Rivet/Professor

Holger Schulz
IPPP Durham

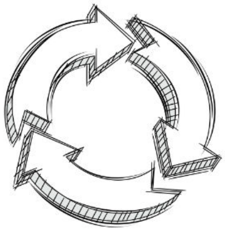
SLAC, 12 May 2017

rivet.hepforge.org

professor.hepforge.org

Introduction

- **Recent big changes in LHC experiment/theory interaction**
  - ⇒ more direct collaboration to improve methods and modelling, starting from SM & QCD, now also Top, Higgs, and BSM
- **Rivet** analysis library is part of this: a lightweight way to exchanging analysis details and ideas
- Implementing a Rivet analysis to complement the data analysis is increasingly expected of LHC analyses. **Everyone benefits!**



Introduction

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

427 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!**

Design philosophy / pragmatics

Rivet operates on HepMC events, intentionally unaware of who made them…so don't "look inside" the event graph.
⇒ reconstruct resonances, dress leptons, avoid partons, etc.

cf. q/g jet discrimination: LO picture is an implementation-dependent cartoon; a useful motivator but incomplete and ill-defined until after hadronization

This "hard work" way is actually simpler – fewer gotchas.
Makes you think about physics & helps find analysis bugs/ambiguities

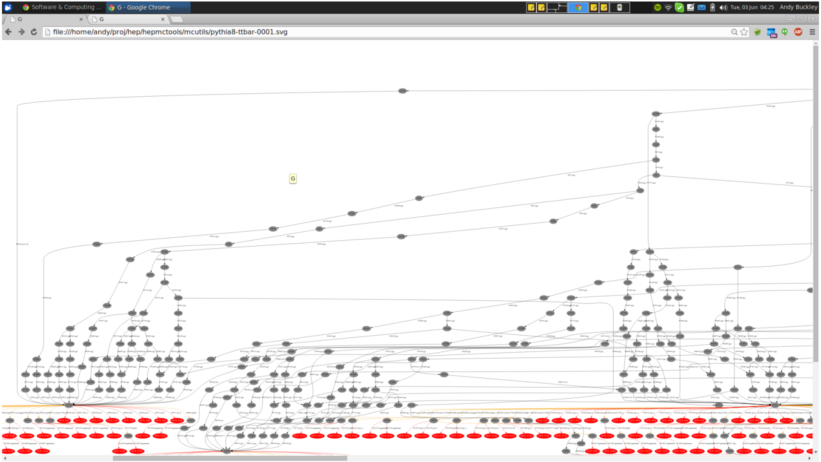
Tech stuff:

- 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

Why wouldn't we want to look at the event graph?!

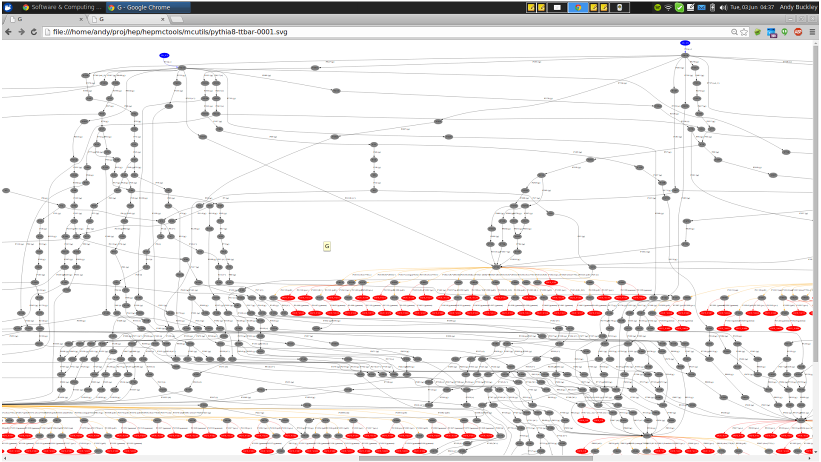
A Pythia8 $t\bar{t}$ event!



Most of this is not standardised: Herwig and Sherpa look *very* different.
But final states and decay chains have to have equivalent meaning.

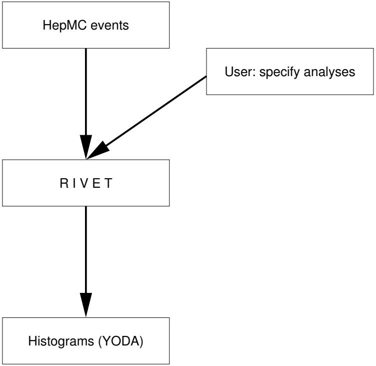
Why wouldn't we want to look at the event graph?!

A Pythia8 $t\bar{t}$ event!

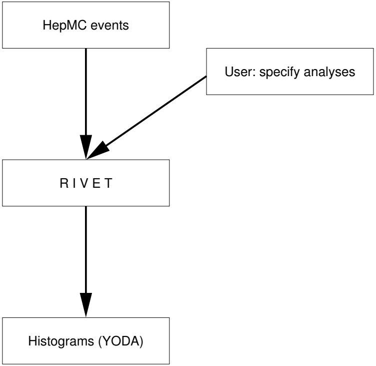


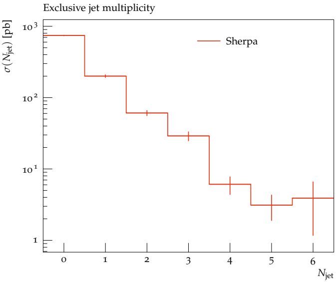
Most of this is not standardised: Herwig and Sherpa look *very* different.
But final states and decay chains have to have equivalent meaning.

Basic principle

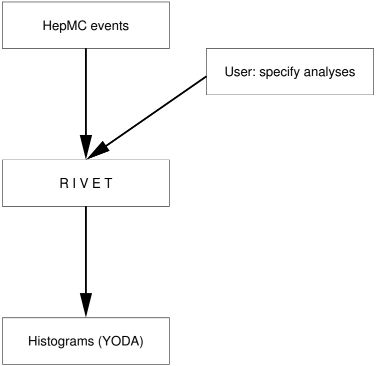


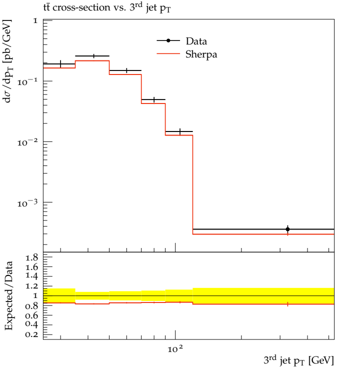
Basic principle





Basic principle





Getting Rivet

Rivet is readily available on the MC4BSM virtual machine.
Some event files are available in `tutorial/Rivet/Events`

Easy to install using our *bootstrap script*:

```
wget http://rivet.hepforge.org/hg/bootstrap/raw-file/2.5.3/rivet-bootstrap
bash rivet-bootstrap
```

Latest version is 2.5.3 Requires C++11

Docker image available:

```
docker pull hepstore/rivet:2.5.3
```

http://rivet.hepforge.org/trac/wiki/Docker

`CVMFS` installations on lxplus.

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

Projections

Major idea: **projections**. They are just observable calculators: given an `Event` object, they *project* out physical observables.

They also automatically cache themselves, to avoid recomputation. This leads to slightly unfamiliar calling code.

Projections were *declared* with a name in `init()` they are then *applied* to the current event in `analyze()`, by the same name.

E.g.

- Final states (Identified, Charged, Visible, ...)
- Jets (All native FastJet algorithms)
- Event shapes (Thrust etc.)
- Missing momentum and DIS kinematics

Selection cuts

Combinable `Cut` objects:

- `FinalState(Cuts::pT > 0.5*GeV && Cuts::abseta < 2.5)`
- `fs.particles(Cuts::absrap < 3 || (Cuts::absrap > 3.2 && Cuts::absrap < 5), cmpMomByEta)`

Can also use cuts on PID and charge:

- `fs.particlesByPt(Cuts::abspid == PID::ELECTRON)`, or
- `FinalState(Cuts::charge != 0)`

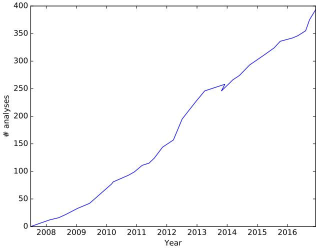
Use of functions/functors for ParticleFinder filtering is coming…

Rivet + fast-sim for BSM searches

BSM analysis coverage

Currently $\sim$ 427 analyses total & $\sim$ 230 LHC alone

- Until recently only 27 dedicated BSM searches – and BSM-sensitive SM measurements , cf. CONTUR talk
- SM focus on unfolded observables, not sufficient for most BSM studies
- Rivet 2.5.0 introduced detector smearing machinery. *For BSM only!*



NB. glitch is Rivet 1.x → 2.x migration.
Note recent acceleration!

- ⇒ 9 more BSM routines in last few months:
  - ATLAS: ICHEP 2016 3-lepton & same-sign 2-lepton, 1-lepton + jets, 1-lepton + many jets, jets + MET; 2015 jets + MET and monojet
  - CMS: ICHEP 2016 jets + MET; 8 TeV α_T + b-jets
  - *Partially* validated – not many cutflows available!
  - Also added tools to help with object filtering, cutflows, etc.
  - Important as real-world examples of how to write BSM routines
- **Rivet is in good shape for preserving new physics searches!**

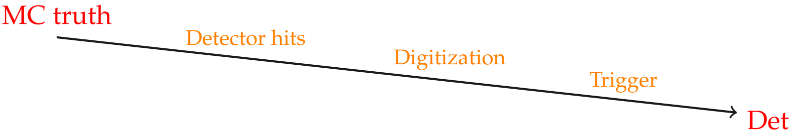
BSM & detector effects

Explicit fast detector simulation vs. smearing/efficiencies

MC truth

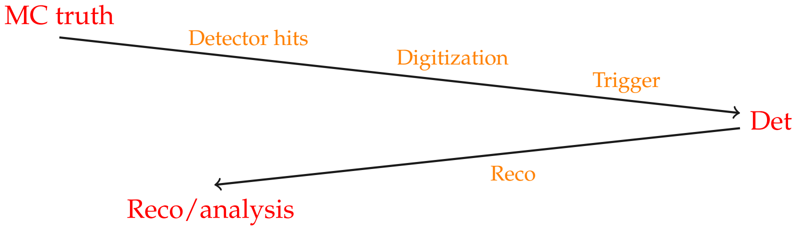
BSM & detector effects

Explicit fast detector simulation vs. smearing/efficiencies



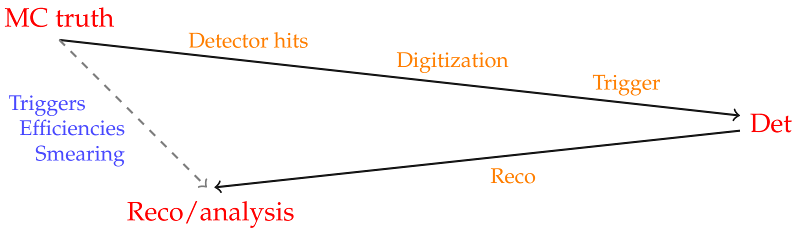
BSM & detector effects

Explicit fast detector simulation vs. smearing/efficiencies



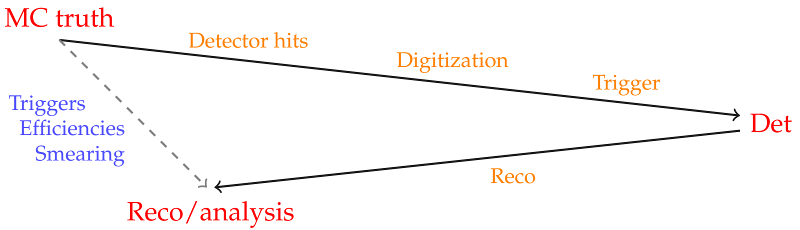
BSM & detector effects

Explicit fast detector simulation vs. smearing/efficiencies



BSM & detector effects

Explicit fast detector simulation vs. smearing/efficiencies

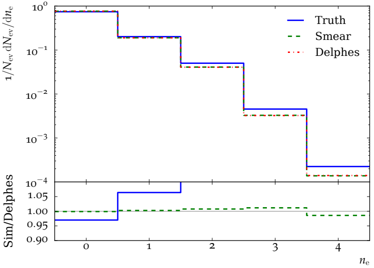


- **Explicit fast-sim takes the "long way round".**
- **Reco already reverses most detector effects!**
- Reco calibration to MC truth: smearing is a few-percent effect
- (Lepton) efficiency & mis-ID functions dominate – and are tabulated in both approaches
- Smearing is more flexible: effs change with phase-space, reco version, run, ... and need to guarantee *stability* for preservation

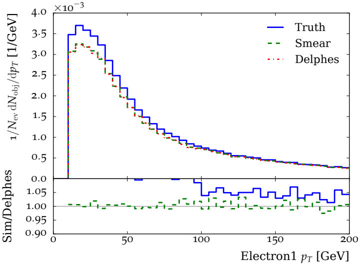
Smearing vs. fast sim vs. MC truth

CMSSM eff/smearing effects from Rivet, in turn using some DELPHES and paper/note calibration functions:

Electron multiplicity



Leading electron p_T

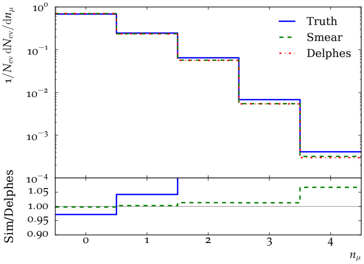


Note major lepton shifts from blue truth to green smeared: difference w.r.t red DELPHES very small

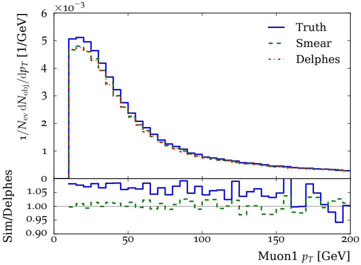
Smearing vs. fast sim vs. MC truth

CMSSM eff/smearing effects from Rivet, in turn using some DELPHES and paper/note calibration functions:

Muon multiplicity



Leading muon p_T



Note major lepton shifts from blue truth to green smeared: difference w.r.t red DELPHES very small

BSM & detector effects (II) ⇒ Rivet 2.5

In addition to last slides, *flexibility* of det-sim is important:

- "Global" fast-sims hence difficult for coverage of multiple experiments, multiple runs, multiple reco calibrations, etc.
- Analysis-specific efficiencies and smearings are more precise and allow use of multiple jet sizes, tagger & ID working points, isolations, … ⇒ **many variations in real analyses**

⇒ Rivet det-sim as effs+smearing, localised per-analysis

Rivet internally caches results, so global effect sim still efficient

- Functions for generic ATLAS & CMS performance in Runs 1 & 2
- Inline or analysis-specific functions easy to write & *chain*
- Eff/smearing functions can be used directly, e.g. for object filtering
- Working on embeddability for multithreaded fitters/samplers.

Selection tools for search analyses

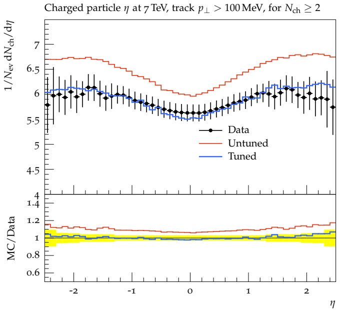
Search analyses typically do a lot more "object filtering" than measurements. Rivet 2.5 provides a lot of tools to make this complex logic expressive:

- Filtering functions: **`filter_select`**, **`filter_discard`** + **`ifilter_*`** in-place variants
- Lots of *functors* for common "stateful" filtering criteria: **`PtGtr(10*GeV), EtaLess(5), AbsEtaGtr(2.5), DeltaRGtr(mom, 0.4)`**
- Cut-flow monitor via **`#include "Rivet/Tools/Cutflow.hh"`**

```
Particles elecs = apply<ParticleFinder>(event, "Electrons").particles(Cuts::pT > 10*GeV);
Jets jets = apply<JetAlg>(event, "Jets").jetsByPt(Cuts::pT > 20*GeV && Cuts::abseta < 2.8);
// Remove electrons within dR = 0.2 of a b-tagged jet
for (const Jet& j : jets)
  if (j.abseta() < 2.5 && j.pT() > 50*GeV && j.bTagged(Cuts::pT > 5*GeV))
    ifilter_discard(elecs, deltaRLess(j, 0.2, RAPIDITY));
// Remove any |eta| < 2.8 jet within dR = 0.2 of a remaining electron
for (const Particle& e : elecs)
  ifilter_discard(jets, deltaRLess(e, 0.2, RAPIDITY));
```

Professor

- Established tool for MC generator tuning, heavily taylored to Rivet
- Replace MC generator response with polynomials in χ^2 minimisation using **`Minuit`**
- $\chi^2(\vec{p}) = \sum_{b}^{\text{Nbins}} w_b \cdot \left(\frac{I_b(\vec{p}) - D_b}{\Delta(\vec{p})} \right)^2$

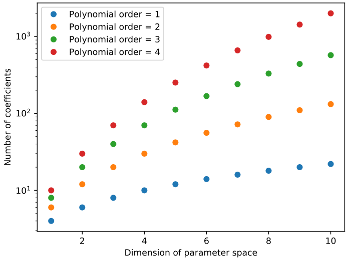


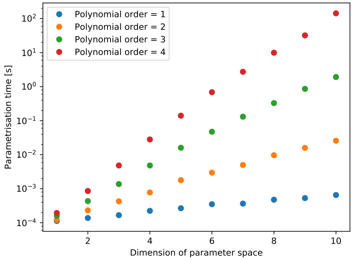
Getting Professor

- Prerequisits: Eigen3 headers, C++ 11 compiler, Python 2.7
- professor.hepforge.org
- Docker image: `docker pull iamholger/professor:2.2.1`
- Try out at http://mybinder.org/repo/iamholger/professor

Professor technicalities

- C++ core functionality, python bindings for everything else
- Least-squares fits of general polynomials to input data in a certain parameter space
- Technically, solving of matrix equation by means of Singular Value Decomposition
- In case of MC, input generation trivial to do in parallel (different points in parameter space)
- Result is fast analytic pseudo-generator

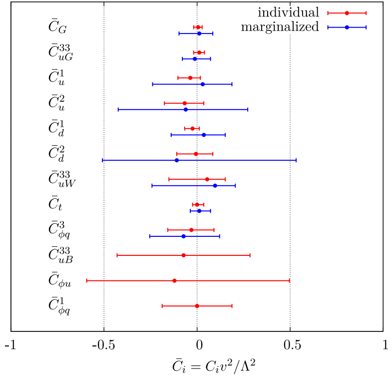




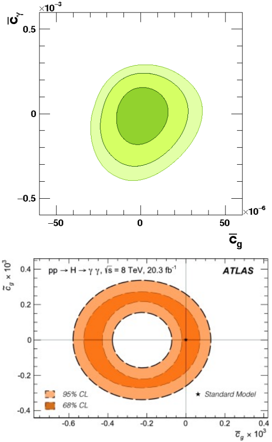
Professor beyond tuning

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

TopFitter

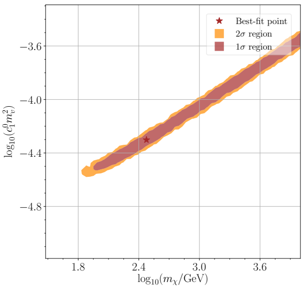


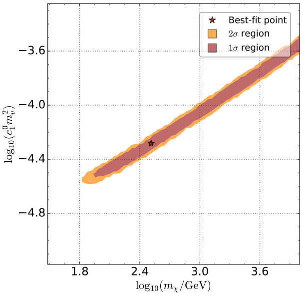
HEFT



Professor beyond collider physics

- Recently got foothold in neutrino MC community, **`Genie`**
- Triggered containerisation of Professor with Docker
- Similarly, Dark Matter direct detection codes: Professor in likelihood evaluation (MultiNest)





BSM challenges for Professor

- More careful checks of validity of polynomial approximation.
  - Partitioning of parameter space in case parameter space too big
  - Need to allow to drop inputs in case of vanishing cross-sections (avoid discontinuities in polynomial fit)
  - Resolutions, jack-knifing
- Usage of other parameterisations, e.g. Gaussian Processes for $\frac{1}{x}$, exponential behaviour

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!
- Strongly encouraged/required by ATLAS (and CMS?) physics groups. Integrated with ATLAS and CMS software
- Now supports detector simulation for BSM search preservation
- Multi-weights, NLO counter-events, and multi-threading all in the pipeline
- **Feedback, questions and getting involved in development all very welcome!**

- Professor:
  - Parametrisation of computationally expensive functions
  - Inputs can always be parallelised in a trivial way
  - Seamless integration into numerical tools **`iminuit`**, **`pymultinest`** through python bindings
  - Immediately available through **`docker`**
- Professor development used to be driven by Rivet/YODA and MC tuning needs.
- BSM requires more care in parametrisation than tuning.