HistFitter

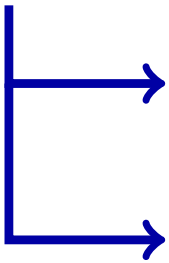
A flexible framework for statistical data analysis

Jeanette Lorenz (LMU München/Excellence Cluster Universe)
Max Baak (CERN)
Geert-Jan Besjes (Radboud University Nijmegen/Nikhef)
David Côté (University of Texas)
Alex Koutsman (TRIUMF)
Dan Short (University of Oxford)

01.09.2014 / ACAT 2014

Introduction

HistFitter: software framework for statistical data analysis.



Built on top of HistFactory/RooFit (construction of parametric models) and RooStats (statistical tests of data)

Consists of a Python part for configuration and a C++ part for CPU-intensive calculations

HistFitter extends RooFit/HistFactory/RooStats in four key areas:

- Programmable framework: performing complete statistical analyses, using a user-defined configuration file
- Analysis strategy: Concepts of analysis control, validation and signal regions deeply woven into the design of HistFitter
- Bookkeeping: HistFitter keeps track of numerous data models - including construction and statistical tests of all of them in an organized way
- Presentation and interpretation: Collection of tools to: determine the statistical significance of signal hypotheses, estimate the quality of likelihood fits, produce tables and plots expressing the results

HistFitter used in numerous analyses (e.g. SUSY searches) of the ATLAS Collaboration at the LHC.

Outline

Data analysis strategy

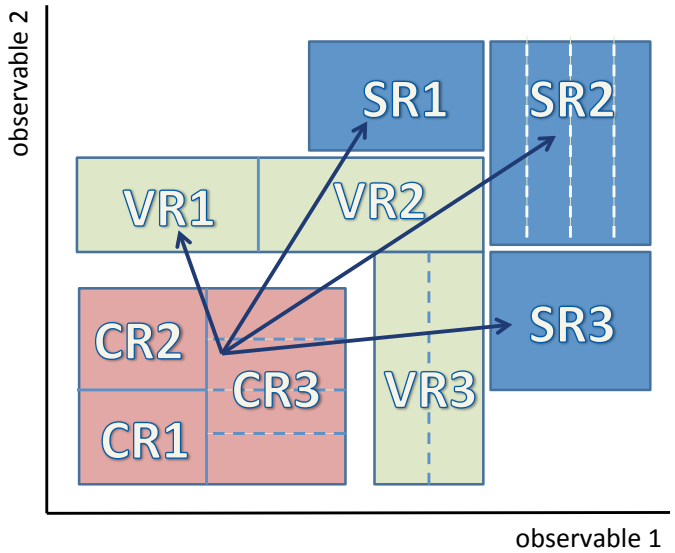
Particle physics experiments analyze large data samples in order to measure properties of fundamental particles and to discover new physical processes.

Data interpreted using external predictions for background and signal components.

→ **HistFitter configures and builds parametric models to describe the observed data, and provides tools to interpret the data.**

Construction and handling of models based on the concept of signal, control and validation regions:

- Signal regions: signal-rich region (SR)
- Control regions: background-rich region, fit simulated backgrounds to data (CR)
- Validation regions: validation of extrapolation (VR)



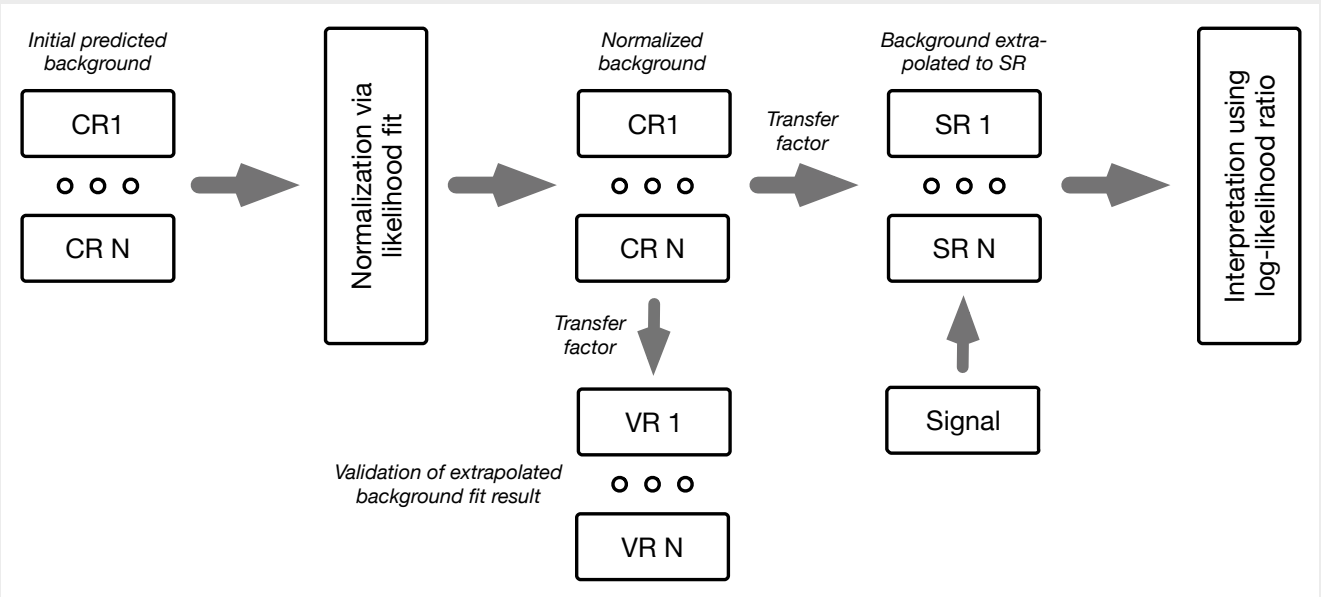
Concept deeply woven into design of HistFitter.

Typical analysis strategy with HistFitter

Model represented by a Probability Density Function (PDF):

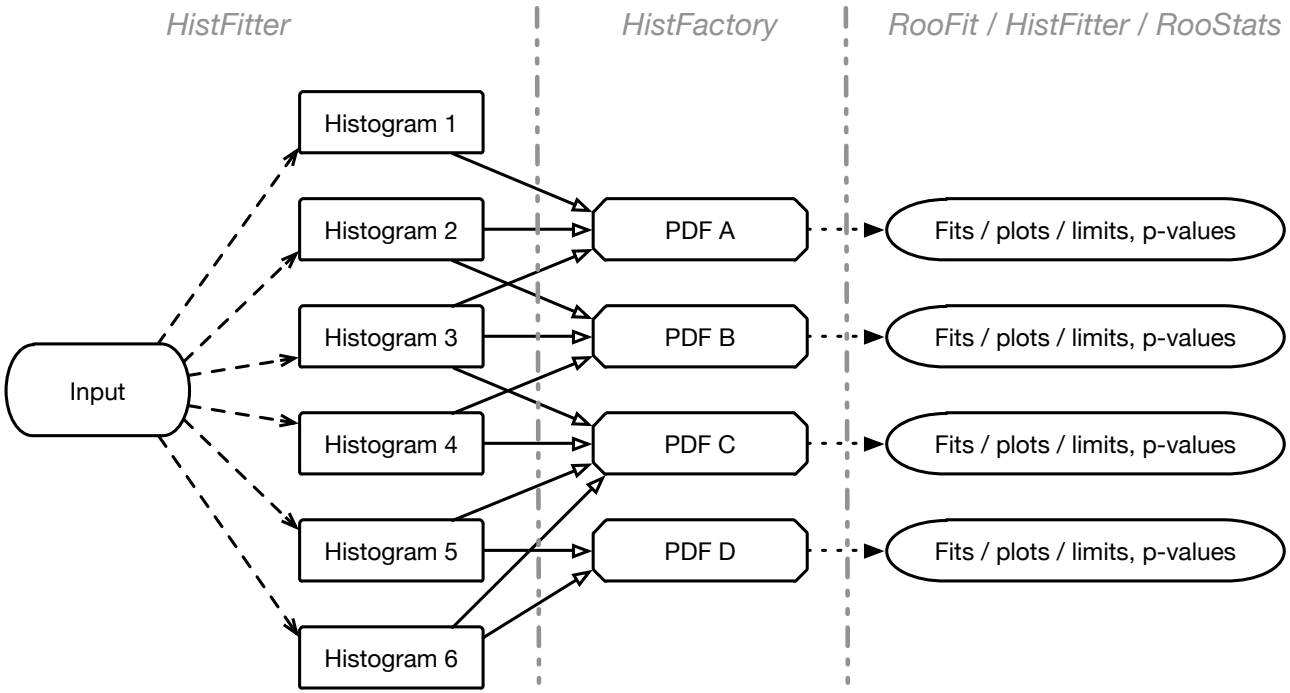
- Parameters are adjusted by a likelihood fit.
- Each CR/VR/SR modeled by separate PDF, combined in simultaneous fit.
- PDF parameters can be shared in different regions.

- Background normalized to data in a fit to data in control regions.
- Extrapolate to validation or signal regions using transfer factors (ratio of expected event count between each control and validation/signal region).



HistFitter software framework

Based on user-defined configuration and raw data as input, the processing sequence of HistFitter consists of three steps:



Histogram construction from input data and configuration

Construction of PDFs from binned histograms

Analysis of models

Configuration and bookkeeping

Substantial bookkeeping and configuration machinery required for presented analysis flow (in particular if working with multiple different signal hypotheses)

Realized through a user-defined Python configuration file which interacts with a configuration manager within HistFitter.

Technical side: configuration manager

- Realized as two singleton objects in Python and in C++
  - The user interacts with the Python configuration manager
- The configuration manager can be understood as “factory of factories”: it organizes and creates so-called `fitConfig` objects
  - The `fitConfig` objects contain the PDF of the studied model along with meta-data giving information about construction, fitting, visualizing and interpretation of the model.
  - A `fitConfig` object is thus an own “factory” and represents one row on the last slide.

Benefit of the configuration manager: histogram recycling - histograms are reused if appearing in different models.

Construction of PDFs

Construction of parametric models as PDFs via HistFactory from binned input histograms

General form of constructed likelihood:

$$L(\boldsymbol{n}, \boldsymbol{\theta}^0 | \mu_{\text{sig}}, \boldsymbol{b}, \boldsymbol{\theta}) = P_{\text{SR}} \times P_{\text{CR}} \times C_{\text{syst}}$$

$\rightarrow$ product of Poisson distributions of event counts in signal and control regions (P_{SR} and P_{CR}) and of additional constraint terms for systematic uncertainties (C_{syst})

Likelihood depends on number of observed events in all regions ($\boldsymbol{n}$), nuisance parameters parameterizing the impact of systematic uncertainties ($\boldsymbol{\theta}$) with their central values $\boldsymbol{\theta}^0$, signal strength μ_{sig} and predictions $\boldsymbol{b}$ for various background sources.

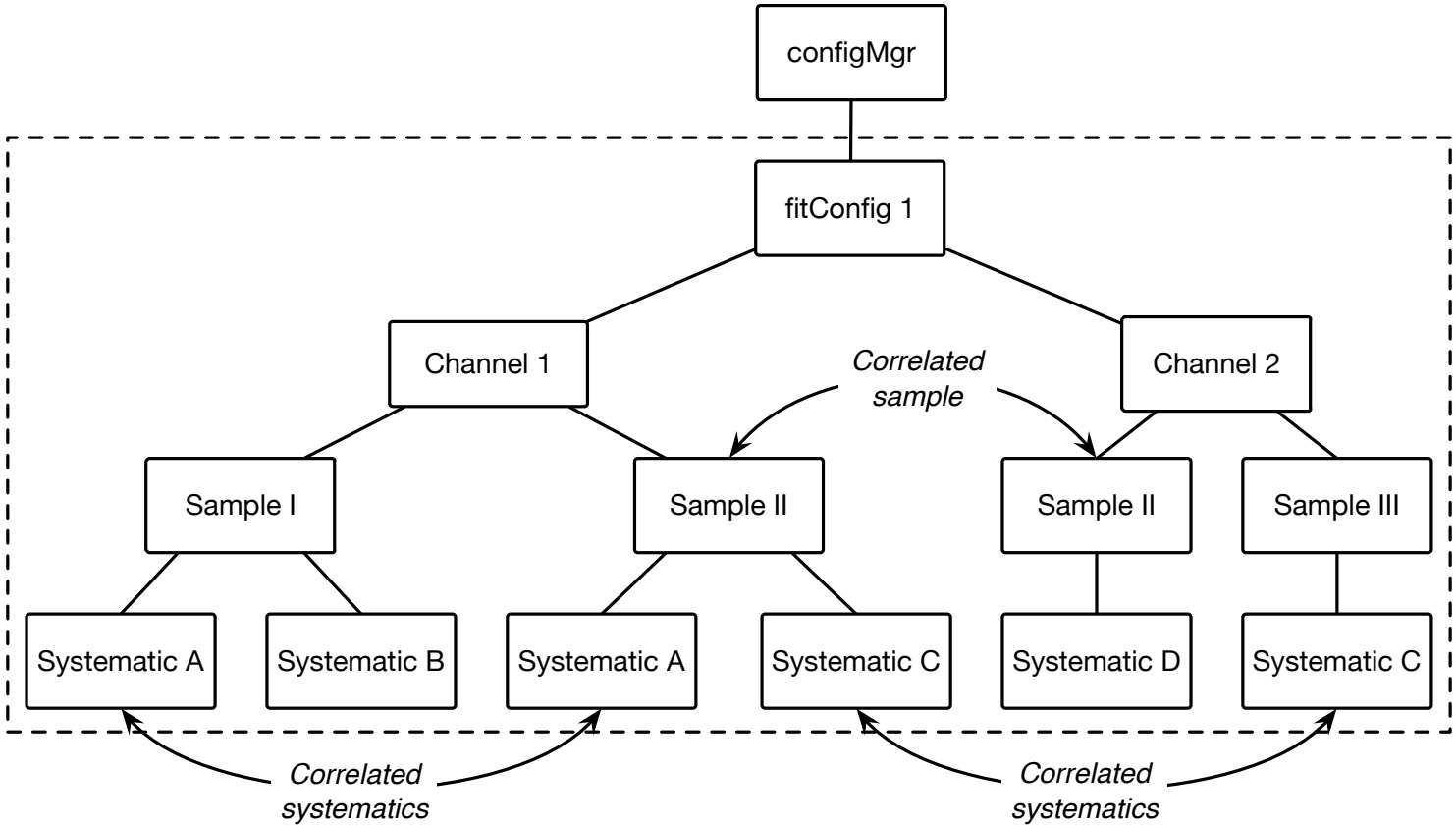
The likelihood thus has multiple building blocks:

- Control, validation, signal regions: called `channel` in HistFitter context
- Signal and background processes: called `samples`
- `Systematic` uncertainties, including statistical, theoretical and experimental uncertainties

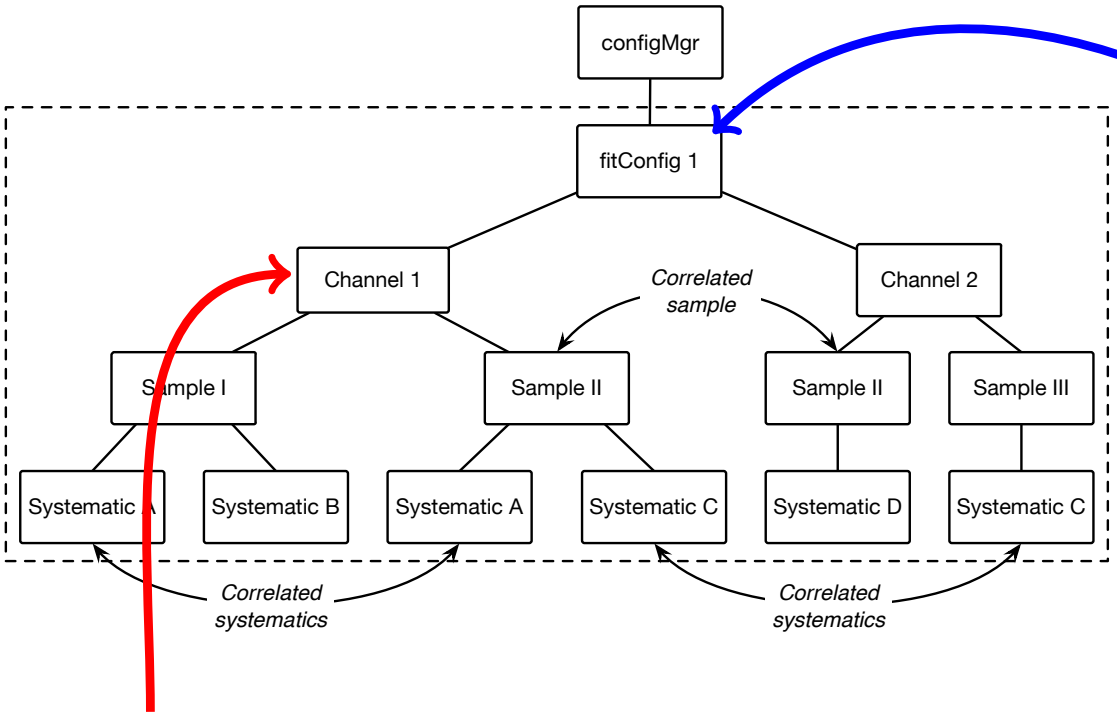
Original HistFactory classes mirrored and extended in HistFitter (in Python)

Fit configuration through `fitConfig` object

The `fitConfig` objects summarize channels, samples and systematics together with links to input histograms (representing the input data)



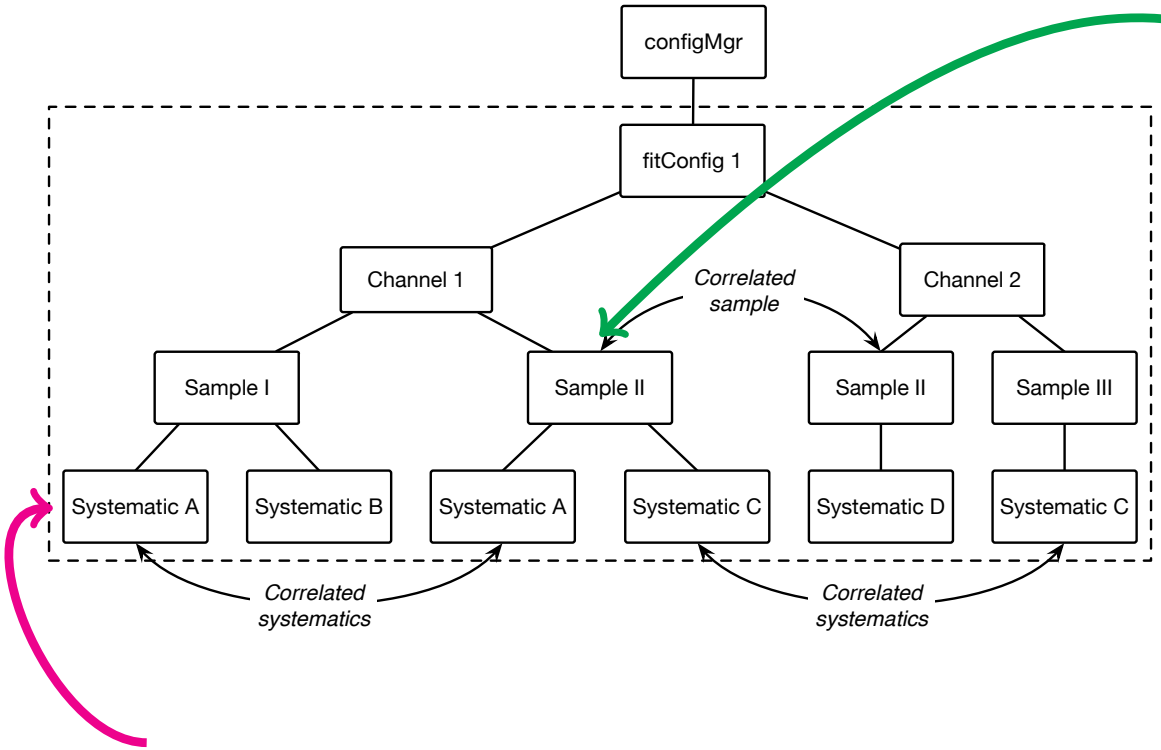
Further properties of the fit configuration



Basic fit configuration can be cloned and extended

- Channels have either one bin or derive from multi-bin histograms.
- Property set: the channel is a control, validation or signal region.

Further properties of the fit configuration



`Sample`:

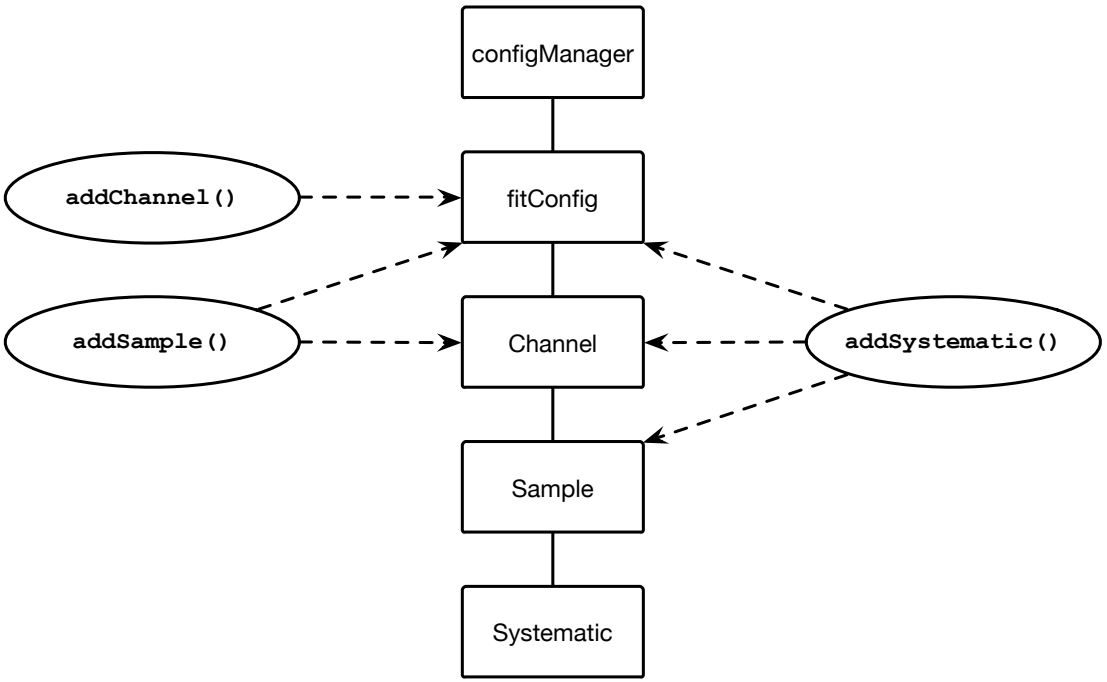
- Corresponds to a component of PDF decorated with HistFitter meta-data.
- Input either ROOT TTree, ROOT TH1 histograms or raw float numbers.
- Samples can be correlated between multiple channels.

`Systematic` class object:

- Systematic uncertainties typically provided as 1σ up and down variations of a nominal histogram.
- Input either ROOT TTree, ROOT TH1 histograms or raw float numbers.
- Systematic uncertainties can be correlated between different samples and channels.
- Different types available (differing in the constraint parametrization and interpolation/extrapolation type).

$\rightarrow$ HistFitter extends here the types present in HistFactory (`overallSys`, `histoSys`, `shapeSys`) by further composite and extended types.

“Trickle-down” mechanism



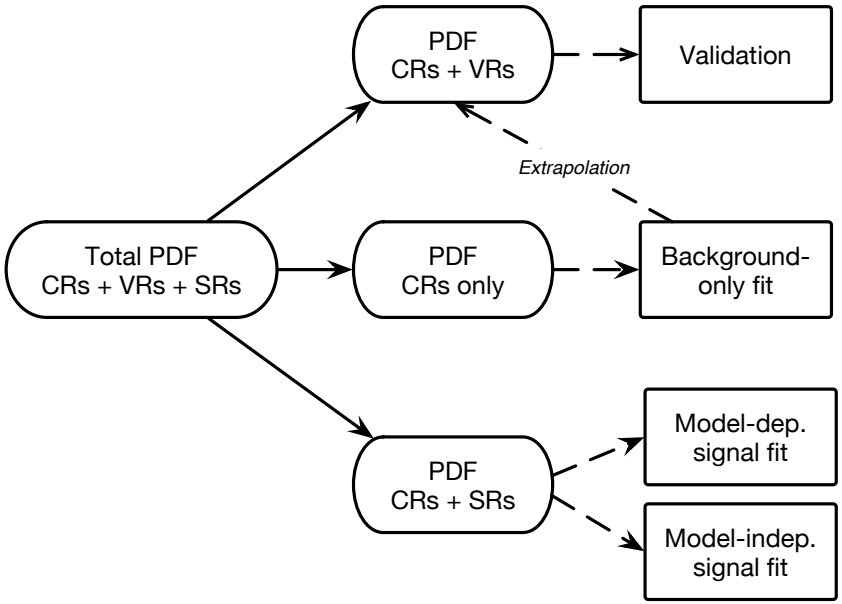
`Channels` are added to a `fitConfig`.

Samples can be added to either a `fitConfig` or a `channel`. If adding to `fitConfig` also added to all depending `channels`.

Similar for **systematics**. Can be added to `fitConfig` and then propagated to all `channels` and `samples`. Or added to a `channel` and then propagated to all depending `samples`. Or just added to a specific `sample`.

$\Rightarrow$ A complicated PDF can be described by few lines of code.

Fit strategy

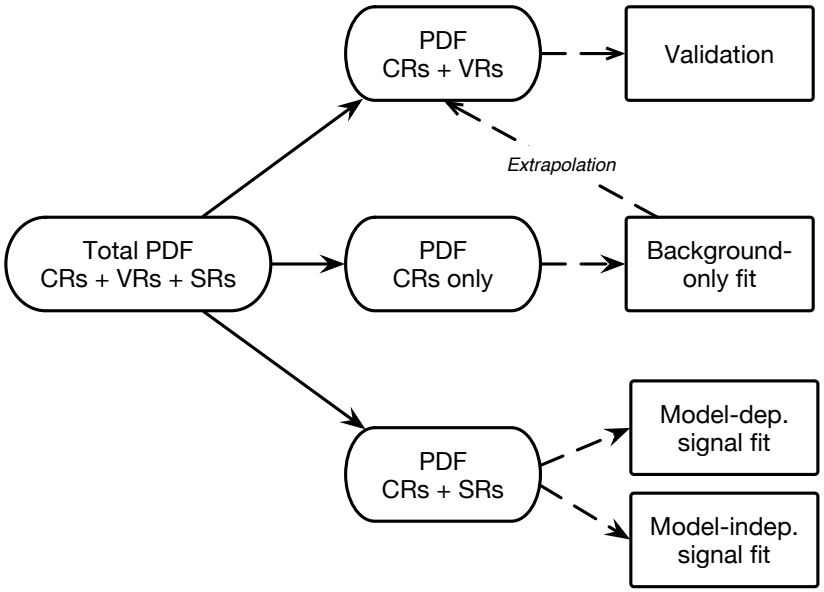


HistFitter supports different fit strategies:

- **Background-only fit:**
  - To estimate background yields in validation and signal regions.
- **Model-dependent signal fit:**
  - To set exclusion limits on a specific signal model in absence of an excess in the signal regions or to measure its properties in case of an excess.
  - Simultaneous use of multiple signal regions possible; signal then constrained in all signal regions (“shape fit”)
- **Model-independent signal fit:**
  - To obtain model-independent upper limits on the number of events beyond the expected number of events in a certain signal region.

Fit setup	Background-only fit	Model-dependent signal fit	Model-independent signal fit
Samples used	backgrounds	backgrounds + signal	backgrounds + dummy signal
Fit regions	CR(s)	CR(s) + SR(s)	CR(s) + SR

Extrapolation and error propagation



Extrapolation into validation and signal regions:

- Deconstruction of full likelihood containing CRs and VRs/SRs into smaller likelihood only containing CRs for use in the background-only fit.
- Incorporation of fitted parameters after background-only fit into full likelihood.
- Evaluation of the extrapolated uncertainty in validation/signal regions through standard error propagation.

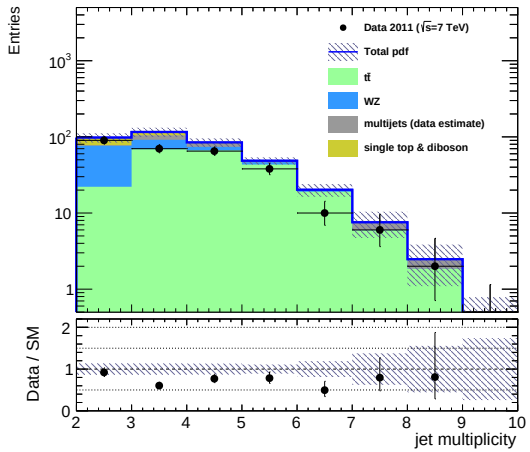
Extrapolation into signal and validation regions particularly rigorous in HistFitter due to use of `RooExpandedFitResult` class:

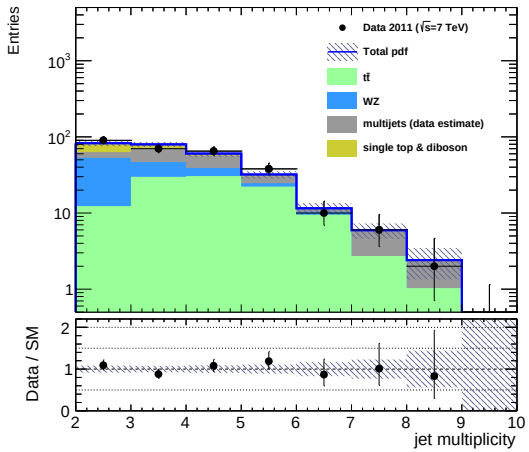
- Standard `RooFitResult` contains only the parameters used in the background-only fit.
- Using instead the `RooExpandedFitResult` class allows to extrapolate **all** parameters, such that a correct evaluation of the uncertainties through error propagation is possible.

Presentation of results

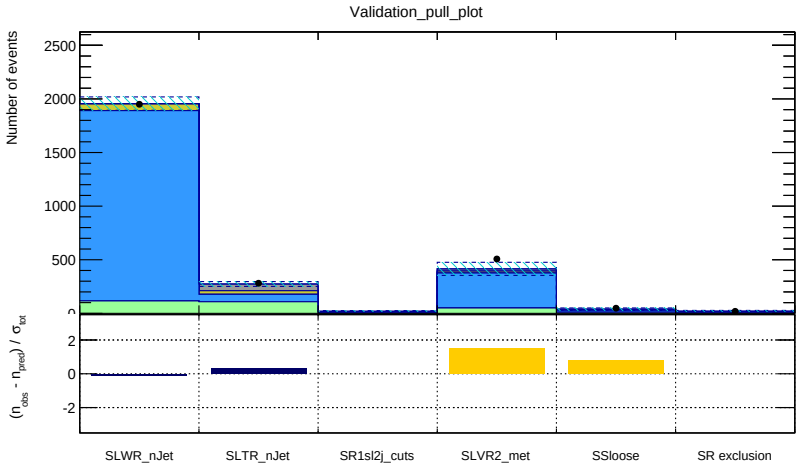
HistFitter includes a collection of tools and functions to aid the presentation of the results:

1. Visualization of fit results in before and after-fit distributions and in pull plots.





Before and after fit plots.



Pull plot:

$$\chi = \frac{n_{\text{obs}} - n_{\text{pred}}}{\sigma_{\text{tot}}}$$

$$\sigma_{\text{tot}} = \sqrt{\sigma^2_{\text{pred}} + \sigma^2_{\text{stat, exp}}}$$

(Example plots without physical meaning)

Presentation of results

HistFitter includes a collection of tools and functions to aid the presentation of the results:

2. Scripts for producing event yields and uncertainty tables.

Signal Region	SR1	SR2
Observed events	16	19
Fitted bkg events	19.54 ± 3.93	20.47 ± 5.14
Fitted Top events	4.02 ± 0.96	4.32 ± 1.04
Fitted V+jets events	9.89 ± 1.86	10.47 ± 1.91
Fitted other background events	1.14 ± 0.15	1.19 ± 0.16
Fitted QCD events	4.49 ± 2.72	4.49 ± 4.24
MC exp. SM events	24.85	26.32
MC exp. Top events	8.42	9.11
MC exp. V+jets events	10.82	11.55
MC exp. other background events	1.13	1.17
Data-driven exp. QCD events	4.49	4.49

Uncertainty of channel	SR1	SR2
Total background expectation	19.54	20.47
Total statistical ($\sqrt{N_{\text{exp}}}$)	$\pm$4.42	$\pm$4.52
Total background systematic	$\pm$3.93 [20.14%]	$\pm$5.14 [25.09%]
QCD background	$\pm$2.66	$\pm$4.20
Statistical uncertainties	$\pm$2.54	$\pm$1.86
Jet Energy Scale	$\pm$1.15	$\pm$1.17
Top yield	$\pm$0.82	$\pm$0.88
Renormalization scale (Top)	$\pm$0.34	$\pm$0.39
V+jets yields	$\pm$0.28	$\pm$0.29
Renormalization scale (V+jets)	$\pm$0.14	$\pm$0.03

(Example tables without physical meaning)

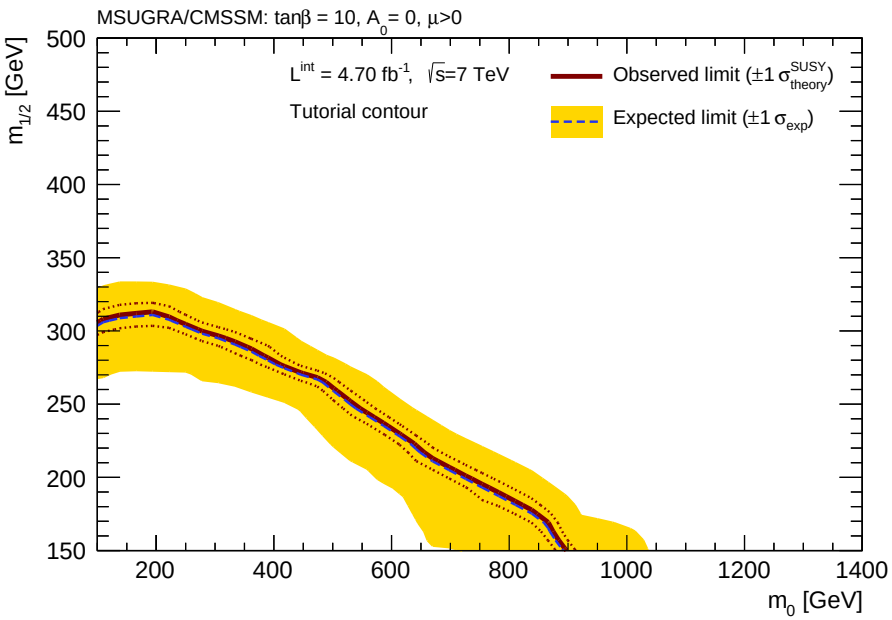
Interpretation signal model dependent

Hypothesis tests for different assumptions and models

HistFitter provides various interpretations/hypothesis tests through calls to the appropriate RooStats functions and classes and offers macros for interpreting the results in plots and tables.

Based on a specific signal model, using the model-dependent signal limit fit:

- **Signal model hypothesis test:**
  - Testing signal strengths of 1 for usually multiple signal models.
  - HistFitter executes the hypothesis test(s), collects the results, provides plotting macros.
- **Signal strength upper limit:**
  - Multiple hypothesis tests for a certain signal model, testing different signal strengths, to determine excluded upper limit on 95% CL.



(Example plot without physical meaning)

Interpretation signal model independent

Interpretations without the dependency on a specific signal model:

- **Model-independent upper limit:**
  - To calculate the 95% CL upper limit on the number of events for any kind of new physics.
  - Using the model-independent fit configuration.
  - HistFitter includes a script for the calculation and for presentation in a table.
- **Background-only hypothesis test (discovery p-value):**
  - Significance of an excess of events in the signal region: probability that a background-only experiment is more signal-like than observed.

Example Upper limit table:

Signal channel	$\langle \epsilon\sigma \rangle^{95}_{\text{obs}}$[fb]	S^{95}_{obs}	S^{95}_{exp}	$p(s=0)$
Example signal region	0.72	3.4	$8.9^{+4.0}_{-2.7}$	0.50

(Table has no physical meaning.)

Summary

Presented the software framework HistFitter which is tailored for statistical analysis.

...programmable framework to build and test data models of nearly arbitrary complexity.

...starting from user-defined input configuration file, and by using HistFactory, RooStats, RooFit, the tool constructs and fits PDFs and provides interpretations by statistical tests

Innovative features:

- Modular configuration interface with trickle-down mechanism which eases the construction of complicated PDFs.
- Built-in concepts of control, validation and signal regions with a particular rigorous statistical treatment for the extrapolation.
- Designed and providing the bookkeeping to work with multiple signal models at once and thus provides an additional level of abstraction.
- Sizable collection of tools and options for presenting end results with a publication-style quality.

A paper about HistFitter is in preparation and will be released in $\sim$ 1 month along with a public release of the tool.

Backup

Different possibilities of implementing systematic uncertainties

Various types available in HistFitter:

Basic systematic methods in HistFactory	
`overallSys`	uncertainty of the global normalization, not affecting the shape
`histoSys`	correlated uncertainty of shape and normalization
`shapeSys`	uncertainty of statistical nature applied to a sum of samples, bin by bin
Additional systematic methods in HistFitter	
`overallNormSys`	overallSys constrained to conserve total event count in a list of region(s)
`normHistoSys`	histoSys constrained to conserve total event count in a list of region(s)
`normHistoSysOneSide`	one-sided normHistoSys uncertainty built from tree-based or weight-based inputs
`normHistoSysOneSideSym`	symmetrized normHistoSysOneSide
`overallHistoSys`	factorized normalization shape and uncertainty, described with `overallSys` and `histoSys` respectively
`overallNormHistoSys`	overallHistoSys in which the shape uncertainty is modeled with a normHistoSys and the global normalization uncertainty is modeled with an overallSys
`shapeStat`	shapeSys applied to an individual sample

Sub-set of the systematic methods available in HistFitter. The methods are specified by a string argument containing a combination of basic HistFactory methods and optional HistFitter keywords: `norm`, `OneSide` and/or `Sym`. Systematic objects can be built with Tree-based, weight-based, Float or histogram input methods in all cases.

Different methods for uncertainty tables

Two methods available:

- Method 1: Calculating the uncertainty propagated to the background prediction by a specific parameter.
- Method 2: Excluding a (or multiple) specific parameters from the fit and refit - the impact of the parameter is then given by $\sigma_{\eta_1} = \sqrt{\left(\sigma_{\text{tot}}^{\text{nominal}}\right)^2 - \left(\sigma_{\text{tot}}^{\eta_1 = C}\right)^2}$