Extended RooFit tutorial

M. J. Zurowski

March 2025

In my experience, the best way to learn RooFit is to decide on a use case, then beg borrow and steal from various RooFit tutorials or ROOT forum posts. This document presents a suggested 'analysis' that gives you a workflow to practice various aspects of RooFit.

Before you start, some useful ROOT links:

- ROOT manual
- RooFit and RooStats introductions
- RooFit and RooStats tutorials - note RooFit are available in both C macros and pyroot while RooStats is largely C macros.

1 Defining and combining PDFs

Construct two PDF objects - an exponential (we can pretend this is the background) and a gaussian (we can pretend this is signal) functions of our observable energy:

$$\begin{aligned} \mathrm{PDF}_{\mathrm{bkg}}(E) &= \exp\left(-E/E_0\right), \\ \mathrm{PDF}_{\mathrm{sig}}(E) &= \exp\left(-\frac{(E-\mu)^2}{2\sigma_E^2}\right). \end{aligned} \tag{1}$$

Here your observable parameter is E, and your nuisance parameters are E_0, μ, and σ_E. Pick initial values for these so that you have a non-trivial overlap between the two functions.
Once these are defined, construct your combined PDF:

$$\mathrm{PDF}_{\mathrm{sig+bkg}}(E) = n_{\mathrm{bkg}}\mathrm{PDF}_{\mathrm{bkg}}(E) + n_{\mathrm{sig}}\mathrm{PDF}_{\mathrm{sig}}(E). \tag{2}$$

Here you will have introduced two additional parameters: n_{bkg} and n_{sig}. If we think about this in a context of a DM search, one can imagine μ as the DM mass we care about and σ_E as the resolution of the detector. If we want to construct a limit curve, for each μ, we are putting some limit on the value of n_{sig} we can see with some statistical significance, and so n_{sig} is our parameter of interest, and n_{bkg} is another nuisance parameter.

Useful tutorials:

- RooFit 101: basic gaussian construction
- RooFit 103: interpreting custom functions
- RooFit 105: 'binding' functions to a PDF
- RooFit 201: creating composite PDFs (i.e., combining multiple PDF objects)

2 Generating data and basic fits

Now that you have a PDF you should choose some 'true' set of values, generate data given these parameters, then perform fits to $\mathrm{PDF}_{\mathrm{sig+bkg}}(E)$ (with everything floating). Not that to do this thoroughly, you should generate data the appropriate number of data points from the signal and background PDFs separately, rather than generating from the combined PDF.
Try this with a few different 'true' values for various parameters. When you're looking at the fit results, don't just look at the direct output from RooFit. Also assess the pulls for your parameters:

$$\mathrm{Pull} = \frac{|x_{\mathrm{true}} - x_{\mathrm{fit}}|}{\Delta x_{\mathrm{fit}}} \tag{3}$$

where Δx_{fit} is the error in the fit. Ideally, this should be less than 1 for a 'good' fit. You should also produce a plot that shows your data and fit model in the same canvas. Include a pull plot as well. Note that there are lots of different fitting methods in RooFit that you can try out.

Useful tutorials:

- RooFit 101: generating data from a PDF, then fitting to it and plotting
- RooFit 106: decorating plots
- RooFit 107: more plotting styles
- RooFit 109: getting χ^2, residuals, and pulls from a fit and plotting them
- RooFit 601: more detailed minimiser fits rather than just calling `fitTo`

3 Bias/pull test

A bias test is used to make sure that your fitting procedure is robust, and that 'on average' you are correctly reconstructing the truth values used to generate the data. There are a few steps you'll need to code up:

1. Set 'true' values for your model. These should stay constant throughout the test.
2. Generate data from your signal and background PDFs. Rather than generating $n_{\mathrm{sig,true}}$ and $n_{\mathrm{bkg,true}}$ events, you should randomly sample the number of events you generate from a Poisson distribution centred on the true values.
3. Perform the fit you wrote for the last section. Compute the pull for all of your parameters.
4. Save the pull values. You can of course save these however you like (e.g., adding to a list in python) but in the spirit of learning ROOT you may want to consider saving them to TH1 objects, or even a TTree with a branch corresponding to each parameter.
5. Repeat steps 2-4 a number of times (say - 1000), and plot the resulting pull distributions.

If your fit method is unbiased, the pull distributions should all be gaussians centred on zero with a standard deviation of 1. If this isn't what you see, it suggests there is something going wrong with your PDF construction, data generation, or fitting procedure. One hint I'll give is that you should make sure that when you repeat the generation of the data, you reset all your parameters to the

true values rather than inheriting the newly found fit values and generating data with those.

Useful tutorials:

- RooFit 801: demonstrates the use of RooMCStudy that performs these cycles of generation and fitting without you needing to write the steps yourself. I've actually never used this myself, but it looks very useful and ultimately potentially better than coding them in yourself. I would suggest you go through the process of explicitly writing the steps yourself at least once to make sure you understand what is going on, and can make sure you input the correct information into a RooMCStudy.

4 Sensitivity calculation

As with fit tests, there are a number of different implementations in RooFit and RooStats you can use. In general, you are always calculating the same thing. If we are constructing a 90% upper limit for a parameter (the one you will do this for is n_{sig}), what you are really saying is for the data set you are looking at, there is a 90% probability that it was generated from a true value of n_{sig} that is below your limit value. The exact way you should interpret this is outside of the scope of this tutorial (see some statistics textbooks if you're really interested) and likely (hah) depends on whether you are more inclined to frequentist vs Bayesian thinking. In short, what you are calculating is the upper edge of a confidence interval:

1. Set 'true' values for your model. Again, these should stay constant throughout the test.
2. Generate data from your signal and background PDFs.
3. Calculate your confidence interval upper limit. This is your upper limit for this data set.

Sensitivity calculations don't actually care about whether you are generating the data from a signal, background, or combined PDF - all it is telling you is a probabilistic statement about a parameter, given this data. How you interpret this statement will depend on what you are feeding into it.

4.1 Exclusion projections

This is probably the way in which you are used to thinking about sensitivity: given some background model, what is the smallest DM cross section we can constrain for a given DM mass (or, for this extended example, what is the smallest n_{sig} we can constrain for each value of μ). If we are making projections (because we don't yet have real data) we also want to recognise/capture statistical uncertainty in our fit. The general steps are:

1. Set 'true' values for your model (again, these true values stay constant during generation).
2. Generate data from your *background* PDF. As for the pull test, the number of events should be randomly generated from a Poisson distribution centred on $n_{\mathrm{bkg,true}}$.
3. Calculate your confidence interval upper limit for your *signal + background* PDF. Save it (as you did for the pull test).
4. Repeat steps 2 and 3 (again, something on the order of 1000 times).
5. Calculate the average UL value from these tests, as well as the standard deviation of the distribution. This gives you your UL and uncertainty in $n_{\mathrm{sig,UL}}$ for this value of μ.

6. If you want to extend this example to create something like the $m_\chi - \sigma_\chi$ limit plots, repeat steps 1-5 for different true values of μ.

Note that before you move onto step 6 here, you may want to perform a coverage test (see the next section) to make sure your limit setting procedure is robust before investing a lot of computational time.

Useful tutorials:

- RooStats IntervalExamples: this shows off four different ways you can construct confidence intervals. However, it also uses RooWorkspace and ModelConfig objects, which you have not yet been introduced to. However, if you look at the documentation of whichever method you want to implement, it will explain how to use it with a RooDataSet and RooPDF object instead.

5 Coverage test

Similar to a bias test, a coverage test is a diagnostic to check that your upper limit calculation is doing what you think it is. I.e., will a 90% upper limit indeed be larger than the true value used to generate the data 90% of the time? The basic steps are:

1. Set 'true' values for your signal and background models (yet again, these true values stay constant during generation).
2. Generate data from your signal and background PDFs, exactly the same as you do for a bias test.
3. Calculate your confidence interval upper limit for your *signal + background* PDF. Save it (I sometimes find it easier to save the ratio of the UL to the true value for plotting purposes).
4. Repeat steps 2 and 3 (again, something on the order of 1000 times).
5. Plot the resulting UL distribution, and calculate how many entries are larger than your true value.

If your 90% UL is working, you should find that indeed, 90% of your pseudo-experiments have correctly chosen an UL that is larger than the true value you set in step 1. If that's not the case, something is going wrong with your limit setting. A common mistake occurs when setting the desired confidence level for the test. Depending on which method you have chosen to implement, the confidence level will refer to a two sided interval. The upper limit of a 90% confidence *interval* would actually correspond to a 95% exclusion limit because you exclude all values inside the interval (90% of values) plus the 5% that fall below this interval. To get a 90% UL you actually want to find the 80% confidence interval (if you assume it's a symmetric distribution).

6 Constraining nuisance parameters

Often you will have constraints or uncertainties associated with a nuisance parameter (or even a parameter of interest). Mathematically, these can be implemented by including a gaussian term in your PDF/likelihood function. This will 'increase' your likelihood for values of these parameters

when they are close to the prior/within the uncertainty, and 'reduce' your likelihood as it floats away from these. If you have a prior on parameter x given by $x_0 \pm \sigma_x$, your likelihood becomes:

$$\mathrm{PDF}_{\mathrm{full}}(E) = \mathrm{PDF}_{\mathrm{sig+bkg}}(E) \times \exp\left(-\frac{(x-x_0)^2}{2\sigma_x^2}\right). \tag{4}$$

Choose some constaint on one (or both) of your nuisance parameters. Adjust your PDF appropriately, and see how your fit results and sensitivity change.

Useful tutorials:

- RooProdPdf reference documents explains how to multiply conditional functions.

7 RooWorkspace

RooWorkspace is a convenient way to deal with complicated models/objects that have a large number of parameters and functions associated with them. In some ways you can think of it as a method to package them all together and label it with a single reference parameter, which makes it possible to write more general functions for statistical tests that take a RooWorkspace as an argument, then unpack all the necessary parameters for the test at hand. You may have noticed that this is the principle used for most of the RooStats tutorials as well as the toy BGM analysis.
Construct a RooWorkspace in which to store the PDFs you have constructed here, then adjust your fit, bias test, sensitivity, and coverage test code to be functions that take this RooWorkspace in as an argument.

Useful tutorials:

- RooFit 502: writing RooWorkspace objects
- RooFit 503: reading RooWorkspace objects

8 Useful statistics references

If you're interested in reading more about statistical tests, I suggest:

- Asymptotic formulae for likelihood-based tests of new physics
- PDG statistics review
- Statistics for Nuclear and Particle Physics
- Statistical Data Analysis
- Systematic uncertainties and profiling
- Uncertainties in RooFit