Analysis preservation with Rivet & model tuning with Professor

Holger Schulz

Computing Round Table, 6 June 2017

https://rivet.hepforge.org

https://professor.hepforge.org

University of CINCINNATI Fermilab

1. Analysis preservation

2. Event generators and Rivet

3. Model tuning with Professor

Analysis preservation

Data analysis

More often than not, physics analyses are single-use

- Measurement of a fundamental parameter
- Exclusion of one particular BSM physics model

Documentation available to “outsiders” limited

- Peer-reviewed paper typically condensed
- Experiment internal notes contain full efficiencies, subtle cuts

$\rightarrow$ it is usually very hard to reimplement analyses based on the publication alone in order to obtain new result with old data.

Analysis preservation

In an ideal world, the analysis team publishes

- all measured data
- all efficiencies, resolutions
- all systematic uncertainties with full correlations
- the logic of the event-by-event analysis

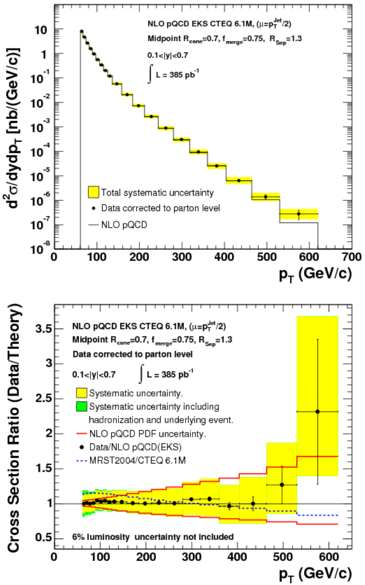
In the real world

- Experiment policy prohibits publication of certain information
- People want to move on

Rivet is a HEP community tool that aims to minimise effort and maximise additional benefit of analysis preservation.

HEPdata (**https://hepdata.net**) is a data base where all kinds of experimental data can be stored centrally.

Example



- **arXiv:hep-ex/0512020**
- 12 year old analysis of jets events with CDF
- Would like to compare the data with theory prediction
- Predictions typically from Monte-Carlo (MC) event generators
  - Validation of MC calculation
  - Test of new model/feature
  - Reinterpretation/limit setting
- Data can be read from plot/tables, better to find it in HEPdata

Event generators and Rivet

Anatomy of a hadron collider event

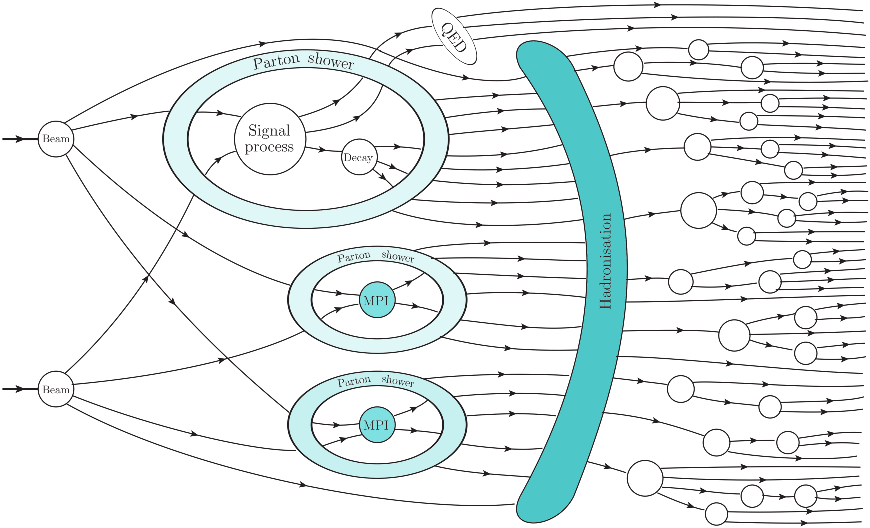


Image: Marek Schönherr

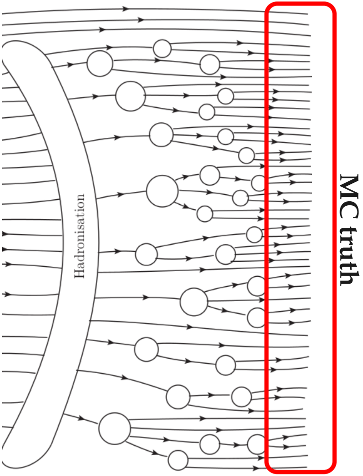
HepMC — standard event format

```
HepMC::Version 2.06.09
HepMC::IO_GenEvent-START_EVENT_LISTING
E 0 -1 -1.000000000000e+00 8.871071040596e-02 7.818608287725e-03 0 0 1 10001 10002 0 5 3.416101592648e+03 7.483422617301e-08 3.416101592648e+03 1.200000000000e+01 0
N 5 "0" "1" "2" "3" "4"
U GEV MM
C 2.846751327207e+02 2.846751327207e+02
F 2 21 4.008893662122e-01 3.729543078303e-02 9.782054163953e+02 1.532449558314e-01 2.594332904812e+00 0 0
V -1 0 0 0 0 0 2 224 0
P 10001 2212 0 0 3.999999889956e+03 4.000000000000e+03 9.382719993929e-01 2 0 0 -1 0
P 10002 2212 0 0 -3.999999889956e+03 4.000000000000e+03 9.382719993929e-01 2 0 0 -1 0
P 10003 52 2.152701458984e+02 -4.008098606740e+01 1.224843865257e+02 2.510978815160e+02 1.000000000000e+01 1 0
P 10004 -52 -1.859496611203e+02 2.683906048726e+02 2.974023320540e+02 4.417679711860e+02 1.000000000000e+01 1 0
P 10005 -211 9.890420965096e-03 -7.232191998081e-02 -4.585734642526e-01 4.848687322907e-01 1.395700000000e-01 1
P 10006 211 -4.875521999232e-02 -6.391682129595e-01 -3.244942277751e+00 3.310595603025e+00 1.395700000000e-01 1
P 10007 -211 -5.419747849145e-01 1.109603099151e+00 -5.358114740881e+00 5.500348085938e+00 1.395700000000e-01 1
P 10008 211 -2.371586367548e-01 9.913161177003e-02 -7.510349282885e-01 8.059804860224e-01 1.395700000000e-01 1
P 10009 211 1.081239345290e+00 -1.119198919259e+00 3.588261440166e-01 1.603097230116e+00 1.395700000000e-01 1 0
P 10010 -2212 -6.650888213379e-02 7.369297556192e-01 3.366876238277e+00 3.572631921424e+00 9.382720000000e-01 1
P 10011 -211 8.982907262553e+00 -1.076204480402e+01 5.447163443754e+00 1.504012302540e+01 1.395699999998e-01 1
P 10012 211 1.093319138358e+00 -1.292226110997e+00 6.619801755435e-01 1.822880302695e+00 1.395700000000e-01 1 0
P 10013 211 2.896119078340e-02 -2.408931921491e-01 -1.202631135574e+00 1.234775167287e+00 1.395700000000e-01 1
P 10014 211 -3.568764257915e-01 -2.361806061069e-01 -3.470102086736e+00 3.499175665676e+00 1.395700000000e-01 1
P 10015 2112 -2.003637330934e+01 -2.382464016977e+01 1.759645813122e+02 1.786994303160e+02 9.395659999907e-01 1
P 10016 22 8.357442917026e-01 -2.581824372676e+00 1.269227052127e+01 1.297913774476e+01 -2.384185791016e-07 1 0
...
```

- Particle 4 vectors
- Vertices and genealogy
- Common format for most event generators
- Typically MB per event

Rivet

- Analysis tool for MC events, generator agnostic via HepMC
- Provides most relevant methods for multi particle final states:
  - Cuts
  - Jets
  - Boson finders
  - Event shapes
  - DIS kinematics
  - …
- Writes out histogram (YODA format)
- External infrastructure: HepData, inspire
- Easy to write new analyses for signal and background estimates from MC
- Implementation and validation of new (data) analyses now largely provided by (LHC) experiments



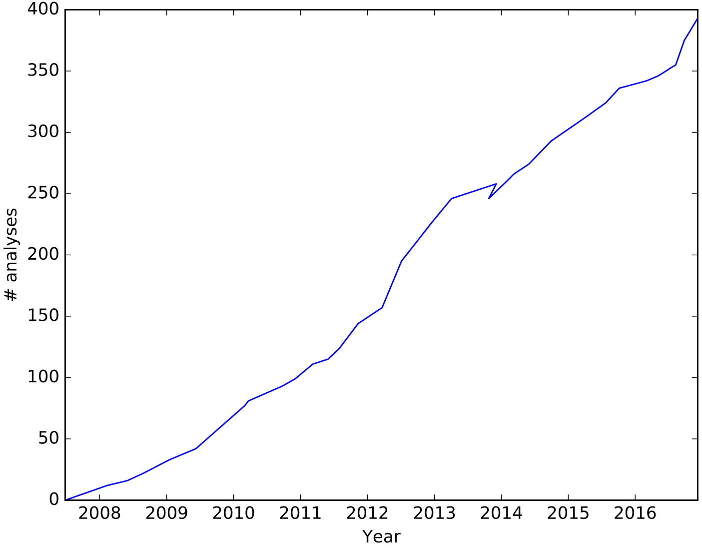
Rivet

Rivet is an analysis system for MC events, and *lots* of analyses

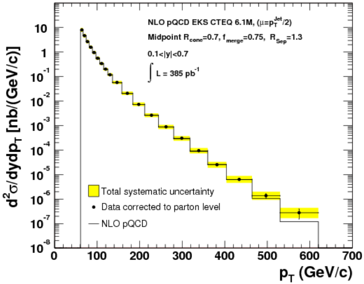
494 built-in, at today's count! 54 are pure MC, and some double/triple-counting

- *Generator-agnostic* for physics & pragmatics
- A quick, easy and powerful way to get physics plots from lots of MC gens
  - Only requirement: use **HepMC** event record
  - Usually via ASCII, but in-memory exchange is faster
- Rivet has become the LHC standard for archiving LHC data analyses
  - Focus on *unfolded* measurements, esp. QCD and EW+QCD, rather than searches
  - But there are BSM studies using it! **And detector simulation now possible**
  - Key input to MC validation and tuning – increasingly comprehensive coverage
  - Also "recasting" of SM and BSM data results on to new/more general BSM model spaces
  - **Add your analyses, too!**

Rivet is community driven



Analysis naming scheme



- **`CDF_2006_S6450792`**
- **`EXPERIMENT_YEAR_INSPIREORSPIRESKEY`**
- **http://inspire-hep.net/record/699933**
- This particular histogram is **`CDF_2006_S6450792/d01-x01-y01`**
- **http://hepdata.cedar.ac.uk/view/ins699933**, table 1, row 1

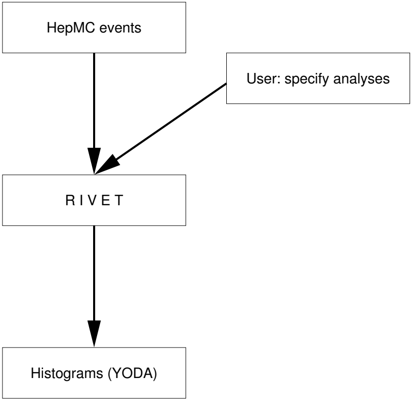
Design philosophy / pragmatics

Rivet operates on HepMC events, intentionally unaware of who made them…so don't "look inside" the event graph.

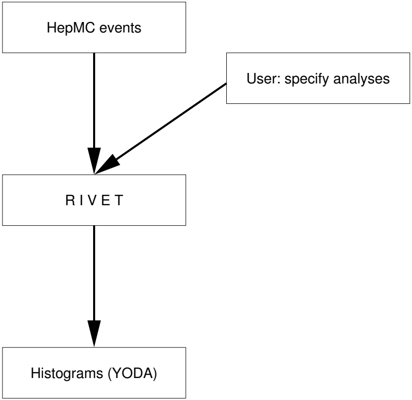
- C++ library with Python interface & scripts
- "Plugins" ⇒ write your analyses without needing to rebuild Rivet
  Trivial from user / analysis author point of view
- Tools to make "doing things properly" easy and default
- Computation caching for efficiency
- Histogram syncing: *keep code clean and clear*

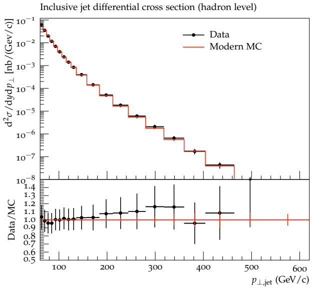
+ helpful developers! New contributors always welcome

Basic principle



Basic principle





Rivet comes with its own plotting tool which is geared towards data-MC comparison.

Making this plot took half an hour.

Output example

```
BEGIN YODA_HISTO1D /CDF_2006_S6450792/d01-x01-y01
IsRef=1
Path=/CDF_2006_S6450792/d01-x01-y01
ScaledBy=4.07524008176626886e-11
Title=
Type=Histo1D
XLabel=
YLabel=
# Mean: 7.582350e+01
# Area: 1.066413e+02
# ID    ID     sumw    sumw2   sumwx   sumwx2  numEntries
Total       Total       1.066413e+02    1.080097e+00    8.085917e+03    6.429205e+05    10561
Underflow   Underflow   0.000000e+00    0.000000e+00    0.000000e+00    0.000000e+00    0
Overflow    Overflow    0.000000e+00    0.000000e+00    0.000000e+00    0.000000e+00    0
# xlow  xhigh  sumw   sumw2  sumwx  sumwx2  numEntries
6.100000e+01    6.700000e+01    3.745851e+01    3.792265e-01    2.387847e+03    1.523267e+05    3708
6.700000e+01    7.400000e+01    2.702184e+01    2.736801e-01    1.896080e+03    1.331523e+05    2676
7.400000e+01    8.100000e+01    1.617063e+01    1.635330e-01    1.246894e+03    9.620820e+04    1599
8.100000e+01    8.900000e+01    1.023432e+01    1.041130e-01    8.658179e+02    7.330429e+04    1018
8.900000e+01    9.700000e+01    6.007102e+00    6.074961e-02    5.564960e+02    5.158416e+04    594
9.700000e+01    1.060000e+02    3.964283e+00    4.009065e-02    4.007805e+02    4.054641e+04    392
1.060000e+02    1.150000e+02    2.154062e+00    2.198850e-02    2.371195e+02    2.611565e+04    215
1.150000e+02    1.250000e+02    1.537171e+00    1.554536e-02    1.837150e+02    2.196950e+04    152
1.250000e+02    1.360000e+02    8.393761e-01    8.488582e-03    1.094036e+02    1.426854e+04    83
1.360000e+02    1.580000e+02    8.090373e-01    8.181766e-03    1.171344e+02    1.698694e+04    80
1.580000e+02    1.840000e+02    2.427112e-01    2.454530e-03    4.090288e+01    6.910398e+03    24
1.840000e+02    2.120000e+02    9.101669e-02    9.204487e-04    1.792065e+01    3.532616e+03    9
2.120000e+02    2.440000e+02    7.079076e-02    7.159045e-04    1.560661e+01    3.443141e+03    7
2.440000e+02    2.800000e+02    4.045186e-02    4.090883e-04    1.019828e+01    2.571608e+03    4
END YODA_HISTO1D
```

Can also be converted to ROOT

Getting Rivet

Easy to install using our *bootstrap script*:

```
wget http://rivet.hepforge.org/hg/bootstrap/raw-file/2.6.0/rivet-bootstrap
bash rivet-bootstrap
```

Latest version is 2.6.0 Requires C++11

Getting Rivet

- `rivet` command line tool to query available analyses
- Can be used as a library (e.g. in big experiment software frameworks)
- Can also be used from the command line to read HepMC ASCII files/pipes: very convenient
- Helper scripts like `rivet-mkanalysis`, `rivet-buildplugin`
- Histogram comparisons, plot web albums, etc. very easy

Docs online at **http://rivet.hepforge.org** – PDF manual, HTML list of existing analyses, and Doxygen. Entries in HEPdata point to existing rivet analyses.

Writing an analysis

Writing an analysis is of course more involved. But the C++ interface is pretty friendly: most analyses are short, simple, and readable – details handled in the library + expressive API functions.

A single C++ file is sufficient. Rivet comes with scripts that generate analysis templates and compile the new code into a shared library (plugin).

Mostly "normal":

- Typical init/exec/fin structure
- Histogram titles, labels, etc.: use `.plot` file
- Rivet's own Particle, Jet and FourMomentum classes: some nice things like `abseta()` and `abspid()`, sorting and filtering
- Use of *projections* for computations, with a bit of magic – this is where the caching happens
- Projections are *declared* with a string name, and later are *applied* using the same name
- Final state projections are central: compute from final state or physical decayed particles

Analysis example

```
void init() {
  FinalState fs;
  declare(FastJets(fs, FastJets::CDFMIDPOINT, 0.7), "ConeFinder");
  _h_jet_pt = bookHisto1D(1, 1, 1);
}


void analyze(const Event& event) {
  const Jets& jets = apply<JetAlg>(event, "ConeFinder").jets(Cuts::pT > 61*GeV);
  foreach (const Jet& jet, jets) {
    if (inRange(jet.absrap(), 0.1, 0.7))
      _h_jet_pt->fill(jet.pT()/GeV, event.weight());
  }
}


void finalize() {
  const double delta_y = 1.2;
  scale(_h_jet_pt, crossSection()/nanobarn/sumOfWeights()/delta_y);
}
```

Rivet and DIS

Rivet is the successor of a similar tool used at HERA, “HZtool”
Unfortunately only a handful of data analyses made it from HZtool to Rivet.

The histogram system is currently limited to 2D data sets. Fully differential DIS distributions are currently a bit of a pain.

Within Rivet there is a clear plan to extend all analysis objects to N-dimensions.

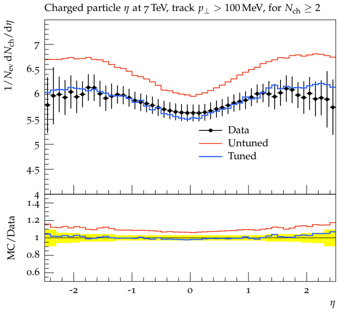
The Rivet team welcomes DIS involvement. Manpower and experimental insight are needed.

There is money to bring PhD students to Glasgow, UK for 3 months to code this extension under supervision of Andy Buckley
http://www.montecarlonet.org — “Short-term studentships”

Model tuning with Professor

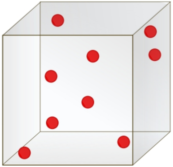
Tuning

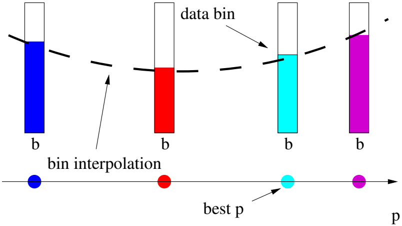
- Realistic events contain physics at low scales where perturbation breaks down (hadronisation, particle decays)
- Rely on model assumptions that introduce many parameters
- Need to find "meaningful" settings for best physics prediction $\rightarrow$ better measurements
- Can be done manually but hard to on reasonable time-scale



Tuning with Professor in a nutshell

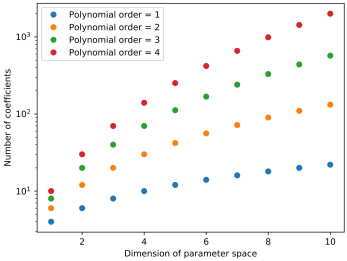
- Sampling: N parameter points in n-dimensional model space
- Run generator and fill histograms (e.g. Rivet) trivial parallel
  $\rightarrow N$ slightly different physics predictions
- For each bin:
  - Multivariate approximation (polynomial, Pade), $I_b(\vec{p})$
- Construct overall (now trivial) $\chi^2(\vec{p}) \approx \sum_{bins} \frac{(D_b - I_b(\vec{p}))^2}{error^2}$
- and numerically *minimise* with **`iminuit`**

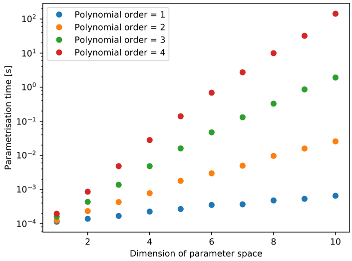




Professor technicalities

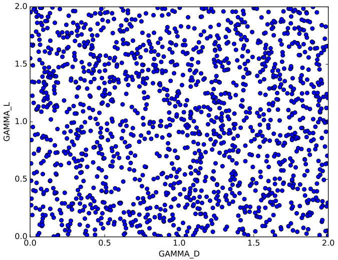
- C++ core functionality, python bindings for everything else
- In case of MC, input generation trivial to do in parallel (different points in parameter space)
- Result is fast analytic pseudo-generator
- Storage of polynomials as plain text file → can use parameterisation in other C++ codes

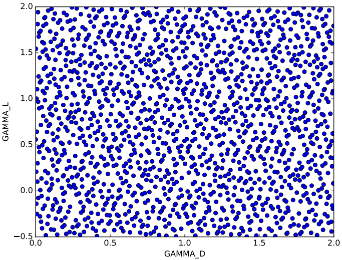




Sampling

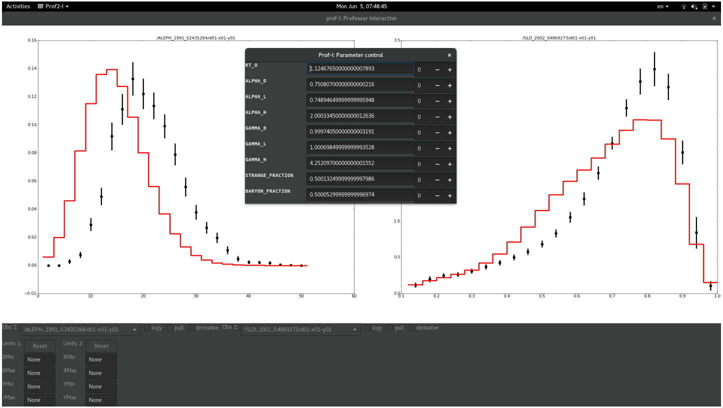
- Define parameter space with input textfile
- Can bias sampling to avoid say unphysical parameter space
- Convenient template instantiation for e.g. generator steering cards
- Random uniform, Sobol, or latin hypercube sampling





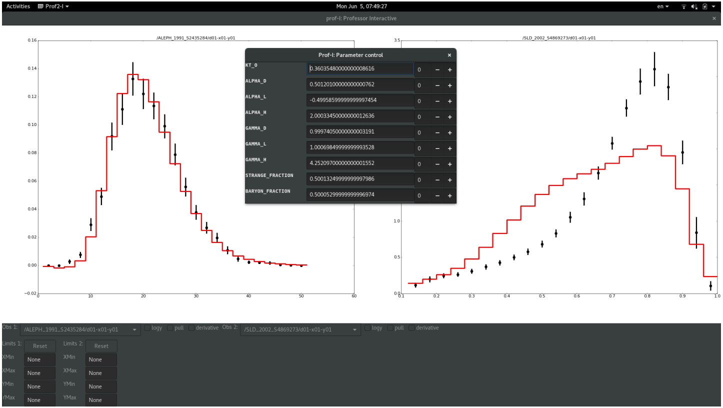
Interactive explorer

- GTK application to interactively “play” with parameterisation
- One slider per parameter, moving them redraws histograms
- Good for intuition building
- Running the event generator would require a few hours wait



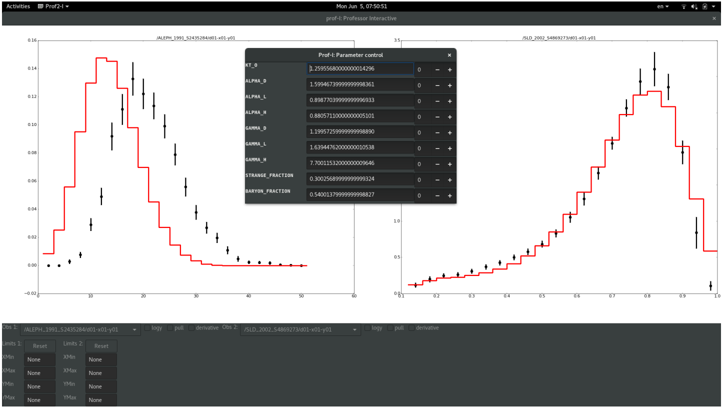
Interactive explorer

- GTK application to interactively “play” with parameterisation
- One slider per parameter, moving them redraws histograms
- Good for intuition building
- Running the event generator would require a few hours wait



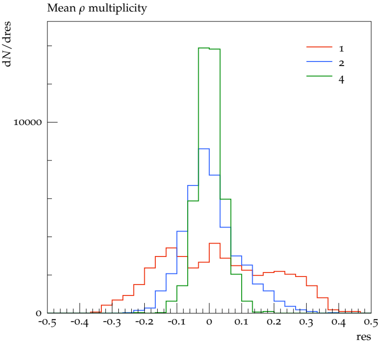
Interactive explorer

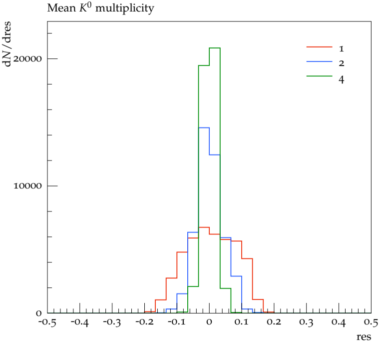
- GTK application to interactively “play” with parameterisation
- One slider per parameter, moving them redraws histograms
- Good for intuition building
- Running the event generator would require a few hours wait



Residuals

- Split input data into training ($\rightarrow I(\vec{p})$) and test ($\rightarrow \mathrm{MC}(\vec{p})$) sample.
- Define residual as distance between the two:
  $\mathrm{res} = \left[I(\vec{p}) - \mathrm{MC}(\vec{p})\right] / I(\vec{p})$
- Put all res into histogram, expect something symmetry around 0
- Helps to gain confidence in approximation.



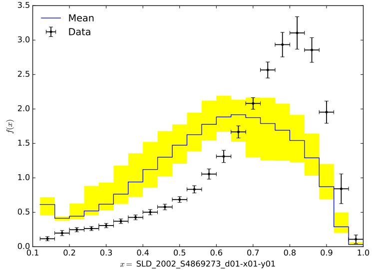


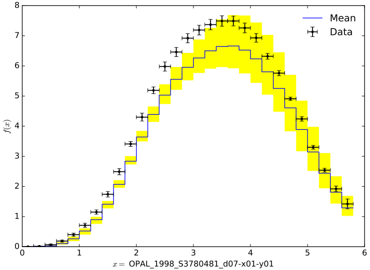
Weights and choice of observables

- Tuning: reasonable measure between model prediction and data
- By default, ad-hoc "chi square" inspired goodness-of-fit

$$\chi^2(\vec{p}) = \sum_{\mathcal{O}} \sum_{b \in \mathcal{O}} w_b \cdot \frac{(f^{(b)}(\vec{p}) - \mathcal{R}_b)^2}{\Delta_b^2(\vec{p})}$$

- Weights w_b neccessary because models not perfect:
  - Exclude regions of observables (e.g. bad coverage, breakdown of polynomial approximation)
  - Force good description of certain observables (at the cost of others)





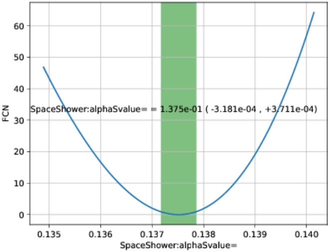
Minimisation

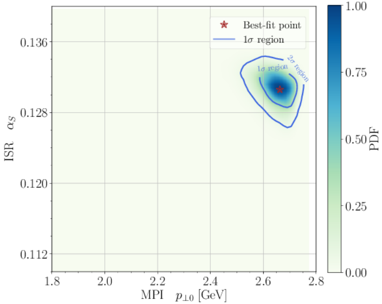
- Tuning is numerical minimisation of goodness-of-fit measure
- We use **`iminuit`** as it is a flexible python wrapper for Minuit
- Input to tuning stage is parameterisation file, text file with weights, directory with data files

```
/ATLAS_2010_S8918562/d03-x01-y01 1            # Set weight to 1   for  each bin of this histo
/ATLAS_2010_S8918562/d05-x01-y01 100          # Set weight to 100 for  each bin of this histo
/ATLAS_2010_S8918562/d07-x01-y01@0:20  10     # Set weight to  10 for  bins with binEDGES in [0,20)
/ATLAS_2010_S8918562/d07-x01-y01@20:40 50     # Set weight to  10 for  bins with binEDGES in [20,50)
/TOTEM_2012_I1115294/d01-x01-y01#0:20  10     # Set weight to  10 for  bins with binINCIDES in [0,20)
/TOTEM_2012_I1115294/d01-x01-y01#20:40 50     # Set weight to  10 for  bins with binINDICES in [20,50)
```

- Output:
  - Text file with minimisation result, covariance matrix etc.
  - File with histograms calculated from parameterisation at this minimum
- Quick turnaround, minimisation seconds to minutes, plots comparing with data seconds
- Usually iterative procedure, look at plots, adjust weights

Examples





Left: profile of model parameter with Minuit and Professor.

Right: likelihood evaluation with Multinest and Professor.

Getting Professor

- Prerequisits: Eigen3 headers, C++ 11 compiler, Python 2.7
- **professor.hepforge.org**

Summary

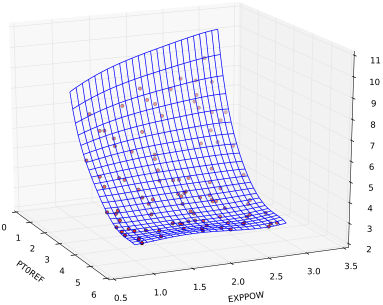
- **Rivet is a user-friendly MC analysis system for prototyping and preserving data analyses**
- Allows theorists to use analyses for model development & testing, and BSM recasting: impact beyond "get a paper out"
- Also a very useful cross-check: quite a few ATLAS analysis bugs have been found via Rivet!
- Well established in LHC community (experiment and theory)
- DIS involvement very welcome!

- Professor:
  - Parametrisation of computationally expensive functions
  - Seamless integration into numerical tools **`iminuit`**, **`pymultinest`** through python bindings $\rightarrow$ tuning and BSM applications

Fitting model

1 bin example, 2 parameters (x,y), 2nd order polynomial

$$\mathrm{MC}_b(\vec{p}) \approx \alpha_0^{(b)} + \sum \beta_i^{(b)}\, p'_i + \sum_{i \le j} \gamma_{ij}^{(b)}\, p'_i\, p'_j$$



$$c^{(\vec{b})} = (\alpha, \beta_x, \beta_y, \gamma_{xx}, \gamma_{xy}, \gamma_{yy})$$

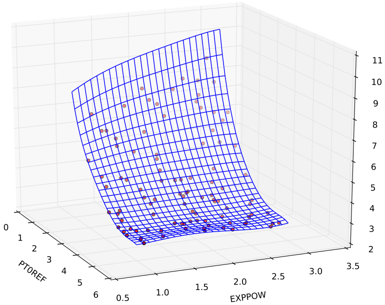
$$\tilde{p}_i = (1, x_i, y_i, x_i^2, x_i y_i, y_i^2)$$

$$\mathrm{MC}_b(\vec{p}) \approx \sum_{i=1}^{N_{\min}(P)} c_i^{(b)}\, \tilde{p}_i$$

Fitting model

1 bin example, 2 parameters (x,y), 2[nd] order polynomial

$$\mathrm{MC}_b(\vec{p}) \approx \alpha_0^{(b)} + \sum \beta_i^{(b)} p'_i + \sum_{i \le j} \gamma_{ij}^{(b)} p'_i p'_j$$



$$c^{(\vec{b})} = (\alpha, \beta_x, \beta_y, \gamma_{xx}, \gamma_{xy}, \gamma_{yy})$$

$$\tilde{p}_i = (1, x_i, y_i, x_i^2, x_i y_i, y_i^2)$$

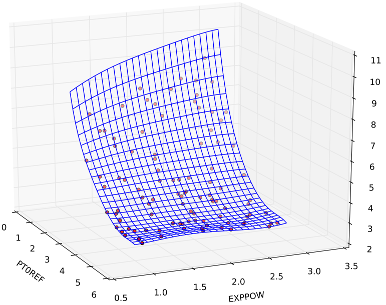
$$\mathrm{MC}_b(\vec{p}) \approx \sum_{i=1}^{N_{\min}(P)} c_i^{(b)} \tilde{p}_i$$

$$\underbrace{\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{pmatrix}}_{\vec{\mathrm{MC}}_b} = \underbrace{\begin{pmatrix} 1 & x_1 & y_1 & x_1^2 & x_1 y_1 & y_1^2 \\ 1 & x_2 & y_2 & x_2^2 & x_2 y_2 & y_2^2 \\ & & & \vdots & & \\ 1 & x_N & y_N & x_N^2 & x_N y_N & y_N^2 \end{pmatrix}}_{\tilde{P} = \{\tilde{p}_i\}} \underbrace{\begin{pmatrix} \alpha_0 \\ \beta_x \\ \beta_y \\ \gamma_{xx} \\ \gamma_{xy} \\ \gamma_{yy} \end{pmatrix}}_{\vec{c}^{(b)}}$$

Fitting model

1 bin example, 2 parameters (x,y), 2^{nd} order polynomial

$$\mathrm{MC}_b(\vec{p}) \approx \alpha_0^{(b)} + \sum \beta_i^{(b)} p'_i + \sum_{i\leq j} \gamma_{ij}^{(b)} p'_i p'_j$$



$$c^{(\vec{b})} = (\alpha, \beta_x, \beta_y, \gamma_{xx}, \gamma_{xy}, \gamma_{yy})$$

$$\tilde{p}_i = (1, x_i, y_i, x_i^2, x_i y_i, y_i^2)$$

$$\mathrm{MC}_b(\vec{p}) \approx \sum_{i=1}^{N_{\min}(P)} c_i^{(b)} \tilde{p}_i$$

$$\underbrace{\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{pmatrix}}_{\vec{\mathrm{MC}}_b} = \underbrace{\begin{pmatrix} 1 & x_1 & y_1 & x_1^2 & x_1 y_1 & y_1^2 \\ 1 & x_2 & y_2 & x_2^2 & x_2 y_2 & y_2^2 \\ & & & \vdots & & \\ 1 & x_N & y_N & x_N^2 & x_N y_N & y_N^2 \end{pmatrix}}_{\tilde{P}=\{\tilde{p}_i\}} \underbrace{\begin{pmatrix} \alpha_0 \\ \beta_x \\ \beta_y \\ \gamma_{xx} \\ \gamma_{xy} \\ \gamma_{yy} \end{pmatrix}}_{\vec{c}^{(b)}}$$

$$c^{(\vec{b})} = \mathcal{I}[\tilde{P}]\vec{\mathrm{MC}}_b.$$

Fitting model

- $\mathcal{I}[\tilde{P}]$ is the pseudo-inverse of $\tilde{P}$
- $\mathcal{I}[\tilde{P}]$ is calculated using singular value decomposition (SVD)
- SVD is least-squares fit
- We need at least as many $\tilde{p}_i$ as there are coefficients

$$\underbrace{\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{pmatrix}}_{\vec{\mathrm{MC}}_b} = \underbrace{\begin{pmatrix} 1 & x_1 & y_1 & x_1^2 & x_1 y_1 & y_1^2 \\ 1 & x_2 & y_2 & x_2^2 & x_2 y_2 & y_2^2 \\ & & & \vdots & & \\ 1 & x_N & y_N & x_N^2 & x_N y_N & y_N^2 \end{pmatrix}}_{\tilde{P}=\{\tilde{p}_i\}} \underbrace{\begin{pmatrix} \alpha_0 \\ \beta_x \\ \beta_y \\ \gamma_{xx} \\ \gamma_{xy} \\ \gamma_{yy} \end{pmatrix}}_{\vec{c}^{(b)}} \qquad c^{(\vec{b})} = \mathcal{I}[\tilde{P}]\vec{\mathrm{MC}}_b$$

- With $c_i^{(b)}$ calculated $\rightarrow$ prediction

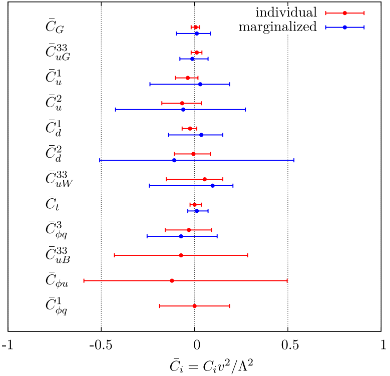
$$\mathrm{MC}_b(\vec{p}) \approx \sum_{i=1}^{N_{\min}(P)} c_i^{(b)}\, \tilde{p}_i$$

  for any $\vec{p}$ in *milliseconds*
- Separate polynomials for central value and uncertainty of a bin

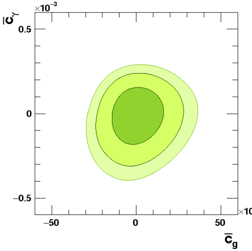
Professor beyond tuning

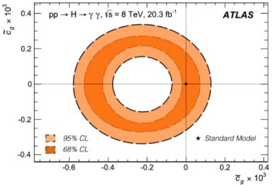
- Instead of fiddling with say hadronisation model parameters, explore BSM parameter space
- Lots of experience can be transferred from tuning to BSM

TopFitter



HEFT





Professor beyond collider physics

- Dark Matter direct detection codes: Professor in likelihood evaluation (MultiNest)