Rivet

Chris Gütschow (UC London) Deepak Kar (Witwatersrand)
Chris Pollard (Glasgow) Holger Schulz (Durham)

July 13 2016, CERN

Contents

What is Rivet?

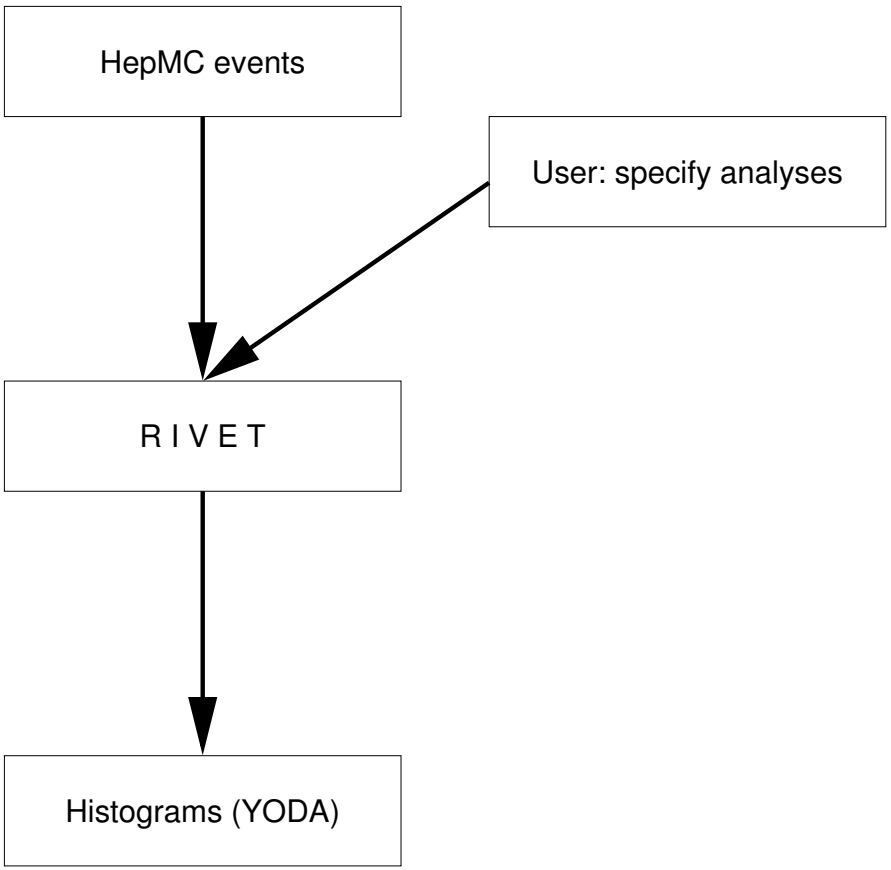
- HEP tool for analyzing Monte Carlo events
- Very capable library of predefined calculators (e.g. event shapes, jets, Z-finder,. . .)
- Analyses intended to be based on physical objects:
  - Final state hadrons
  - Jets (FastJet)
  - Muons, Electrons (dressed)
  - Bosons reconstructed from particles (rather than taken from event record)
- $\rightarrow$ allows for fair comparison with (unfolded) collider data

- Core written in robust C++, elegant and flexible Python scripts for the rest

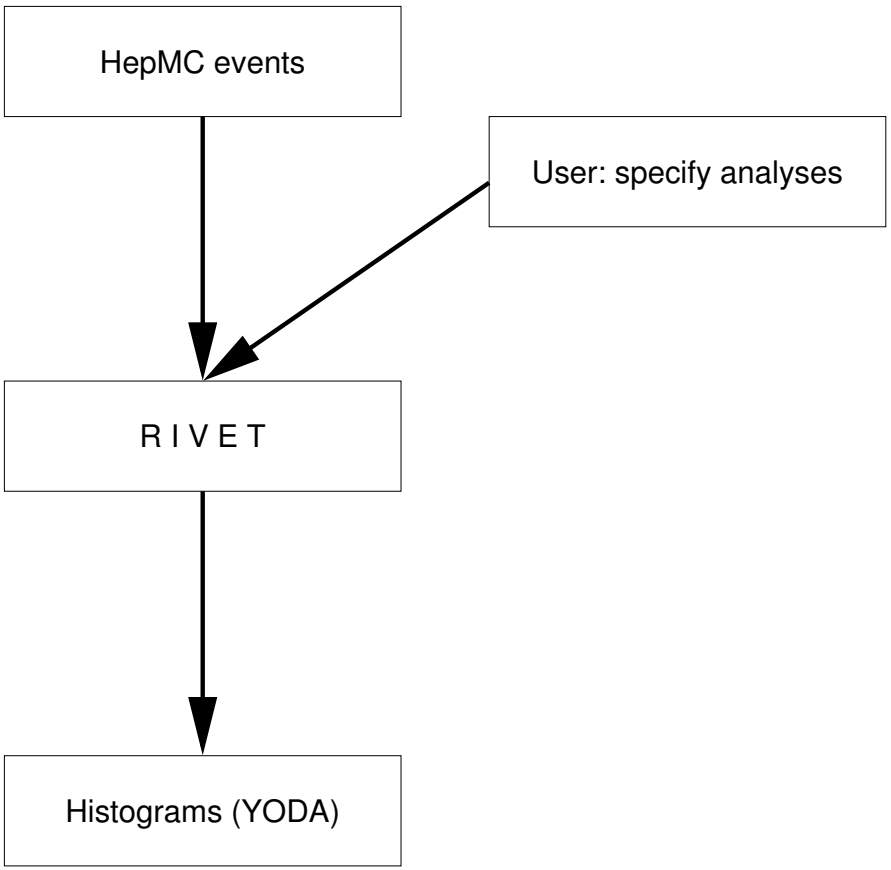
Why use Rivet?

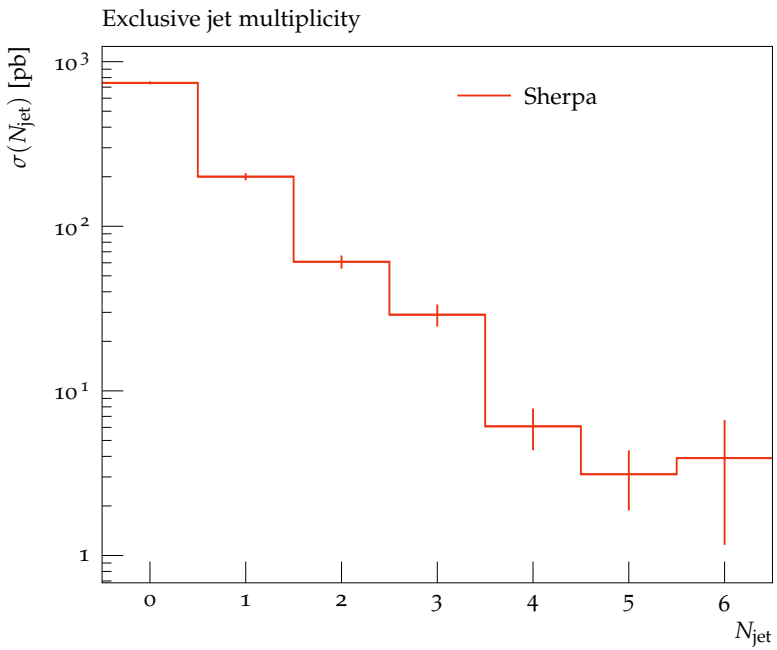
- Generator independent code due to "industry standard" for simulated events (HepMC)
- Version 2.5 contains $\sim$ 350 Analyses (195 LHC)
  - Monte Carlo validation and tuning, data preservation
  - Lots of code examples to get inspired
- Plugin system for new analyses:
  - One file of C++ with clear structure
  - Scripts prepare templates and turn C++ into shared library, download data from HepData if available
  - Very simple to plan new data analyses, test new observables etc.
  - Perfect as independent cross-check for e.g. CMS or ATLAS analysis code
- Lightweight histogramming (YODA — excellent to use in Python), quite ok plotting
- Exploration of BSM physics (UFO + Herwig7/Sherpa)

Basic principle

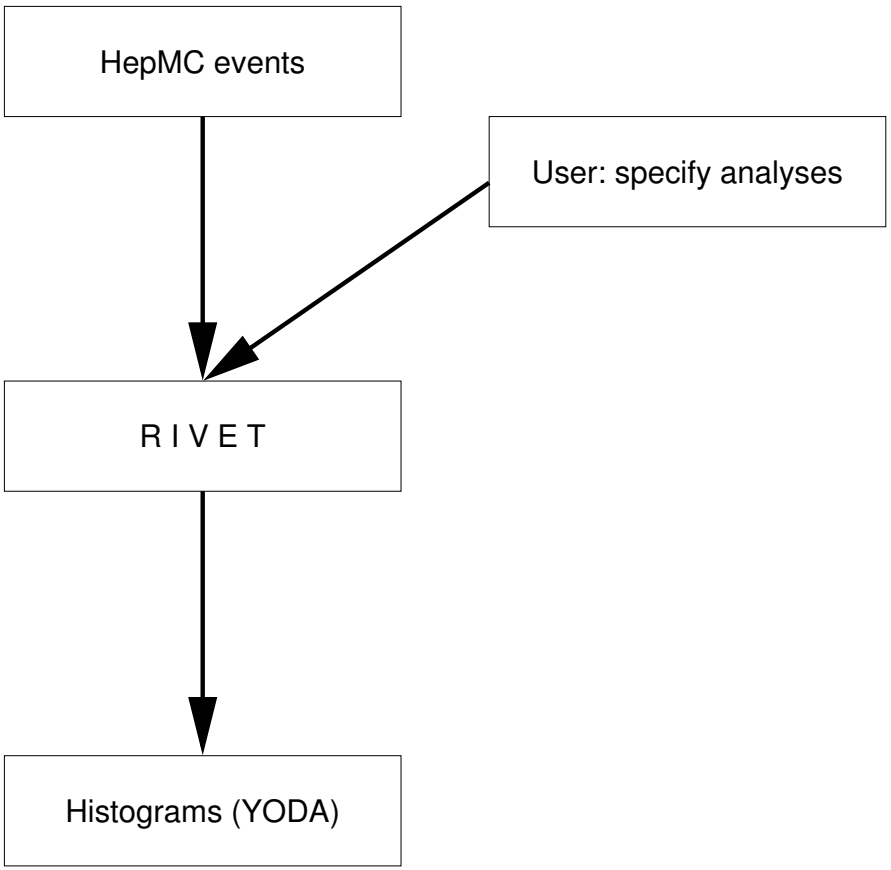


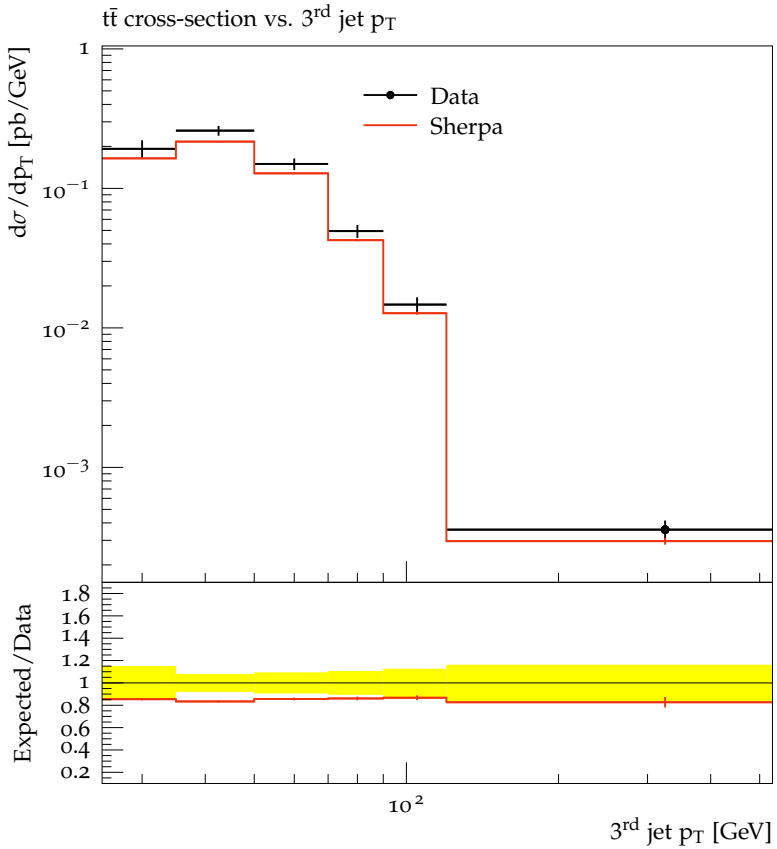
Basic principle





Basic principle





Events

- Several event files are available on lxplus
- 13 TeV Z+jets (muon) inclusive
- 13 TeV Z+jets (muon) with Zpt>200 GeV
- 13 TeV Dark Matter model
  `https://feynrules.irmp.ucl.ac.be/wiki/DMsimp`
- All events are fully showered, hadronised and underlying event

```
1 /afs/cern.ch/work/h/hschulz/public/Events
```

Hands on 1 — on lxplus

```
source /afs/cern.ch/work/h/hschulz/public/setupRivet2.5.0.sh // Set up Rivet 2.5
    with Python 2.7 from LCG

rivet -h

rivet --list-analyses // List precompiled analyses shipped with Rivet

rivet --show-analysis CMS_2015_I1310737 // Detailed info for particular analysis

rivet --show-analysis MC_ZJETS_MU
rivet --show-analysis ATLAS_2015_CONF_2015_041

rivet -a ATLAS_2015_CONF_2015_041 -a MC_ZJETS_MU -H Output.yoda
    INPUTFILE

rivet -a CMS_2015_I1310737 -H Output2.yoda --ignore-beams  INPUTFILE

rivet-mkhtml -h
rivet-mkhtml Output.yoda:'LegendLabel' --mc-errs
```

Analysis skeleton

- `rivet-mkanalysis ANANAME` prepares skeleton
- `rivet-buildplugin RivetANANAME.so ANANAME.cc` compiles analysis plugin
- `rivet EVENTS.hepmc -a ANANAME --pwd` finds and runs analysis plugin

```
1
  void init() {
3     // Declare analysis containers (jets etc), histograms here
  }
5
  void ANANAME::analyze(const Event & e) {
7     // Executed once per event, calc observables, fill histograms here
  }
9
  void ANANAME::finalize() {
11    // Normalise, divide histograms here
  }
13
  private:
15    // Histogram pointers go here
```

Hands on 1 — on lxplus

```
source /afs/cern.ch/work/h/hschulz/public/setupRivet2.5.0.sh // Set up Rivet 2.5
    with Python 2.7 from LCG

rivet -h

rivet --list-analyses // List precompiled analyses shipped with Rivet

rivet --show-analysis CMS_2015_I1310737 // Detailed info for particular analysis

rivet --show-analysis MC_ZJETS_MU
rivet --show-analysis ATLAS_2015_CONF_2015_041

rivet -a ATLAS_2015_CONF_2015_041 -a MC_ZJETS_MU -H Output.yoda
    INPUTFILE

rivet -a CMS_2015_I1310737 -H Output2.yoda --ignore-beams  INPUTFILE

rivet-mkhtml -h
rivet-mkhtml Output.yoda:'LegendLabel' --mc-errs
```

Analysis skeleton

- `rivet-mkanalysis ANANAME` prepares skeleton
- `rivet-buildplugin RivetANANAME.so ANANAME.cc` compiles analysis plugin
- `rivet EVENTS.hepmc -a ANANAME --pwd` finds and runs analysis plugin

```
1
  void init() {
3     // Declare analysis containers (jets etc), histograms here
  }
5
  void ANANAME::analyze(const Event & e) {
7     // Executed once per event, calc observables, fill histograms here
  }
9
  void ANANAME::finalize() {
11    // Normalise, divide histograms here
  }
13
  private:
15    // Histogram pointers go here
```

- http://rivet.hepforge.org/code/dev/annotated.html

Analysis objects

```
1  void init() {
       _hist=bookHisto1D("hist_met", 14, 0., 700);
3      _p1d=bookProfile1D("JetPt_vs_JetEta", 50, 10, 120);
       _c=bookCounter('passedCutXY');
5  }

7  void ANANAME::analyze(const Event & e) {
       double weight =  e.weight()
9      _hist->fill(MET, weight);
       _p1d->fill(jet.pt(), jet.eta(), weight);
11     _c->fill(weight);
   }
13 void ANANAME::finalize() {
       _hist->scale(crossSection()/femtobarn*20.3/sumOfWeights());
15 }
   private:
17   Histo1DPtr _hist;
     Profile1DPtr _p1d;
19   CounterPtr _c;
```

Note: Everything that is booked is automatically written out to file.

Projections

- Predefined calculators
- Declaration in `init()`, application to event ("project") in `analyze`

```
#include "Rivet/Projections/FastJets.hh"

void init() {
  FinalState fs;
  FastJets jetpro(fs, FastJets::ANTIKT, 0.4);
  declare(jetpro, "Jets");
}

void ANANAME::analyze(const Event & e) {
  const Jets& jets = apply<FastJets>(e, "Jets").jetsByPt(Cuts::pT>30*GeV);
  foreach(const Jet& j, jets) {
      if (j.bTagged()) { // Is there a ghost associated B-hadron
          //
      }
  }
}
```

FinalStates

- ChargedFinalState
- NeutralFinalState
- UnstableFinalState
- IdentifiedFinalState
- VetoedFinalState
- DISFinalState
- VisibleFinalState
- HadronicFinalState

```
#include "Rivet/Projections/FastJets.hh"
#include "Rivet/Projections/IdentifiedFinalState.hh"
#include "Rivet/Projections/VetoedFinalState.hh"

void init() {
  IdentifiedFinalState neutrinoFS(fs);
  neutrinoFS.acceptNeutrinos();
  declare(neutrinoFS, "Neutrinos");

  VetoedFinalState jetinput;
  jetinput.addVetoOnThisFinalState(neutrinoFS);

  FastJets jetpro(jetinput, FastJets::ANTIKT, 0.4);
  declare(jetpro, "Jets");
}
```

Plotting

- rivet-mkhtml MC1.yoda:'Signal1' MC2.yoda:'Signal2' -o plots
- .plot file

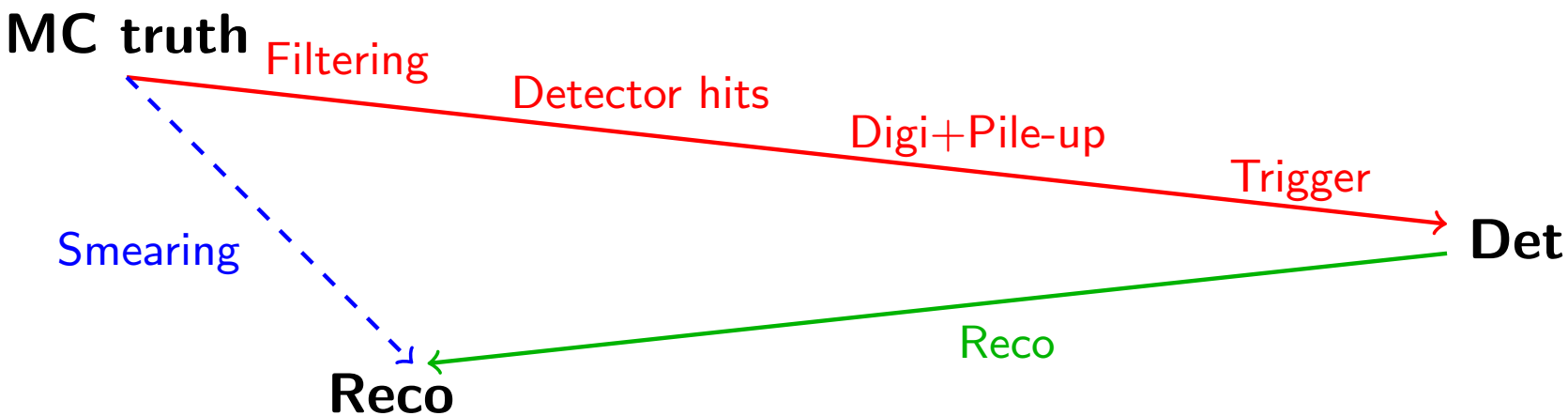
```
BEGIN PLOT /ATLAS_2015_I1343107/d18-x01-y01
XLabel=$E^{\rm{miss}}_T$ [GeV]
YLabel=Events
XMin=150
END PLOT
```

New in Rivet 2.5

- Even simpler projections syntax
- Basic modelling of detector repsonses
- No more boost dependency (pure C++11)

Modelling detector response

- Huge wealth of searches available — would like to preserve properly to allow amongst others reinterpration
- Data preservation of non-unfolded analyses: approximate repsonse (can be made public)
- Need to encode efficiencies and response of physics objects



Implementation

```
1  inline Jet MY_JET_SMEAR_ATLAS_RUN1(const Jet& j) {
     static random_device rd;
3    static mt19937 gen(rd());
     // Const fractional resolution for now
5    static const double resolution = 0.03;
     // Smear by a Gaussian centered on 1 with width given by the (fractional) resolution
7    normal_distribution<> d(1., resolution);
     const double fsmear = max(d(gen), 0.);
9    if (j.mass() <0) return Jet(FourMomentum::mkXYZM(j.px()*fsmear, j.py()*fsmear,
       j.pz()*fsmear, 0.));
     return Jet(FourMomentum::mkXYZM(j.px()*fsmear, j.py()*fsmear, j.pz()*fsmear, j.
       mass()));
11 }
```

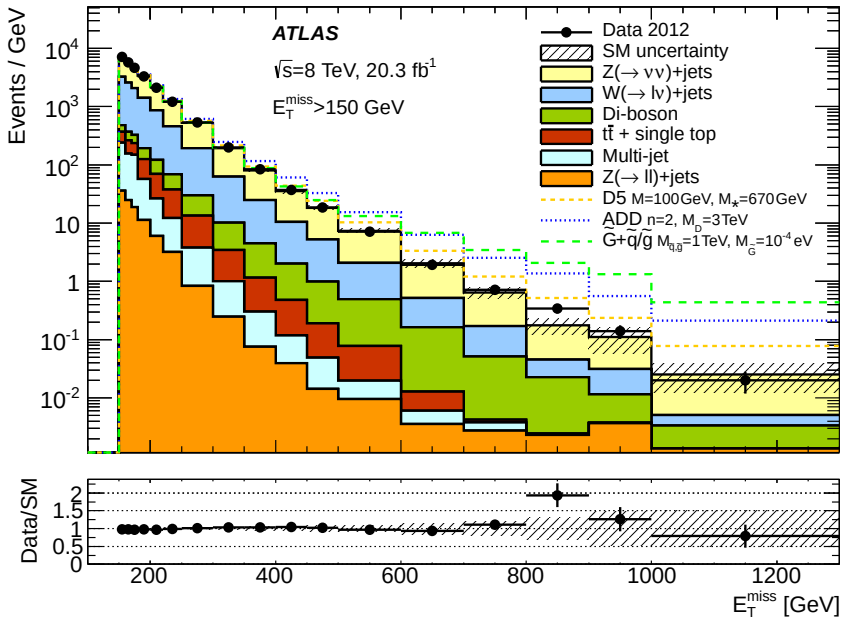
- Some adhoc jet resolution smearing

Implementation

```
// This is in init()
    FastJets fj(FinalState(Cuts::abseta < 5), FastJets::ANTIKT, 0.4);
    declare(fj, "Jets");

    SmearedJets sj(fj, MY_JET_SMEAR_ATLAS_RUN1,
      [](const Jet& j){ return j.bTagged() ? 0.7*(1 - exp(-j.pT()/(10*GeV))) :
    0.01; } );
    declare(sj, "SmearedJets");

    _hist_met = bookHisto1D(18,1,1); // Binning read from data file entry d18-x01-
    y01

// This is in analyze()
    const Jets& jets = apply<JetAlg>(event, "SmearedJets").jetsByPt(Cuts::pT >
    30.0*GeV && Cuts::abseta < 4.5);
```

- The SmearedJet projection takes a smearing and an efficieny function
- The latter uses C++11 lambda syntax (any function pointer would work)

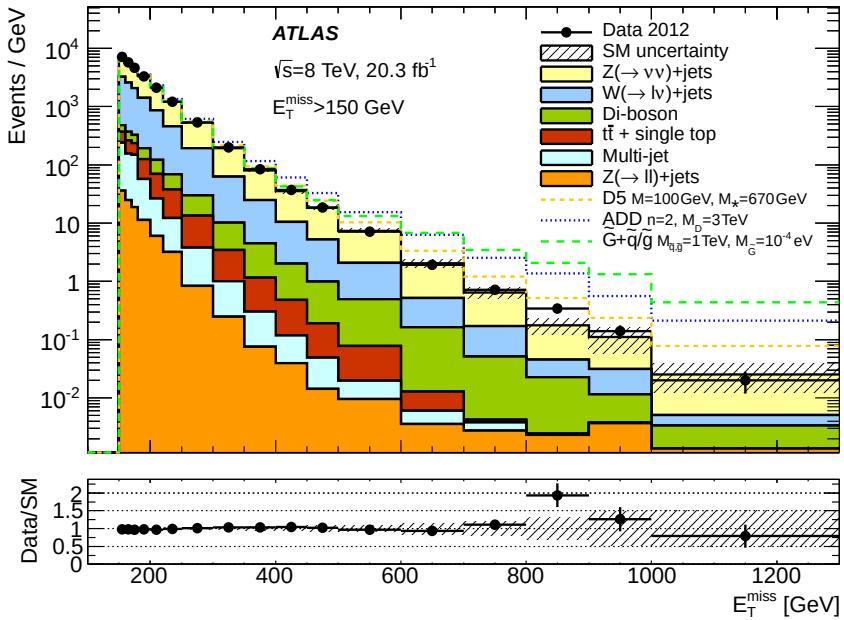
Hands on 2



- Monojet search (`https://arxiv.org/pdf/1502.01518v2.pdf`)
- Data in signal region and total background available on HepData
- Prepared skeleton (`rivet-mkanalysis ATLAS_2015_I1343107`)

- See `/afs/cern.ch/work/h/hschulz/public/Monojet`
- `rivet-buildplugin RivetATLAS_2015_I1343107.so ATLAS_2015_I1343107.cc`
- `rivet --pwd -a ATLAS_2015_I1343107 INFILE`

Hands on 2



- Monojet search (`https://arxiv.org/pdf/1502.01518v2.pdf`)
- Data in signal region and total background available on HepData
- Prepared skeleton (`rivet-mkanalysis ATLAS_2015_I1343107`)

Preselection
Primary vertex
$E_{\mathrm{T}}^{\mathrm{miss}} > 150$ GeV
Jet quality requirements
At least one jet with $p_{\mathrm{T}} > 30$ GeV and $\lvert\eta\rvert < 4.5$
Lepton and isolated track vetoes
Monojet-like selection
The leading jet with $p_{\mathrm{T}} > 120$ GeV and $\lvert\eta\rvert < 2.0$
Leading jet $p_{\mathrm{T}}/E_{\mathrm{T}}^{\mathrm{miss}} > 0.5$
$\Delta\phi(\mathrm{jet}, \mathbf{p}_{\mathrm{T}}^{\mathrm{miss}}) > 1.0$

Final remarks

- `rivet.hepforge.org` Getting started — super convenient bootstrap script

- BSM tool chain extremely easy via UFO + Herwig7/Sherpa + Rivet