Document version 0

July 30, 2009

RooStats

User's Guide

Kyle Cranmer, Gregory Schott, Lorenzo Moneta, Wouter Verkerke

With contributions from:

Danilo Piparo, Giacinto Piacquadio, Maurizio Pierini, George H. Lewis, Alfio Lazzaro, Mario Pelliccioni

Contents

1 Introduction

The RooStats project aims to provide a comprehensive, flexible, and validated suite of statistical tools within ROOT. Early on in the project it was decided to leverage the data modeling approach developed in RooFit, which is already well-established within high-energy physics and beyond. Thus, RooStats can be seen as providing high-level statistical tools, while RooFit provides the core data modeling language as well as many low-level aspects of the functionality. In the ongoing process of developing RooStats, RooFit is also undergoing rapid development.

One of the major goals of RooStats is to provide a unified framework for different statistical methods. Early on in the project it was demonstrated that RooFit is well suited for the three major types of statistical inference:

Classical / Frequentist This "school" of statistics restricts itself to making statements of the form "probability of the data given the hypothesis". The definition of probability in this context is based on a limit of frequencies of various outcomes. In that sense it is objective.

Bayesian This "school" of statistics allows one to make statements of the form "probability of the hypothesis given the data", which requires a prior probability of the hypothesis. Often the definition of probability in this context is a "degree of belief".

Likelihood-based This intermediate approach does not require a prior for the hypothesis, but also is not guaranteed to poses the properties that frequentists methods aim to achieve (or achieve by construction). These approaches do "obey the likelihood principle" (as do Bayesian methods), while frequentist methods do not.

The developers of RooStats appreciate that there are many different approaches to answering the same types of problems. There are pros and cons of various techniques, and we aim to provide a framework which can accommodate any of them.

One Model, Many Methods

A common scenario that we hope to address with RooStats is the comparison of different statistical approaches for the same statistical problem. Without a unified framework, these comparisons are complicated by the fact that each method must: a) re-create the model (eg. probability density function(s) for the data) and b) represent the data itself. Having redundant modeling and data representation is error prone and often complicates the comparison (or makes it practically impossible).[1] In that sense, RooStats aims to be like TMVA[2], providing utilities to easily try and compare multiple statistical techniques. By relieving the technical overhead associated to these types of comparisons, the hard work can go into improved modeling of the problem at hand and asking better questions.

[1] Of course, these types of cross-checks can also be very useful!

[2] The Toolkit for MultiVariate Analysis is also distributed in ROOT http://tmva.sourceforge.net.

Types of Statistical Questions

One of the first steps in any statistical analysis is to carefully pose the question that one wishes to answer. Most of these questions can be classified as follows:

Parameter Estimation Find the most likely ('best fit') values of the parameters of a model given data.

Hypothesis Testing Based on the data accept or reject a given hypothesis. Often one tests a null hypothesis against an alternative. When the hypothesis has no free parameters it is called 'simple' and when it has free parameters it is called 'composite'.

Confidence intervals Find a region in the parameter space that is consistent with the data. In the frequentist setting, one desires for this interval or region to 'cover' the true parameter point with a specified probability (or confidence). In the Bayesian setting, one wishes for the interval or region to contain some fixed amount of the posterior probability.

Goodness of Fit Quantify how well a model fits the data, without reference to an alternative.

RooStats provides tools for each of these broad class of questions in addition to some miscellaneous utilities. The design of the software is explicitly organized around these broad classes of questions: for instance the interface `IntervalCalculator` is common to all tools that produce `ConfidenceIntervals`.

Combining Results & Digital Publishing

Combining results from multiple experiments in order to enhance sensitivity of a measurement or improve the power of a hypothesis test is common. The challenge of combining results is primarily logistical, since a proper combination requires low-level information from each experiment be brought together to form one large statistical test. Again, this is hindered by the fact that the ingredients to the combination are heterogeneous (eg. different formats, technologies, and conventions).

The major advancement that was made by the RooStats project is the concept of the *workspace*. The power of the workspace is that it allows one to save data and an arbitrarily complicated model to disk in a ROOT file. These files can then be shared or archived, and they provide all the low-level ingredients necessary for a proper combination in a unified framework. The `RooWorkspace` provides this low-level functionality, thus it is technically part of RooFit (along with its documentation and several tutorial macros).

This form of digital publishing has great potential, consider a few examples: It allows for one to publish likelihood functions in n-dimensions instead of resorting to 2-dimensional contours. It allows for one to interface the likelihood function to even higher-level software packages (eg. extraction of fundamental lagrangian parameters from experimental measurements). It allows for one to generate toy data for the observables given any parameter point, which is necessary for a truly frequentist calculation. It allows for combinations with other experiments as already mentioned.

1.1 Getting Started

Since December 2008, RooStats has been distributed in the ROOT release since version 5.22 (December 2008). To use RooStats, you need a version of ROOT greater than 5.22, but you will probably want the most recent ROOT version since the project is developing quickly.

Option 1) Download the binaries for the latest ROOT release

You can download the most recent version of ROOT here: http://root.cern.ch/

Option 2) Check out and build the ROOT trunk

If you prefer to build ROOT from source,

```
svn co http://root.cern.ch/svn/root/trunk root
```

then build and install ROOT via (you may want different configure options)

```
configure --enable-roofit
make
make install
```

Option 3) Check out and build the RooStats branch

If you need a development or bug-fix that is not yet in a ROOT release, you can download the most recent version of the code from ROOT's subversion repository. To check it out, go to some temporary directory and type:

```
 svn co https://root.cern.ch/svn/root/branches/dev/roostats root
```

then build and install ROOT via (you may want different configure options)

```
configure --enable-roofit
make
make install
```

For more information about building ROOT from source, see the ROOT webpage: http://root.cern.ch/drupal/content/installing-root-source.

1.2 Other Resources

The RooStats Web Page:
https://twiki.cern.ch/twiki/bin/view/RooStats/

The Root User's Guide:
http://root.cern.ch/drupal/content/users-guide

The RooFit User's Guide:
ftp://root.cern.ch/root/doc/RooFit_Users_Manual_2.91-33.pdf

RooFit Tutorials:
http://root.cern.ch/root/html/tutorials/roofit/

RooStats Tutorials:
http://root.cern.ch/root/html/tutorials/roostats/

1.3 Terminology used in this guide

model a probability density function that describes some observables. We use the term model for both parametric models (eg. a Gaussian is parametrized by a mean and standard deviation) and non-parametric models (eg. histograms or KEYS pdfs).

observable(s) quantities that are directly measured by an experiment and present in a data set. The distribution of the observables are predicted by the model. Models are normalized such that the integral of the model over the observables is 1.

auxiliary observable observables that are come from an auxiliary experiment (eg. a control sample or a preceding experiment).

parameter of interest quantities used to parametrize a model that are 'interesting' in the sense that one wishes to estimate their values, place limits on them, etc (eg. masses, cross-sections, and the like).

nuisance parameter quantities used to parametrize a model that are uncertain but not 'interesting' in the above sense (eg. background normalization, shape parameters associated to systematic uncertainties, etc.)

control sample a data set independent of the main measurement (defining auxiliary observables) often used to constrain nuisance parameters by simultaneous considering it together with the main measurement.0

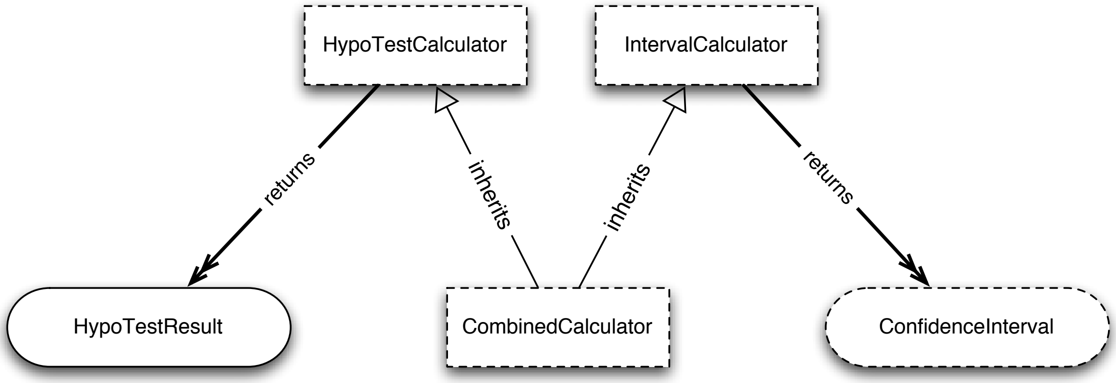

Figure 1: **default**

1.4 Types of Statistical Tools in RooStats

1.5 Design Philosophy: Mapping Mathematics to Software

Mathematical representation:

$$G(x|\mu, \sigma)$$

Representation in RooFit / RooStats code:

```
{
  // Make observable and parameters
  RooRealVar x("x","x", 150,100,200);
  RooRealVar mu("mu","#mu", 150,130,170);
  RooRealVar sigma("sigma","#sigma", 5,0,20);

  // make a simple model
  RooGaussian G("G","G",x, mu, sigma);

  // make graph to represent model (using GraphViz and latex formatting)
  G.graphVizTree("GaussianModel.dot", ":",true, true);
}
```

Graphical Representation

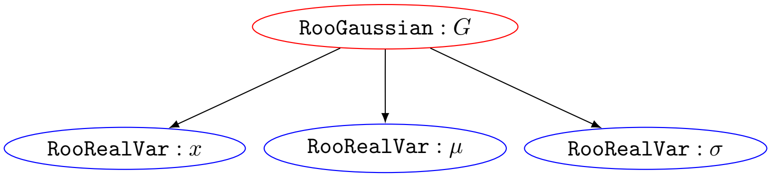

Figure 2: test

2 Parameter Estimation

3 Test Statistics and Sampling Distributions

3.1 TestStatistic interface and implementations

We added a new interface class called TestStatistic. It defines the method Evaluate(data, parameterPoint), which returns a double. This class can be used in conjunction with the ToyMCSampler class to generate sampling distributions for a user-defined test statistic.

The following concrete implementations of the TestStatistic interface are currently available

ProfileLikelihoodTestStatReturns the log of profile likelihood ratio. Generally a powerful test statistic. NumEventsTestStatReturns the number of events in the dataset. Useful for number counting experiments. DebuggingTestStat Simply returns a uniform random number between 0,1. Useful for debugging. SamplingDistribution and the TestStatSampler interface and implementations

We introduced a "result" or data model class called SamplingDistribution, which holds the sampling distribution of an arbitrary real valued test statistic. The class also can return the inverse of the cumulative distribution function (with or without interpolation).

We introduced an interface for any tool that can produce a SamplingDistribution, called TestStatSampler. The interface is essentially GetSamplingDistribution(parameterPoint) which returns a SamplingDistribution based on a given probability density function. We foresee a few versions of this tool based on toy Monte Carlo, importance sampling, Fourier transforms, etc. The following concrete implementation of the TestStatSampler interface are currently available

ToyMCSamplerUses a Toy Monte Carlo approach to build the sampling distribution. The pdf's generate method to generate is used to generate toy data, and then the test statistic is evaluated at the requested parameter point. DebuggingSampler Simply returns a uniform distribution between 0,1. Useful for debugging. NeymanConstruction and FeldmanCousins

A flexible framework for the Neyman Construction was added in this release. The NeymanConstruction is a concrete implementation of the IntervalCalculator interface, but it needs several additional components to be specified before use. The design factorizes the choice

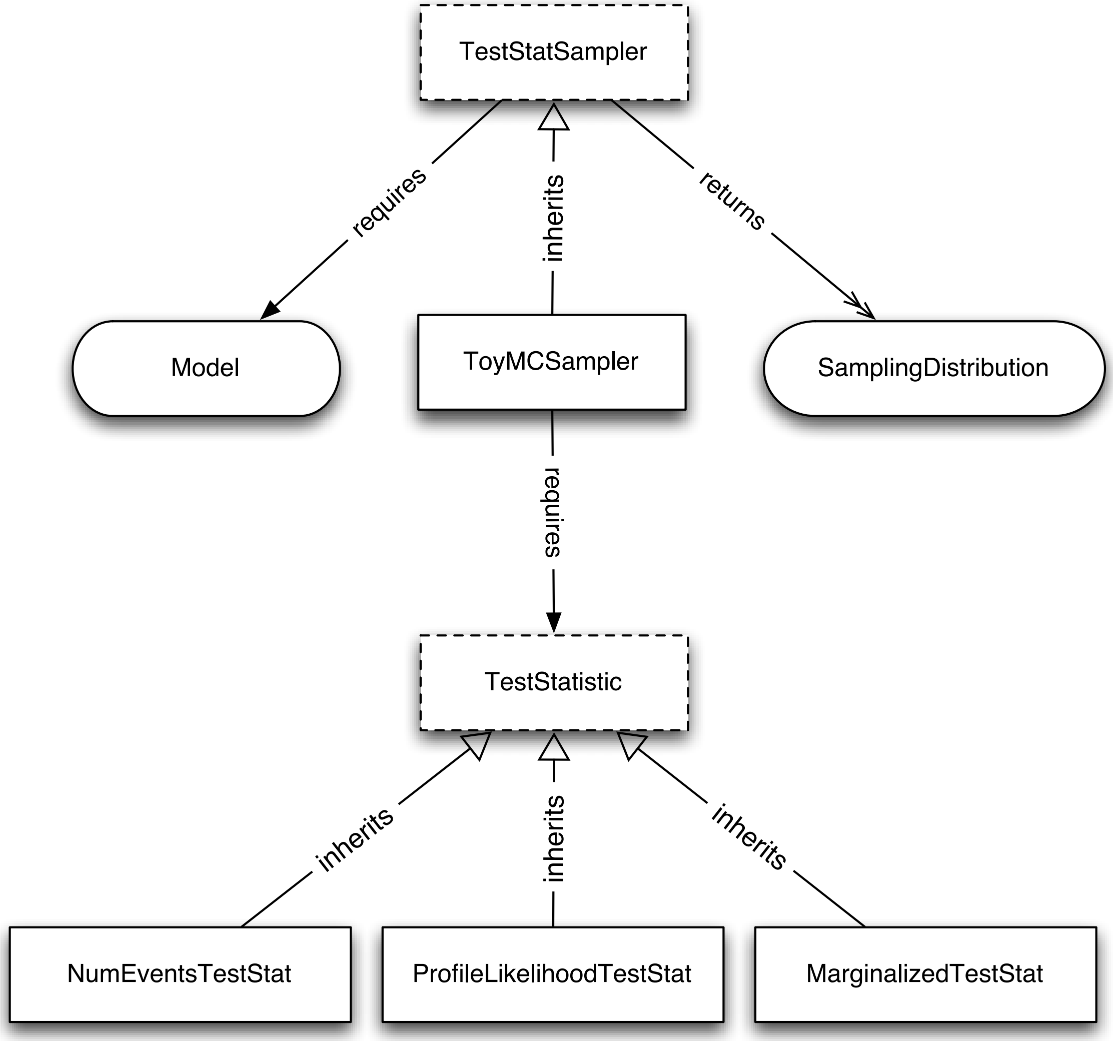

Figure 3: **default**

of the parameter points to be tested, the choice of the test statistic, and the generation of sampling distribution into separate parts (described above). Finally, the NeymanConstruction class is simply in charge of using these parts (strategies) and constructing the confidence belt and confidence intervals. The ConfidenceBelt class is still under development, but the current version works fine for producing ConfidenceIntervals. We are also working to make this class work with parallelization approaches, which is not yet complete.

The FeldmanCousins class is a separate concrete implementation of the IntervalCalculator interface. It uses the NeymanConstruction internally, and enforces specific choices of the test statistic and ordering principle to realize the Unified intervals described by Feldman and Cousins in their paper Phys.Rev.D57:3873-3889,1998.

4 Hypothesis Testing

5 Confidence Intervals

5.1 Profile Likelihood Ratio (the method of MINOS)

5.2 Neyman Construction

5.3 Feldman-Cousins

5.4 Neyman Construction with nuisance parameters

5.5 The "Profile Construction"

6 Goodness of Fit

7 Coverage Studies

8 Utilities

8.1 The Number Counting PDF Factory

8.2 RooStatsUtils

8.2.1 Standalone number counting functions

8.2.2 Converting between p-values and significance

8.3 SPlot

This initial description of `RooStats::SPlot` is taken directly from the documentation of http://root.cern.ch/root/html/TSPlot.html. It mainly describes the method, which is common to both the RooStats implementation and `TSPlot`. The main difference between the implementations is that the RooStats implementation allows one to use arbitrary models created with RooFit's data modeling language.

A common method used in High Energy Physics to perform measurements is the maximum Likelihood method, exploiting discriminating variables to disentangle signal from background. The crucial point for such an analysis to be reliable is to use an exhaustive list of sources of events combined with an accurate description of all the Probability Density Functions (PDF).

To assess the validity of the fit, a convincing quality check is to explore further the data sample by examining the distributions of control variables. A control variable can be obtained for instance by removing one of the discriminating variables before performing again the maximum Likelihood fit: this removed variable is a control variable. The expected distribution of this control variable, for signal, is to be compared to the one extracted, for signal, from the data sample. In order to be able to do so, one must be able to unfold from the distribution of the whole data sample.

The TSPlot method allows to reconstruct the distributions for the control variable, independently for each of the various sources of events, without making use of any a priori knowledge on this variable. The aim is thus to use the knowledge available for the discriminating variables to infer the behaviour of the individual sources of events with respect to the control variable.

TSPlot is optimal if the control variable is uncorrelated with the discriminating variables.

A detail description of the formalism itself, called ${}_s\mathcal{P}lot$

M. Pivk and F. R. Le Diberder, Nucl. Inst. Meth. A (in press), physics/0402083

8.3.1 The method

The ${}_s\mathcal{P}lot$ technique is developped in the above context of a maximum Likelihood method making use of discriminating variables.

One considers a data sample in which are merged several species of events. These species represent various signal components and background components which all together account for the data sample. The different terms of the log-Likelihood are:

N the total number of events in the data sample,

$\mathrm{N_s}$ the number of species of events populating the data sample,

$\mathrm{N_i}$ the number of events expected on the average for the i^{th} species,

$\mathrm{f}_i(y_e)$ the value of the PDFs of the discriminating variables y for the i^{th} species and for event e,

x the set of control variables which, by definition, do not appear in the expression of the Likelihood function L

The extended log-Likelihood reads:

$$\mathcal{L} = \sum_{e=1}^{N} \ln \Big\{ \sum_{i=1}^{\mathrm{N_s}} N_i \mathrm{f}_i(y_e) \Big\} - \sum_{i=1}^{\mathrm{N_s}} N_i \ . \tag{1}$$

From this expression, after maximization of L with respect to the N_i parameters, a weight can be computed for every event and each species, in order to obtain later the true distribution $\mathbf{M}_i(x)$ of variable x If n is one of the $\mathrm{N_s}$ species present in the data sample, the weight for this species is defined by:

$$\boxed{{}_s\mathcal{P}_\mathrm{n}(y_e) = \frac{\sum_{j=1}^{\mathrm{N_s}} \mathbf{V}_{\mathrm{n}j} \mathrm{f}_j(y_e)}{\sum_{k=1}^{\mathrm{N_s}} N_k \mathrm{f}_k(y_e)}} \ , \tag{2}$$

where $\mathbf{V}_{\mathrm{n}j}$ is the covariance matrix resulting from the Likelihood maximization. This matrix can be used directly from the fit, but this is numerically less accurate than the direct computation:

$$\mathbf{V}^{-1}_{\mathrm{n}j} \ = \ \frac{\partial^2(-\mathcal{L})}{\partial N_\mathrm{n} \partial N_j} \ = \ \sum_{e=1}^{N} \frac{\mathrm{f}_\mathrm{n}(y_e)\mathrm{f}_j(y_e)}{(\sum_{k=1}^{\mathrm{N_s}} N_k \mathrm{f}_k(y_e))^2} \ . \tag{3}$$

The distribution of the control variable x obtained by histogramming the weighted events reproduces, on average, the true distribution $\mathbf{M}_\mathrm{n}(x)$.

The class TSPlot allows to reconstruct the true distribution $\mathbf{M}_\mathrm{n}(x)$ of a control variable x for each of the $\mathrm{N_s}$ species from the sole knowledge of the PDFs of the discriminating variables $\mathrm{f}_i(y)$ The plots obtained thanks to the TSPlot class are called ${}_s\mathcal{P}lots$

8.3.2 Some properties and checks

Beside reproducing the true distribution, ${}_s\mathcal{P}lots$ bear remarkable properties:

Each x distribution is properly normalized:

$$\sum_{e=1}^{N} {}_s\mathcal{P}_\mathrm{n}(y_e) \ = \ N_\mathrm{n} \ . \tag{4}$$

For any event:

$$\sum_{l=1}^{\mathrm{N_s}} {}_s\mathcal{P}_l(y_e) \ = \ 1 \ . \tag{5}$$

That is to say that, summing up the $\mathrm{N_s}$ ${}_s\mathcal{P}lots$ one recovers the data sample distribution in x and summing up the number of events entering in a ${}_s\mathcal{P}lot$ for a given species, one recovers the yield of the species, as provided by the fit.

$$\sigma[N_{\rm n}\ {}_s\tilde{\rm M}_{\rm n}(x)\delta x] \ = \ \sqrt{\sum_{e\subset\delta x} ({}_s\mathcal{P}_{\rm n})^2}\ . \tag{6}$$

reproduces the statistical uncertainty on the yield $N_{\rm n}$, as provided by the fit: $\sigma[N_{\rm n}] \equiv \sqrt{\mathbf{V}_{\rm nn}}$ Because of that and since the determination of the yields is optimal when obtained using a Likelihood fit, one can conclude that the ${}_s\mathcal{P}lot$ technique is itself an optimal method to reconstruct distributions of control variables.

- A maximum Likelihood fit is performed to obtain the yields N_i of the various species. The fit relies on discriminating variables y uncorrelated with a control variable x the later is therefore totally absent from the fit.
- The weights ${}_s\mathcal{P}$ are calculated using Eq. **??** where the covariance matrix is taken from Minuit.
- Histograms of x are filled by weighting the events with ${}_s\mathcal{P}$
- Error bars per bin are given by Eq. **??**.

The ${}_s\mathcal{P}lots$ reproduce the true distributions of the species in the control variable x within the above defined statistical uncertainties.

8.4 Bernstein Correction

BernsteinCorrection is a utility in RooStats to augment a nominal PDF with a polynomial correction term. This is useful for incorporating systematic variations to the nominal PDF. The Bernstein basis polynomails are particularly appropriate because they are positive definite.

This tool was inspired by the work of Glen Cowan together with Stephan Horner, Sascha Caron, Eilam Gross, and others. The initial implementation is independent work. The major step forward in the approach was to provide a well defined algorithm that specifies the order of polynomial to be included in the correction. This is an emperical algorithm, so in addition to the nominal model it needs either a real data set or a simulated one. In the early work, the nominal model was taken to be a histogram from Monte Carlo simulations, but in this implementation it is generalized to an arbitrary PDF (which includes a RooHistPdf). The algorithm basically consists of a hypothesis test of an nth-order correction (null) against a n+1-th order correction (alternate). The quantity q = -2 log LR is used to determine whether the n+1-th order correction is a major improvement to the n-th order correction. The distribution of q is expected to be roughly χ^2 with one degree of freedom if the n-th order correction is a good model for the data. Thus, one only moves to the n+1-th order correction of q is relatively large. The chance that one moves from the n-th to the n+1-th order correction when the n-th order correction (eg. a type 1 error) is sufficient is given by the Prob($\chi^2_1 >$ threshold). The constructor of this class allows you to directly set this tolerance (in terms of probability that the n+1-th term is added unnecessarily).

9 Tutorials

Several new tutorials were added for RooStats. They are located in your `$ROOTSYS` area under `tutorials/roostats`. They can be run from the command line by typing

```
root $ROOTSYS/tutorials/roostats/<tutorial>
```

(note you can also add a + to the end of the command to have the macro compiled).

For your convenience, you can browse them online:
RooFit Tutorials: http://root.cern.ch/root/html/tutorials/roofit/
RooStats Tutorials: http://root.cern.ch/root/html/tutorials/roostats/

rs301_splot.C Demonstrates use of RooStats sPlot implementation

rs401c_FeldmanCousins.C Demonstrates use of FeldmanCousins interval calculator with a Poisson problem, reproduces resulst from table IV and V of the original paper Phys.Rev.D57:3873-3889,1998.

rs401d_FeldmanCousins.C Demonstrates use of FeldmanCousins interval calculator with the neutrino oscillation toy example described in the original paper Phys.Rev.D57:3873-3889,1998. Reproduces figure 12.

rs_bernsteinCorrection.C Demonstrates use of BernsteinCorrection class, which corrects a nominal PDF with a polynomial to agree with observed or simulated data.