RooUnfold: unfolding framework and algorithms

Tim Adye[1], *Kerstin Tackmann*[2], *Fergus Wilson*[1]

[1]Rutherford Appleton Laboratory, Science and Technology Facilities Council, Harwell Science and Innovation Campus, Didcot OX11 0QX, United Kingdom
[2]CERN, CH-1211 Geneva 23, Switzerland

DOI: will be assigned

1 RooUnfold package aims and features

The RooUnfold package [1] was designed to provide a framework for different unfolding algorithms. This approach simplifies the comparison between algorithms and has allowed common utilities to be written.

Currently RooUnfold implements the iterative Bayes [2], Singular Value Decomposition (SVD) [3], and correction factors methods. In these cases RooUnfold was interfaced to existing code, which was then adapted and improved in-situ. We plan to continue this process with other algorithms.

The package is designed around a simple object-oriented approach, implemented in C++, and using existing ROOT [4] classes. RooUnfold defines classes for the different unfolding algorithms, which inherit from a common base class, and a class for the response matrix. The response matrix object is independent of the unfolding, so can be filled in a separate "training" program.

RooUnfold can be linked into a stand-alone program, run from a ROOT/CINT script, or executed interactively from the ROOT prompt. The response matrix can be initialised using existing histograms or matrices, or filled with build-in methods (these can take care of the normalisation when inefficiencies are to be considered). The results can be returned as a histogram with errors, or a vector with full covariance matrix.

The framework also takes care of handling multi-dimensional distributions (with ROOT support for 1–, 2–, and 3–dimensional histograms), different binning for measured and truth distributions, variable binning, and the option to include or exclude of under- and over-flows. These details are handled by the framework, so don't have to be implemented for each algorithm. However different bin layouts may not produce good results for algorithms that rely on the global shape of the distribution (SVD).

Common utility routines are provided for displaying the results with resolutions, pulls, and χ^2. A toy Monte Carlo (MC) test framework is also provided, allowing selection of different MC probability distribution functions (PDF) and parameters, comparing different binning, and performing the unfolding with the different algorithms and varying the unfolding regularisation parameters. Tests can be performed with 1D, 2D, and 3D distributions. A few example tests are presented in section 4.

2 C++ classes

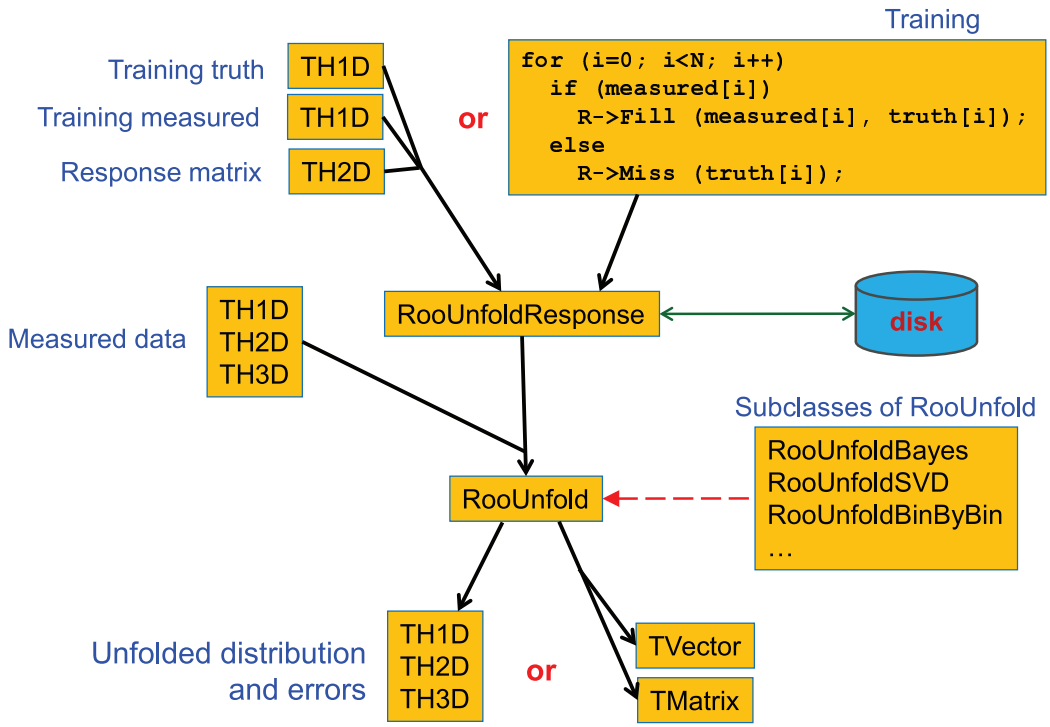

Figure 1: RooUnfold classes

Figure 1 summarises how the ROOT and RooUnfold classes are used together. The RooUnfoldResponse object can be constructed using a 2D response histogram (TH2D) and 1D truth and measured projections (these are required to determine the effect of inefficiencies). Alternatively, RooUnfoldResponse can be filled directly with the

`response->Fill(`x_{measured}`,` x_{true}`)`

and

`response->Miss(`x_{true}`)`

methods, where the `Miss` method is used to count an event that was not measured and should be counted towards the inefficiency.

The RooUnfoldResponse object can be saved to disk using the usual ROOT input/output streamers. This allows the easy separation in separate programs of MC training from the unfolding step.

A RooUnfold object is constructed using a RooUnfoldResponse object and the measured data. It can be constructed as a RooUnfoldBayes, RooUnfoldSVD, or RooUnfoldBinByBin object, depending on the algorithm required.

The results of the unfolding can be obtained as ROOT histograms (TH1D, TH2D, or TH3D) or as a ROOT vector (TVectorD) and covariance matrix (TMatrixD). The histogram will include just the diagonal elements of the error matrix, which may be misleading given the significant correlations that can occur if there is much bin-to-bin migration.

3 Unfolding algorithms

3.1 Iterative Bayes' theorem

The RooUnfoldBayes algorithm uses the method described by Giulio D'Agostini in [2].

Repeated application of Bayes' theorem is used to invert the response matrix. Regularisation is achieved by stopping iterations before reaching "true" (but wildly fluctuating) inverse. The regularisation parameter is just the number of iterations. In principle, this has to be tuned according to the sample statistics and binning. In practice, the results are fairly insensitive to the precise setting used.

RooUnfoldBayes takes the training truth as its initial prior, rather than a flat distribution, as described by D'Agostini. This should not bias result once we have iterated, but could reach an optimum after fewer iterations.

This implementation takes account of multinomial errors on the data sample but not, by default, uncertainties in the response matrix due to finite MC statistics. That calculation can be very slow, and usually the training sample is much larger than the data sample.

RooUnfoldBayes does not normally do smoothing, since this has not been found to be necessary and can, in principle, bias the distribution. Smoothing can be enabled with an option.

3.2 Singular Value Decomposition

RooUnfoldSVD uses the method of Andreas Höcker and Vakhtang Kartvelishvili [3] and was originally implemented for the *BABAR* $B \to X_u l\nu$ measurement [5].

The response matrix is inverted using singular value decomposition, which allows for a linear implementation of the unfolding algorithm. The normalisation to the number of events is retained in order to minimize uncertainties due to the size of the training sample. Regularisation is performed using a smooth cut-off on small singular value contributions, which correspond to high-frequency fluctuations.

The regularisation needs to be tuned for the particular type of distribution, number of bins, and approximate sample size in order to not generate any significant bias due to the choice of the training sample while retaining small statistical fluctuations in the unfolding result.

The unfolded error matrix includes the contribution of uncertainties on the response matrix due to finite MC training statistics, though these are usually quite small.

3.3 Simple correction factors

A very simple algorithm using bin-by-bin correction factors, with no inter-bin migration is included to aid comparison.

4 Examples

Examples of toy MC tests generated by RooUnfoldTest are shown in Figures 2–4.

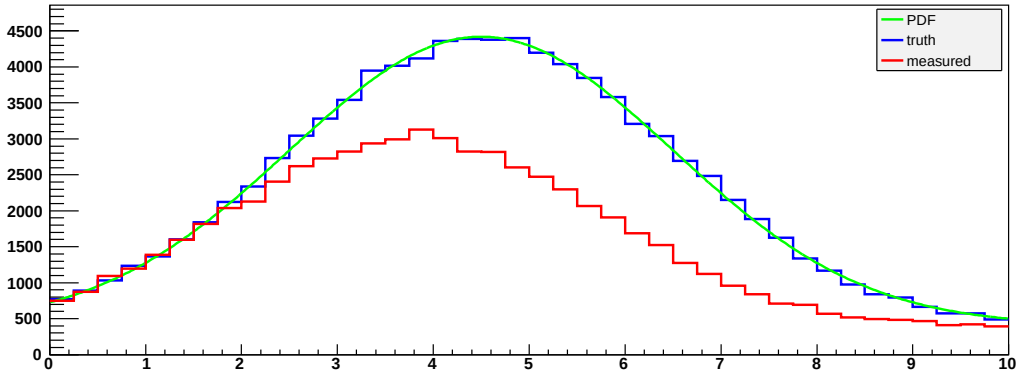

Figure 2: Example of a toy MC training run. A Gaussian PDF on a flat background (green curve) is used to generate a training "truth" sample (upper histogram in blue). This is then smeared, shifted, and a variable inefficiency applied to produce the "measured" distribution (lower histogram in red).

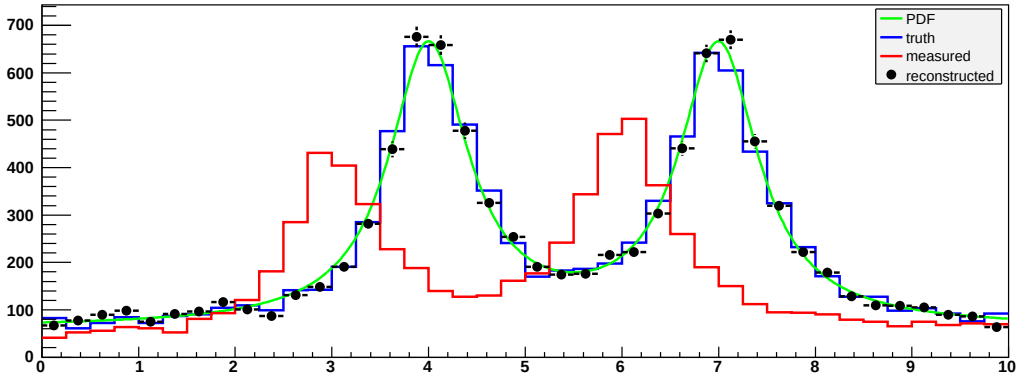

Figure 3: Unfolding with the Bayes algorithm. A double Breit-Wigner PDF on a flat background (green curve) is used to generate a training "truth" sample (upper histogram in blue). The same resolution effects as were used in training (Figure 2) were applied to produce the "measured" distribution (lower histogram in red). Applying the Bayes algorithm with 3 iterations on this latter gave the unfolded result (black points), shown with errors from the diagonal elements of the error matrix.

5 Status and Plans

RooUnfold was first developed in the *BABAR* software environment. It was released standalone in 2007. Since then, it has been used by physicists from many different particle physics,

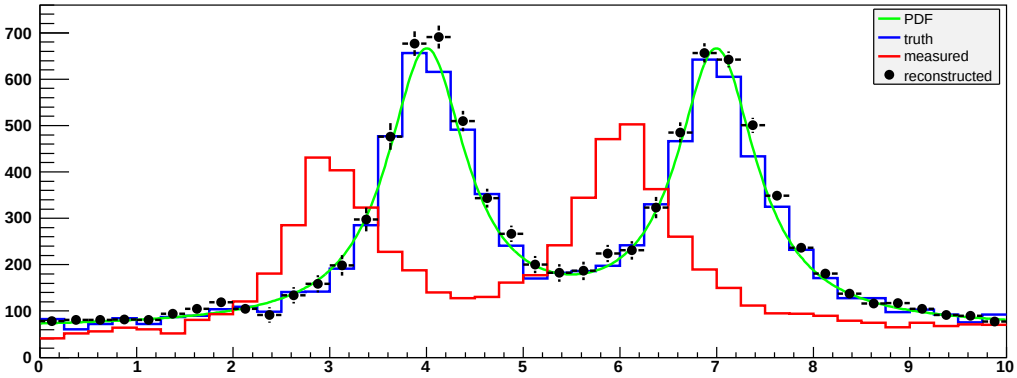

Figure 4: Unfolding with the SVD algorithm ($k = 20$) on the same training and test samples as described in Figures 2 and 3.

particle-astrophysics, and nuclear physics groups. Questions, suggestions, and even a few bug reports from users have prompted new versions with fixes and improvements.

We plan further improvements, namely to

- provide additional input/output methods (eg. RooFit [6] datasets);
- add common tools, useful for all algorithms, such as automating the validation and selection of regularisation parameters;
- interface with additional algorithms; and
- publish RooUnfold as an official ROOT package as part of the RooStats collection [7].

References

[1] The RooUnfold package and documentation are available from `http://hepunx.rl.ac.uk/~adye/software/unfold/RooUnfold.html`

[2] G. D'Agostini, "A Multidimensional unfolding method based on Bayes' theorem," Nucl. Instrum. Meth. A **362** (1995) 487.

[3] A. Hocker and V. Kartvelishvili, "SVD Approach to Data Unfolding," Nucl. Instrum. Meth. A **372** (1996) 469 [arXiv:hep-ph/9509307].

[4] R. Brun and F. Rademakers, "ROOT: An object oriented data analysis framework," Nucl. Instrum. Meth. A **389** (1997) 81. See also `http://root.cern.ch/`.

[5] K. Tackmann [*BABAR* Collaboration], "Determination of the b-quark mass and nonperturbative parameters in semileptonic and radiative penguin decays at *BABAR*," Eur. Phys. J. A **38** (2008) 137 [arXiv:0801.2985 [hep-ex]].

[6] W. Verkerke and D. P. Kirkby, "The RooFit toolkit for data modeling," arXiv:physics/0306116.

[7] RooStats Wiki, `https://twiki.cern.ch/twiki/bin/view/RooStats/WebHome`