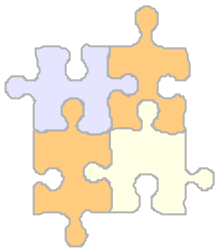


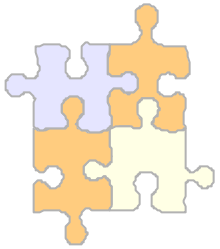
RooFit Tutorial – Fitting and Generating

Wouter Verkerke (UC Santa Barbara)
David Kirkby (UC Irvine)

Overview

- WARNING - This tutorial is incomplete
  - Relevant parts of the retired ‘advanced’ tutorial have been moved here.

Generating Toy MC events



Prototyping your ToyMC off real data

Using RooMCStudy to efficiently generate (and fit) many samples

Wouter Verkerke, UCSB

PDF generator interface has two invocation methods

- Regular
  - All observables generated by the PDF

```
genData = pdf.generate(<observables>,numEvent) ;
```

- With prototype data
  - Some observables modeled by a prototype dataset, other generated by PDF

```
RooDataSet protoData("D","D",<protoObservables>") ;
// fill protoDataSet ;
genData = pdf.generate(<genObservables>,protoData) ;
```

  - *Generator cycles through dataset*, loading proto-observables in PDF.
  - *PDF generates other observables* → *all correlations preserved*
  - By default #events is also taken from proto dataset.
    If overruled, generated dataset may not exactly reproduce proto data, unless #events = n * #proto-events

Generating ToyMC for sin2beta fit

- CP/mixing PDF doesn't model per-event errors & tagCats
  - Must use prototype data when generating
- Where to get the prototype data for per-event errors etc…?
  - Pre-generate proto data with separate PDF

Add desired #entries for each tagCat by hand

Generate per-event errors from separate PDF (same #evts)

Add per-event errors column to proto data

```
// Generate tagging categories
RooDataSet proto("pD","pD",tagCat) ;
tagCat = "Lep" ; for (i=0;i<400 ;i++) proto.add(tagCat) ;
tagCat = "Kao" ; for (i=0;i<1600;i++) proto.add(tagCat) ;
tagCat = "NT1" ; for (i=0;i<250 ;i++) proto.add(tagCat) ;
tagCat = "NT2" ; for (i=0;i<1500;i++) proto.add(tagCat) ;

// Generate a set of event errors
RooBifurGauss gerr("gerr","error distribution",dterr,
                   RooRealConstant::value(0.1),
                   RooRealConstant::value(0.3),
                   RooRealConstant::value(0.8)) ;
RooDataSet *errdata = gerr.generate(dterr,proto.numEntries());

// Add per-event error column to protoData
proto.merge(errdata) ;
```

Generating ToyMC for sin2beta fit

- Where to get the prototype data for per-event errors etc…?
  - ***Use actual distribution from data***

```
// Select per-event errors and tagCat from actual data
RooDataSet* protoData = cpData->reduce(RooArgSet(dterr,tagCat));
```

- Prototype data can be provided for *any* observable, *overriding the PDF*s intrinsic distribution
  - Example: force B^0/B^0-bar distribution of actual data

```
// Select per-event errors, tagCat and tag flavour from data
RooDataSet* protoData =
                cpData->reduce(RooArgSet(dterr,tagCat,tagFlav));

// Generate deltat from pdf with tag flavour from data
RooDataSet* genData = cpPdf->generate(deltat,protoData)
```

- Generate ToyMC dataset that mimic all properties of data
  - Use prototype data for *everything except* dt and mB
  - Tagging breakdown, #B0/B0bar, per-event errors, run numbers all taken from data

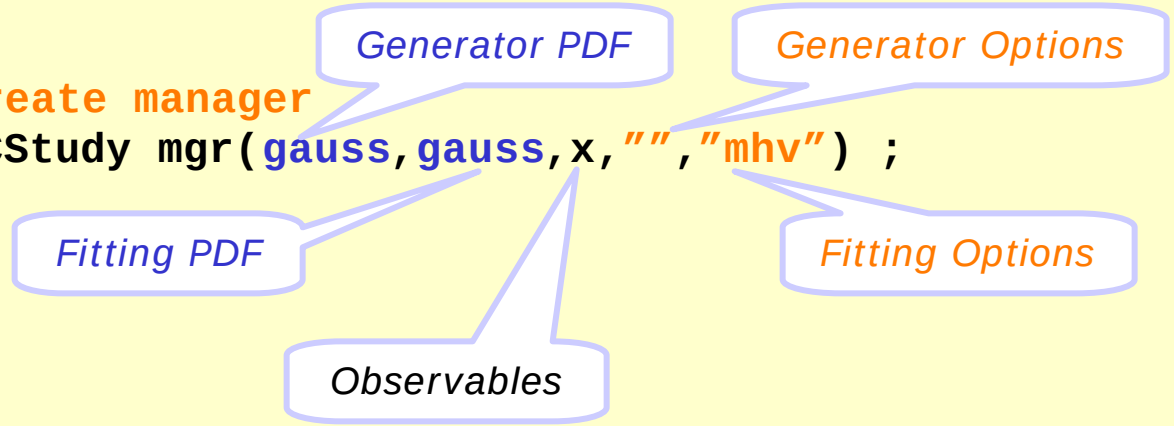
How to *efficiently* generate multiple sets of ToyMC?

- Use **RooMCStudy** class to manage generation and fitting
- Generating features
  - *Generator overhead only incurred once*
    ® Efficient for large number of small samples
  - Optional Poisson distribution for #events of generated experiments
  - Optional automatic creation of ASCII data files
- Fitting
  - Fit with generator PDF or different PDF
  - Fit *results* (floating parameters & NLL) automatically *collected in summary dataset*
- Plotting
  - Automated plotting for distribution of parameters, parameter errors, pulls and NLL

A **RooMCStudy** example

- Generating and fitting a simple PDF

```
// Setup PDF
RooRealVar x("x","x",-5,15) ;
RooRealVar mean("mean","mean of gaussian",-1) ;
RooRealVar sigma("sigma","width of gaussian",4) ;
RooGaussian gauss("gauss","gaussian PDF",x,mean,sigma) ;

// Create manager
RooMCStudy mgr(gauss,gauss,x,"","mhv") ;

// Generate and fit 1000 experiments of 100 events each
mgr.generateAndFit(1000,100) ;
RooMCStudy::run: Generating and fitting sample 999
RooMCStudy::run: Generating and fitting sample 998
RooMCStudy::run: Generating and fitting sample 997
…
```

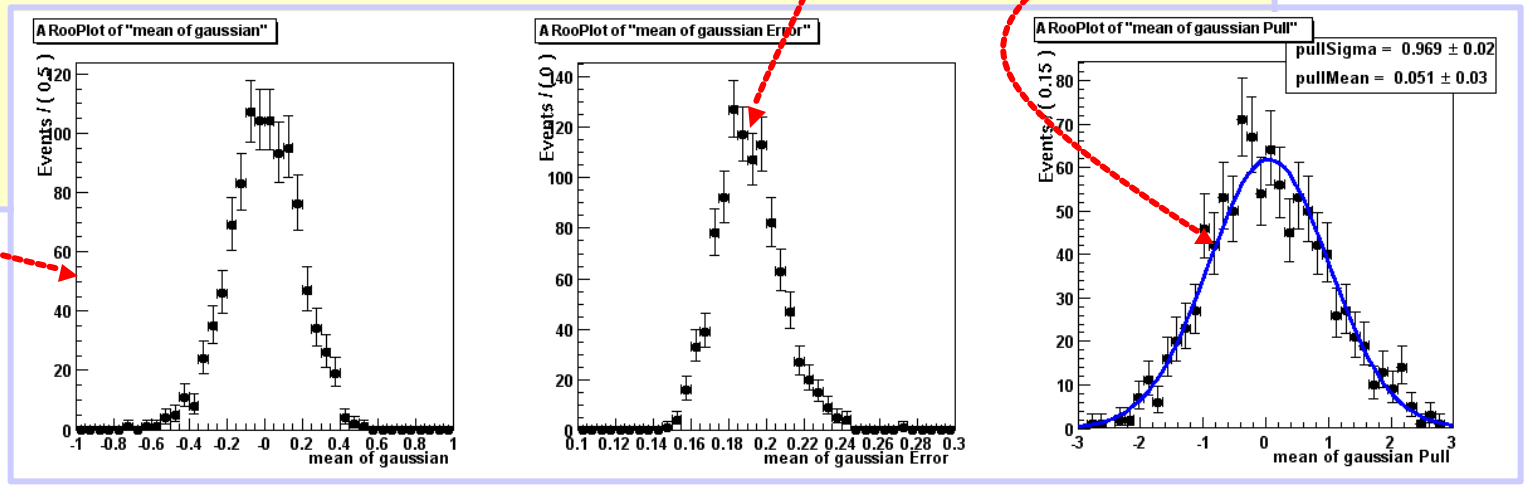


A **RooMCStudy** example

- Plot the distribution of the value, error and pull of *mean*

```
// Plot the distrution of the value
RooPlot* mframe = mean.frame(-2,0) ;
mgr.plotParamOn(mframe) ;
mframe->Draw() ;

// Plot the distrution of the error
RooPlot* meframe = mgr.plotError(mean,0.,0.1) ;
meframe->Draw() ;

// Plot the distrution of the pull
RooPlot* mpframe = mgr.plotPull(mean,-3,3,40,kTRUE) ;
mpframe->Draw() ;
```

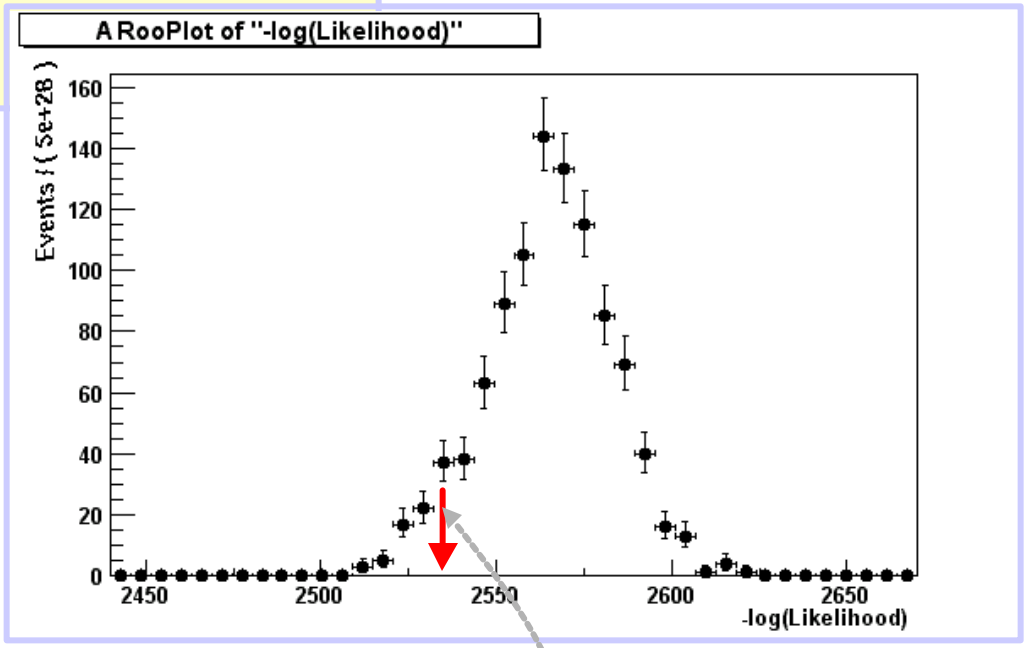
Add Gaussian fit



A **RooMCStudy** example

- Plot the distribution of –log(L)

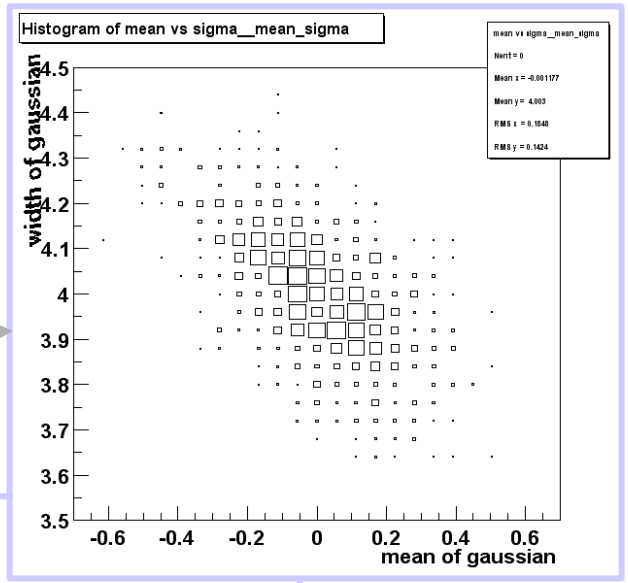
```
// Plot the distribution of the NLL
mgr.plotNLL(mframe) ;
mframe->Draw() ;
```



- All plots are regular **RooPlot**s and can be further decorated
  - E.g add arrow indicating value of fit to actual data

A **RooMCStudy** example

- For other uses, use summarized fit results in **RooDataSet** form



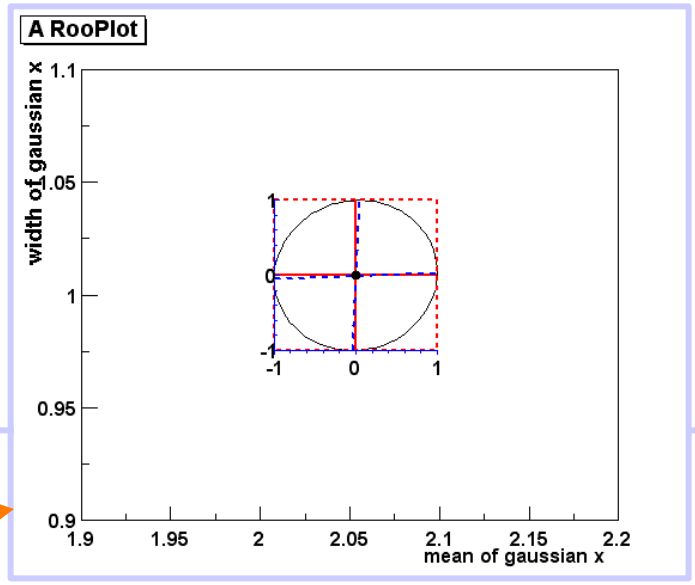

```
mgr.fitParDataSet().get(10)->Print("v") ;
RooArgSet:::
  1) RooRealVar::mean      :  0.14814 +/- 0.191 L(-10 - 10)
  2) RooRealVar::sigma     :  4.0619 +/- 0.143 L(0 - 20)
  3) RooRealVar::NLL       :  2585.1 C
  4) RooRealVar::meanerr   :  0.19064 C
  5) RooRealVar::meanpull  :  0.77704 C
  6) RooRealVar::sigmaerr  :  0.14338 C
  7) RooRealVar::sigmapull :  0.43199 C

TH2* h = mean.createHistogram("mean vs sigma",sigma) ;
mgr.fitParDataSet().fillHistogram(h,RooArgList(mean,sigma)) ;
h->Draw("BOX") ;
```

Pulls and errors have separate entries for easy access and plotting

A **RooMCStudy** example

- If the “r” fit option is supplied
  the **RooFitResult** output of each fit is be saved

```
mgr.fitResult(10)->Print(“v”) ;

  RooFitResult: minimized NLL value: 2585.13, estimated distance to minimum: 3.18389e-06

    Floating Parameter  InitialValue    FinalValue +/-  Error     GblCorr.
  --------------------  ------------  --------------------------  --------
                  mean    0.0000e+00    1.4814e-01 +/-  1.91e-01  0.597596  <none>
                 sigma    4.0000e+00    4.0619e+00 +/-  1.43e-01  0.597596  <none>

mgr.fitResult(10)->correlation("sigma")->Print("v") ;

RooArgList::C[sigma,*]: (Owning contents)
  1) RooRealVar::C[sigma,mean]  : -0.597596 C
  2) RooRealVar::C[sigma,sigma] :  1.0000 C


RooPlot* frame = new RooPlot(…)
mgr.fitResult(10)->plotOn(frame,meanx,
                          sigmax,"ME12VHB") ;
```



Using RooMCStudy for sin2b

- sin2b fit is relatively expensive (~30 minutes)
  - Don't want do 1000 consecutive fits on 1 processor…
  - *Use only generator part*
- Example: generating 1000 toys for goodness-of-fit

```
// Create manager
RooMCStudy mgr(cpmixPdf,cpmixPdf,RooArgSet(dt,mB),
               "","" ,protoData) ;

// Generate 1000 ToyMC sets in memory
mgr.generate(1000,0,kTRUE) ;

// Generate 1000 ToyMC sets in memory
mgr.generate(1000,0,kFALSE,"output/cpfit_%04d.dat") ;
```

Observables to generate with PDF

Prototype data (with all other observables

Keep generated datasets in memory

Warning: may be large!

Write out each dataset immediately to ASCII file. Do not retain in memory

General accept/reject generating caveats

- Monolithic multi-dimensional PDFs have *high startup overhead*
  - Need to find maximum function value empirically
  - *Large number* of samples needed
    - 1D – 1000, 2-D 100,000, 3-D 10,000,000
    - Initial samples will be reused for generation, but usually requests need <10M samples
  - Potential solution:
    If more efficient generation technique exists, *implement that method in PDF* and advertise it
    (see 'writing PDF yourself' section')

- *Pathological* distributions may be *sampled incorrectly*
  - *Narrow peak* may be missed by initial sampling (<1% of phase-space)
  - Otherwise generation is *very inefficient* in nearly all of phase space
  - Solution: generate your dataset in parts
    - Generate peak and non-peak areas separately, append afterwards