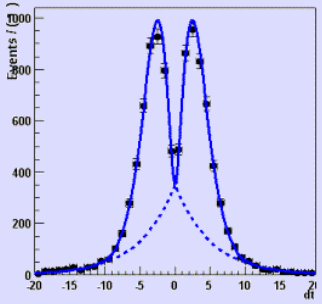
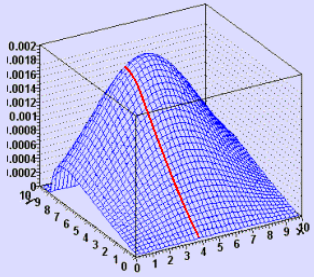
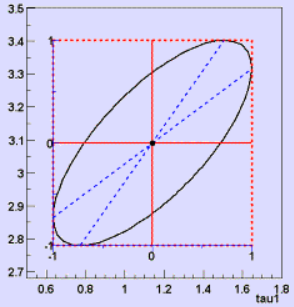






RooFit

A tool kit for data modeling in ROOT

Wouter Verkerke (NIKHEF)

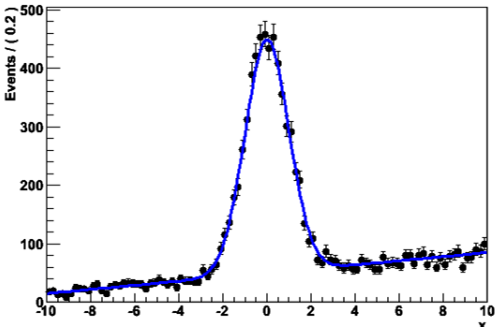
1. Introduction & overview
2. Template models (HistFactory)
3. The workspace
4. Performance tuning

What is Roofit?

- RooFit is a language to formulate models to describe your data
- Relates to end-game of (nearly) all HEP physics: statistical analysis.
- Original focus on complex model, new focus also on low-statistics problems

Simple model → Complex model

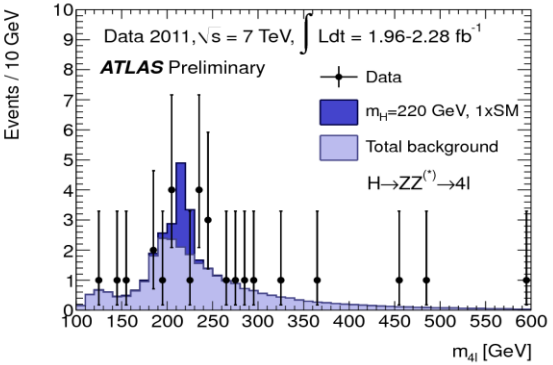
High statistics

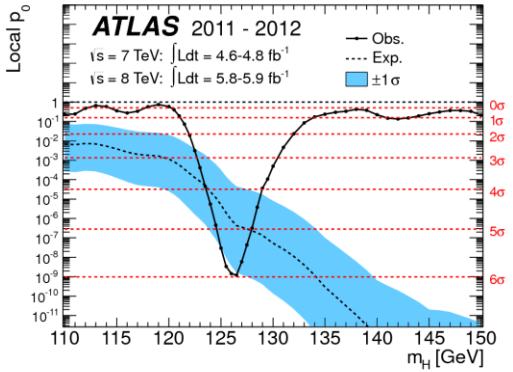


$$f_{sig} \times \text{SigSel}(m;\bar{p}_{sig}) \times \text{SigDecay}(t;\vec{q}_{sig},\sin(2\beta)) \otimes \text{SigResol}(t \mid dt;\vec{r}_{sig}) + (1-f_{sig}) \times \text{BkgSel}(m;\bar{p}_{bkg}) \times \text{BkgDecay}(t;\vec{q}_{bkg}) \otimes \text{BkgResol}(t \mid dt;\vec{r}_{bkg})$$

Measurement of CP violation at BaBar

Low statistics

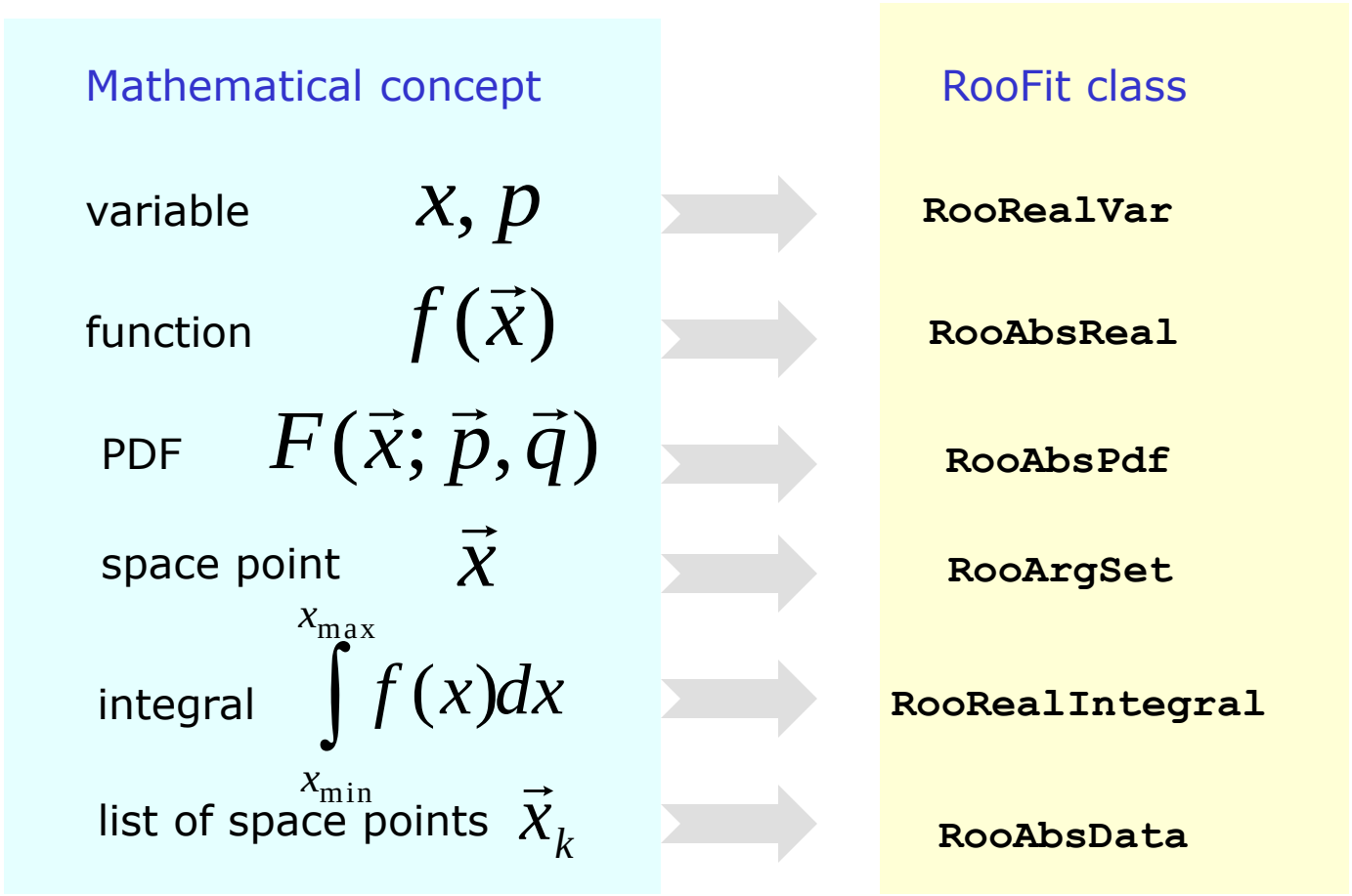




Discovery of Higgs boson at LHC

How does it work – code structure

- Key concept: represent individual elements of a mathematical model by separate C++ objects

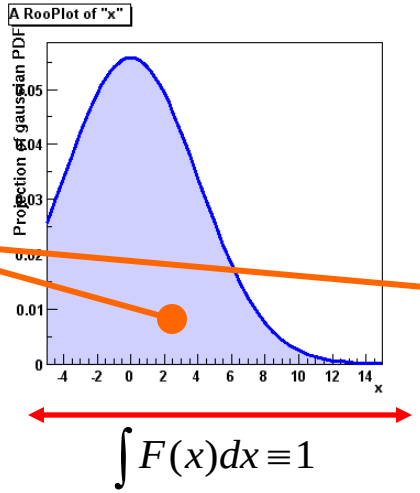


How does it work – mathematical model

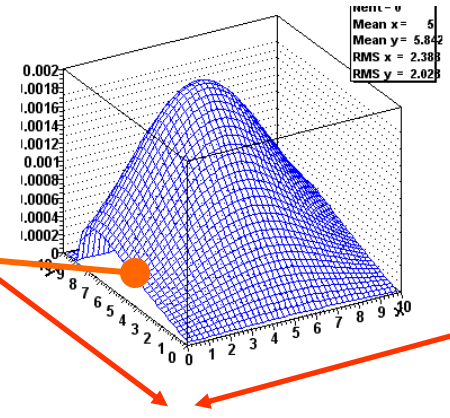
- Focus on specific models – **probability density functions**

- Defining feature:

$$\int f(\vec{x},\vec{p})\,d\vec{x} = 1, \qquad f(\vec{x},\vec{p}) \geq 0$$



$$\int F(x)dx \equiv 1$$

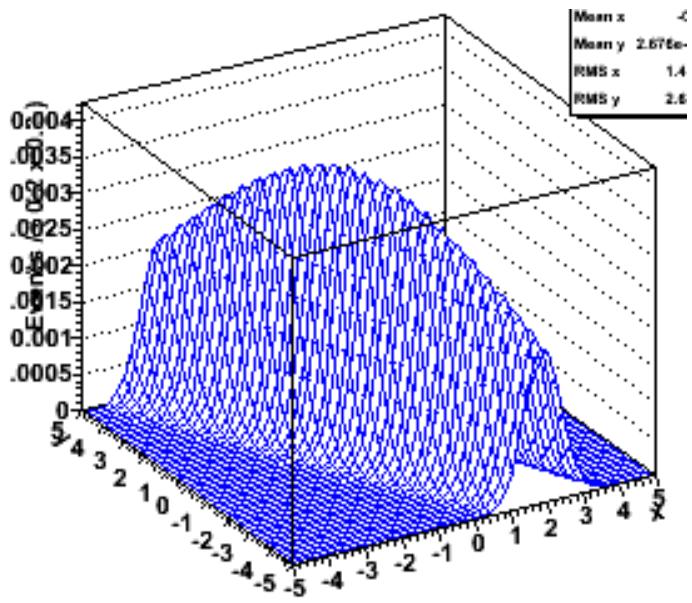


$$\int F(x,y)dxdy \equiv 1$$

- Normalization condition introduces extra complication in formulation of models, but has important advantages
  - Directly usable for formal statistical techniques (needed fot low stats)
  - Easier interpretation of model parameters
  - Easier construction of complex models
- RooFit provides built-in support for normalization, taking away down-side for users, leaving upside

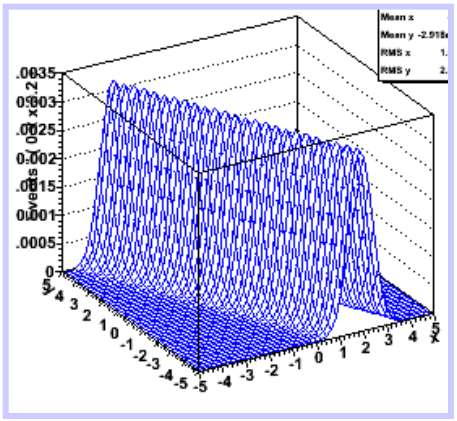
An example of model construction with pdfs

- Take following model f(x,y):
  what is the analytical form?

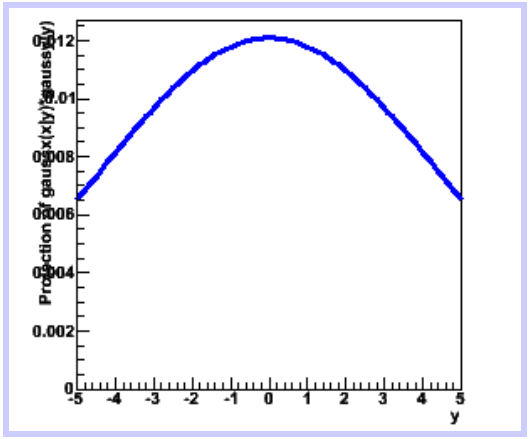


- Trivially constructed with (conditional) probability density functions!

Gauss f(**x**|a***y**+b,1)



Gauss g(**y,0,3**)

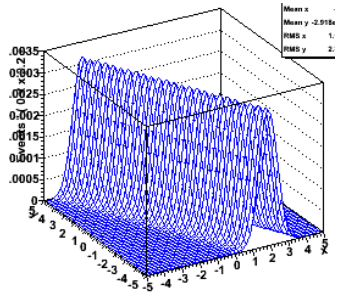


F(**x**,**y**) = f(**x**|**y**)*g(**y**)

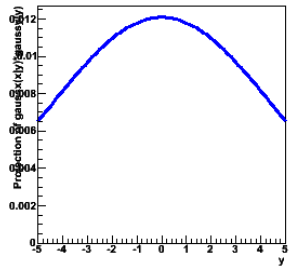
Coding a model in RooFit

- Construct each ingredient with a single line of code

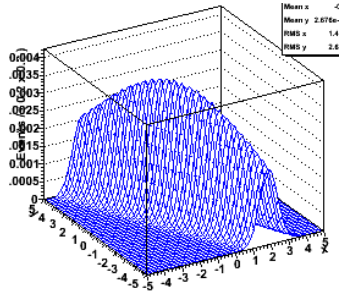
Gauss f(**x**,a*y+b,1)



Gauss g(**y,0,3**)



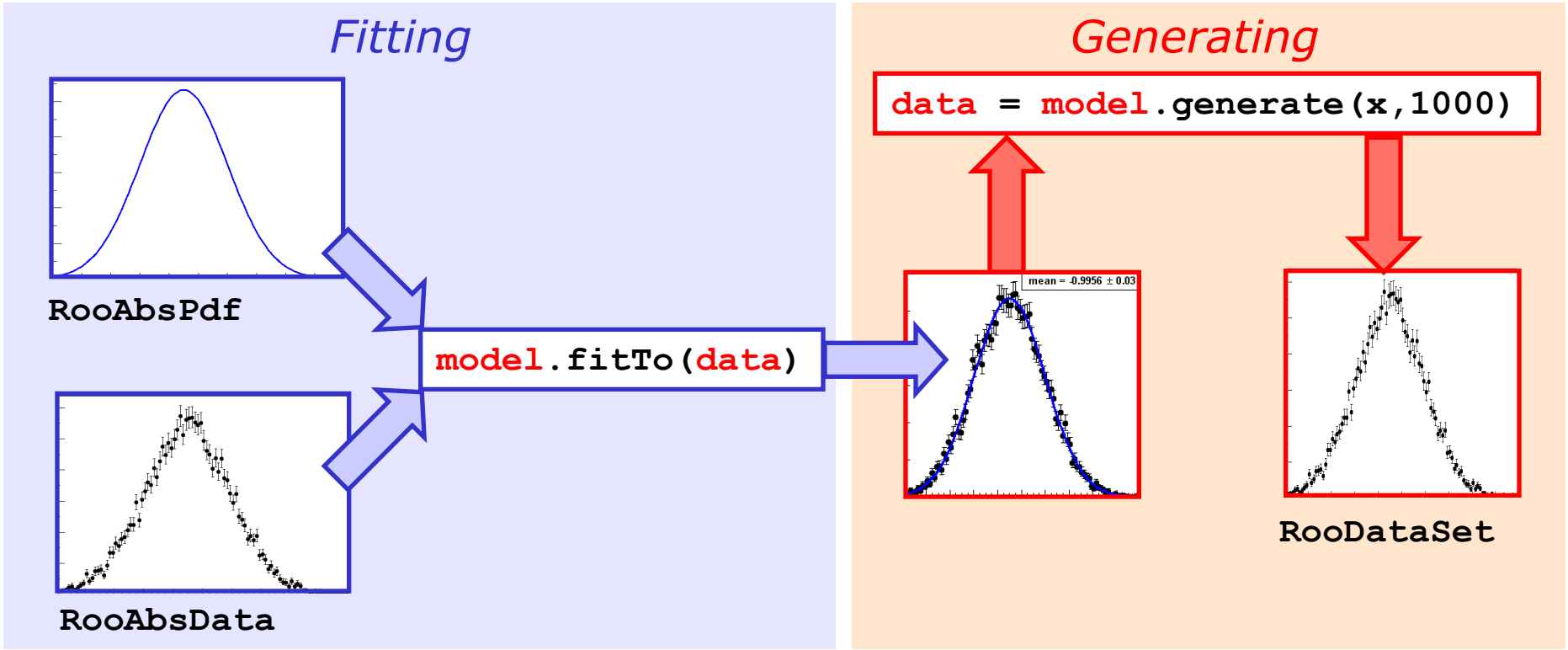
F(**x**,**y**) = f(**x**|**y**)*g(**y**)



```
RooRealVar x("x","x",-10,10) ;
RooRealVar y("y","y",-10,10) ;
RooRealVar a("a","a",0) ;
RooRealVar b("b","b",-1.5) ;

RooFormulaVar m("a*y+b",a,y,b) ;
RooGaussian f("f","f",x,m,C(1)) ;

RooGaussian g("g","g",y,C(0),C(3)) ;

RooProdPdf F("F","F",g,Conditional(f,y)) ;
```

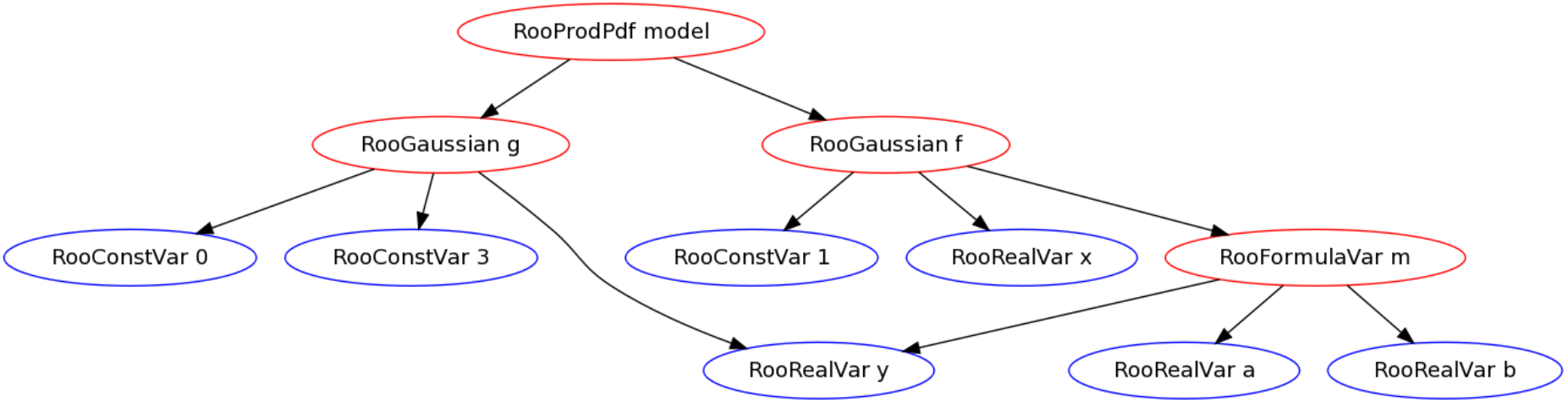
Using a model

- Can perform (unbinned) ML fit, generation of toy data from each model with one-line operation



The structure of a model visualized

- Graph of client-server connections between variables and functions



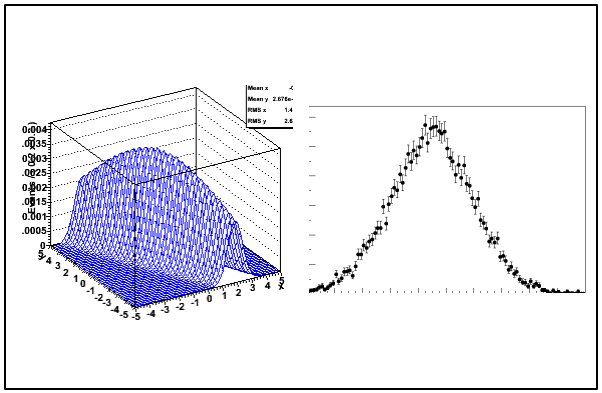

```
RooRealVar x("x","x",-10,10) ;
RooRealVar y("y","y",-10,10) ;
RooRealVar a("a","a",0) ;
RooRealVar b("b","b",-1.5) ;
RooFormulaVar m("a*y+b",a,y,b) ;
RooGaussian f("f","f",x,m,C(1)) ;
RooGaussian g("g","g",y,C(0),C(3)) ;
RooProdPdf model("model","model",g,Conditional(f,y)) ;
```

Graph made with F.graphVizTree("model.dot")

RooFit developments for LHC (Higgs analysis)

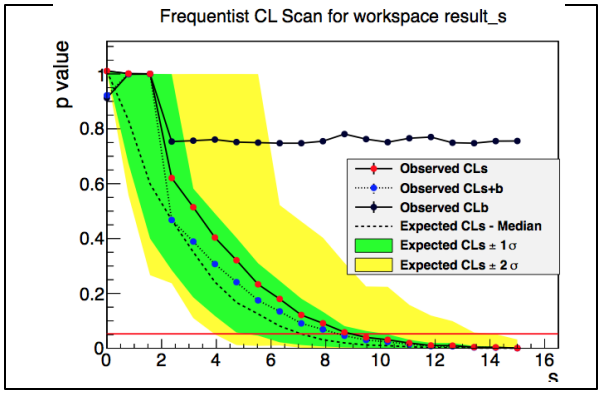
Class RooWorkspace

Simplify packaging and sharing of models



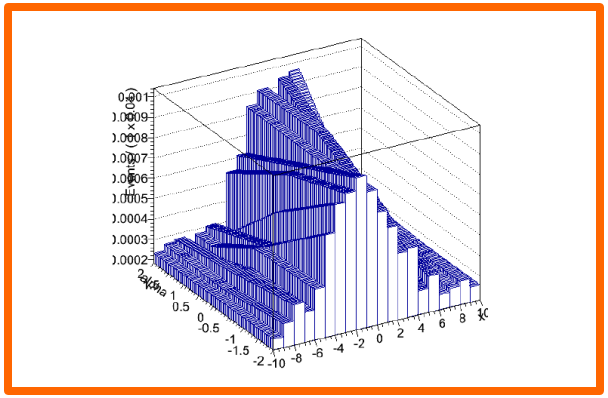
RooStats toolkit

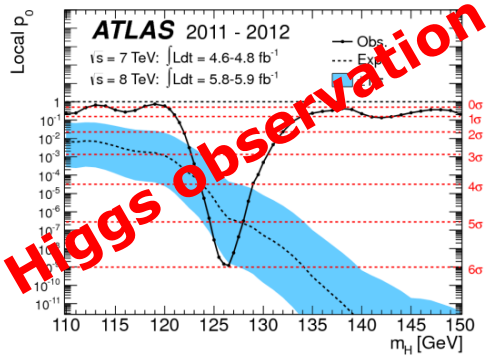
Statistical tests based on likelihoods from RooFit models



HistFactory package

Constructing models from Monte Carlo templates

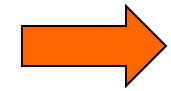




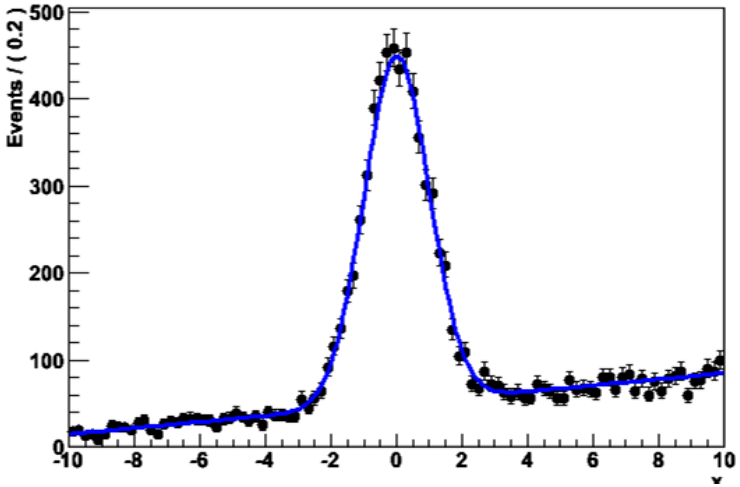
HistFactory – a new class of pdfs

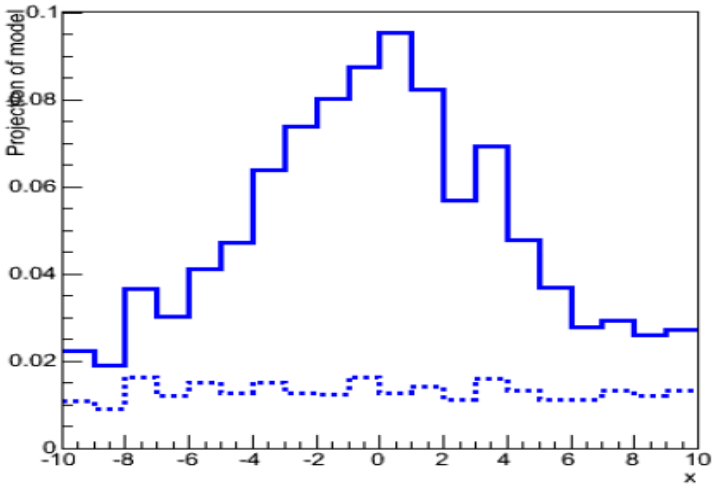
- Focus of RooFit traditionally on analytical models
  - Assumes you can formulate signal/background in an analytical form
  - Often possible in e+e- experiments,
    shapes for hadron colliders cumbersome → **rely on MC simulation**

Analytical form:
Gaussian+Polynomial



Template form:
Histogram (discrete)

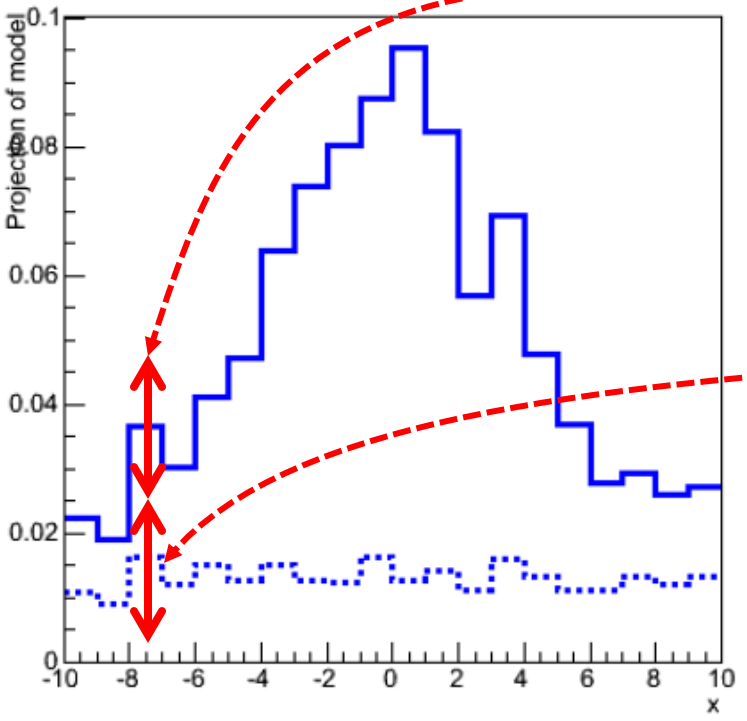




K. Cranmer, G. Lewis, L. Moneta, A. Shibata, and W. Verkerke, *HistFactory: A tool for creating statistical models for use with RooFit and RooStats*, CERN-OPEN-2012-016 (2012).
http://cdsweb.cern.ch/record/1456844.

HistFactory: modeling uncertainties on templates

- Histogram-shaped model not new (RooHistPdf), but key issue in physics analysis is in modeling uncertainties on this model (RooHistPdf has no degrees of freedom)
  - HistFactory: Model MC statistical uncertainties by allowing each histogram bin to float.
  - Then constrain rate in each bin with a Poisson distribution based on MC event count ("subsidiary measurements")



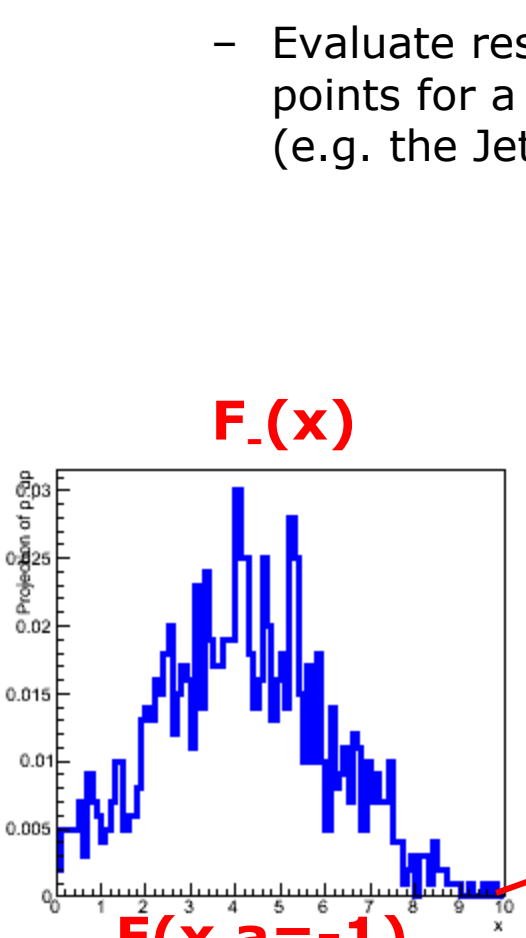
$$L(\tilde{y}_{0,s},\ldots,\tilde{y}_{n,s}) = \prod_{i=0}^{n} P_s(\tilde{y}_{i,s} \mid \theta_{i,s}) P_b(\tilde{y}_{i,b} \mid \theta_{i,b})$$

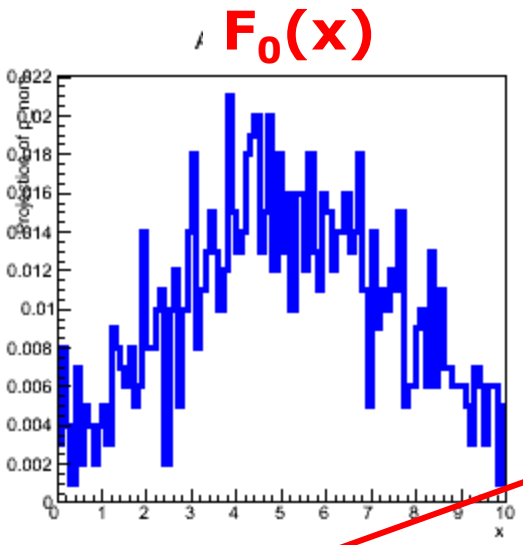
Signal simulation event count for bin i

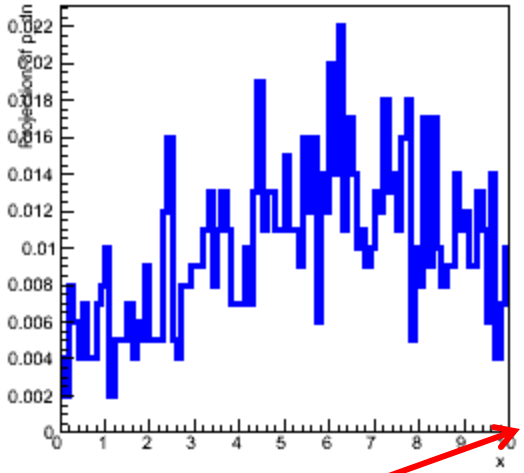
Background simulation event count for bin i

HistFactory: modeling uncertainties on templates

- MC simulation also have ***systematic* uncertainties**. Model these with 'template morphing approach' $F_+(x)$
  - Evaluate response of MC at 3 operating points for a systematic uncertainty (e.g. the Jet energy scale)



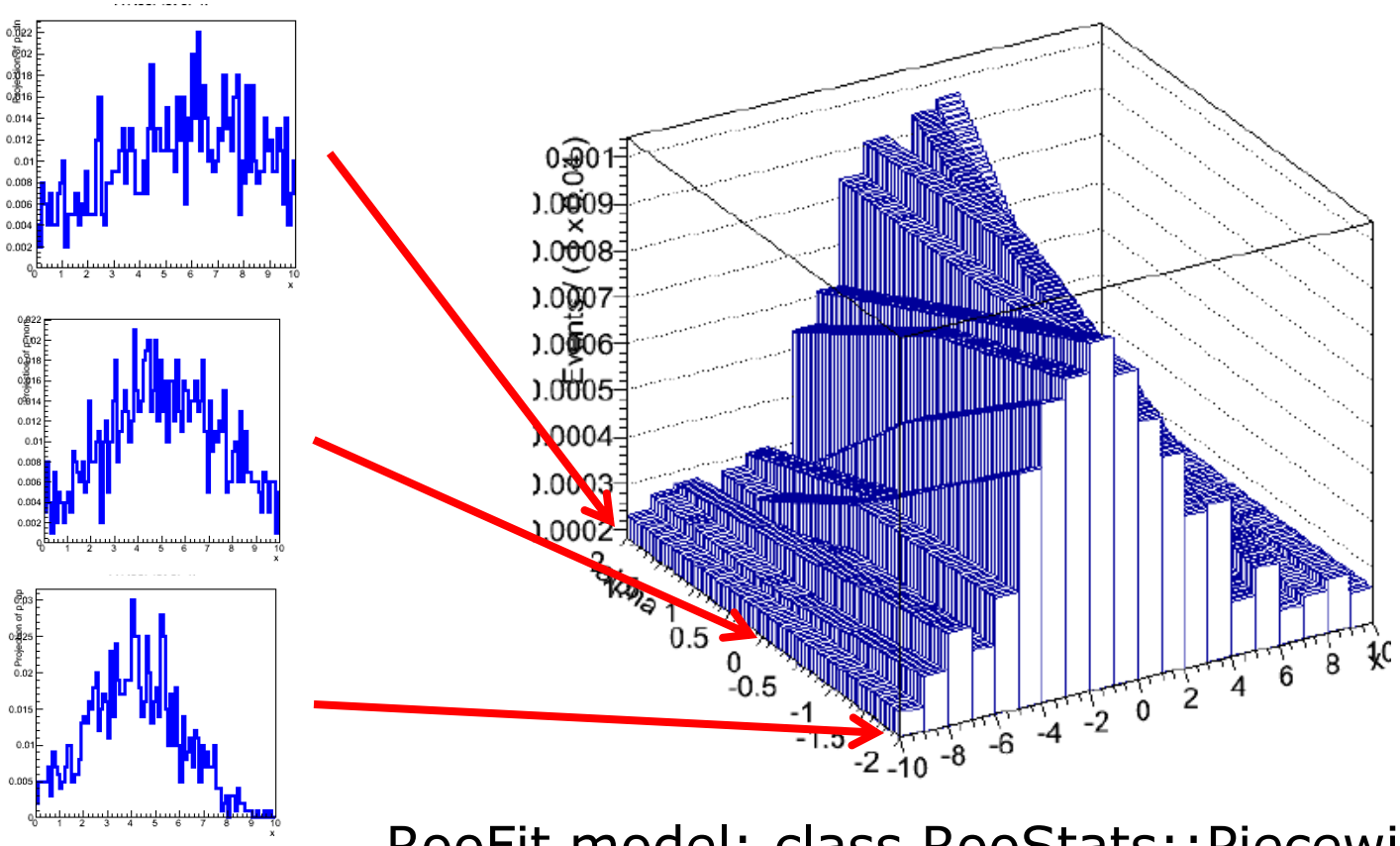




Solution: construct a model **F(x,a)** that **interpolates** (bin-by-bin) between the histograms introducing a newly introduced nuisance parameter

Simplest version: vertical interpolation

- Note template morphing is not a uniquely defined problem. Several practical solutions exist.
- The simplest (conceptually and practically) is vertical interpolation bin-by-bin



RooFit model: class RooStats::PiecewiseInterpolation

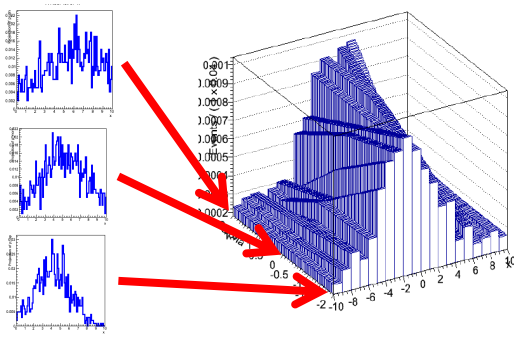
Describing MC simulation uncertainties – putting it together

- Example: MC template with **J**et **E**nergy **S**cale uncertainty

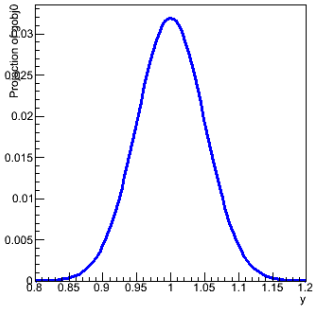
Measured JES

JES NP

$$L(\vec{x}, \tilde{\theta}_{JES} \mid s, \theta_{JES}) = Morph(\vec{x} \mid \theta_{JES}) \cdot Gauss(\tilde{\theta}_{JES} \mid \theta_{JES}, \sigma(\tilde{\theta}_{JES}))$$



'Effect of JES on observable distribution'



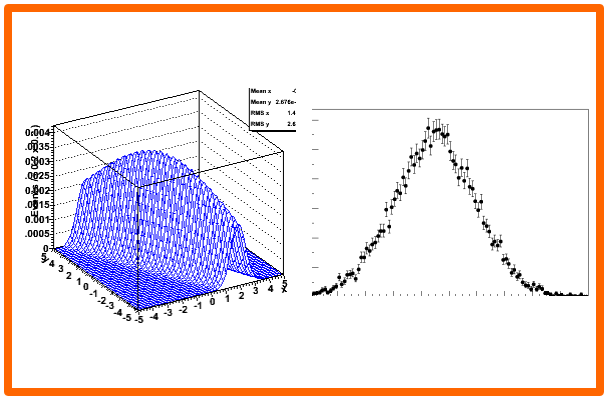
'Knowledge of JES uncertainty'

- HistFactory provides, in addition to PDF, a structured framework to construct such models in RooFit for an arbitrary number of signal,bkg components and systematic uncertainties

RooFit developments for LHC (Higgs analysis)

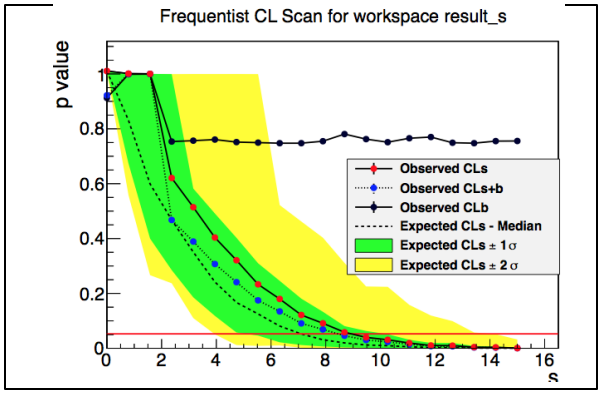
Class RooWorkspace

Simplify packaging and sharing of models



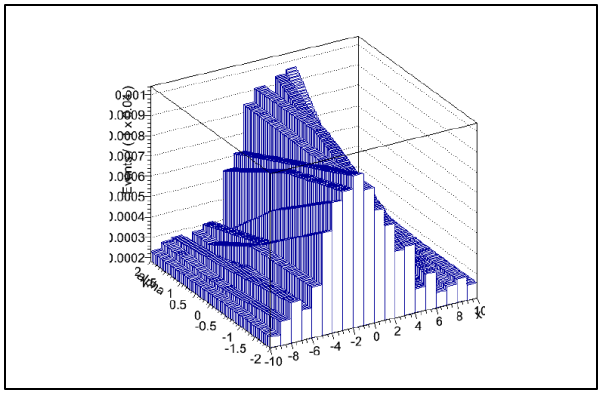
RooStats toolkit

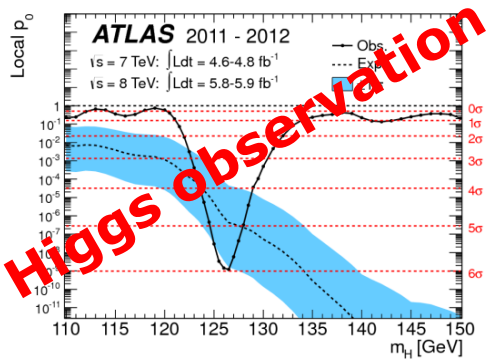
Statistical tests based on likelihoods from RooFit models



HistFactory package

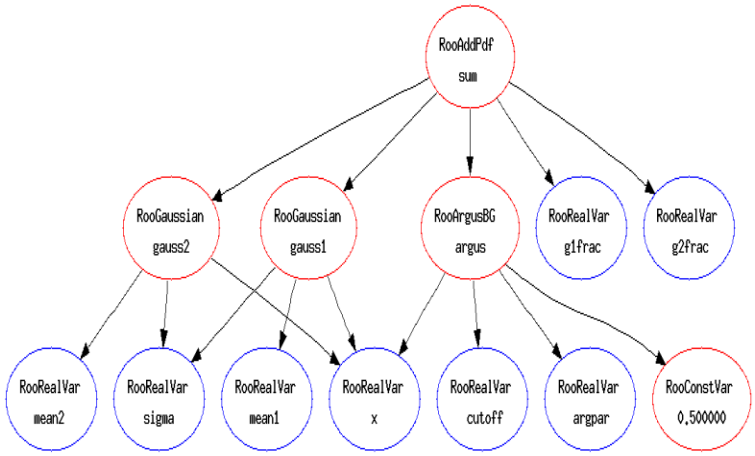
Constructing models from Monte Carlo templates



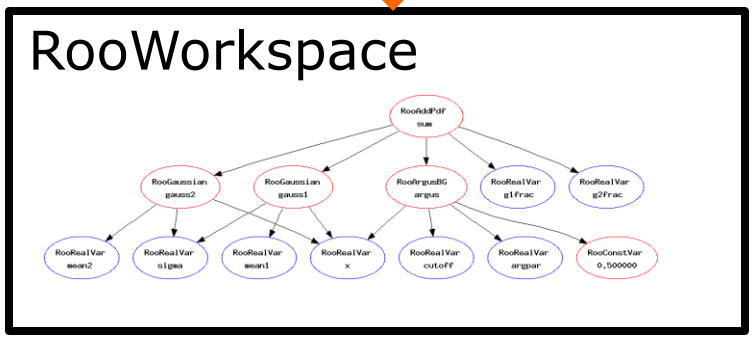


The workspace

- **The workspace concept has revolutionized the way people share and combine analysis**
  - You can give somebody an analytical likelihood of a (potentially very complex) physics analysis in a way to the easy-to-use, provides introspection, and is easy to modify.



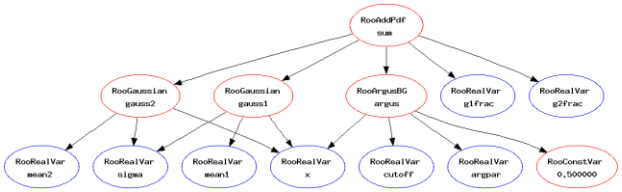

```
RooWorkspace w("w") ;
w.import(sum) ;
w.writeToFile("model.root") ;
```

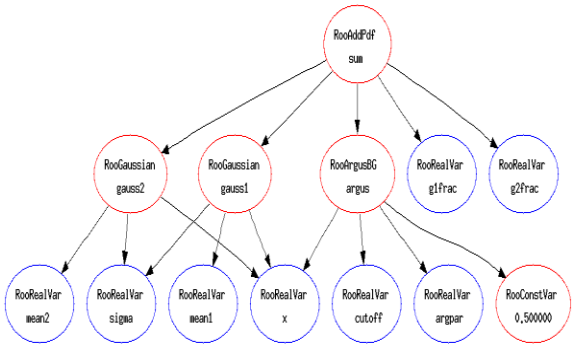


model.root

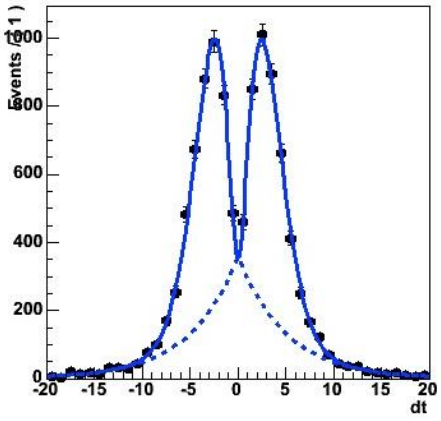
Using a workspace

RooWorkspace



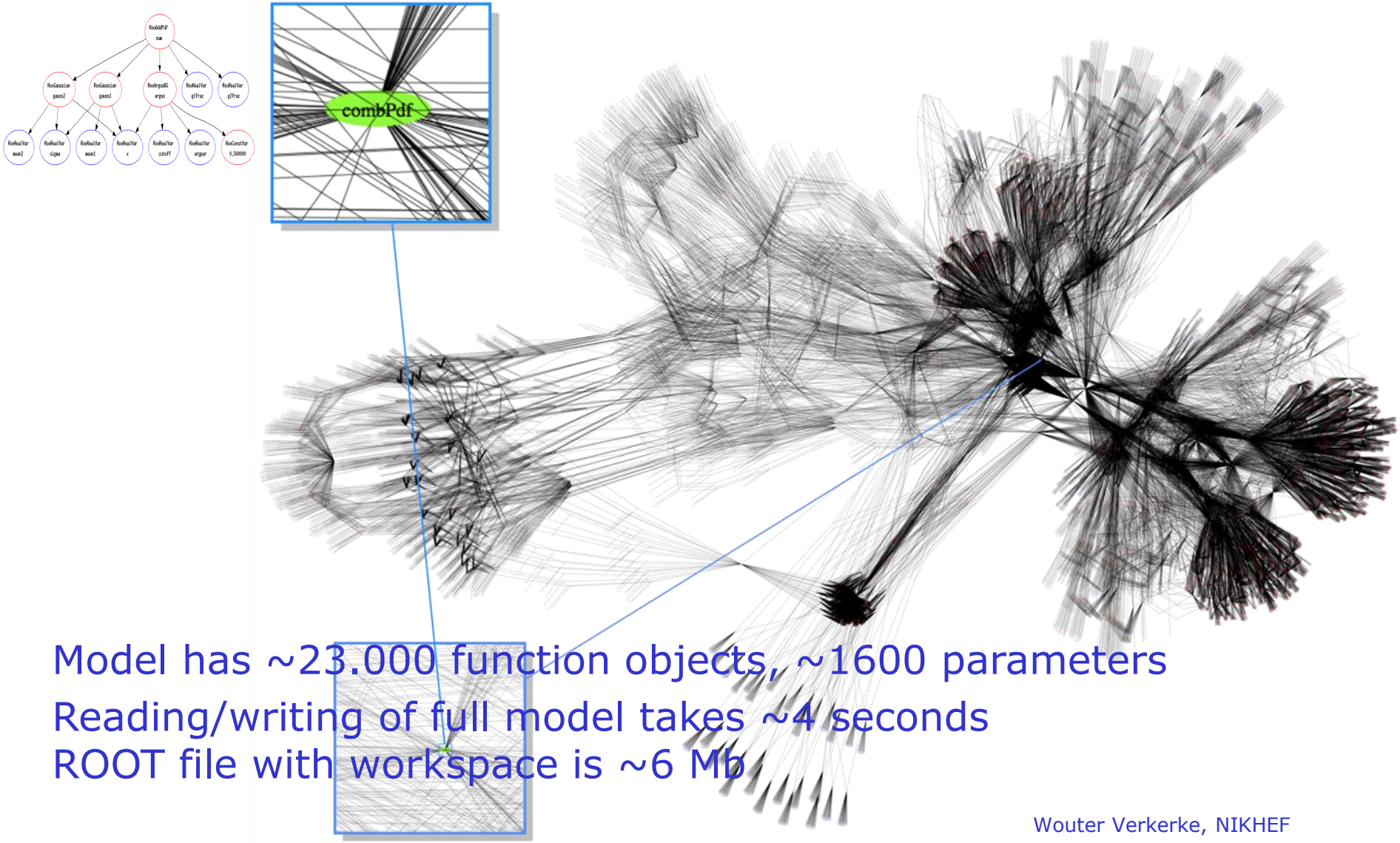



```
// Resurrect model and data
TFile f("model.root") ;
RooWorkspace* w = f.Get("w") ;
RooAbsPdf* model = w->pdf("sum") ;
RooAbsData* data = w->data("xxx") ;

// Use model and data
model->fitTo(*data) ;

RooPlot* frame = w->var("dt")->frame() ;
data->plotOn(frame) ;
model->plotOn(frame) ;
```



How well does it scale?

- Graph of the full ATLAS Higgs combination model



Model has ~23.000 function objects, ~1600 parameters

Reading/writing of full model takes ~4 seconds

ROOT file with workspace is ~6 Mb

What else can be in the workspace?

- Probability density models and all their components
  - All components (pdf, function, variable) individually accessible
  - Method exists to export all workspace contents to a CINT namespace (correctly typed) → simplifies interactive access

```
w->exportToCint("w") ;
w::model.fitTo(w::data) ;
```

  - Named sets of parameters,
    snapshot of values of sets of parameters
  - **`RooStats::ModelConfig`** objects (construct a uniquely defined statistical problem that can be computed by RooStats from a pdf and a dataset)
  - Code of custom RooFit classes that are note in the release. Recognition of custom classes and import of corresponding code fully automated

```
w->importClassCode() ;
```

Importing class code

- Reading in the class code

```
root [0] RooRealVar x("x","x",-0,10) ;
root [2] RooExample f("f","f",x) ; // CUSTOM CLASS
root [3] RooWorkspace w("w") ;

root [4] w.import(f) ;
root [5] w.importClassCode()
[#1] INFO:ObjectHandling -- RooWorkspace::autoImportClass(w) importing code of
class RooExample from /project/atlas/Users/verkerke/roofit/workdir/./RooExample.cxx
and /project/atlas/Users/verkerke/roofit/workdir/./RooExample.h
root [6] w.writeToFile("example.root") ;
```

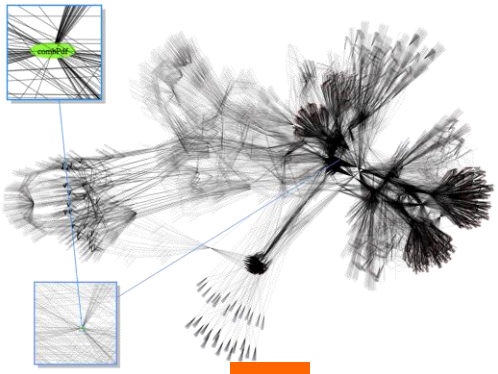
- Opening a workspace with custom code

```
tevere:pts/16 ~/roofit/workdir$root -l example.root
root [1] w
[#1] INFO:ObjectHandling -- RooWorkspace::CodeRepo::compileClasses() creating code export directory
.wscode.2e33dd96-8b42-11e2-9717-bcb910c0beef.w to extract coded embedded in workspace
[#1] INFO:ObjectHandling -- RooWorkspace::CodeRepo::compileClasses() Extracting declaration code of
class RooExample, file .wscode.2e33dd96-8b42-11e2-9717-bcb910c0beef.w/RooExample.h
[#1] INFO:ObjectHandling -- RooWorkspace::CodeRepo::compileClasses() Extracting implementation code
of class RooExample, file .wscode.2e33dd96-8b42-11e2-9717-bcb910c0beef.w/RooExample.cxx
[#1] INFO:ObjectHandling -- RooWorkspace::CodeRepo::compileClasses() Compiling code unit
RooExample to define class RooExample
Info in <TUnixSystem::ACLiC>: creating shared library /project/atlas/Users/verkerke/roofit/workdir/
.wscode.2e33dd96-8b42-11e2-9717-bcb910c0beef.w/RooExample_cxx.so
(class RooWorkspace*)0x8081010
```

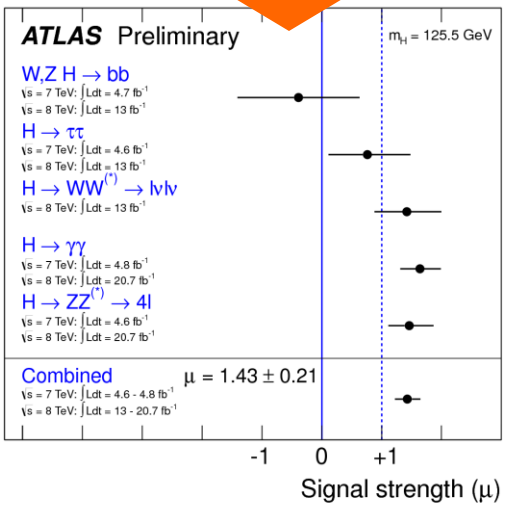
Collaborative analyses with workspaces

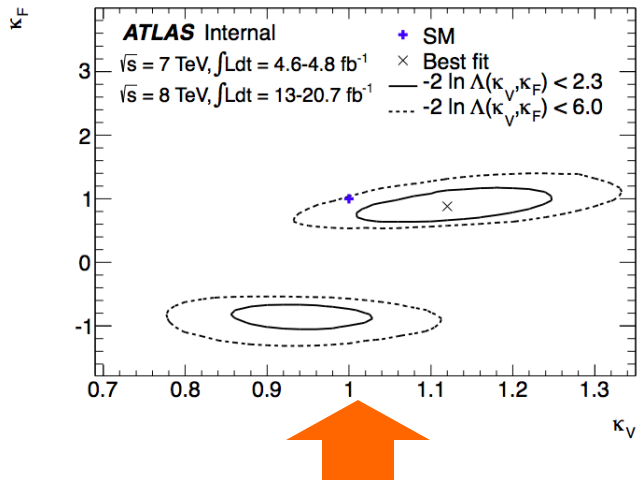
- Workspaces allow to share and modify very complex analyses with very little *technical* knowledge required
- Example: Higgs coupling fits

Full Higgs model



Signal strength in 5 channels





$$\sigma(gg \to H) * \mathrm{BR}(H \to \gamma\gamma) \sim \frac{\kappa_F^2 \cdot \kappa_\gamma^2(\kappa_F, \kappa_V)}{0.75 \cdot \kappa_F^2 + 0.25 \cdot \kappa_V^2}$$

$$\sigma(qq' \to qq'H) * \mathrm{BR}(H \to \gamma\gamma) \sim \frac{\kappa_V^2 \cdot \kappa_\gamma^2(\kappa_F, \kappa_V)}{0.75 \cdot \kappa_F^2 + 0.25 \cdot \kappa_V^2}$$

$$\sigma(gg \to H) * \mathrm{BR}(H \to ZZ^{(*)}, H \to WW^{(*)}) \sim \frac{\kappa_F^2 \cdot \kappa_V^2}{0.75 \cdot \kappa_F^2 + 0.25 \cdot \kappa_V^2}$$

$$\sigma(qq' \to qq'H) * \mathrm{BR}(H \to ZZ^{(*)}, H \to WW^{(*)}) \sim \frac{\kappa_V^2 \cdot \kappa_V^2}{0.75 \cdot \kappa_F^2 + 0.25 \cdot \kappa_V^2}$$

$$\sigma(qq' \to qq'H, VH) * \mathrm{BR}(H \to \tau\tau, H \to b\bar{b}) \sim \frac{\kappa_V^2 \cdot \kappa_F^2}{0.75 \cdot \kappa_F^2 + 0.25 \cdot \kappa_V^2},$$

Confidence intervals on Higgs fermion, v-boson couplings

Reparam in terms of fermion, v-boson scale factors

Collaborative analyses with workspaces

- How do you do this *in practice*?
- Write functions expressions corresponding to new parameterization

$$\sigma(gg \to H) * \mathrm{BR}(H \to \gamma\gamma) \sim \frac{\kappa_F^2 \cdot \kappa_\gamma^2(\kappa_F, \kappa_V)}{0.75 \cdot \kappa_F^2 + 0.25 \cdot \kappa_V^2}$$

```
RooFormulaVar mu_gg_func("mu_gg_func",
                         "(KF2*Kg2)/(0.75*KF2+0.25*KV2)",
                         KF2,Kg2,KV2) ;
```

- Edit existing model

```
w.import(mu_gg_func) ;
w.factory("EDIT::newmodel(model,mu_gg=mu_gg_gunc)") ;
```

*Top node of **modified** Higgs combination pdf*

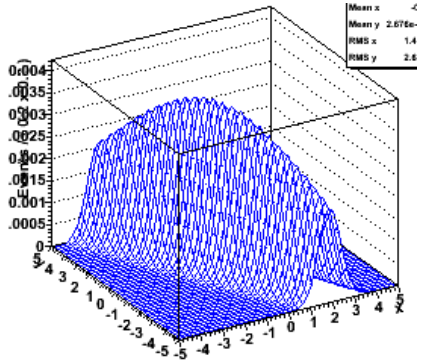
*Top node of **original** Higgs combination pdf*

***Modification prescription** replace parameter mu_gg with function mu_gg_func everywhere*

The workspace factory

- Example shown in previous slide uses 'factory' to construct objects in workspace.
  - Compact special language designed to quickly write RooFit pdfs
  - Derives directly from constructor syntax of RooFit objects, works for all users classes (Uses ROOT introspection)
  - Operators classes have special names and syntax for easier understanding (PROD,SUM,EDIT,…)

```
RooRealVar x("x","x",-10,10) ;
RooRealVar y("y","y",-10,10) ;
RooRealVar a("a","a",0) ;
RooRealVar b("b","b",-1.5) ;
RooFormulaVar m("a*y+b",a,y,b) ;
RooGaussian f("f","f",x,m,C(1)) ;
RooGaussian g("g","g",y,C(0),C(3))
RooProdPdf model("model","model",g,Conditional(f,y)) ;
```

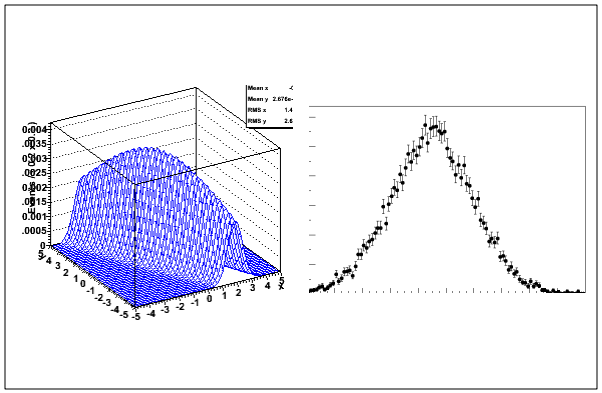


```
w.factory("PROD::model(Gaussian::f(x[-10,10],
          expr::m('a*y+b',a[1],b[0],y[-10,10]),1)|y,
          Gaussian::g(y,0,3)))
```

RooFit developments for LHC (Higgs analysis)

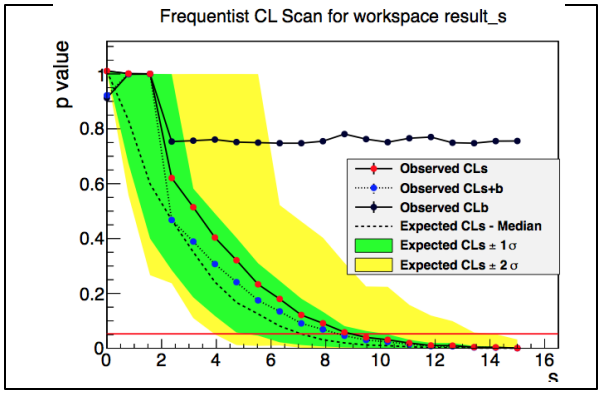
Class RooWorkspace

Simplify packaging and sharing of models



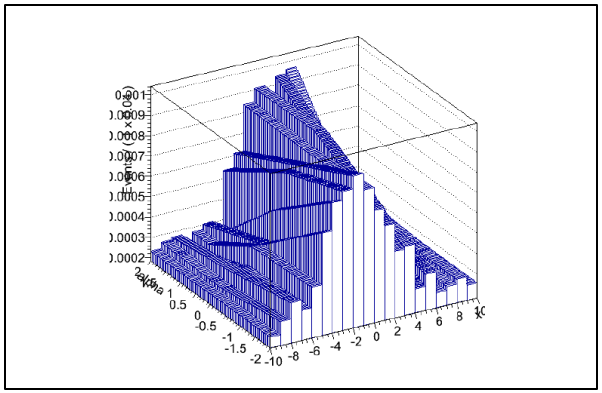
RooStats toolkit

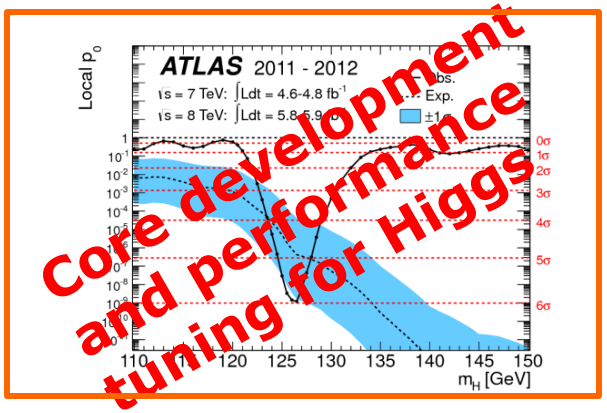
Statistical tests based on likelihoods from RooFit models



HistFactory package

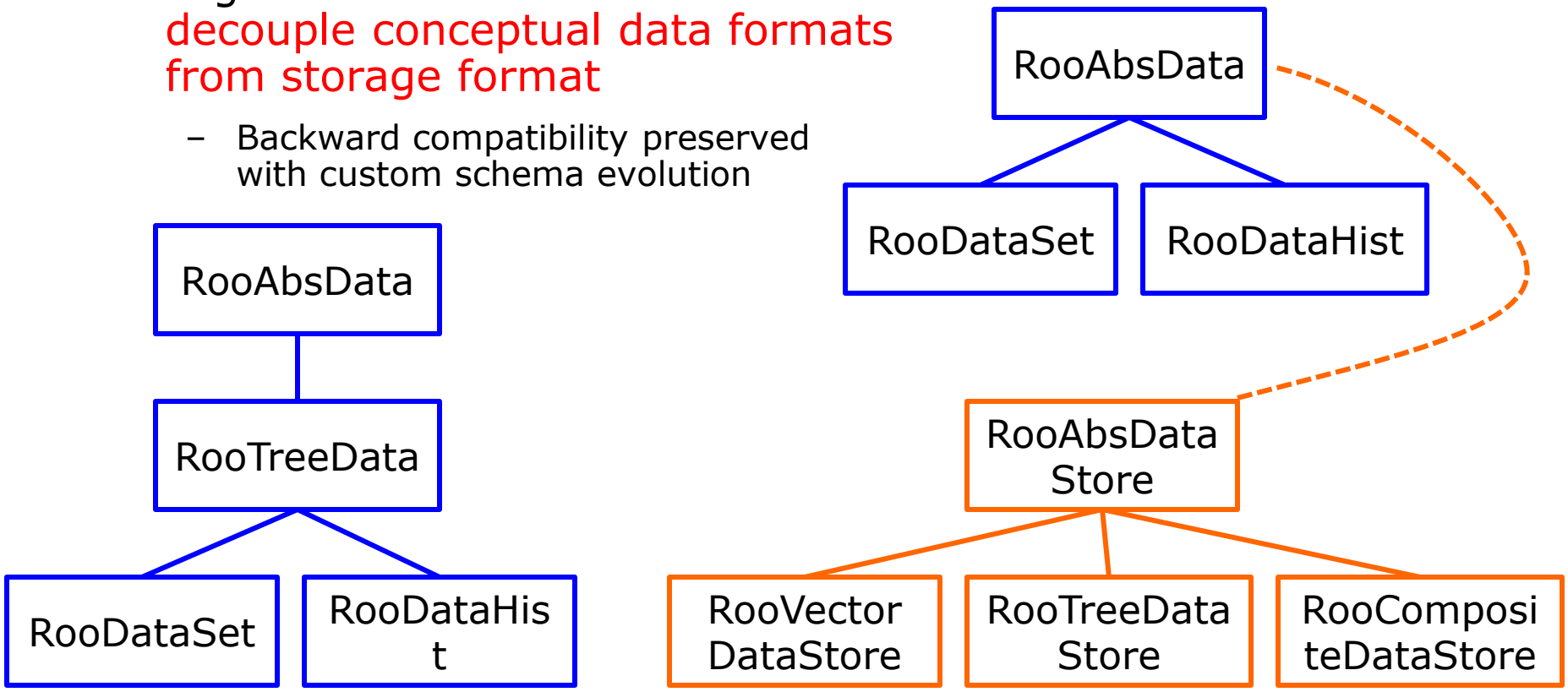
Constructing models from Monte Carlo templates





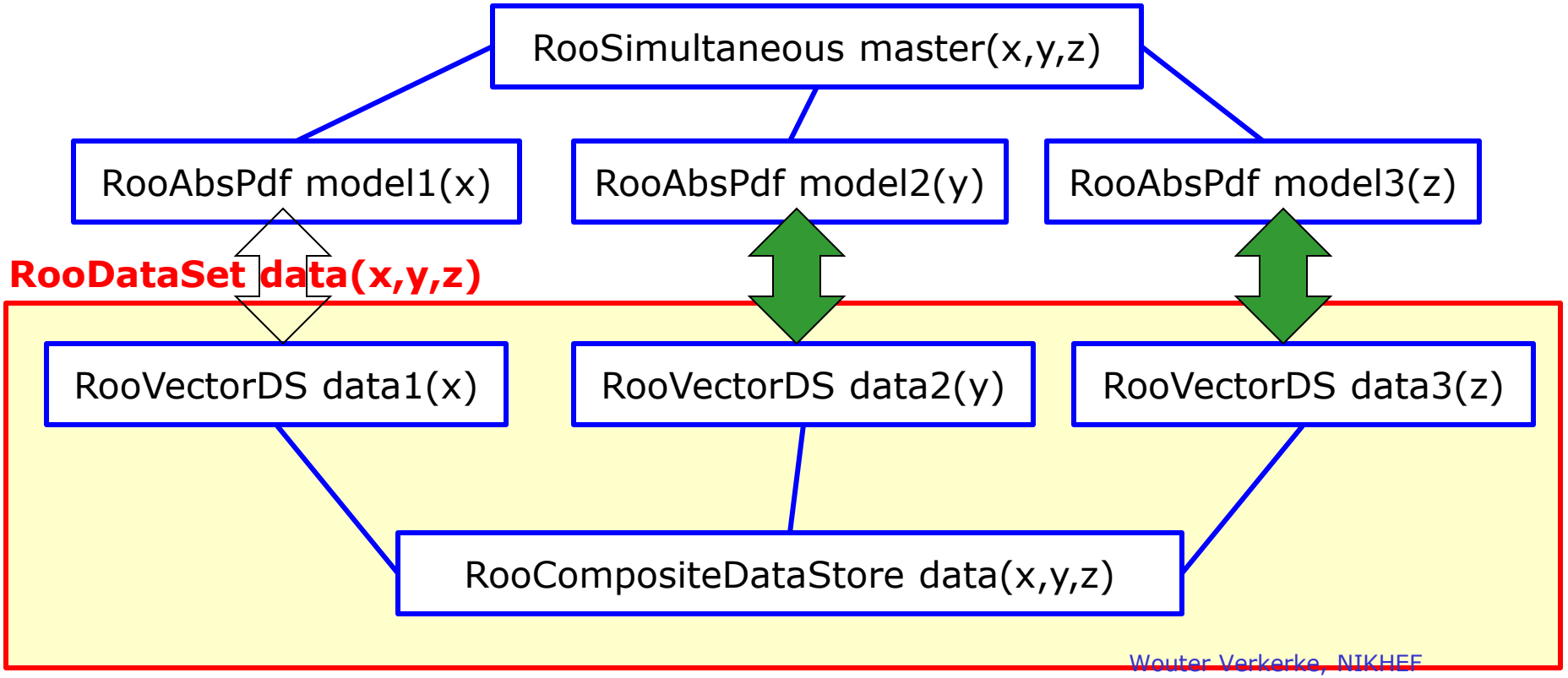
Optimization, parallelization, vectorization

- RooFit datasets have been migrated from TTree-based storage to std::vector based storage
  - Speedup in data reading inside likelihood by factor 40!
  - Essential TTree functionality of disk-based datasets not needed for likelihood fits
- Migrated class structure to decouple conceptual data formats from storage format
  - Backward compatibility preserved with custom schema evolution



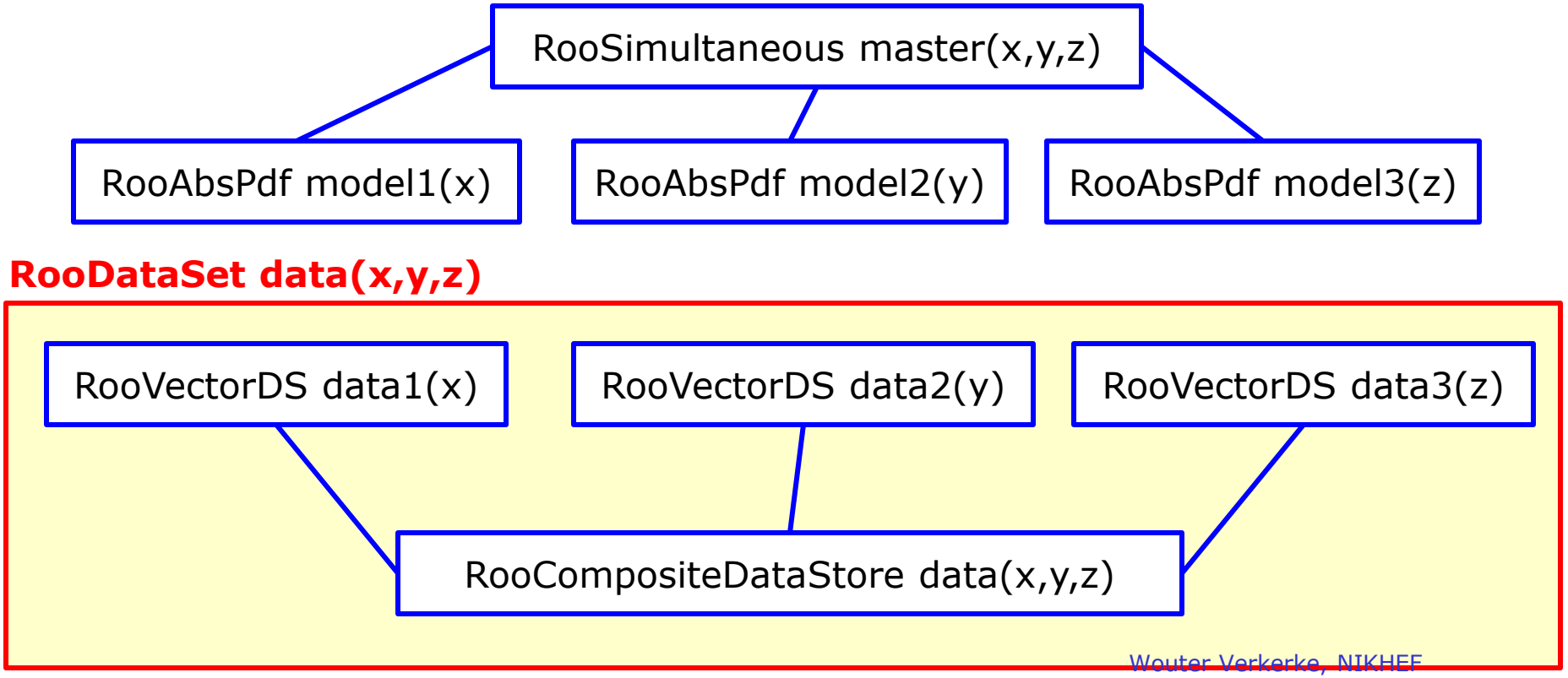
Composite datasets vs simultaneous pdfs

- Use of simultaneous pdfs quite common – you fit >1 sample at the same time
- Concept of composite datasets nicely complements this
  - Used to do without, but sizeable overhead is 'splitting' of a monolithic dataset in preparation of joint likelihood



Composite datasets vs simultaneous pdfs

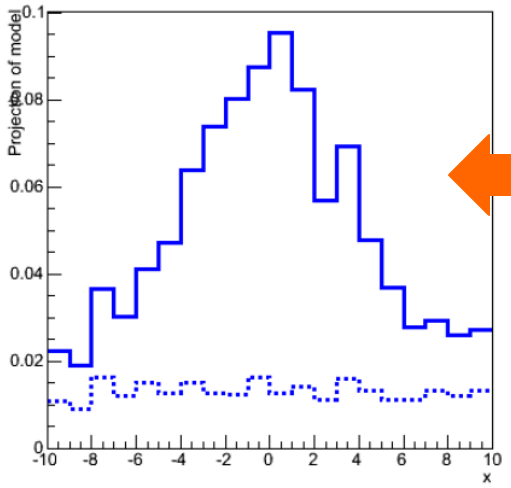
- Can decide *separately for each component* if it is binned/unbinned (→supports hybrid datasets)
- All-binned scenario is much more memory-efficient (3 x 1D histogram instead of 1 x 3D histogram)



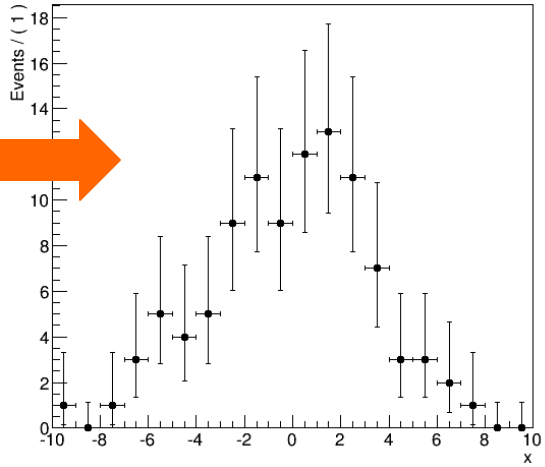
New use cases for binned data

- When the pdf is a template model – unbinned data makes no sense (pdf value is the same for all data points inside a bin of the template pdf)
- New: automatically generate binned data when pdf is 'binned'
  → Binned generation is computationally much more efficient
  - Not tied to a particular pdf class: 'binned-ness' and bin boundaries are advertised through virtual methods.
  - Addition, multiplication 'transparent' to binned-ness properties

Template model



Binned data

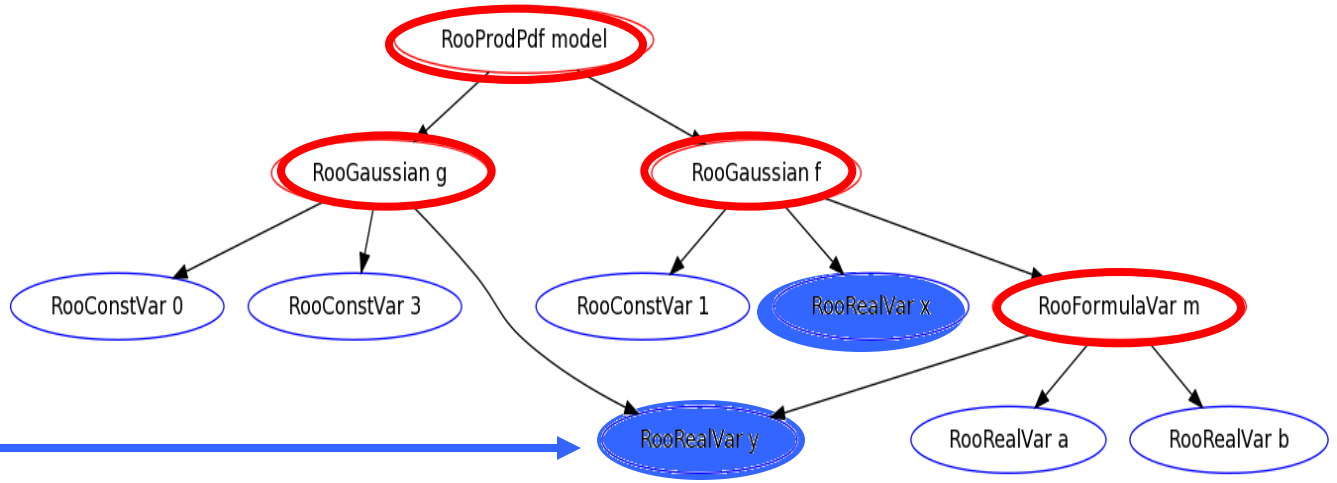


Optimization of likelihood calculations

- Likelihood evaluates pdf at all data points

$$-\log L(\vec{p}) = -\sum_{i=0\ldots n} \log f(\vec{x}_i, \vec{p})$$

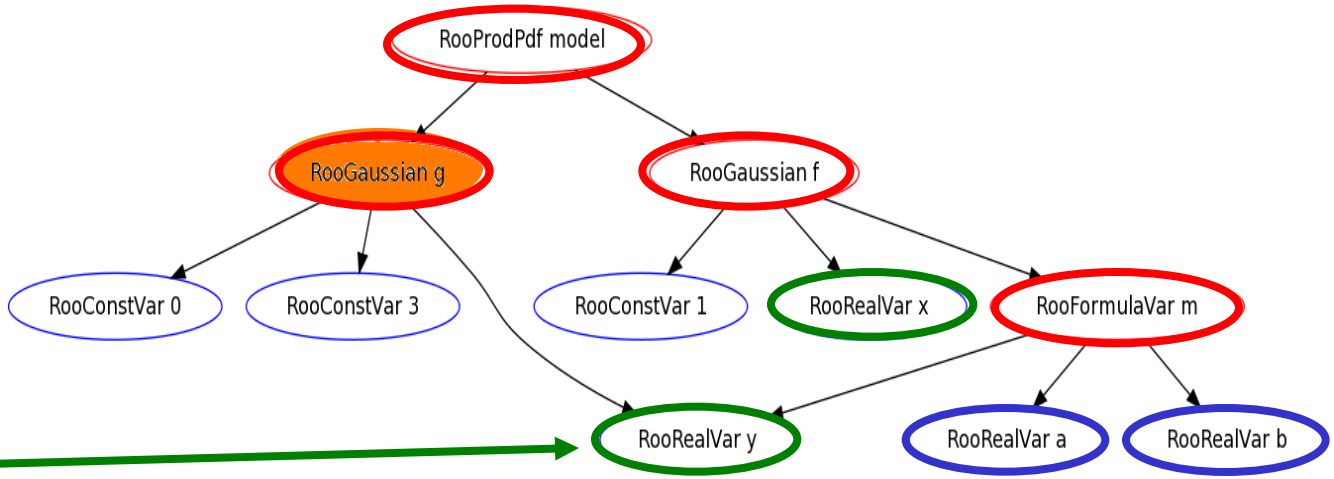
X	Y
1	2
2	3
0	1
-1	5
3	6
7	6
-3	-2



Level-1 optimization of likelihood calculation

- Constant terms (depend only on observables, not on parameters) are precalculated, added to dataset

X	Y	g
1	2	1.5
2	3	2.7
0	1	1.2
-1	5	0.6
3	6	9.8
7	6	3.4
-3	-2	5.7



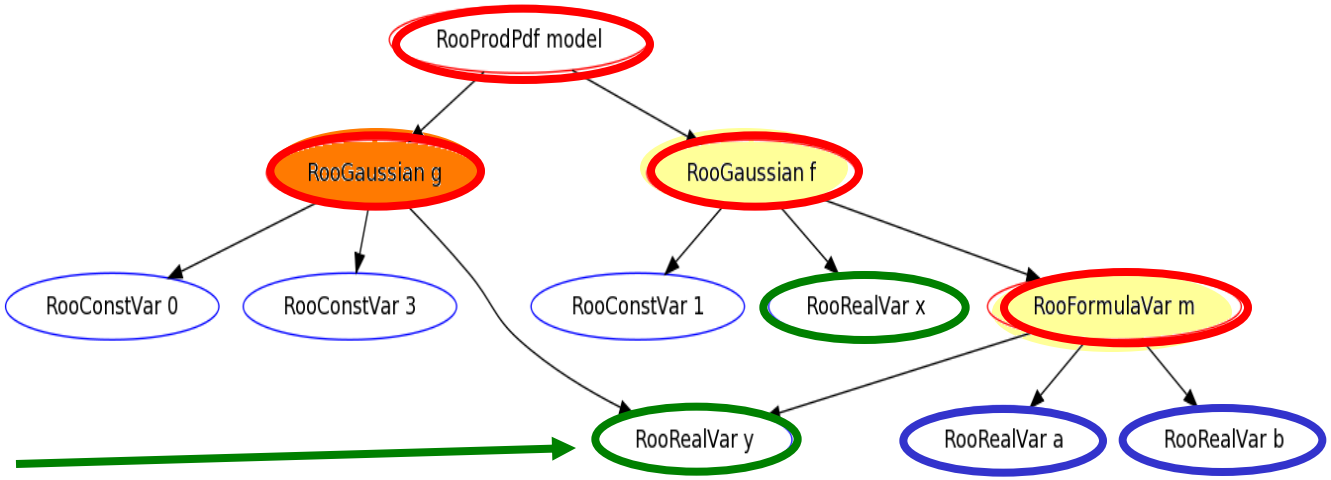
Level-2 optimization of likelihood calculation

- Precalculate all components, with conditional update strategy

Depends on m

Depends on a,b

X	Y	g	f	m
1	2	1.5	..	..
2	3	2.7	..	..
0	1	1.2	..	..
-1	5	0.6	..	..
3	6	9.8	..	..
7	6	3.4	..	..
-3	-2	5.7	..	..



Why does level-2 optimization help?

- Most minuit calls vary one parameter at a time (to calculate derivative of likelihood) → Computed cached values of most components will stay valid

prevFCN = 5170.289989 FCN=5170.53 FROM MIGRAD STATUS=INITIATE 6 CALLS 7 TOTAL
prevFCN = 4495.931306 a=0.9961, b=0.106, c=0.06274,
prevFCN = 3936.921265 a=0.9967,
prevFCN = 3936.938281 a=0.9954,
prevFCN = 3936.907905 a=0.9965,
prevFCN = 3936.933086 a=0.9956,
prevFCN = 3936.911321 a=0.9961, b=0.108,
prevFCN = 3937.05644 b=0.104,
prevFCN = 3936.790003 b=0.1074,
prevFCN = 3937.014478 b=0.1046,
prevFCN = 3936.829929 b=0.106, c=0.06845,
prevFCN = 3936.934463 c=0.05703,
prevFCN = 3936.911648 c=0.06688,
prevFCN = 3936.930463 c=0.05861,
prevFCN = 3936.913944 a=1, b=-0.02103, c=0.02074,
prevFCN = 3936.613348 a=0.9982, b=0.04018, c=0.04096,

Only a changes, caches depending on b,c remain valid

Only b changes, caches depending on a,c remain valid

Only c changes, caches depending on b,c remain valid

- Fraction of likelihood calls with 1-parameter change increases ~linearly with number of parameters

From level-2 optimization to vectorization

- Resequencing of calculation in full level-2 optimization results in 'natural ordering' for complete vectorization

Level-1 sequence

$m(y_0)$
$f(m_0)$
$g(x_0)$
$Model(f_0,g_0)$

$m(y_1)$
$f(m_1)$
$g(x_1)$
$Model(f_1,g_1)$

$m(y_2)$
$f(m_2)$
$g(x_2)$
$Model(f_2,g_2)$

Level-2 sequence

$m(y_0)$
$m(y_1)$
$m(y_2)$

$f(m_0)$
$f(m_1)$
$f(m_2)$

$g(x_0)$
$g(x_1)$
$g(x_2)$

$Model(f_0,g_0)$
$Model(f_1,g_1)$
$Model(f_2,g_2)$

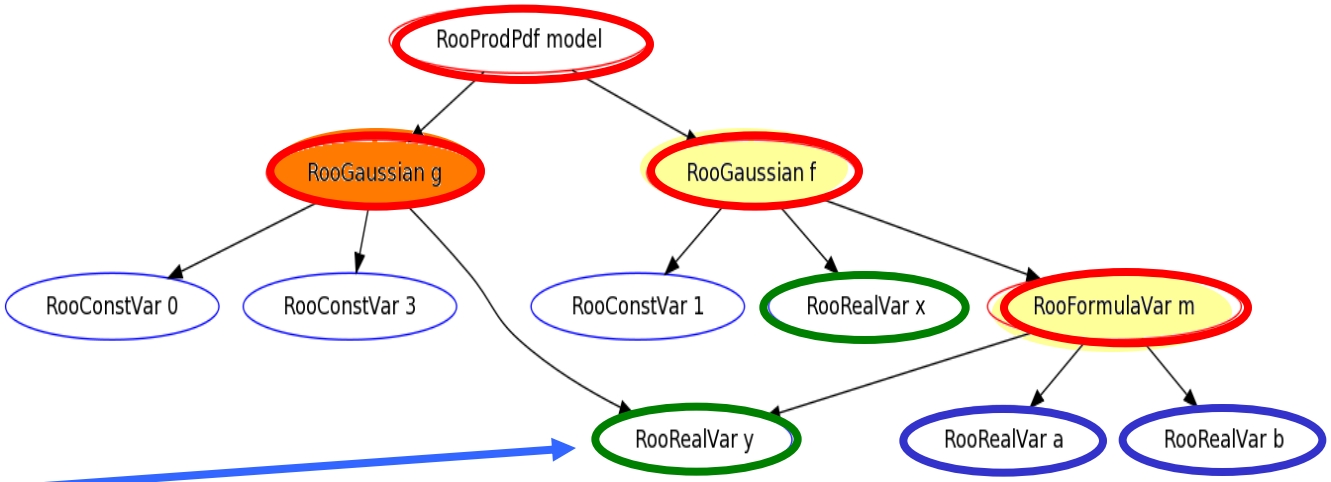
Level-2 optimization of likelihood calculation

- Pass complete vector x to m::evaluate()

Depends on m

Depends on a,b

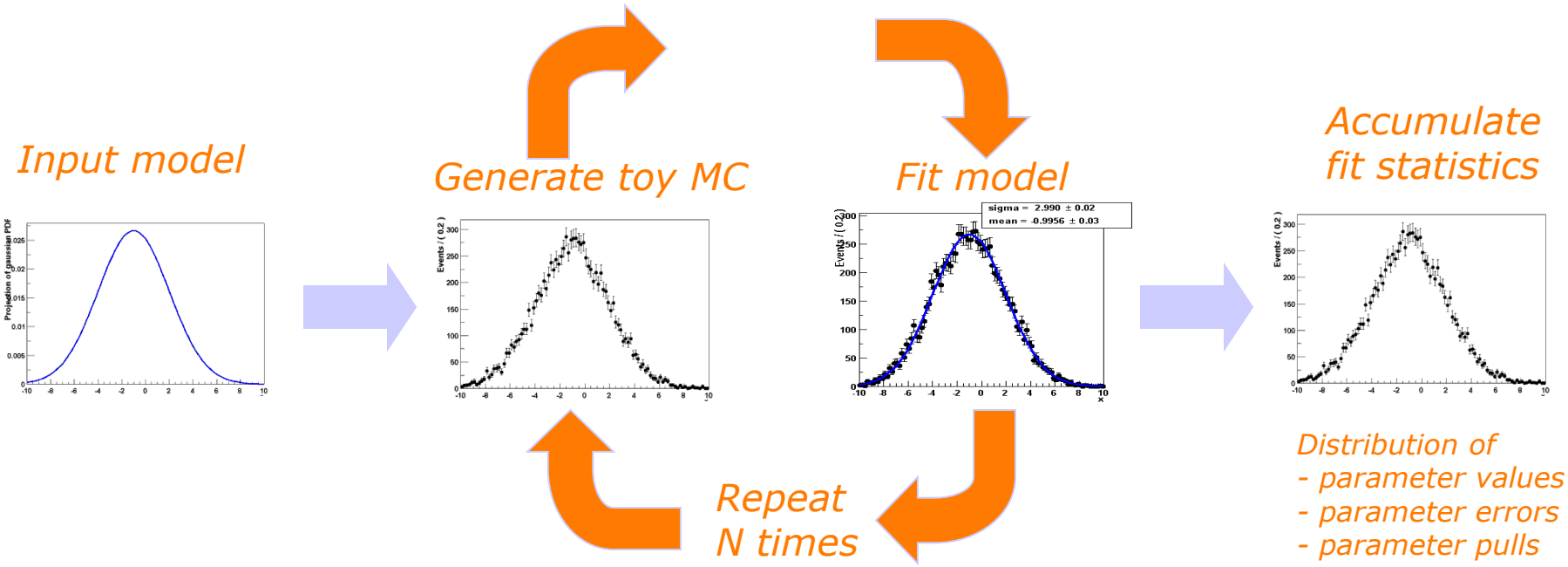
X	Y	g	f	m
1	2	1.5	..	..
2	3	2.7	..	..
0	1	1.2	..	..
-1	5	0.6	..	..
3	6	9.8	..	..
7	6	3.4	..	..
-3	-2	5.7	..	..



Proxy class inside function object m can be trivially adapted to 'see' full vector x instead of 'current value'

Parallelization using PROOF

- Simple parallelization of likelihood calculation using NumCPU(n) option of RooAbsPdf::fitTo() very popular, but restricted to likelihood calculations
- Another common CPU-intensive task are toy studies



- Ideally parallelizable, now have generic interface to PROOF(-lite) to parallelize loop

Parallelization using PROOF

- Code demo

```
// Create workspace with p.d.f
RooWorkspace* ww = new RooWorkspace("ww") ;
ww->factory("Gaussian::g(x[-10,10],mean[-10,10],sigma[3,0.1,10])") ;

// Configure what should be done for one 'toy' (generate & fit)
RooGenFitStudy gfs ;
gfs.setGenConfig("g","x",NumEvents(1000)) ;
gfs.setFitConfig("g","x",PrintLevel(-1)) ;

// Manager object that drives PROOF
RooStudyManager mgr(*ww,gfs) ;

// Requires parallel execution on PROOF-lite (farm name="")
mgr.runProof(100,"",kFALSE) ;

// Print the collected summary data
gfs.summaryData()->Print() ;
```

NB: No need to 'place' data: workspace contains complete problem definition, streamed via PROOF I/O channels

Summary & Outlook

- Large number of improvements made in RooFit in the past 2 years, largely driven by needs for Higgs discovery
  - Workspace concept for sharing and persisting models and code
  - Factory language for rapid construction, editing of complex models (e.g. in-situ editing to combine, reparameterize Higgs channels)
  - More advanced optimizations of the likelihood calculation
  - Hybrid binned/unbinned datasets
  - New types of pdfs (MC template shapes) → In separate package HistFactory (K.Cranmer et al. + WV)
  - Framework for parallelization of statistical tasks (generate+fit toy cycles) with back-end to PROOF, batch facilities
- Lot of 'core engineering' done to enable powerful new features – but potential not yet fully unlocked: convenient high-level user interface often missing (todo)

Outlook

- Goal for the next year(s) - Stability
  - No major outstanding new features
  - But several powerful new features that already exist will become easily available to end-user (hybrid datasets, PROOF parallelization, more workspace factory features)
  - Cleanup of some conceptual inconsistencies in handling of counting-only data (requires a separate data type)
  - Further work on optimization algorithms
  - Backward compatibility of code and files: ROOT files with models persisted in RooWorkspaces should be readable in all future versions
    - Extensive regression testing in place to make sure that code is backward compatible and new code is bug-free – but it is very difficult to capture all use cases!
- Documentation
  - **More tutorial macros and documentation will come**