Journal of Physics: Conference Series **664** (2015) 072004 doi:10.1088/1742-6596/664/7/072004

HistFitter: a flexible framework for statistical data analysis

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

[1] Niels Bohr Institute, University of Copenhagen, Blegdamsvej 17, 2100 Copenhagen, Denmark
[2] CERN, Route de Meyrin 385, 1217 Meyrin, Switzerland
[3] University of Texas at Arlington, 701 South Nedderman Drive, Arlington, TX, 76019, United States
[4] TRIUMF, 4004 Wesbrook Mall, Vancouver, BC V6T 2A3, Canada
[5] Fakultät für Physik, LMU München, Am Coulombwall 1, 85748 Garching, Germany
[6] Excellence Cluster Universe, Boltzmannstr. 2, 85748 Garching, Germany
[7] Department of Physics, Denys Wilkinson Building, University of Oxford, Keble Road, Oxford, OX1 3RH, United Kingdom

E-mail: geert-jan.besjes@cern.ch

Abstract. HistFitter is a software framework for statistical data analysis that has been used extensively 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.

HistFitter interfaces with the statistics tools HistFactory and RooStats to construct parametric models and to perform statistical tests of the data, and extends these tools in four key areas. The key innovations are to weave the concepts of control, validation and signal regions into the very fabric of HistFitter, and to treat these with rigorous methods. Multiple tools to visualize and interpret the results through a simple configuration interface are also provided.

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, in which data of proton-proton collisions produced by the Large Hadron Collider at CERN are analyzed. HistFitter consists of a C++ part, for CPU-intensive calculations, and a Python part, used for configuration purposes. The tool is built on top of the software packages ROOT [3, 4], RooFit [5], HistFactory [6] and RooStats [7]. RooFit and HistFactory are used to build parametric models, RooFit to fit those models and RooStats to perform statistical tests.

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

- it offers a programmable framework to perform complete statistical analyses starting from a user-defined configuration file;

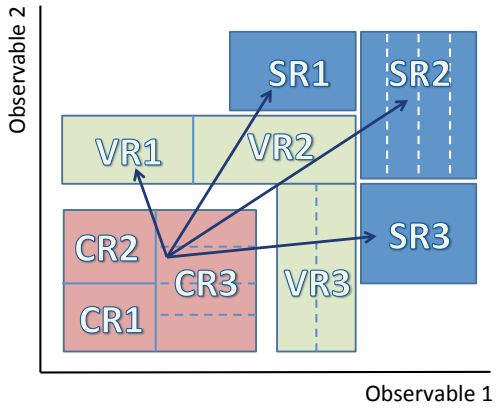

Figure 1. Schematic analysis strategy with control regions (CR1 - CR3) used to constrain the background predictions in the signal regions (SR1 – SR3). The obtained predictions are validated using three validation regions (VR1 – VR3).

- typical analysis strategies in particle physics rely on control, validation and signal regions in the phase space of one or more quantities. These types of regions are deeply woven into the design of HistFitter;
- it keeps track of numerous data models including the construction and statistical tests in an organized way;
- it provides a collection of tools for interpreting results of statistical analyses. These include tools to determine the statistical significance of signal hypotheses, to estimate the quality of likelihood fits and tools for production plots and tables that summarize results.

2. Data analysis strategy with HistFitter

In particle physics experiments, large samples of data are analyzed to measure properties of fundamental particles or to discover new physical processes. In order to interpret the measurements, predictions for background and signal processes must be compared. This is done though statistical models that describe the observed data. HistFitter configures and builds such parametric models and provides various tools to help in the interpretation of the data.

In the construction and handling of these 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 that are defined through selections on kinematic variables, for example the transverse momentum of one of the particles in an event. They should be designed to be enriched in potential signal of interest in comparison to the predicted background level.

In order 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 now 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. The extrapolation is verified using validation regions, located in phase space between the control and signal regions. A major benefit of these extrapolations is that systematic uncertainties that are identical in the ratios of predicted to observed numbers of events in the control and signal regions no longer impact the final systematic uncertainty on the total background prediction.

The parametric model that describes the data is represented by a Probability Density Function (PDF). PDFs are built for every control, validation and signal region using HistFactory [6]. Since the regions are statistically independent, the PDFs can be fitted simultaneously to data, adjusting their parameters in the process. An important feature in the analysis strategy of

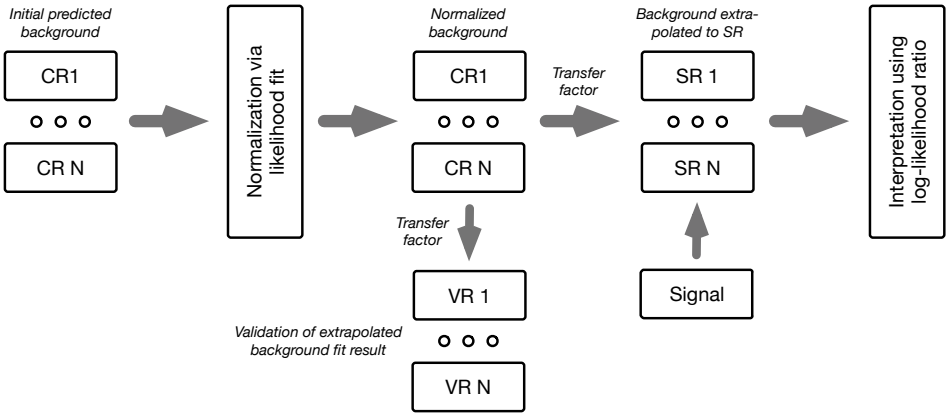

Figure 2. A typical analysis strategy flow with HistFitter involving the control regions CR 1 – CR N, validation regions VR 1 – VR N and signal regions SR 1 – SR N. Predictions from specific signal models can be used in the signal region to test their compatibility with the observed data.

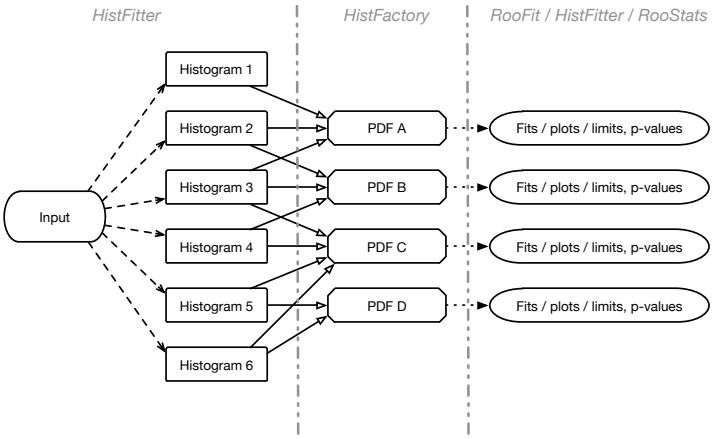

Figure 3. The HistFitter software framework utilizes HistFactory to build its probability density functions. It relies on tools in RooFit and RooStats to perform fits and hypothesis tests, and provides user-friendly tools to present the results.

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.

The HistFitter software framework is designed based on the analysis strategy in Figure 2 and illustrated in Figure 3. 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, the calculations of limits. The results can be presented easily in plots and tables. These steps utilize tools both from RooFit and RooStats, as well as HistFitter-specific tools that are described in Sections 4 and 5.

3. Configuration and construction of PDFs

The analysis flow presented requires substantial bookkeeping and an extended configuration machinery. Inn particular if working with hundreds of different models this is the case, as for example for many different hypotheses for new physics beyond the Standard Model.

In HistFitter, an analysis configuration consists of a user-defined configuration file that interacts with a configuration manager. The configuration manager is implemented as two

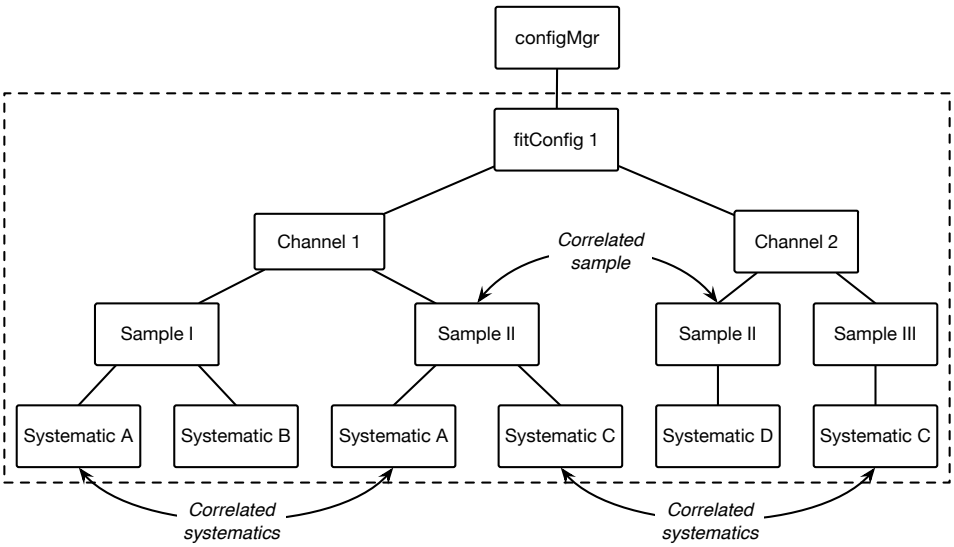

Figure 4. Illustration of an example fit strategy in HistFitter. Samples can be shared between various channels and systematics can be shared between various samples, allowing easy configuration of complex models.

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

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

$$\begin{aligned} L(\boldsymbol{n}, \boldsymbol{\theta}^0 | \mu_{\rm sig}, \boldsymbol{b}, \boldsymbol{\theta}) &= P_{\rm SR} \times P_{\rm CR} \times C_{\rm syst} \\ &= P(n_S | \lambda_S(\mu_{\rm sig}, \boldsymbol{b}, \boldsymbol{\theta})) \times \prod_{i \in {\rm CR}} P(n_i | \lambda_i(\mu_{\rm sig}, \boldsymbol{b}, \boldsymbol{\theta})) \times C_{\rm syst}(\boldsymbol{\theta}^0, \boldsymbol{\theta}) \,, \quad (1) \end{aligned}$$

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 $\boldsymbol{n}$ in all regions, nuisance parameters $\boldsymbol{\theta}$ that parameterize the impact of systematic uncertainties with their central values $\boldsymbol{\theta}^0$, a signal strength $\mu_{\rm sig}$ and the predictions $\boldsymbol{b}$ for the 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 are `Samples` in these channels. Statistical, experimental and theoretical uncertainties on processes are implemented as `Systematics`.

HistFitter extends and mirrors the classes in HistFactory for these building blocks. The three building blocks are put together through a `fitConfig` object in the construction of each PDF, which links them to the input data. A schematic overview is given in Figure 4. A `fitConfig` object can have multiple `channels` that may be one- or multi-bin and can be either control, validation or signal regions. `Samples`, corresponding to components of the PDF decorated with meta-data, are attached to a `Channel`. They can also be correlated between multiple `Channels`. The data input for the sample can be provided as ROOT `TTree` or `TH1`, and also as floating-point numbers. `Systematics` attached to a `Sample` are typically provided by 1σ up and down

Table 1. Subset 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, floating-point numbers or histogram input methods in all cases.

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

variations w.r.t. the nominal histogram of the `Sample`. They can be correlated between multiple `Samples`. `Systematics` can be implemented using different methods of interpolation between up and down histograms and extrapolations. These methods are listed in Table 1.

HistFitter allows the description of a complicated PDF by few lines of code through a 'trickle-down mechanism': `Samples` added to a `fitConfig` object are added to all `Channels`. Similarly, all `Systematics` added to a `fitConfig` or `Channel` are propagated down to the level of `Samples`. This design, 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. These are listed in Table 2.

The *background-only* fit is used to estimate the number of events in the signal and/or validation regions through an extrapolation from the control regions. It constrains the PDF in only 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 over the number of events given by the background-only hypothesis. 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 allowed on top of the expected number of events in a specific, non-binned, signal region.

Table 2. Summary of the different fit strategies possible with HistFitter.

Fit setup:	*Background-only fit*	*Model-dependent signal fit*	*Model-independent signal fit*
Purpose	search for excess	limit setting	upper limit on N_{sig}
Samples used	backgrounds	backgrounds + signal	backgrounds + dummy signal
Fit regions	CR(s)	CR(s) + SR(s)	CR(s) + SR

Other signal regions are not allowed in this fit. No assumption is made on the signal, which is only allowed to appear in the signal region and not in any of the control regions. These constraints lead to the necessity of exactly one single-bin signal region in this fit.

The results of the fits can be easily presented in multiple ways. HistFitter provides scripts to produce tables and plots indicating the before and after fit event yields in the regions, as well as scripts to produce pull plots that illustrate the agreement of the fitted background estimates with the observed data. Examples of such plots are given in Figures 5 and 6.

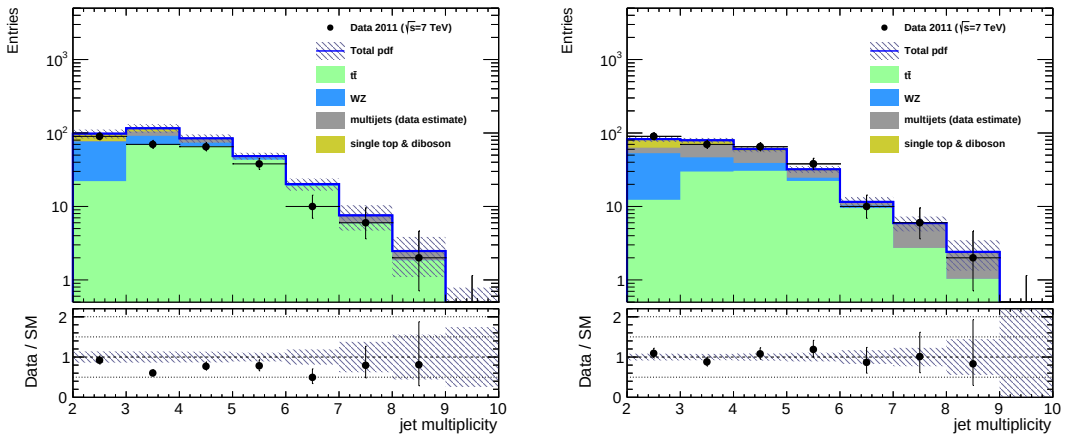

Figure 5. The jet multiplicity distribution in a control region before (left) and after (right) fitting it to the observed data. The benefit of constraining background predictions to data in control regions is readily apparent.

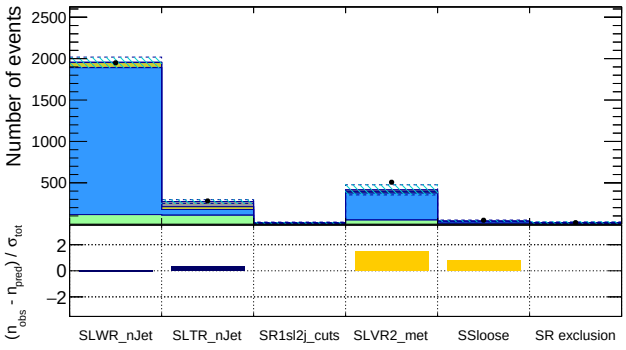

Figure 6. The agreement between estimated background yields and observed data in the validation and/or signal regions (top) can be illustrated in a pull plot (bottom). The data in signal regions can be blinded to aid the optimization of control region definitions.

5. Interpretation of results

In order to draw conclusions whether observed data is compatible with a certain hypothesis, statistical tests must be performed. The observed data can be interpreted in hypothesis tests in HistFitter through calls to the appropriate functions and classes of RooStats [7]. HistFitter also provides macros for the production of plots and tables to display the results.

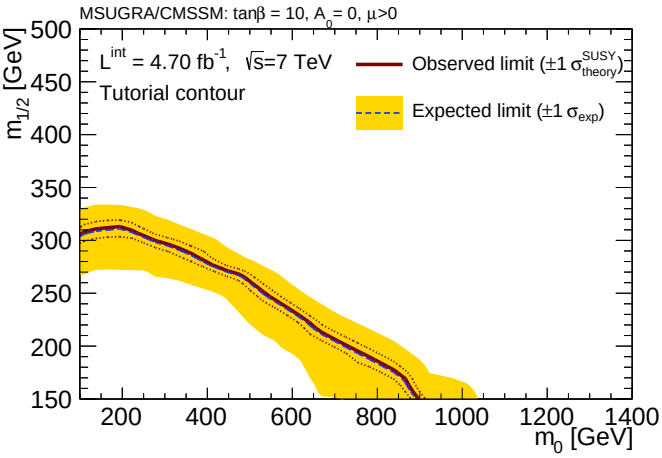

Figure 7. Exclusion limits in a class of supersymmetric models that are characterized by the parameters on the x- and y-axes. Models below the red observed limit are excluded. The result is derived through hypothesis tests that compare signal models to the observed data; the lines are an interpolation between the results of these tests.

The interpretations are based on either the model-dependent or model-independent signal fit. Limits on a particular signal model can be placed in two ways using the model-dependent fit strategy. Exclusion limits on this specific model can be derived in a hypothesis test, given a null hypothesis and a measurement. Alternatively, an upper limit on the allowed signal strength given the data can be derived through repeated hypothesis tests with varying signal strengths. Figure 7 shows a graphical example of the exclusion limits derived for a particular set of models, as a function of two parameters that determine the models' properties.

6. Summary

We have presented HistFitter, a software framework for statistical analyses. HistFitter is a programmable framework used to build, fit and test data models of nearly arbitrary complexity. Starting from a user-defined configuration file and using HistFactory and RooFit functionality PDFs are configured, constructed and fit. Interpretations of the constructed models are obtained through statistical tests that rely on functionality available in the RooStats software package.

Among the innovative features of HistFitter is the modular configuration interface, with a trickle-down mechanism that eases the construction of complicated PDFs. HistFitter is designed to provide the bookkeeping to work with hundreds of different signal models at once, providing an additional level of abstraction and ease of configuration 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 commonly 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.

7. Public release

The HistFitter software package is publicly available through `http://cern.ch/histfitter` and requires ROOT release v5.34.20 or greater. The page contains a description of the source code, a tutorial on how to set up an analysis, and working examples of how to run HistFitter.

References

[1] Baak M, Besjes G J, Côte D, Koutsman A, Lorenz J *et al.* 2015 *Eur. Phys. J.* **C 75** 153 (*Preprint* `1410.1280`)

[2] ATLAS Collaboration (ATLAS Collaboration) 2008 *JINST* **3** S08003
[3] Brun R and Rademakers F 1997 *Nucl. Instrum. Meth.* **A 389** 81–86
[4] Antcheva I, Ballintijn M, Bellenot B, Biskup M, Brun R *et al.* 2011 *Comput. Phys. Commun.* **182** 1384–1385
[5] Verkerke W and Kirkby D P 2003 *eConf* **C0303241** MOLT007 (*Preprint* `physics/0306116`)
[6] Cranmer K, Lewis G, Moneta L, Shibata A and Verkerke W (ROOT Collaboration) 2012 *CERN-OPEN-2012-016*
[7] Moneta L, Belasco K, Cranmer K S, Kreiss S, Lazzaro A *et al.* 2010 *PoS* **ACAT2010** 057 (*Preprint* `1009.1003`)