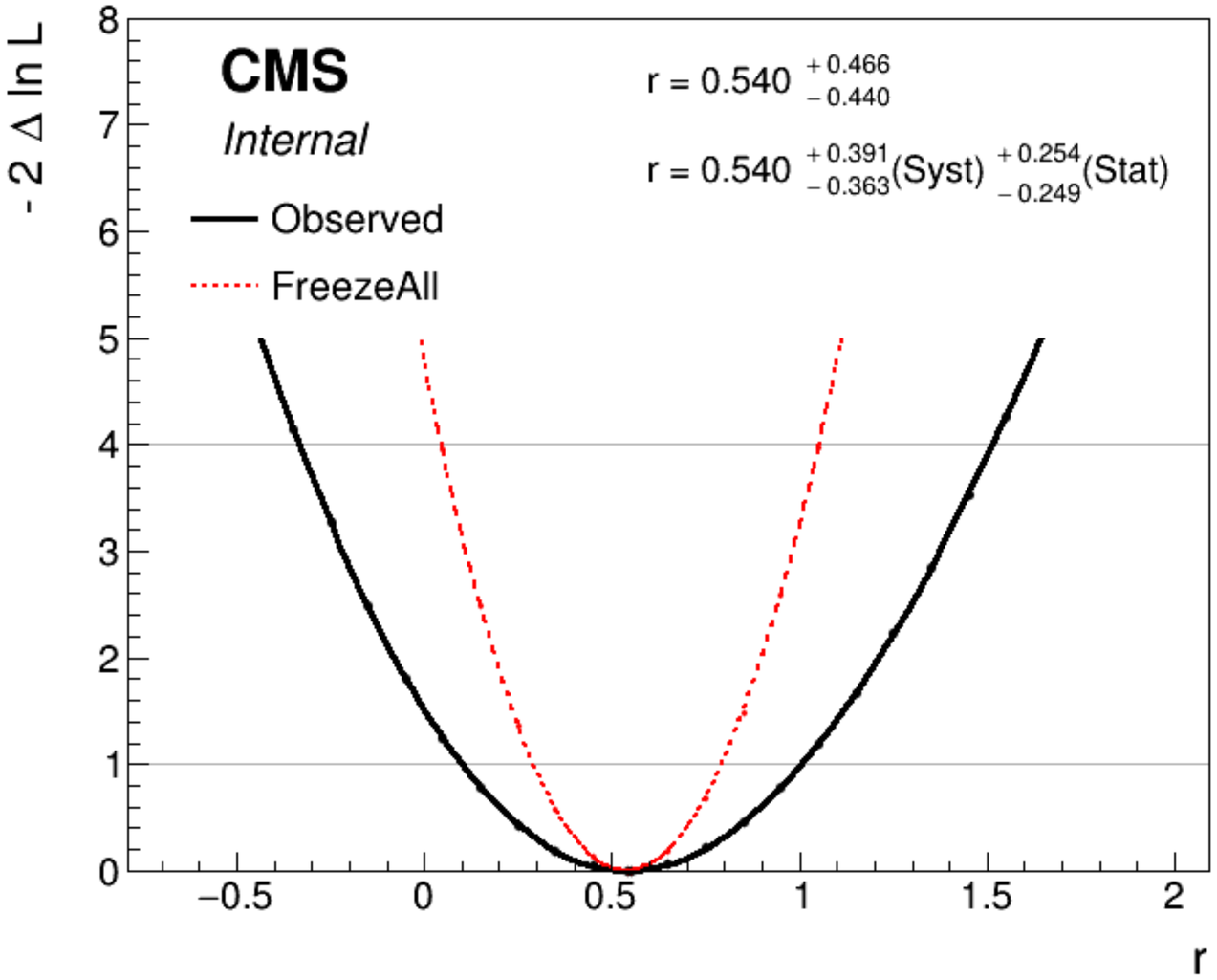
- Works with many methods, e.g. with `-M MultiDimFit` to produce likelihood scans:

```
combineTool.py -d ws.root -M MultiDimFit --algo grid --points 50 --rMin 0 --rMax 1 --job-mode condor -split-points 10 --sub-opts='+JobFlavour="workday"' --task-name mytask -n mytask
```

Particularly useful for GoodnessOfFit, Impacts, limits with toys

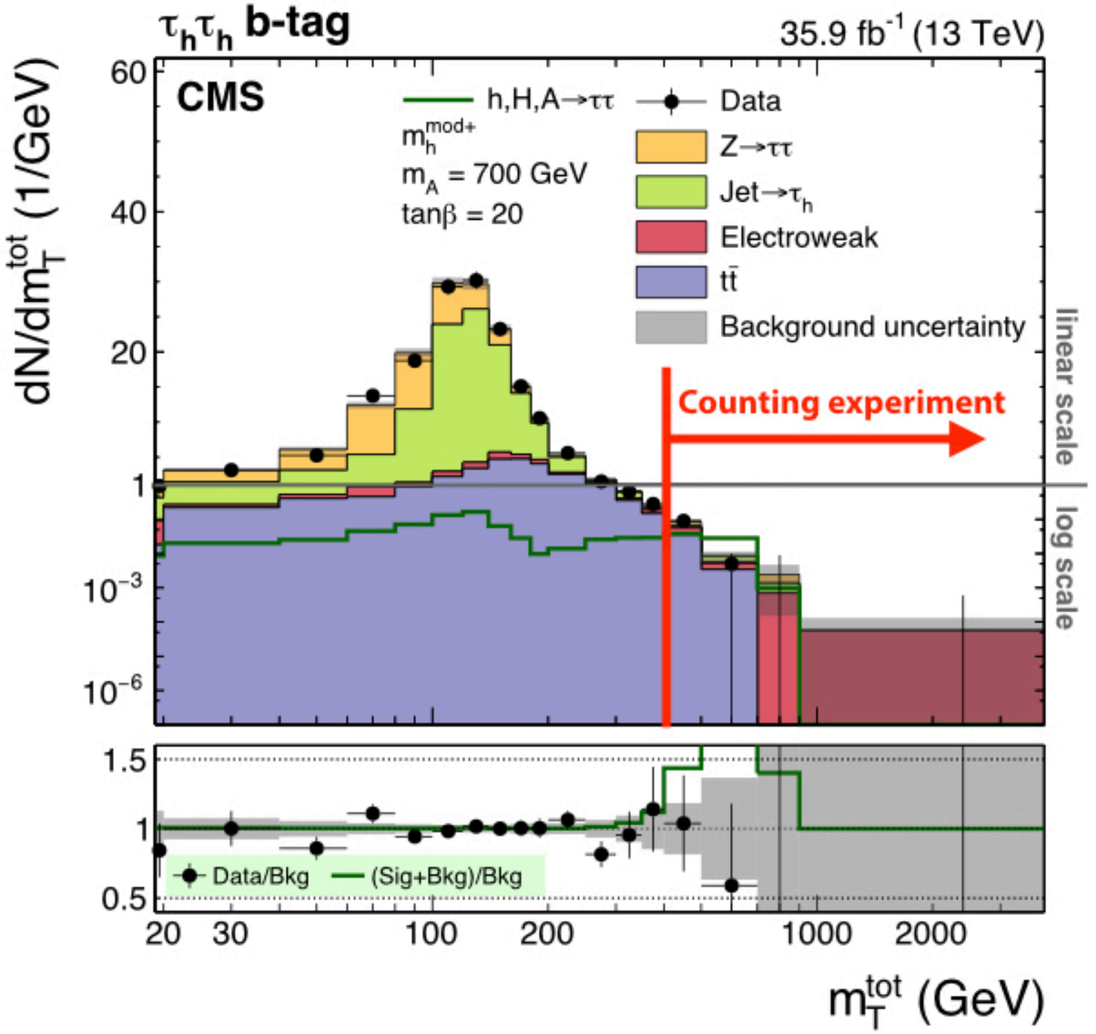
Plotting with combineTools

- Likelihood scans:
  - plot1DScan.py: uses the output of MultiDimFit
  - `--other` option can be used to load the scans with systematic uncertainties fixed, useful when producing scans with stat. and stat.+syst. uncertainties, full uncertainty breakdown to separate the contribution of various sources [documentaion]



Plotting with combineTools

- Observable distributions:
  - FitDiagnostics using --shapes option [link]
  - With PostFitShapesFromWorkspace
  - For post-fit fit shapes the output of FitDiagnostic should be provided with `-f fitDiagnostic.root ; --sampling` option should be used to take into account NP correlations



Questions?

Tutorial

I. Follow [Combine setup instructions]

Getting started

We need to set up a new CMSSW area and checkout the combine package:

```
source /cvmfs/cms.cern.ch/cmsset_default.sh
cmsrel CMSSW_11_3_4
cd CMSSW_11_3_4/src
cmsenv
git clone https://github.com/cms-analysis/HiggsAnalysis-CombinedLimit.git HiggsAnalysis/CombinedLimit
cd HiggsAnalysis/CombinedLimit

cd $CMSSW_BASE/src/HiggsAnalysis/CombinedLimit
git fetch origin
git checkout v9.2.0
cd $CMSSW_BASE/src/
```

We will also make use another package, `CombineHarvester`, which contains some high-level tools for working with combine. The following command will download the repository and checkout just the parts of it we need for this tutorial:

```
bash <(curl -s https://raw.githubusercontent.com/cms-analysis/CombineHarvester/main/CombineTools/scripts/sparse-checkout-https.sh)
```

Now make sure the CMSSW area is compiled:

```
scramv1 b clean; scramv1 b
```

Finally, move to the working directory for this tutorial, which contains all the inputs needed to run the exercises:

```
cd $CMSSW_BASE/src/HiggsAnalysis/CombinedLimit/data/tutorials/longexercise/
```

II. Move to the first exercise here

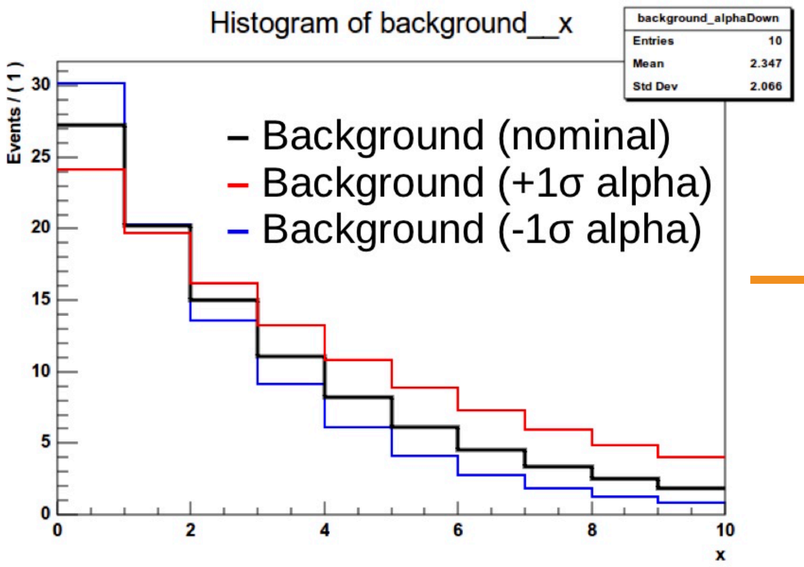
Main Features of Combine (Long Exercises)

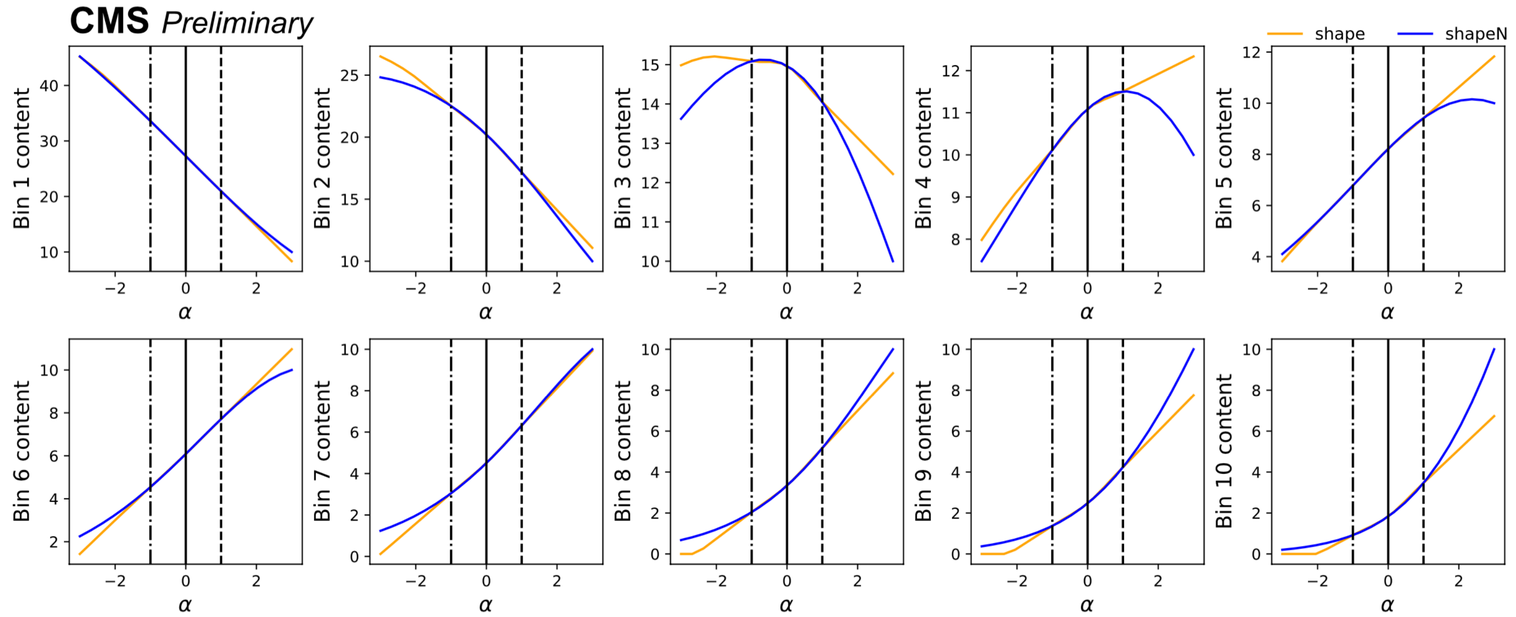
This exercise is designed to give a broad overview of the tools available for statistical analysis in CMS using the combine tool. COMBINE is a high-level tool for building `RooFit` / `RooStats` models and running common statistical methods. We will cover the typical aspects of setting up an analysis and producing the results, as well as look at ways in which we can diagnose issues and get a deeper understanding of the statistical model. This is a long exercise - expect to spend some time on it especially if you are new to COMBINE. If you get stuck while working through this exercise or have questions specifically about the exercise, you can ask them on this mattermost channel. Finally, we also provide some solutions to some of the questions that are asked as part of the exercise. These are available here.

Backup

Template morphing

[details]





Extrapolation with **shape**

$$f_b(\vec{\theta}) = y_b^0 + \sum_s F(\theta_s, \delta_b^{s,+}, \delta_b^{s,-}, \epsilon_s)$$

Extrapolation with **shapeN**

$$f_b(\vec{\theta}) = \exp\left(\ln(f_b^0) + \sum_s F(\theta_s, \Delta_b^{s,+}, \Delta_b^{s,-}, \epsilon_s)\right)$$

$$F(\theta) = \begin{cases} \frac{1}{2}\theta' \left((\delta^+ - \delta^-) + \frac{1}{8}(\delta^+ + \delta^-)J(\bar{\theta})\right), & -q < \theta < q; \\ \theta'\delta^+, & \theta \geq q; \\ -\theta'\delta^-, & \theta \leq -q; \end{cases}$$

$$\theta' = \theta\epsilon,\ \bar{\theta} = \theta'/q$$

$$J(\bar{\theta}) = (3\bar{\theta}^5 - 10\bar{\theta}^3 + 15\bar{\theta})$$

Continuous first and second derivative

Model building: Physics Model

[Details]

- Simple process scaling:
  - By default single POI assigned to all processes marked as signal in the datacards (“r”)
  - Multiple POIs can be assigned with [multiSignalModel] :

```
text2workspace.py -P HiggsAnalysis.CombinedLimit.PhysicsModel:multiSignalModel  --PO verbose
--PO 'map=.*/sig1:r_sig1[1,0,10]' --PO ‘map=.*/sig2:r_sig2[1,0,20]' datacard.txt -o ws.root
```

  - Mapping: `--PO 'map=bin/process:parameter'`

- More complicated: Interference, κ, EFT (signal processes are parametrized)
  - Use existing model [location]:

```
text2workspace.py datacard -P HiggsAnalysis.CombinedLimit.PythonFile:modelName
```

    e.g for Higgs couplings κ-model: `-P HiggsAnalysis.CombinedLimit.HiggsCouplings:c7` [link]

  - Create your own model in CombinedLimit/python directory; example given here on a model with interference: [tutorial], [code]

Model building: datacards (CombineHarvester)

[Details]

C++ package with [python interface] to create and modify datacards

1. Analysis categories ch::Categories
2. Signal and background processes ch::CombineHarvester::AddProcesses
3. Systematic uncertainties ch::CombineHarvester::AddSyst
4. Extract the related shape inputs from ROOT files ch::CombineHarvester::ExtractShapes
5. The input shapes/yields can be modified:
   - modifying signal processes to different cross sections (ch::Process::set_rate), change types (signal/background)
   - (de)correlating systematic uncertainties (renaming) ch::CombineHarvester::RenameSystematic
6. Exporting to the text datacard format and creating the associated ROOT shape file(s): ch::CardWriter

Examples: [Example 1] [Example 2]

[Example 2 in python]

Model building: datacards (CombineHarvester)

[Details]

C++ package with [python interface] to create and modify datacards

1. Analysis categories ch::Categories
2. Signal and background processes ch::CombineHarvester::AddProcesses
3. Systematic uncertainties ch::CombineHarvester::AddSyst
4. Extract the related shape inputs from ROOT files ch::CombineHarvester::ExtractShapes
5. The input shapes/yields can be modified:
   - modifying signal processes to different cross sections (ch::Process::set_rate), change types (signal/background)
   - (de)correlating systematic uncertainties (renaming) ch::CombineHarvester::RenameSystematic
6. Exporting to the text datacard format and creating the associated ROOT shape file(s): ch::CardWriter

Parse already existing datacard and apply necessary changes: ch::CombineHarvester::ParseDatacard

Examples: [Example 1] [Example 2]

[Example 2 in python]