HistFitter - A flexible framework for statistical data analysis

J M Lorenz[1,2], M Baak[3], G J Besjes[4,5], D Côté[6], A Koutsman[7] and D Short[8]

[1] Fakultät füer Physik, LMU München, Am Coulombwall 1, 85748 Garching, Germany
[2] Excellence Cluster Universe, Boltzmannstr. 2, 85748 Garching, Germany
[3] CERN, Route de Meyrin 385, 1217 Meyrin, Switzerland
[4] IMAPP, Radboud University Nijmegen, P.O. Box 9010, 6500 GL Nijmegen, The Netherlands
[5] Nihkef, Science Park 105, 1098 XG Amsterdam, The Netherlands
[6] The University of Texas at Arlington, Department of Physics, Box 19059, Arlington, Texas 76019
[7] TRIUMF, 4004 Wesbrook Mall, Vancouver, BC V6T 2A3, Canada
[8] Department of Physics, Denys Wilkinson Building, University of Oxford, Keble Road, Oxford, OX1 3RH, Great Britain

E-mail: `Jeanette.Lorenz@physik.uni-muenchen.de`

Abstract. We present a software framework for statistical data analysis, called *HistFitter*, that has extensively been used in the ATLAS Collaboration to analyze data of proton-proton collisions produced by the Large Hadron Collider at CERN. Most notably, HistFitter has become a de-facto standard in searches for supersymmetric particles since 2012, with some usage for Exotic and Higgs boson physics. HistFitter coherently combines several statistics tools in a programmable and flexible framework that is capable of bookkeeping hundreds of data models under study using thousands of generated input histograms. The key innovations of HistFitter are to weave the concepts of control, validation and signal regions into its very fabric, and to treat them with rigorous statistical methods, while providing multiple tools to visualize and interpret the results through a simple configuration interface.

1. Introduction

HistFitter [1] is a software framework for statistical data analysis extensively used in the ATLAS Collaboration [2] in (mainly) searches for supersymmetric particles, analyzing data of proton-proton collisions produced by the Large Hadron Collider at CERN. HistFitter consists of a C++ part for CPU-intensive calculations and of a Python part for configuration purposes. The tool is built on top of the packages ROOT [6, 7], RooFit [5], HistFactory [3] and RooStats [4], where RooFit and HistFactory are used to build parametric models, RooFit to fit those models and RooStats to perform statistical tests.

HistFitter extends the functionalities of RooFit, HistFactory and RooStats in four key areas:

- HistFitter offers a programmable framework to perform complete statistical analyses starting from a user-defined configuration file.
- Typical analysis strategies in particle physics using control, validation and signal regions are deeply woven into the design of HistFitter.

- HistFitter keeps track of numerous data models including the construction and statistical tests in an organized way.
- HistFitter provides a collection of tools for interpreting results of statistical analyses such as determining the statistical significance of signal hypotheses, estimating the quality of likelihood fits and for producing plots and tables presenting the results.

2. Data analysis strategy with HistFitter

Particle physics experiments analyze large samples of data to measure properties of fundamental particles or to discover new physical processes. The data is usually interpreted using external predications for background and signal processes which are summarized in statistical models aiming to describe the observed data. HistFitter configures and builds such parametric models and provides various tools to help in the interpretation of the data.

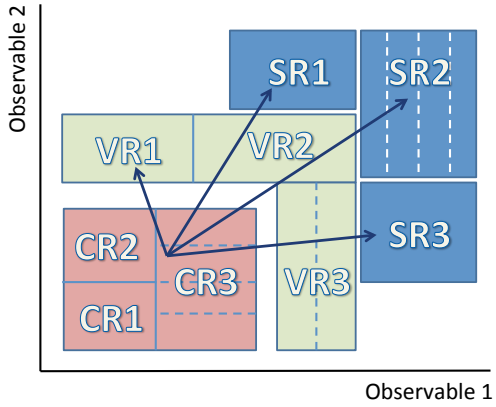

Figure 1. Schematic analysis strategy with control (CR1 - CR3), validation (VR1 - VR3) and signal regions (SR1 - SR3).

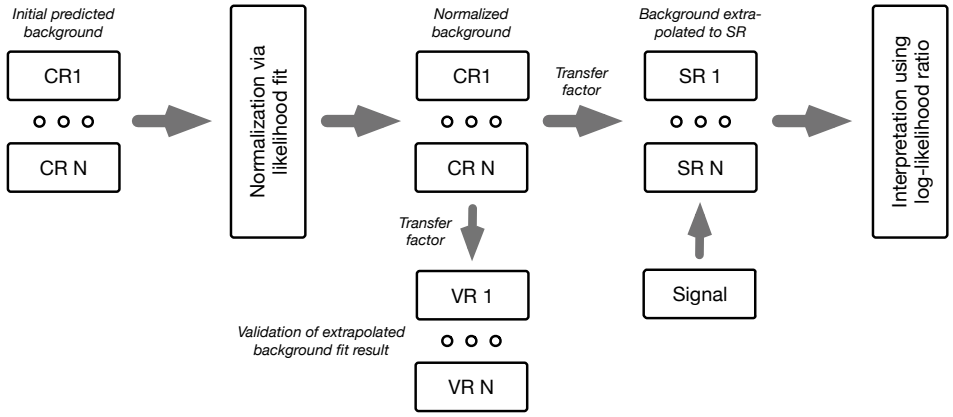

Figure 2. A typical analysis strategy flow with HistFitter involving the control regions CR 1 - CR N, validation regions VR 1 - VR N and signal regions SR 1 - SR N.

In the construction and handling of the models HistFitter deeply relies on the concept of statistically independent control, validation and signal regions as illustrated in Figures 1 and 2. Signal regions are regions in a phase space, defined by cutting on kinematic variables, that are enriched in potential signal in comparison to the predicted background level. To estimate the background in signal regions in a semi-data-driven way, control regions enriched in background

are defined. The predicted background is normalized to data in the control regions using a likelihood fit to data. The normalized background model is extrapolated to a signal region using a transfer factor, which is the ratio of expected event counts between each control and signal region. This extrapolation is verified in validation regions located between control and signal regions.

The parametric model describing the data is represented by a Probability Density Function (PDF). PDFs are built for every control, validation and signal region using HistFactory [3]. The regions being statistically independent, the PDFs can be fitted simultaneously to data, adjusting the parameters of the PDFs. An important point in the analysis strategy of HistFitter is the possibility to share parameters of the PDFs in different regions. This allows the consistent use of information on signal and background processes and systematic uncertainties in all regions.

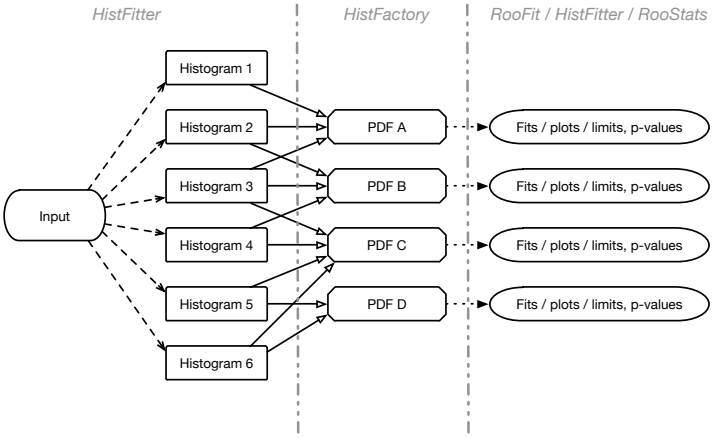

Figure 3. The HistFitter software framework.

The HistFitter software framework, illustrated in Figure 3, is designed based on the analysis strategy in Figure 2. Starting from a user-defined configuration and input data, binned histograms are created by HistFitter in a first processing step. In a second step, PDFs are constructed based on the histograms using HistFactory tools. In subsequent steps, the model is analyzed by performing fits, calculating limits and presenting the results in plots and tables. These steps use tools both from RooFit and RooStats, but also HistFitter specific tools as discussed in Section 4 and 5.

3. Configuration and construction of PDFs

The analysis flow presented requires substantial bookkeeping and an extended configuration machinery, in particular if working with hundreds of different models e.g. for different hypotheses for new physics beyond the Standard Model.

The configuration consists of a user-defined configuration file which interacts with a configuration manager within HistFitter. The configuration manager is realized as two singleton objects, one in Python with which the user interacts, the other one in C++. It organizes and creates `fitConfig` objects. A `fitConfig` object contains the PDF of a certain model along with meta-data providing information about fitting, visualization and interpretation of the model. The `fitConfig` object thus represents one row in Figure 3 and acts as factory of a model. The configuration manager thus functions as 'factory of factories'.

PDFs are constructed using HistFactory. The resulting likelihood has the general form of

$$L(\boldsymbol{n}, \boldsymbol{\theta}^0 | \mu_{\rm sig}, \boldsymbol{b}, \boldsymbol{\theta}) = P_{\rm SR} \times P_{\rm CR} \times C_{\rm syst}, \quad (1)$$

being the product of Poisson distributions of event counts in signal and control regions ($P_{\rm SR}$ and $P_{\rm CR}$) and of additional constraint terms for systematic uncertainties ($C_{\rm syst}$). The 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 has three different building blocks: `channels`, `samples` and `systematics`. `Channels` include all control, validation and signal regions. Signal and background processes compose the `samples` and `systematic` uncertainties summarize statistical, experimental and theoretical uncertainties.

HistFitter extends and mirrors the classes in HistFactory for these building blocks as illustrated in the following.

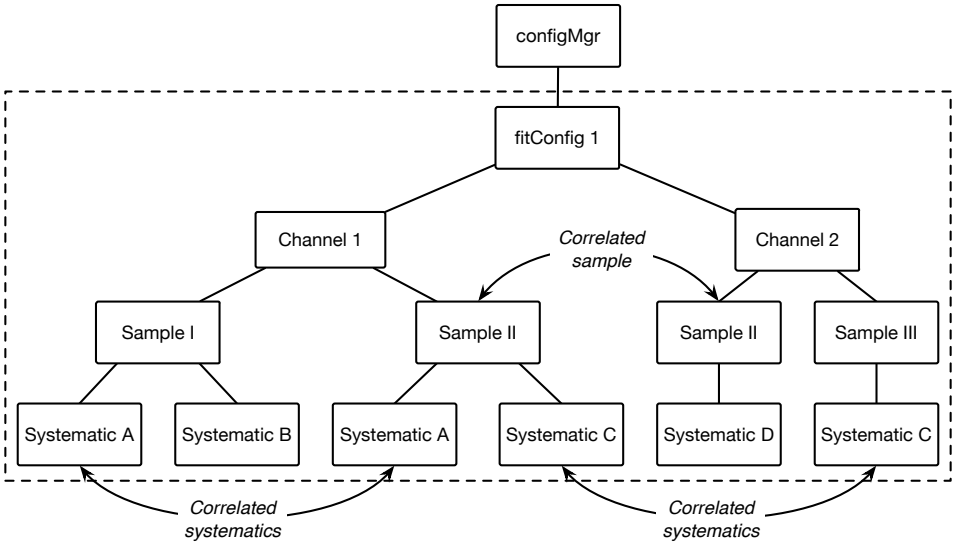

Figure 4. Illustration of an example fit strategy in HistFitter.

The three building blocks are put together by a `fitConfig` object in the construction of each PDF, as illustrated in Figure 4, and links them to the input data. A `fitConfig` object can have multiple `channels` that may be one- or multi-bin and are either control, validation or signal regions. `Samples`, corresponding to components of the PDF decorated with meta-data, are attached to a `channel`, and may also be correlated between multiple `channels`. The data input for the sample may be provided as ROOT `TTree` or `TH1` or also as float numbers by the user. `Systematics` attached to a `sample` and possibly correlated between `samples` are typically provided by 1σ up and down histograms of the nominal histogram of the sample(s). `Systematics` can be implemented using different methods of interpolation between up and down histograms and extrapolations. HistFitter allows the description of a complicated PDF by few lines of code through the 'trickle-down mechanism': this mechanism takes care of adding `samples` that were added to a `fitConfig` object also to all depending `channels`. Similarly, all `systematics` added to a `fitConfig` or `channel` are also propagated to the depending `channels` and `samples`.

The trickle-down mechanism together with the concepts of the `fitConfig` class and the configuration manager eases the construction of complex analyses setups considerably.

4. Fit strategies and presentation of fit results

HistFitter provides different fit strategies to fit the PDF as illustrated in Table 1. The *background-only* fit using only control regions and background samples aims for estimating the

background yields in validation and signal regions by extrapolating from the control regions. The *model-dependent signal fit* is used to derive exclusion limits on a specific model or to measure the properties of an excess. This fit strategy uses background and signal samples in control and signal regions. The simultaneous fitting of multiple signal regions is possible and often used to increase the exclusion power ('shape fit'). The *model-independent signal fit* is used to derive model-independent limits on the number of events beyond the expected number of events in a specific non-binned signal region (other signal regions are not allowed in this fit). No assumption is made on the signal which may only appear in the signal region but not in the control regions.

Table 1. Summary of the different fit strategies possible with HistFitter.

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

The results of the fits can be presented in e.g. tables or plots indicating the before and after fit event yields in the regions or in pull plots illustrating the agreement of the fitted background estimates with the observed data. Examples are given in Figures 5 and 6.

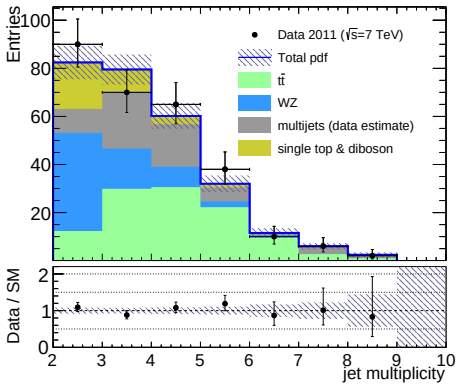

Figure 5. The jet multiplicity distribution after fitting it to data.

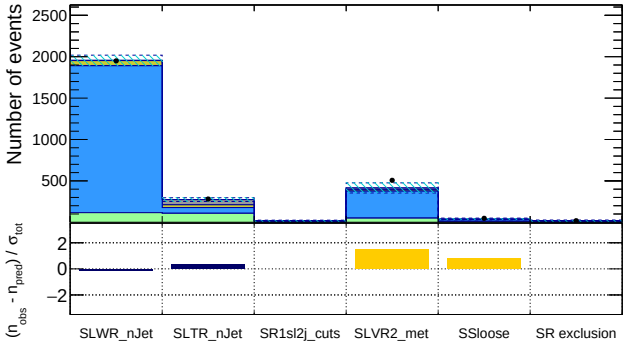

Figure 6. The agreement between estimated background yields and observed data (top) can be illustrated in a pull plot (bottom).

5. Interpretation of results

The observed data can be interpreted in hypothesis tests in HistFitter through calls to the appropriate functions and classes of RooStats [4]. HistFitter also provides macros for plots and tables to display the results.

The interpretations are based on either the model-dependent or model-independent signal fit. Using e.g. the model-dependent fit strategy, limits on a certain signal model can be placed by either deriving exclusion limits on this specific model in comparison to a background assumption in a hypothesis test or by evaluating an upper limit on the allowed signal strength given the data. Figure 7 shows an example giving the exclusion limits as function of parameters important for a specific model class.

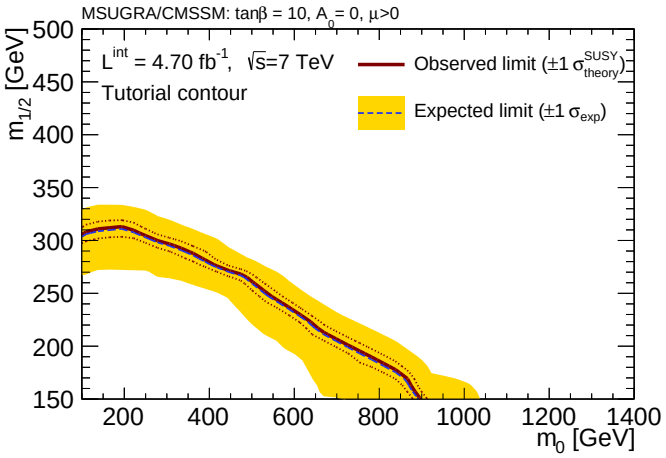

Figure 7. Exclusion limits in a class of supersymmetric models depending on the parameters on the x- and y-axes. Models below the red observed limit are excluded by hypothesis tests comparing a background-only with background+signal models.

6. Summary

We have presented a software framework for statistical analyses, called HistFitter. This programmable framework is used to build, fit and test data models of nearly arbitrary complexity. Starting from a user-defined configuration file and using HistFactory and RooFit functionalities PDFs are configured, constructed and fit. Interpretations of the models are possible through statistical tests using RooStats functionalities.

Among the innovative features of HistFitter is the modular configuration interface with a trickle-down mechanism which eases the construction of complicated PDFs. HistFitter is designed to provide the bookkeeping to work with hundreds of different signal models at once and thus provides an additional level of abstraction over existing tools. Built-in concepts of control, validation and signal regions allow a particular rigorous statistical treatment of extrapolations from control to signal regions as used in numerous background estimation techniques in particle physics. In addition, HistFitter offers a sizable collection of tools for presenting final results in a publication-style quality.

References

[1] Baak M, Besjes G J, Côté D, Koutsman A, Lorenz J and Short D 2014 HistFitter software framework for statistical data analysis *Preprint* arXiv:1410.1280 (`http://cern.ch/histfitter`)

[2] ATLAS Collaboration 2008 it JINST **3** S08003 (`http://atlas.web.cern.ch/Atlas/Collaboration`)

[3] ROOT Collaboration, Cranmer K, Lewis G, Moneta L, Shibata A and Verkerke W 2012 *Preprint* CERN-OPEN-2012-016

[4] Moneta L et al. 2010 *PoS* **ACAT2010** 057 *Preprint* arXiv:1009.1003

[5] Verkerke W and Kirkby D 2003 *eConf* **C0303241** (*Preprint* arXiv:physics/0306116)

[6] Brun R and Rademakers F 1997 *Nucl.Instrum.Meth.* **A389** 81

[7] Antcheva I et al. 2011 *Comput.Phys.Commun.* **182** 1384