Statistical model building at the LHC

Wouter Verkerke (Nikhef & University of Amsterdam)
The HEP analysis workflow illustrated

Simulation of high-energy physics process

Simulation of ‘soft physics’ physics process

Simulation of ATLAS detector

LHC data

Prob(data|SM)

P(m_4l|SM[m_{H}])

Observed m_4l

Analysis Event selection

Reconstruction of ATLAS detector
How is Higgs discovery different from a simple fit?

**Gaussian + polynomial**

\[ L(\bar{N} | \mu, \bar{\theta}) = \prod_i \text{Poisson}(N_i | f(x_i, \mu, \bar{\theta})) \]

"inside ROOT"

ML estimation of parameters \( \mu, \bar{\theta} \) using MINUIT (MIGRAD, HESSE, MINOS)

\[ \mu = 5.3 \pm 1.7 \]

**Higgs combination model**
How is Higgs discovery different from a simple fit?

Gaussian + polynomial

\[ L(\tilde{N} | \mu, \tilde{\theta}) = \prod_i \text{Poisson}(N_i | f(x_i, \mu, \tilde{\theta})) \]

```
ROOT TH1
ROOT TF1
```

“inside ROOT”

ML estimation of parameters \( \mu, \tilde{\theta} \) using MINUIT (MIGRAD, HESSE, MINOS)

\[ \mu = 5.3 \pm 1.7 \]

Likelihood Model
orders of magnitude more complicated. Describes

- \( O(100) \) signal distributions
- \( O(100) \) control sample distr.
- \( O(1000) \) parameters representing syst. uncertainties

Frequentist confidence interval construction and/or p-value calculation not available as ‘ready-to-run’ algorithm in ROOT

Wouter Verkerke, NIKHEF
How is Higgs discovery different from a simple fit?

**Gaussian + polynomial**

**Higgs combination model**

\[
L(\tilde{N} | \mu, \theta) = \prod_i \text{Poisson}(N_i | f(x_i, \mu, \theta))
\]

ML estimation of parameters $\mu, \theta$ using MINUIT (MIGRAD, HESSE, MINOS)

$\mu = 5.3 \pm 1.7$

Wouter Verkerke, NIKHEF
How is Higgs discovery different from a simple fit?

Gaussian + polynomial

\[ L(\hat{N} | \mu, \bar{\theta}) = \prod_i \text{Poisson}(N_i | f(x_i, \mu, \bar{\theta})) \]

"inside ROOT"

Model Usage phase (use \( L(x|H) \) to make statement on \( H \))

ML estimation of parameters \( \mu, \bar{\theta} \) using MINUIT (MIGRAD, HESSE, MINOS)

ROOT TH1

ROOT TF1

Wouter Verkerke, NIKHEF
**How is Higgs discovery different from a simple fit?**

<table>
<thead>
<tr>
<th>Gaussian + polynomial</th>
<th>Higgs combination model</th>
</tr>
</thead>
</table>

**Design goal:**

- Separate **building of Likelihood model** as much as possible from statistical analysis using the Likelihood model
- More modular software design
- ‘Plug-and-play with statistical techniques
- Factorizes work in collaborative effort

**ML estimation of parameters $\mu, \theta$ using MINUIT** (MIGRAD, HESSE, MINOS)

$\mu = 5.3 \pm 1.7$

*Wouter Verkerke, NIKHEF*
The idea behind the design of RooFit/RooStats/HistFactory

- Modularity, Generality and flexibility

- Step 1 – Construct the likelihood function $L(x|p)$

RooFit, or RooFit+HistFactory

- Step 2 – Statistical tests on parameter of interest $p$

Procedure can be Bayesian, Frequentist, or Hybrid), but always based on $L(x|p)$

RooStats

- Steps 1 and 2 are conceptually separated, and in Roo* suit also implemented separately.
The idea behind the design of RooFit/RooStats/HistFactory

- Steps 1 and 2 can be ‘physically’ separated (in time, or user)
- **Step 1** – Construct the likelihood function \( L(x|p) \)

RooFit, or RooFit+HistFactory

RooWorkspace

- **Step 2** – Statistical tests on parameter of interest \( p \)

RooStats

Wouter Verkerke, NIKHEF
The benefits of modularity

- Perform different statistical tests on exactly the same model

RooFit, or RooFit+HistFactory

\[ \downarrow \]

RooWorkspace

\[ \downarrow \]

“Simple fit”
(ML Fit with HESSE or MINOS)

RooStats
(Frequentist with toys)

RooStats
(Frequentist asymptotic)

RooStats
Bayesian MCMC

Wouter Verkerke, NIKHEF
RooFit
RooFit – Focus: coding a probability density function

- Focus on one practical aspect of many data analysis in HEP: How do you formulate your p.d.f. in ROOT
  - For ‘simple’ problems (gauss, polynomial) this is easy
  - But if you want to do unbinned ML fits, use non-trivial functions, or work with multidimensional functions you quickly find that you need some tools to help you

- The RooFit project started in 1999 for data modeling needs for BaBar collaboration initially, publicly available in ROOT since 2003
RooFit core design philosophy

- Mathematical objects are represented as C++ objects

<table>
<thead>
<tr>
<th>Mathematical concept</th>
<th>RooFit class</th>
</tr>
</thead>
<tbody>
<tr>
<td>variable ( x )</td>
<td>RooRealVar</td>
</tr>
<tr>
<td>function ( f(x) )</td>
<td>RooAbsReal</td>
</tr>
<tr>
<td>PDF ( f(x) )</td>
<td>RooAbsPdf</td>
</tr>
<tr>
<td>space point ( \vec{x} )[x_{\text{max}}]</td>
<td>RooArgSet</td>
</tr>
<tr>
<td>integral ( \int_{x_{\text{min}}}^{x_{\text{max}}} f(x)dx )</td>
<td>RooRealIntegral</td>
</tr>
<tr>
<td>list of space points</td>
<td>RooAbsData</td>
</tr>
</tbody>
</table>

Wouter Verkerke, NIKHEF
Data modeling – Constructing composite objects

- Straightforward correlation between mathematical representation of formula and RooFit code

Math: \( \text{gauss}(x, m, \sqrt{s}) \)

RooFit diagram:

- \( \text{RooRealVar } x \)
- \( \text{RooRealVar } m \)
- \( \text{RooRealVar } s \)
- \( \text{RooFormulaVar } \text{sqrts} \)
- \( \text{RooGaussian } g \)

RooFit code:

1. \( \text{RooRealVar } x(“x”, “x”, -10, 10); \)
2. \( \text{RooRealVar } m(“m”, “mean”, 0); \)
3. \( \text{RooRealVar } s(“s”, “sigma”, 2, 0, 10); \)
4. \( \text{RooFormulaVar } \text{sqrts}(“\text{sqrts}”, “\sqrt{\text{s}}”, \text{s}); \)
5. \( \text{RooGaussian } g(“g”, “\text{gauss}”, x, m, \text{sqrts}); \)
RooFit core design philosophy

- A special container class owns all objects that together build a likelihood function

**Math**

\[\text{Gauss}(x, \mu, \sigma)\]

**RooFit diagram**

- RooWorkspace (keeps all parts together)

**RooFit code**

```cpp
RooRealVar x("x","x",-10,10) ;
RooRealVar m("m","y",0,-10,10) ;
RooRealVar s("s","z",3,0.1,10) ;
RooGaussian g("g","g",x,m,s) ;
RooWorkspace w("w") ;
w.import(g) ;
```

New feature for LHC
Populating a workspace the easy way – “the factory”

- The **factory** allows to fill a workspace with pdfs and variables using a simplified scripting language

<table>
<thead>
<tr>
<th>Math</th>
<th>RooWorkspace</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gauss($x,\mu,\sigma$)</td>
</tr>
</tbody>
</table>

**RooFit diagram**

- RooRealVar $x$
- RooRealVar $m$
- RooRealVar $s$
- RooGaussian $g$

**RooFit code**

```cpp
RooWorkspace w("w") ;
w.factory("Gaussian::g(x[-10,10],m[-10,10],s[3,0.1,10])");
```

**New feature for LHC**
The power of pdf as building blocks – adaptability

- RooFit classes do not require their parameter arguments to be variables, one can plug in functions as well
- Allows trivial customization, extension of probability models

```cpp
class RooGaussian

Gauss(x | µ, σ)  Gauss(x | µ ⋅ (1 + 2α), σ)
```

Introduce a response function for a systematic uncertainty

```cpp
// Original Gaussian
w.factory("Gaussian::g1(x[80,100],m[91,80,100],s[1])")

// Gaussian with response model in mean
w.factory("expr::m_response("m*(1+2alpha)",m,alpha[-5,5])")
; w.factory("Gaussian::g1(x,m_response,s[1])")
```

NB: “expr” operates builds an interpreted function expression on the fly
The power of pdf as building blocks – Advanced algorithms

• Example: a ‘kernel estimation probability model’
  - Construct smooth pdf from unbinned data, using kernel estimation technique

Sample of events | Gaussian pdf for each event | Summed pdf for all events | Adaptive Kernel: width of Gaussian depends on local event density

• Example

```c
w.import(myData,Rename(“myData”)) ;
w.factory(“KeysPdf::k(x,myData)”);```

• Also available for n-D data
Powerful operators – Fourier convolution

- Convolve any two arbitrary pdfs with a 1-line expression

```cpp
w.factory("Landau::L(x[-10,30],5,1)") :
w.factory("Gaussian::G(x,0,2)") ;

w::x.setBins("cache",10000) ; // FFT sampling density
w.factory("FCONV::LGf(x,L,G)") ; // FFT convolution
```

- Exploits power of FFTW package available via ROOT
  - Hand-tuned assembler code for time-critical parts
  - Fast! unbinned ML fit to 10,000 events take O(1) second.
Constructing & using the likelihood function

- Plot the likelihood function versus a parameter

```cpp
RooAbsReal* nll = w::model.createNLL(data);
RooPlot* frame = w::param.frame();
nll->plotOn(frame, ShiftToZero());
```

- Maximum Likelihood estimation of parameters and variance

```cpp
RooMinimizer m(*nll);

// ML Parameter estimation
m.minimize("Minuit2","migrad");

// Variance estimation
m.hesse();

// Alternatively – all this in one line
pdf->fitTo(*data);
```
Working with the likelihood function

- Also profile likelihood ratio can be represented as RooFit function,

\[
\lambda(\mu) = \frac{L(\mu, \hat{\theta}(\mu))}{L(\hat{\mu}, \hat{\theta})}
\]

```
RooAbsReal* nll = w::model.createNLL(data) ;
RooAbsReal* L = nll->createProfile(w::mu) ;
L->plotOn(frame) ;
```
Parametric template models, modeling systematic uncertainties
From empirical probability models to simulation-based models

- Important feature of LHC hadronic physics is that distributions usually don’t follow simple analytical shapes

Unbinned analytical probability model

(Geant) Simulation-driven binned template model

- But concept of simulation-driven template models can also be extended to include parameters (corresponding to systematic uncertainties)

Wouter Verkerke, NIKHEF
The HEP analysis workflow illustrated

Simulation of high-energy physics process

Simulation of ‘soft physics’ physics process

Soft Theory uncertainties

Simulation of ATLAS detector

Detector modelling uncertainties

Analysis Event selection

Reconstruction of ATLAS detector

P(m_4|SM[m_H])

Observed m_4l

LHC data

Hard Theory uncertainties

prob(data|SM)
Modeling of shape systematics in the likelihood

- Effect of any systematic uncertainty that affects the shape of a distribution can in principle be obtained from MC simulation chain
  - E.g. obtain histogram templates for distributions at '+1\sigma' and '-1\sigma' settings of systematic effect

```
\hat{\mu}, \hat{\theta} \quad \lambda(\mu) = \frac{L(\mu, \hat{\theta}(\mu))}{L(\hat{\mu}, \hat{\theta})}
```

Challenge: construct an empirical response function based on the interpolation of the shapes of these three templates.

Wouter Verkerke, NIKHEF
Need to interpolate between template models

- Need to define ‘morphing’ algorithm to define distribution $s(x)$ for each value of $\alpha$
Piecewise linear interpolation

- Simplest solution is piece-wise linear interpolation for each bin.
Visualization of bin-by-bin linear interpolation of distribution

Wouter Verkerke, NIKHEF
Introduce notion of subsidiary measurement

- Construct a full model of a systematic uncertainty
  1. Model of physics measurement that includes description of response to NP alpha using template morphing approach
  2. Subsidiary measurement that describes external knowledge (‘prescription’) of the systematic uncertainty

\[ L_{\text{full}}(x \mid \mu, \alpha) = L_{\text{phys}}(x \mid \mu, \alpha) \cdot L_{\text{subs}}(\alpha) \]

Often \( L(\alpha) \) is a unit Gaussian

\[ L_{\text{subs}}(\alpha) = \text{Gauss}(0 \mid \alpha, 1) \]

Response function must be chosen to match definition of alpha

(e.g. templates at \( \alpha = \pm 1 \) sampled at variation of \( \pm 1\sigma \) of underlying uncertain parameter

Wouter Verkerke, NIKHEF
Example 2: binned $L$ with syst

- Example of template morphing systematic in a binned likelihood

$$s_i(\alpha, \ldots) = \begin{cases} 
  s_i^0 + \alpha \cdot (s_i^+ - s_i^0) & \forall \alpha > 0 \\
  s_i^0 + \alpha \cdot (s_i^- - s_i^0) & \forall \alpha < 0
\end{cases}$$

$$L(\vec{N} | \alpha, \vec{s}^-, \vec{s}^0, \vec{s}^+) = \prod_{\text{bins}} P(N_i | s_i(\alpha, s_i^-, s_i^0, s_i^+)) \cdot G(0 | \alpha, 1)$$

// Import template histograms in workspace
w.import(hs_0, hs_p, hs_m);

// Construct template models from histograms
w.factory("HistFunc::s_0(x[80,100],hs_0)");  
w.factory("HistFunc::s_p(x,hs_p)");  
w.factory("HistFunc::s_m(x,hs_m)");

// Construct morphing model
w.factory("PiecewiseInterpolation::sig(s_0,s_-,m,s_p,alpha[-5,5])");

// Construct full model
w.factory("PROD::model(ASUM(sig,bkg,f[0,1]),Gaussian(0,alpha,1))");
Modeling multiple sources of systematic uncertainty

- Modeling multiple sources of systematic uncertainty affecting a single measurement is also easy

\[
L(N,0|s,\alpha_{\text{JES}},\alpha_{\text{ISR}}) = P(N|s + b(1 + 0.1\alpha_{\text{JES}} + 0.05\alpha_{\text{ISR}})) \cdot G(0|\alpha_{\text{JES}},1) \cdot G(0|\alpha_{\text{ISR}},1)
\]

Joint response function for both systematics

One subsidiary measurement for each source of uncertainty
Modeling systematic uncertainties in multiple channels

- Single sources of systematic uncertainty can also coherently affect multiple physics measurements
  - Example – Likelihood of two Poisson counting measurements
  - Magnitude and sign effect does not need to be same

\[
L(N_A, N_B \mid s, \alpha_{JES}) = P(N_A \mid s \cdot f_A + b_A (1 + 0.1\alpha_{JES})) \\
\times P(N_B \mid s \cdot f_B + b_B (1 - 0.3\alpha_{JES})) \\
\times G(0 \mid \alpha_{JES}, 1)
\]
Modeling template simulation uncertainties

- Uncertainties from finite simulation statistics can be described in the same way.

For templates stacking multiple samples, can choose to simplify and only introduce single Poisson for sum of samples ‘Beeston-Barlow light’

For each bin:

\[
L(N, N_{MC} \mid s, b) = \text{Poisson}(N \mid s + b) \cdot \text{Poisson}(N_{MC} \mid \tau \cdot b)
\]

Constant factor \( \tau = L(\text{MC})/L(\text{data}) \)
Example 4 – Beeston-Barlow light

- Beeston-Barlow-(lite) modeling of MC statistical uncertainties

\[ L(\tilde{N} | \tilde{\gamma}) = \prod_{\text{bins}} P(N_i | \gamma_i(\tilde{s}_i + \tilde{b}_i)) \prod_{\text{bins}} P(\tilde{s}_i + \tilde{b}_i | \gamma_i(\tilde{s}_i + \tilde{b}_i)) \]

---

```cpp
// Import template histogram in workspace
w.import(hs);

// Construct parametric template models from histograms
// implicitly creates vector of gamma parameters
w.factory("ParamHistFunc::s(hs)") ;

// Product of subsidiary measurement
w.factory("HistConstraint::subs(s)") ;

// Construct full model
w.factory("PROD::model(s,subs)") ;
```

- Wouter Verkerke, NIKHEF
Example 5 – BB-lite + morphing

- Template morphing model with Beeston-Barlow-lite MC statistical uncertainties

\[ s_i(\alpha, \ldots) = \begin{cases} 
  s_i^0 + \alpha \cdot (s_i^+ - s_i^-) & \forall \alpha > 0 \\
  s_i^0 + \alpha \cdot (s_i^- - s_i^+) & \forall \alpha < 0 
\end{cases} \]

\[ L(\tilde{N} | \tilde{s}, \tilde{b}) = \prod_{bins} P(N_i | \gamma_i \cdot [s_i(\alpha, s_i^-, s_i^0, s_i^+) + b_i]) \prod_{bins} P(\tilde{s}_i + \tilde{b}_i | \gamma_i \cdot [\tilde{s}_i + \tilde{b}_i]) G(0 | \alpha, 1) \]

// Import template histograms in workspace
w.import(hs_0,hs_p,hs_m,hb) ;

// Construct parametric template morphing signal model
w.factory("ParamHistFunc::s_p(hs_p)") ;
w.factory("HistFunc::s_m(x,hs_m)") ;
w.factory("HistFunc::s_0(x[80,100],hs_0)") ;
w.factory("PiecewiseInterpolation::sig(s_0,s_,m,s_p,alpha[-5,5])") ;

// Construct parametric background model (sharing gamma’s with s_p)
w.factory("ParamHistFunc::bkg(hb,s_p)") ;

// Construct full model with BB-lite MC stats modeling
w.factory("PROD::model(ASUM(sig,bkg,f[0,1]),HistConstraint({s_0,bkg}),Gaussian(0, alpha, 1))") ;
Some further considerations on morphing & modeling systematics
Non-linear interpolation options

- Piece-wise *linear* interpolation can lead to kink in response functions that may result in pathological likelihood functions.

- A variety of other polynomial interpolation options exist that mitigate such effects.

$L(\alpha > 0)$ predicts $\alpha < 0$

$L(\alpha < 0)$ predicts $\alpha > 0$

*Wouter Verkerke, NIKHEF*
Non-linear interpolation options

• Piece-wise linear interpolation can lead to kink in response functions that may result in pathological likelihood functions.

• A variety of other polynomial interpolation options exist that mitigate such effects.

Wouter Verkerke, NIKHEF
Limitations of piece-wise linear interpolation

- Bin-by-bin interpolation not well suited for distributions with large shifts in mean

Note double peak structure around \(|\alpha|=0.5\)
Other morphing algorithms - Moment morphing

- Alternative morphing algorithms aim to improve performance on templates with strongly shifting means
- Core idea: apply transformation $x \rightarrow x'$ to observable of template distribution $T(x)$ that linearly adjust moments

$$T_i(x) \rightarrow T_i(x'_i(x, \alpha))$$

where the linear functions $x'_i$ are chosen such that
- the mean and variance of each $T'_i$ are identical at a given value of $\alpha$
- the mean and variance of each $T'_i$ vary linearly between templates $i$ and $i+1$

- For the special case of interpolation between Gaussian templates with linearly shifting mean and variance moment morphing interpolation represent the exact solution
Yet another morphing strategy – ‘Moment morphing’

• But also works well on ‘difficult’ distributions

• Good computational performance
  – Calculation of moments of templates is expensive, but done only once
  – Per-event calculation very fast (just linear algebra)

• Multi-dimensional moment morphing interpolation strategies exist

There are other morphing algorithms to choose from

- Gaussian varying width
- Gaussian varying mean
- Gaussian to Uniform (this is conceptually ambiguous)

n-dimensional morphing?

✔

✗

✔

--Wouter Verkerke, NIKHEF, 42--
How many nuisance parameters for one systematic?

- Some systematic uncertainties are not captured well by one nuisance parameter.
- Written prescription often not clear on number of nuisance parameters:
- Does “the JES uncertainty is 5% for all jets” mean one NP

\[ \text{i.e. JES miscalibration is coherent for all jets} \]
\[ \rightarrow \text{You can calibrate high } p_T \text{ jets with a low } p_T \text{ jet sample} \]
How many nuisance parameters for one systematic?

- Some systematic uncertainties are not captured well by one nuisance parameter.
- Written prescription often not clear on number of nuisance parameters:
- Or does “the JES uncertainty is 5% for all jets” mean 5 NPs?

Jet Energy Scale miscalibration

\[ \alpha_{\text{JES1}}, \alpha_{\text{JES2}}, \alpha_{\text{JES3}}, \alpha_{\text{JES4}}, \alpha_{\text{JES5}} \]

5%

\[ \uparrow \]

i.e. JES miscalibration is not coherent across \( p_T \) but still has 5% uncertainty for each \( p_T \) bin
How many nuisance parameters for one systematic?

- Decision on #NPs mostly a physics/measurement issue
- Underlying measurement (‘calibration’) may introduce **correlations** in NPs → must be reflected in subsidiary measurement $L_{\text{subs}}(a_1 \ldots a_n)$
- Can also perform **eigenvalue decomposition** first → sample response in physics model in eigenvalue basis → diagonal structure of $L_{\text{subs}}$
- Can also **merge weak eigenvalues** to reduce number of NPs used in physics model

For details see:
ATL-PHYS-PUB-2015-014
Modeling theory systematic uncertainties

- Difficulties are not in the modeling procedure, but in quantifying what precisely we know

- **Difficulty 1 – What is distribution of the subsidiary measurement?**

- **Easy example** – Top cross-section uncertainty

\[ L_{full}(s, \sigma_{tt}) = \text{Poisson}(N_{SR} \mid s + \varepsilon_{tt} \cdot \sigma_{tt}) \cdot \text{Gauss}(\tilde{\sigma}_{tt} \mid \sigma_{tt}, 0.08) \]

“Cross-section Uncertainty is 8%” \( \rightarrow \) Gaussian subsidiary with 8% uncertainty?

- **Difficult example** – QCD Factorization scale uncertainty

\[ L_{full}(s, \sigma_{tt}) = \text{Poisson}(N_{SR} \mid s + b(\alpha_{FS})) \cdot F(\tilde{\alpha}_{FS} \mid \alpha_{FS}) \]

“Vary Factorization Scale by x0.5 and x2” \( \rightarrow \) Implied \( F(\alpha) \) is probably not Gaussian

So what distribution was meant?

- Wouter Verkerke, NIKHEF
Modeling theory systematic uncertainties

• Difficulty 2 – What are the *parameters* of the systematic model?

• Easy example – QCD Factorization scale uncertainty

\[ L_{\text{full}}(s, \sigma_t) = \text{Poisson}(N_{SR} \mid s + b(\alpha_{FS})) \cdot F(\bar{\alpha}_{FS} \mid \alpha_{FS}) \]

  – One parameter: the factorization scale → Clearly described and connected to the underlying theory model
  – You can ask yourself if there are additional uncertainties in the theory model (renormalization scale etc), this a valid, but distinct issue.

• Difficult example – Hadronization/Fragmentation model

  – Prescription: run different showering MC generators (e.g. HERWIG and PYTHIA) and take difference results as systematic uncertainty
  – “2 point systematic” → How do you model this in the likelihood?
  – Morphing interpolation works technically, but is uncertainty really captured by one parameter?
Diagnostics on systematic uncertainty modeling

- Given that final model is joint model of many physics measurements and many subsidiary measurements
  → How to understand which measurement constrains what NP?

- Choice of convention in NPs helps:
  → e.g. for NPs with unit Gaussian subsidiary measurement:
    if estimate from joint model is 0±1 then only L(subs) constrained NP
    if estimate is e.g. 0.5 ± 0.2 then one or more L(phys) provide stronger constraint than L(subs)  → Need to check carefully that modeling assumptions are OK.

- **NP ranking plots help diagnose**  →
Working together with workspaces
The workspace

- The workspace concept has revolutionized the way people share and combine analysis
  - **Completely** factorizes process of building and using likelihood functions
  - 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.

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

Wouter Verkerke, NIKHEF
// 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) ;
Analysis chain identical for highly complex (Higgs) models

- **Step 1** – Construct the likelihood function \( L(x|p) \)

- **Step 2** – Statistical tests on parameter of interest \( p \)

```
RooWorkspace* w=TFile::Open("L.root")->Get("w") ;
RooAbsPdf* model = w->pdf("model") ;
pdf->fitTo(data, GlobalObservables(w->set("MC_Glob")),
             Constrain(*w->st("MC_NuisParams")) ;
```
Workspaces power collaborative statistical modelling

• Ability to persist complete\(^(*)\) Likelihood models has profound implications for HEP analysis workflow
  – \(^(*)\) Describing signal regions, control regions, and including nuisance parameters for all systematic uncertainties

• Anyone with ROOT (and one ROOT file with a workspace) can re-run any entire statistical analysis out-of-the-box
  – About 5 lines of code are needed
  – Including estimate of systematic uncertainties

• Unprecedented new possibilities for cross-checking results, in-depth checks of structure of analysis
  – Trivial to run variants of analysis (what if ‘Jet Energy Scale uncertainty’ is 7% instead of 4%). Just change number and rerun.
  – But can also make structural changes a posteri. For example, rerun with assumption that JES uncertainty in forward and barrel region of detector are 100% correlated instead of being uncorrelated.
Collaborative statistical modelling

- As an experiment, you can effectively **build a library of measurements**, of which the full likelihood model is preserved for later use
  - Already done now, experiments have such repositories of workspace files

- Building of **combined likelihood models greatly simplified.**
  - Start from persisted components. No need to (re)build input components.
  - No need to know how individual components were built, or are internally structured. Just need to know meaning of parameters.
  - Combinations can be produced (much) later than original analyses.
  - Even analyses that were never originally intended to be combined with anything else can be included in joint likelihoods at a later time

Wouter Verkerke, NIKHEF
Higgs discovery strategy – add everything together

Dedicated physics working groups define search for each of the major Higgs decay channels ($H \rightarrow WW$, $H \rightarrow ZZ$, $H \rightarrow \tau\tau$ etc).

Output is physics paper or note, and a RooFit workspace with the full likelihood function.

Assume SM rates

$$L(\mu, \bar{\theta}) = L_{H \rightarrow WW}(\mu_{WW}, \bar{\theta}) \cdot L_{H \rightarrow \gamma\gamma}(\mu_{\gamma\gamma}, \bar{\theta}) \cdot L_{H \rightarrow ZZ}(\mu_{ZZ}, \bar{\theta}) \cdot ...$$

A small dedicated team of specialists builds a combined likelihood from the inputs. Major discussion point: naming of parameters, choice of parameters for systematic uncertainties (a physics issue, largely)
The benefits of modularity

- Technically very straightforward to combine measurements

RooFit, or RooFit+HistFactory

RooWorkspace

Higgs channel 1

Combiner

Higgs channel 2

Lightweight software tool using RooFit editor tools (~500 LOC)

RooWorkspace

Higgs Combination

RooStats
Workspace persistence of really complex models works too!

Atlas Higgs combination model (23,000 functions, 1600 parameters)

Model has ~23,000 function objects, ~1600 parameters
Reading/writing of full model takes ~4 seconds
ROOT file with workspace is ~6 Mb
With these combined models the Higgs discovery plots were produced…

\[ L_{\text{ATLAS}}(\mu, \theta) = \]

Neyman construction with profile likelihood ratio test
More benefits of modularity

- Technically very straightforward to reparametrize measurements

RooFit, or RooFit+HistFactory

Standard
Higgs combination

Reparametrization step does not modify workflow

BSM
Higgs combination

RooWorkspace

Reparametrize

RooWorkspace

Lightweight software tool using RooFit editor tools

RooStats

Wouter Verkerke, NIKHEF
BSM Higgs constraints from reparametrization of SM Higgs Likelihood model

Simplified MSSM \((\tan\beta, m_A)\)

Minimal composite Higgs \((\xi)\)

Imposter model \((M, \epsilon)\)

Two Higgs Double Model \((\tan\beta, \cos(\alpha-\beta))\)

Portal model \((m_X)\)

\((ATLAS-CONF-2014-010)\)
An excursion – Collaborative analyses with workspaces

• How can you reparametrize existing Higgs likelihoods in practice?

• Write functions expressions corresponding to new parameterization

\[
\sigma(gg \rightarrow H) \ast BR(H \rightarrow \gamma\gamma) \sim \frac{\kappa_F^2 \cdot \kappa_\gamma^2(\kappa_F, \kappa_\gamma)}{0.75 \cdot \kappa_F^2 + 0.25 \cdot \kappa_\gamma^2}
\]

```c
w.factory("expr::mu_gg_func('(KF2*Kg2)/
(0.75*KF2+0.25*KV2)',
KF2,Kg2,KV2) ;

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

• Import transformation in workspace, edit existing model

Wouter Verkerke, NIKHEF
Higher level model building
Higher-level model building tools

- RooFit is designed to efficiently express models at the level of mathematical formulation.

- For complex LHC data analyses, a lot of logistics needs to happen to formulate models at this level
  - $O(10)$ samples corresponding to physics processes
  - $O(10-100)$ systematic uncertainties
  - $O(1-100)$ channels

- Useful to have a higher-level language to express physics concepts into a mathematical model

- Several tools exist (with similar functionality), e.g.
  - HistFactory (developed by ATLAS, part of RooFit in ROOT distribution)
  - Combine (developed by CMS, publicly available)
Higher-level tool examples: combine & HistFitter

https://cms-hcomb.gitbook.io/combine/

http://histfitter.web.cern.ch/histfitter/
HistFactory language elements

- Hierarchy of concepts for description of one measurement channel

Beeston-Barlow-lite MC statistical uncertainties

(Theory) sample normalization

Template morphing shape systematic
Higher-level model building tools

• Use of particular high-level model building tools depends often on personal preference and/or collaboration
  – In CMS ‘combine’ tool is dominant
  – In ATLAS a variety of generic (HistFitter, HistFactory) and physics-specific high-level modeling tools is used.
  – Writing new high-level modeling frameworks is easy, either directly on top of RooFit (string-driven factory interface) or via HistFactory is easy

• All high-level modeling tools write models in the same language: RooFit workspaces.
  – All can be used, inspected in the same way
  – All can be edited *a posteriori*, (for combination building, reparametrization)
  – Facilitates combinations between experiments (e.g. ATLAS/CMS Higgs combination)
The benefits of modularity

- Perform different statistical test on exactly the same model

RooFit, or RooFit+HistFactory

RooWorkspace

“Simple fit”
(ML Fit with HESSE or MINOS)

RooStats
(Frequentist with toys)

RooStats
(Frequentist asymptotic)

RooStats
Bayesian MCMC

Wouter Verkerke, NIKHEF
Maximum Likelihood estimation as simple statistical analysis

• **Step 1** – Construct the likelihood function \( L(x|p) \)

```cpp
RooWorkspace w("w") ;
w.factory("Gaussian::sig(x[-10,10],m[0],s[1])");
w.factory("Chebychev::bkg(x,a1[-1,1])");
w.factory("SUM::model(fsig[0,1]*sig,bkg)");
w.writeToFile("L.root") ;
```

• **Step 2** – Statistical tests on parameter of interest \( p \)

```cpp
RooWorkspace* w=TFile::Open("L.root")->Get("w") ;
RooAbsPdf* model = w->pdf("model") ;
pdf->fitTo(data) ;
```

Wouter Verkerke, NIKHEF
But fundamental techniques can be complicated to execute...

- Example of confidence interval calculation with Neyman construction
  - Need to construct ‘confidence belt’ using toy MC. Intersection observed data with belt defined interval in POI with guaranteed coverage
  
  \[ x = 3.2 \]

  \[ t_\mu(x, \mu) = -2 \log \frac{L(x | \mu)}{L(x | \hat{\mu})} \]

- Expensive, complicated procedure, but completely procedural once Likelihood and parameter of interest are fixed
  
  → Can be wrapped in a tool that runs effectively ‘out-of-the-box’

Wouter Verkerke, NIKHEF
Formulating a unique defined statistics question

- A workspace is a container that can contain any number of models and datasets → No uniquely defined statistical inference task that a tool can perform
- Solution: Introduce special ModelConfig object stored in the workspace that unique defines the problem

```cpp
RooStats::ModelConfig mc("ModelConfig", &w);

// Define the pdf, the POI and observables
mc.SetPdf(*w.pdf("model"));
mc.SetParametersOfInterest(*w.var("mu"));
mc.SetObservables(*w.var("Nobs"));

// import model in the workspace
w.import(mc);
```
Offset advanced control over details of statistical procedure (use of CLS, choice of test statistic, boundaries...)

```cpp
// create first HypoTest calculator (N.B null is s+b model)
FrequentistCalculator fc(*data, *bModel, *sbModel);

// configure ToyMCSampler and set the test statistics
ToyMCSampler *toymcs = (ToyMCSampler*)fc.GetTestStatSampler();
ProfileLikelihoodTestStat profll(*sbModel->GetPdf());
// for CLs (bounded intervals) use one-sided profile likelihood
profll.SetOneSided(true);
toymcs->SetTestStatistic(&profll);

HypoTestInverter calc(*fc);
calc.UseCLs(true);

// configure and run the scan
calc.SetFixedScan(npoints,pomin,poimax);
HypoTestInverterResult * r = calc.GetInterval();

// get result and plot it
double upperLimit = r->UpperLimit();
double expectedLimit = r->GetExpectedUpperLimit(0);

HypoTestInverterPlot *plot = new HypoTestInverterPlot("hi","",r);
plot->Draw();
```
RooStats Example: Frequentist upper limit calculation

```
// create first HypoTest calculator (N.B null is s+b model)
FrequentistCalculator fc(*data, *bModel, *sbModel);

// configure ToyMCSampler and set the test statistics
ToyMCSampler *toymcs = (ToyMCSampler*)fc.GetTestStatSampler();
ProfileLikelihoodTestStat profll(*sbModel->GetPdf());
// for CLs (bounded intervals) use one-sided profile likelihood
profll.SetOneSided(true);
toymcs->SetTestStatistic(&profll);

HypoTestInverter calc(*fc);
calc.UseCLs(true);
// configure and run the scan
calc.SetFixedScan(npoints,pomin,pomax);
HypoTestInverterResult * r = calc.GetInterval();

// get result and plot it
double upperLimit = r->UpperLimit();
double expectedLimit = r->GetExpectedUpperLimit(0);
HypoTestInverterPlot *plot = new HypoTestInverterPlot("hi","",r);
plot->Draw();
```

Offset advanced control over details of statistical procedure (use of CLS, choice of test statistic, boundaries...)

$f(q_{\mu} \mid \mu')$

Tool to construct test statistic distribution

$q_{\mu}(\mu')$

The test statistic to be used for the calculation of p-values
RooStats Example: Frequentist upper limit calculation

```
// create first HypoTest calculator (N.B null is s+b model)
FrequentistCalculator fc(*data, *bModel, *sbModel);

// configure ToyMCSampler and set the test statistics
ToyMCSampler *toymcs = (ToyMCSampler*)fc.GetTestStatSampler();
ProfileLikelihoodTestStat profl1(*sbModel->GetPdf());
// for CLs (bounded intervals) use one-sided profile likelihood
profl1.SetOneSided(true);

// set test statistic
proflls->SetTestStatistic(&profl1);

HypoTestInverter calc(*fc);
calc.UseCLs(true);

// configure and run the scan
calc.SetFixedScan(npnodes, poimin, poimax);
HypoTestInverterResult *r = calc.GetInterval();

// get result and plot it
double upperLimit = r->UpperLimit();
double expectedLimit = r->GetExpectedUpperLimit(0);

HypoTestInverterPlot *plot = new HypoTestInverterPlot("hi","",r);
plot->Draw();
```

Offset advanced control over details of statistical procedure (use of CLS, choice of test statistic, boundaries...)

Tool to scan over values of \( \mu \) to find a \( q_\mu \) that results in a p-value of 0.05 (for 95% C.L.)
RooStats Example: Frequentist upper limit calculation

Offset advanced control over details of statistical procedure (use of CLS, choice of test statistic, boundaries...)

```c
// create first HypoTest calculator (N.B null is s+b model)
FrequentistCalculator fc(*data, *bModel, *sbModel);

// configure ToyMC Sampler and set the test statistics
ToyMC Sampler *toyMcs = (ToyMC Sampler*)fc.GetTestStatSampler();

ProfileLikelihoodTestStat proflli(*sbModel->GetPdf());
// for CLs (bounded intervals) use one-sided profile likelihood proflli.SetOneSided(true);
toyMcs->SetTestStatistic(&proflli);

HypoTestInverter calc(*fo);
calc.UseCLs(true);

// configure and run the scan
calc.SetFixedScan(npoin, poinmin, poinmax);
HypoTestInverterResult * r = calc.GetInterval();

// get result and plot it
double upperLimit = r->UpperLimit();
double expectedLimit = r->GetExpectedUpperLimit(0);

HypoTestInverterPlot *plot = new HypoTestInverterPlot("hi", "", r);
plot->Draw();
```

Options are
1) FC-style test stat \( q_{\mu} \)
2) CLS: calculate p-value from \( q_{\mu} \) divide by p-value of bkg hypothesis in scan for 95% point.

Optionally choose a technique to avoid spurious exclusions (all at 95% C.L. signal excluded due to low fluctuation)
RooStats Example: Frequentist upper limit calculation

Offset advanced control over details of statistical procedure (use of CLS, choice of test statistic, boundaries...)

```cpp
// create first HypoTest calculator (N.B null is s+b model)
FrequentistCalculator fc(*data, *bModel, *sbModel);

// configure ToyMCSampler and set the test statistics
ToyMCSampler *toymcs = (ToyMCSampler*)fc.GetTestStatSampler();
ProfileLikelihoodTestStat profll(*sbModel->GetPdf());
// for CLs (bounded intervals) use one-sided profile likelihood profll.SetOneSided(true);
toymcs->SetTestStatistic(&profll);

HypoTestInverter calc(*fc);
calc.UseCLs(true);

// configure and run the scan
calc.SetFixedScan(npoints,pomin,poimax);
HypoTestInverterResult * r = calc.GetInterval();

// get result and plot it
double upperLimit = r->UpperLimit();
double expectedLimit = r->GetExpectedUpperLimit(0);
HypoTestInverterPlot *plot = new HypoTestInverterPlot("hi","",r);
plot->Draw();
```

Run calculation
Extract result
Make optional plot
RooStats Example: Frequentist upper limit calculation

Offset advanced control over details of statistical procedure (use of CLS, choice of test statistic, boundaries...)

AsymptoticCalculator calculates \( p \)-values for given hypothesis \( \mu \)

\( \text{CLS ratio divides } \frac{p(s+b)}{p(b)} \)

Hypothesis inverter finds \textit{intersection} of CLS with target \( p \)-value (0.10) for 90% C.L.
Using asymptotic distributions

- Full frequentist calculation can be time consuming. If not needed, it is trivial to swap full calculator for fast asymptotic calculator that assumes known asymptotic distributions

```c
// create first HypoTest calculator (N.B. null is s+b model)
FrequentistCalculator fc(*data, *bModel, *sbModel);
```

```c
// first create HypoTest calculator (N.B. null is s+b model)
AsymptoticCalculator asympCalc(*data, *bModel, *sbModel);
```
Summary

- At LHC standardized modeling tools (RooFit) are used extensively. For LHC a lot of new functionality was developed in RooFit
  - (Parameterized) template models
  - Workspace & persistence

- Parameterized template models are used extensively at ATLAS+CMS
  - Lots of tools and diagnostic techniques were developed (morphing algorithms, ranking plots…)
  - Standard technique to represent all systematic uncertainties as NPs in models

- Great benefit from common language (workspace) in which all models are expressed
  - Easy combination & a posteriori editing
  - Preservation of exact likelihoods
  - Interoperability of models built by variety of user-developed high-level modeling packages

- RooStats calculator provide universal methods for limit and interval calculation for all RooFit-expressed models
  - Bayesian & frequentist (toys & asymptotic)