Intro to the Rivet Analaysis Toolkit

Chris Pollard

University of Glasgow

2016 02 23

A brief introduction to Rivet I

What exactly is Rivet?

- Robust Independent Validation of Experiment and Theory*
- Toolkit for validation of MC generators* written in a combination of C++ and python
- Collection of experimental analyses for MC generator development, validation, and tuning*
- Also provides convenient infrastructure for implementing new analyses*
- This infrastructure is also very useful for testing new analysis ideas/techniques; selecting "good" objects and events (even complex ones) can be a simple one line of c++.

*borrowed directly from Rivet webpage,

`http://rivet.hepforge.org/`

A brief introduction to Rivet II

Additionally...

- Provides histogram storage and rendering through YODA—Yet more Objects for Data Analysis
- Convenience scripts for histogram comparisons and plotting (make-plots, rivet-cmphistos, rivet-mkhtml)...
- ...although there is a very nice python API available for histogram and object manipulation
- Runs on industry-standard (generator independent!!!) HepMC files
- Excellent means of validating MC samples at truth level
- Predict fiducial cross sections
- It has become the standard for propagating unfolded measurements from experiment to theory community at the LHC (and beyond).

A brief introduction to Rivet III

What is Rivet *not*?

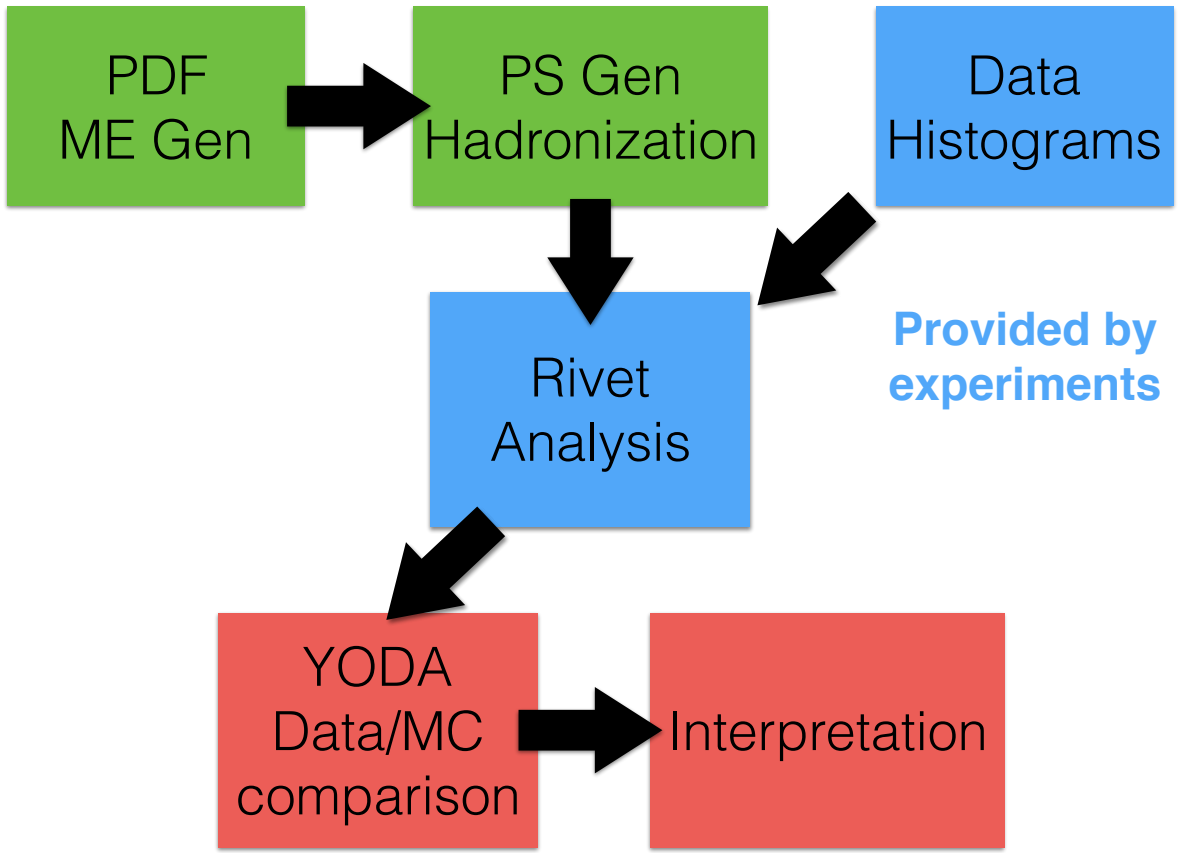
- ROOT. ROOT is a very useful piece of software, but Rivet doesn't have the same use-case.
- Detector simulation. Unless you add your own smearing or efficiency simulation, you are dealing directly with what the generators give you—usually stable particles (more on this later).
- Perfect. We are still improving the available methods and APIs, adding features as new techniques or needs arise, and finding and fixing bugs.
- Exclusive. If you have an improvement idea (or better: implementation), please let us know! We are happy to get feedback from users on what works and doesn't for their needs. (rivet@projects.hepforge.org)

For example...

Let's say that I'm a developer for a parton shower generator, and there was a recent experimental analysis that is sensitive to parameters in my PS model.
How can I use the experimental result?

- Obtain the selection in the form of a Rivet analysis (hopefully already in a release!)
- Obtain the unfolded observable distributions from the experiment (also hopefully in the Rivet release!)
- Generate MC events corresponding to the process probed in the experiment; shower through your PS model using various parameter values (output to hepmc format).
- Run the Rivet analysis over your hepmc files; compare to experimental data.
- You don't have to be an expert in that particular analysis—it should "just work" because it has been validated by the experimentalists!

Flow


PDF
ME Gen
PS Gen
Hadronization
Data
Histograms
Rivet
Analysis
Provided by
experiments
YODA
Data/MC
comparison
Interpretation

A brief Rivet example I

Let's look at a quick example anaysis (that isn't so simple that it's trivial): reconstructing 1-lepton $t\bar{t}$ events
Here's what we need:

- High-p_T (> 25 GeV) leptons
- High-p_T (> 25 GeV) jets
- E_{T}^{miss}

We get these through "projections," which are maps from the initial HepMC event to what you're interested in—final state particles, (b-)jets, events shapes, etc. These are the building blocks of Rivet analyses.

This example is largely based on the MC_TTBAR analysis included with Rivet: http://rivet.hepforge.org/code/1.8.2/a00592_source.html

A brief Rivet example: init()

We need to fill out three methods in our analysis: init(), analyze(), and finalize(). init() is where histograms need to be booked and projections need to be registered:

```
/// Set up projections and book histograms
void init() {

    // A FinalState is used to select particles within |eta| < 4.2 and with pT
    // > 25 GeV, out of which the ChargedLeptons projection picks only the
    // electrons and muons, to be accessed later as "LFS".
    FinalState fs = FinalState(Cuts::abseta < 4.2 & Cuts::pT > 25*GeV);
    ChargedLeptons lfs(fs);
    addProjection(lfs, "LFS");

    // A second FinalState is used to select all particles in |eta| < 4.2,
    // with no pT cut. This is used to construct jets and measure missing
    // transverse energy.
    VetoedFinalState vfs(FinalState(Cuts::abseta < 4.2));

    // But we veto any high-pt charged leptons
    vfs.addVetoOnThisFinalState(lfs);

    addProjection(FastJets(vfs, FastJets::ANTIKT, 0.4), "Jets");

    // Finally, we need the event MET.
    addProjection(MissingMomentum(vfs), "MissingET");

    // Booking of histograms
    _h_njets = bookHisto1D("jet_mult", 11, -0.5, 10.5);
    _h_nbjets = bookHisto1D("bjet_mult", 11, -0.5, 10.5);

}
```

A brief Rivet example: analyze()

The analyze() method holds projections of interesting quantities, event selection, and histogram filling.

```
void analyze(const Event& event) {
    const double weight = event.weight();

    // Use the "LFS" projection to require at least one hard charged
    // lepton.
    const ChargedLeptons& leptons = applyProjection<ChargedLeptons>(event, "LFS");

    if (leptons.chargedLeptons().size() != 1) {
        MSG_DEBUG("Event failed lepton multiplicity cut");
        vetoEvent;
    }

    const MissingMomentum& met = applyProjection<MissingMomentum>(event, "MissingET");

    if (met.vectorEt().mod() < 30*GeV) {
        MSG_DEBUG("Event failed missing ET cut");
        vetoEvent;
    }
```

A brief Rivet example: analyze()

The analyze() method holds projections of interesting quantities, event selection, and histogram filling.

```
    // Use the "Jets" projection to check that there are at least 4 jets of
    // any pT. Getting the jets sorted by pT ensures that the first jet is the
    // hardest, and so on.
    const FastJets& jetpro = applyProjection<FastJets>(event, "Jets");
    const Jets jets = jetpro.jetsByPt(25*GeV);

    if (jets.size() < 4) {
        MSG_DEBUG("Event failed jet multiplicity cut");
        vetoEvent;
    }

    _h_njets->fill(jets.size(), weight);

    // Sort the jets into b-jets and light jets.
    Jets bjets, ljets;
    foreach (const Jet& jet, jets) {
        if (jet.bTagged())
            bjets.push_back(jet);
    }

    _h_nbjets->fill(bjets.size(), weight);
}
```

A brief Rivet example: finalize()

Any post-processing of outgoing information (e.g. histograms) should be performed in finalize().

```
void finalize() {

    double norm = crossSection()/sumOfWeights();
    scale(_h_njets, norm);
    scale(_h_nbjets, norm);
}
```

A brief Rivet example: building, running, and output

Running an analysis like this is relatively simple:

- Build:
  ```
  $ rivet-buildplugin RivetMC_TTBAR.so MC_TTBAR.cc
  ```
- Run:
  ```
  $ rivet --pwd -a MC_TTBAR -H output.yoda input.hepmc
  ```
- Make a webpage with all plots:
  ```
  $ rivet-mkhtml output.yoda -o myPrettyPlots/
  ```

Obviously a lot of details have been left out here, but the "GettingStarted" page has more complete instructions: https://rivet.hepforge.org/trac/wiki/GettingStarted

Recent Rivet developments

There have been some nice additions to the toolkit over the last year:

- Truth-level *b*-tagged jets using ghost-associated B hadrons
- Cuts system for projections:
  `Cuts::abseta < 2.5 && Cuts::pT > 25*GeV`
- New projections, e.g. FinalPartons, which returns all quarks and gluons immediately before hadronization.
- Numerous (about 15) new experimental analyses in each point release
- Preferred truth definitions outlined in recent experimental preprints are largely in line with the default behavior of Rivet methods.

Current and future Rivet developments I

Of course, there are a few things on the horizon as well:

- Automatic handling of events with multiple weights (PDF, scale variations, etc.)
- Automatic handling of NLO sub-events (multiple HepMC events which combine into one collision event)
- In both cases, users will deal with only one copy of each histogram in their analyses as they have in the past, but multi-weighted events and sub-events will be accounted for behind the scenes.
- The output will include e.g. one histogram per PDF set, but there will be an option to only store the nominal histograms.
- First pass is available in Rivet3.0alpha1!

Current and future Rivet developments II

Of course, there are a few things on the horizon as well:

- Combining and extending runs: users should be able to 1) check histograms while an analysis is running and 2) continue a run with a new HepMC file—output will automatically be combined correctly.
- Improved/simpler plotting of YODA objects, in particular moving to matplotlib-based plotting instead of LaTeX.

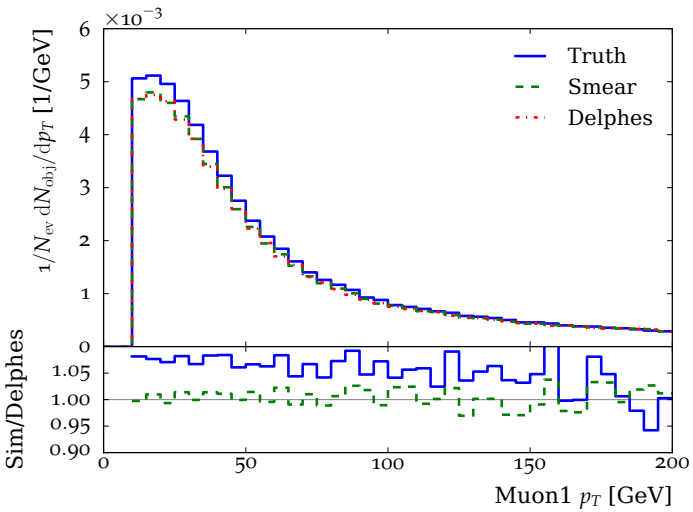
Feedback from the community regarding their usefulness and implementation is welcome!

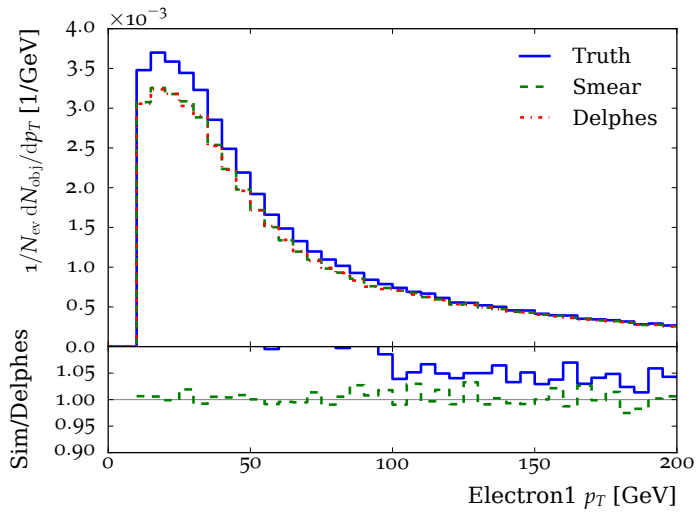
Future Rivet developments: BSM I

- The majority of Rivet analyses provided in releases are for measurements rather than searches.
- However: the Rivet authors are very interested in providing tools to make BSM phenomenology simpler with the infrastructure that is already in place.
- The main idea: analyses would provide as part of their Rivet analyses
  - parameterized object reconstruction efficiencies,
  - parameterized object resolutions,
  - data and background distributions of interest (or data minus background) after reconstruction.
- N.B.: the detector folding is provided *by the experiment for each analysis object in a particular analysis setting*!

Future Rivet developments: BSM II

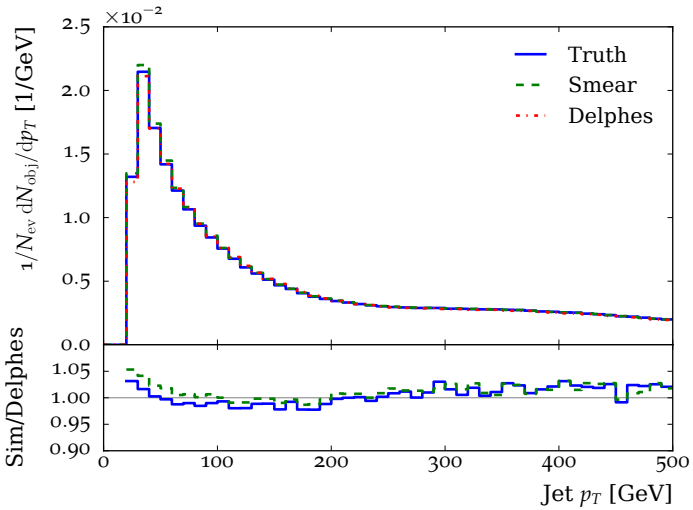
- The strategy outlined above differs from fast detector simulations, e.g. Delphes.
- We think the experiments can provide more precise truth → reco folding than external software.
- For many objects in common topologies fastsim and efficiency/smearing are basically identical.

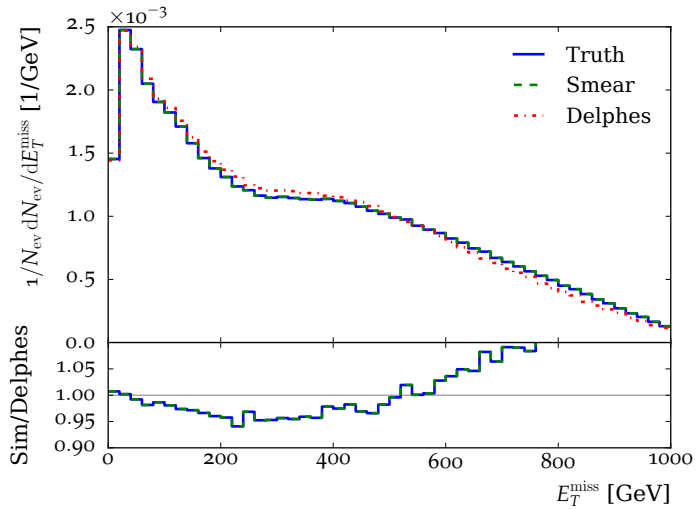




Future Rivet developments: BSM III

- For more complicated objects (jets, E_{T}^{miss}, close-by objects, etc.) and topologies (boosted tops/bosons, etc.) it is very non-trivial to validate external fast simulations programs.
- . . . can we convince the experiments to provide this information? We already have $>$ 200 analyses distributed with Rivet (several BSM) . . .
- There is already a fork of Rivet (Atom) that focuses on BSM searches: expertise exists.





Play time (hopefully)

Before we run an analysis included in the release, let's be sure everything is in place...

```
$ # only if you aren't using the virtual machine
$ mkdir -p ~/Rivet && cd ~/Rivet
$ git clone https://github.com/YETIUK/RivetTutorial.git
$ # also for the virtual machine
$ cd ~/Rivet/RivetTutorial
$ git pull
```

Play time (hopefully)

Now, make sure Rivet is in your `PATH` and run an analysis!

```
$ source ../Rivet-2.4.0/rivetenv.sh
$ rivet --ignore-beams -a ATLAS_2013_I1230812 \
    -H ATLAS_2013_I1230812.yoda PythiaZll.hepmc
```

We are running a 7 TeV ATLAS Z+jets analysis on (13 TeV) Pythia Z+jets events! More info on the analysis can be had from

```
$ rivet --show-analysis ATLAS_2013_I1230812
$ rivet --list-analyses
```

Play time (hopefully)

Our Rivet run should have resulted in a new yoda file. Let's make some plots...

```
$ rivet-mkhtml --mc-errs ATLAS_2013_I1230812.yoda -o myplots/
```

This should make a folder with an html file inside of it, as well as another folder with pdf and png files. Take a look.

```
$ firefox myplots/index.html
```

Play time (hopefully)

Now let's take a look at the analysis we've prepared, LHADA_ZJETS.cc.
Be sure to note what is going on in the init(), analyze(), and finalize() methods.
To build this is relatively straightforward:

```
$ rivet-buildplugin LHADA_ZJETS.cc
```

You should now have a .so file in your directory.

Play time (hopefully)

Try running the analysis over those Z+jets events.

```
$ rivet --pwd -a LHADA_ZJETS -H LHADA_ZJETS.yoda \
    PythiaZll.hepmc
```

Take a look at the new yoda file.

Play time (hopefully)

Now we can make some plots.

```
$ rivet-mkhtml --mc-errs LHADA_ZJETS.yoda -o lhadaplots/
$ firefox lhadaplots/index.html
```

Play time (hopefully)

Try modifying/extending the analysis yourself!

- add jet and Z p_{T} histograms (remember to alter init(), analyze(), and finalize()!!)
- add a $\Delta\eta(Z, j)$ histogram
- lower the minimum jet p_{T} cut to 10 GeV
- require $p_{\mathrm{T},Z} > 15$ GeV
- add dimuon events
- explore...