Data Analysis and Fitting with ROOT

LIP Internship Program

Guilherme Soares
guilherme.m.s.soares@tecnico.ulisboa.pt

Laboratory of Instrumentation and Experimental Particle Physics

14 July 2022

Index

Introduction

This tutorial is divided into 3 parts :

- Draw the dimuon mass spectrum of CMS (short)
- Use RooFit on the J/ψ peak and determine its mass
- Fit a different resonance (if we have time)

We will use **real CMS data**

We will follow a python **notebook template** in python.

The only thing you need is **internet** and a **Google colab** account.

If you prefer to do it in C++, everything can be found in **this link**.

Handling the CMS Data

Data stored in Zjet.root is in the form of **TLorentzVectors**.

4-vector for position (X, Y, Z, T) or energy (Px, Py, Pz, E) (no actual distinction for the methods)

Performs operations between 4-vectors automatically, and can give quantities of interest like mass easily (TLorentzVector : :M()).

It cannot be utilized as an automatic variable for plotting within ROOT (There is no TTree : :Draw(TLorentzVector) option, so we need to loop over all events and fill histograms manually.)

What are we seeing in the histogram ? Is there a way to get better insight into the spectrum ?

Handling the CMS Data

Lets try to branch out of the template!!!

Look at a few values for the branch "*event*". Why are the values like that?

Plot the mass spectrum of the dimuon events **in a better scale**. Compare it to the mass spectrum for the *muonp_4* and *muonn_4* branches.

If there is enough time :

Check the TLorentzVector class and make use of its functions to create a new TLorentzVector that is the sum of muonP and muonN. Plot this new vector's mass and compare it to the dimuon spectrum.

Fitting a histogram with RooFit - Principles

We will be utilizing **RooFit** to perform our fits. There are other ways to perform fits with ROOT, like using TH1's fitting functions or using TFormula and TGraphs, but we only have so much time.

The template shows a gaussian fit being applied to the J/ψ peak with an associated exponential for the expected background.

RooFit has a plethora of specific classes that transform usual ROOT objects into a format that RooFit can work with. The primary ones are :

- **RooDataHist** / RooDataSet : has the data
- **RooRealVar** : defines the variables
- RooFit **Functions** : functions to be used in the fit

Fitting a histogram with RooFit - Creating the Dataset

Step 1 : Creating the Histogram

Start by defining our TH1F object with the limits of the peak, and filling it with the relevant data from our TLorentzVectors.

Create the **RooDataHist** object through RooDataHist("name", "title", RooArgList, *TH1F). Here, RooArgList is a specific object that contains all the variables of our dataset. In our case we only want the **mass** of the particles.

Variables are defined through a RooRealVar object. We create the **"mass"** with RooRealVar("name", "title", minimum, maximum, "units").

Fitting a histogram with RooFit - Creating the Fitting Model

Step 2 : Creating the Fitting Functions

There are 3 primary items :

- Signal
- Background
- Amount of events that belong to each one (Normalization)

RooFit has a plethora of functions, such as RooExponential. However, we need to first define the variable of our data (mass that we already defined), but also define the free parameters of our functions as RooRealVar objects.

The exponential function only has one free parameter, since the normalizing factor is included in the amount of events : **lambda**, the exponential factor.

We define this through RooRealVar("name", "title", estimation, min, max)

As for the background function, we utilize RooExponential("name","title",real_var,parameter_var)

The signal gaussian function is done in a similar way, but we use RooGaussian and now we have 2 variables : **mean**, **sigma**.

Fitting a histogram with RooFit - Fitting the Data

Step 3 : Combining the Functions

Need to join our signal and background functions into a single model.

Additionally, we need to define the amount of events predicted. Hence the last 2 RooRealVar objects n_signal and n_back.

We are now ready to create our final model "model" = RooAddPdf("name", "title", RooArgList(functions), RooArgList(amount of events))

Last Step : Fitting Proper

We are now all set, and to fit the model to our data one simply uses

RooAddPdf : :fitTo(RooDataHist) $\rightarrow$ model.fitTo(dh)

The rest of the code is for plotting the different RooFit models, the RooDataHist and a nice legend to accompany it.

Fitting a histogram with RooFit - Do It Yourself

Now that you know how to perform a fit, **choose a different peak** and perform a **fit with a gaussian** function for the signal **and a polynomial** as the background.

What is the mass of the chosen peak ? And what about its statistical error ?

Now try to perform a different fit to the same peak. This time use **2 gaussians** as the function for the signal and keep the **polynomial** as the background.

Which of the 2 models describes the data the best ? **Physics** results always show a statistical and a systematic error. How can you **extrapolate the systematic error** from these fits ?

Unbinned Fits - Principles

As the name implies we do not bin data.

The advantage against the histogram fitting is that we **do not "compress" data** by binning. The downside is that we cannot perform a traditional fit on a histogram that has infinitely small bins.

We transform it into an optimization problem

We define a likelihood function $L(\vec{x}, \vec{\theta})$ that depends on the parameter space from our data points $\vec{x} = \{x_0, x_1, ..., x_n\}$ and also on the free parameters from our fitting models $\theta = \{\theta_0, \theta_1, ..., \theta_m\}$.

The likelihood function is usually written in the form of $L(\vec{x}, \vec{\theta}) = -log[P(\vec{x}, \vec{\theta})]$, and we then use iterative methods (Minuit2 in ROOT) to obtain the $\vec{\theta}$ that minimizes $\sum_{\text{dataset}} L(\vec{x}, \vec{\theta})$.

Unbinned Fits - Creating the Dataset

Performing an unbinned fit with RooFit is very similar to a binned one. However, now we utilize a **RooDataSet** object in order to store our data.

Initializing the RooDataSet object is slightly more complicated.

First we define an empty RooDataSet, that has a RooRealVar "mass", and then fill it one entry at a time.

Unlike for the TH1F objects, values outside of the specified range for the variable are counted as being on the limit, damaging the quality of the data for the fit. Given this we need to fill the RooDataSet we just what we want.

With the RooDataSet created, the rest of the process is the same, and we can easily copy the code from our previous block in order to perform the fit.

As you can see, plotting a RooDataSet yields a RooDataHist-like plot.

The End

Congrats. You are now able to discover the next fundamental particle and collect a Noble prize !