Conditional PDFs

- A conditional probability density function answers the question, “Given y, what is the probability density of x?”
  - y is the conditional observable.

- This is not a 2-dimensional probability density.
  - That would be, “What is the combined probability density of x and y?”

- In effect, the observable y gets demoted from the argument of a function to a parameter of it.
Example

\[ \text{ARGUS}(m_{ES}, m_{ES}^0) = m_{ES} \sqrt{1 - \left(\frac{m_{ES}}{m_{ES}^0}\right)^2} \cdot e^{-\xi \left[1 - \left(\frac{m_{ES}}{m_{ES}^0}\right)^2\right]} \]

- Let \( m_{ES} \) be defined in the range \([5.2, 5.3]\) GeV/c\(^2\) and let \( \xi = -20 \).
- Given an endpoint of 5.29 GeV/c\(^2\) what is the probability to find \( m_{ES} \) in the range \([5.275, 5.285]\) GeV/c\(^2\)?
  0.07573
- If the endpoint is 5.2981 GeV/c\(^2\), the probability becomes 0.08777.
A RooPlot of "mES"

- Endpoint 5.29
- Endpoint 5.2981

mES
Relation to 2D PDFs

- On the first slide I said that a conditional PDF $p(x|y)$ is not a 2D PDF.
  - We’ll denote the 2D PDF by $p(x,y)$.
- However, these things are related:
  $$p(x,y) = p(x|y) \cdot p(y)$$
- To get the full 2D PDF you need to know something about the distribution of $y$, whereas with the conditional version, you don’t care.
Effect on Normalization

- When you want to get something useful out of your PDF, it needs to be normalized.
- To normalize a regular 2D PDF $p(x,y)$ you divide by
  $$\iint p(x, y) dx dy$$
- However, since a conditional PDF $p(x|y)$ treats $y$ as a parameter, you only divide a conditional PDF by
  $$\int p(x, y) dx$$
Fitting

- Your likelihood function is the product of normalized probability densities of all events in your data set:

$$\mathcal{L} = \prod_i^N \frac{p(x_i, y_i)}{\iint p(x, y) dx dy} = \frac{1}{\iint p(x, y) dx dy} \prod_i^N p(x_i, y_i)$$

- If your PDF is just $p(x,y)$, then the constant denominator is irrelevant to maximizing the likelihood.

- However…
Fitting

- If you’re using $p(x|y)$, then each event is weighted by $\frac{1}{\int p(x, y_i)dx}$ and your likelihood function becomes

$$
\mathcal{L} = \prod_{i}^{N} \frac{p(x_i, y_i)}{\int p(x, y_i)dx}
$$
Fitting

- In RooFit, you can tell it to treat certain observables as conditional when you fit your PDF:

```cpp
RooRealVar x; RooRealVar y; RooDataSet ds;
RooSomePdf myPdf("myPdf", "myPdf", x, y);
myPdf.fitTo(ds,
    ConditionalObservables(RooArgSet(y)));
```
Plotting

- The different normalization required when using conditional observables also affects plotting.
- To project out an observable from a 2D PDF, you just integrate:

\[
p(x) = \frac{\int p(x, y) dy}{\iiint p(x, y) dx dy}
\]

- However, if \( y \) is conditional, then it’s a parameter – not something you integrate over.
Plotting

- Since we can’t integrate over $y$, we average.
- Instead of
  $$ p(x) = \frac{\int p(x, y)\,dy}{\iint p(x, y)\,dx\,dy} $$
- We take an average over the data to get:
  $$ p(x) = \frac{1}{N} \sum_{i=1}^{N} \frac{p(x, y_i)}{\int p(x, y_i)\,dx} $$
Plotting

- In RooFit, you can tell it to average over the data when plotting:
  ```cpp
  RooRealVar x; RooRealVar y; RooDataSet ds;
  RooSomePdf myPdf("myPdf", "myPdf", x, y);
  RooPlot* myPlot = x.frame();
  myPdf.plotOn(myPlot,
                ProjWData(RooArgSet(y), ds));
  ```

- Specifying the RooArgSet of conditional observables is crucial – otherwise you get an average over ALL of the observables in the dataset, which is probably not what you want.

- If this method takes too long to calculate, you can bin your conditional observable in a RooDataHist and pass that as the data set argument.
Toy MC

- Since a conditional PDF knows nothing about the distribution of its conditional observable, that information has to come from somewhere else.
  - Can’t just generate toy MC from your PDF.
- If you really don’t know anything about the observable, you can take it from your actual data set or the centrally-produced MC.
  - Can run into statistics issues if you want to do a lot of toys.
- Otherwise, you can make up a reasonable PDF for it, and generate a prototype dataset from that PDF.
To specify conditional observables and prototype data in RooFit, declare your RooMCStudy like this:

```cpp
RooAbsPdf theFullPdf;
RooArgSet allObservables;
RooDataSet* theYDistr = yPdf.Generate(nEvt);
RooMCStudy myStudy(theFullPdf, allObservables,
  ConditionalObservables(RooArgSet(y)),
  ProtoData(theYDistr));
```
Pitfalls

- If your conditional observable has a different distribution for different species (like signal and background), then your fit can be biased.
  - For a simple example, see arXiv:physics/0401045v1 [physics.data-an].
- Not really a problem with the $m_{ES}$ endpoint example, but conceivably a problem in other applications.
  - You might notice this in embedded toy studies.
  - It can be explicitly tested in pure toys by generating with different distributions of your conditional observable for signal and background, then fitting with the conditional PDFs.
Summary

- Some observables have to be treated on an event-by-event basis.
- This affects the normalization of your PDFs, and therefore impacts the way you fit, plot, and validate your fitter.
  - Watch out for conditional observables that look very different between signal and background events.
    - If they are so different, maybe you should use this as a discriminating observable in your fit!
- RooFit has built-in methods to properly deal with conditional observables, but you have to tell it how to handle them.